

---

# **AWS Certified Solutions Architect – Associate**

***Release 1.0***

**Jose Antonio Alvarez Cubero**

**Nov 03, 2020**



---

## Contents

---

<b>1</b>	<b>Contents:</b>	<b>1</b>
1.1	Preface . . . . .	1
1.2	Introduction . . . . .	2
1.3	Storage services . . . . .	21
1.4	Amazon EC2 . . . . .	108
1.5	Database services . . . . .	124
1.6	Networking in AWS . . . . .	156
1.7	AWS Identity and Access Management (IAM) . . . . .	185
1.8	Elasticity, High Availability, and Monitoring . . . . .	192
1.9	Automation . . . . .	209
1.10	Caching . . . . .	214
1.11	Building decoupled architectures . . . . .	220
1.12	Microservices and serverless architectures . . . . .	231
1.13	RTO/RPO and backup recovery setup . . . . .	239
1.14	Optimizations and review . . . . .	239
1.15	Appendix . . . . .	240
1.16	Domains . . . . .	251
<b>2</b>	<b>Indices and tables</b>	<b>267</b>





## 1.1 Preface

### 1.1.1 Authors

- Jose Antonio Alvarez Cubero

#### Who should read this book?

#### Expected Audience

The intended audience for this book is those with a general need to understand main AWS services. While AWS certification candidates may experience the largest benefit from this content, the materials included herein may be of use to a much wider audience, especially given modern industry trends towards AWS cloud computing technologies.

There are many elements in this book that explore topics outside the typical contents of AWS certifications. For example the use of AWS python SDK can be viewed as a development-oriented task, however has specific relevance to cloud computing configuration and function, taking a very different approach than traditional CLI-based interface configuration.

#### Organization of this book

This book has the same modules as the official course **Architecting on AWS**.

#### Purpose of this book

This book complements the material given in the course **Architecting on AWS** with additional information and illustrates how to obtain programmatically relevant data about AWS services.

## Resources for preparing the exam

Practice questions

Solutions Architect Associate Bootcamp

## 1.2 Introduction

### 1.2.1 Cloud computing

In this section, I outline the definition of cloud computing given by NIST and map its essential characteristics, service and deployment models with the AWS offering.

#### Definition

The National Institute of Standards and Technology (NIST) comprehensively outlined the definition of Cloud Computing in their September 2011 [NIST SP 800-145](#) publication:

*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

What is cloud computing?

This cloud model is composed of five essential characteristics, three service models, and four deployment models.

#### Essential Characteristics

##### 1. On-demand self-service

*A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.*

To access Amazon Web Services (AWS) Cloud services, you can use the AWS Management Console, the AWS Command Line Interface (CLI), or the AWS Software Development Kits (SDKs). These services are provisioned automatically without any human intervention from AWS staff.

When you create a user, that user has no permissions by default. To give your users the permissions they need, you attach policies to them. It applies also to controlling user access to the AWS Management Console, AWS CLI or AWS SDKs.

##### 2. Broad network access

*Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).*

The AWS Management Console has been designed to work on tablets as well as other kinds of devices:

- Horizontal and vertical space is maximized to show more on your screen.
- Buttons and selectors are larger for a better touch experience.

The AWS Management Console is also available as an app for Android and iOS. This app provides mobile-relevant tasks that are a good companion to the full web experience. For example, you can easily view and manage your existing Amazon EC2 instances and Amazon CloudWatch alarms from your phone.

You can download the AWS Console mobile app from Amazon Appstore, Google Play, or iTunes.

### 3. Resource pooling

*The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.*

secGlobalInfrastructure

### 4. Rapid elasticity

*Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.*

secAWSAutoScaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Currently, the supported applications are: secECS, secEMR, secEC2, secAppStream, secDynamoDB, secRDS, secSageMaker, secComprehend.

Any service that you build with adjustable resource capacity can be automatically scaled using the new Custom Resource Scaling feature of Application Auto Scaling. To use Custom Resource Scaling, you register an HTTP endpoint with the Application Auto Scaling service, which will use that endpoint to scale your resource.

### 5. Measured service

*Cloud systems automatically control and optimize resource use by leveraging a metering capability [1]\_ at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.*

secCloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. A namespace is a container for CloudWatch metrics. For the list of AWS namespaces, see [AWS Services That Publish CloudWatch Metrics](#).

CloudWatch ServiceLens enhances the observability of your services and applications by enabling you to integrate traces, metrics, logs, and alarms into one place. ServiceLens integrates CloudWatch with AWS XRay to provide an end-to-end view of your application to help you more efficiently pinpoint performance bottlenecks and identify impacted users. A service map displays your service endpoints and resources as “nodes” and highlights the traffic, latency, and errors for each node and its connections. You can choose a node to see detailed insights about the correlated metrics, logs, and traces associated with that part of the service. This enables you to investigate problems and their effect on the application.

You can use Event history in the secCloudTrail console to view, search, download, archive, analyze, and respond to account activity across your AWS infrastructure. This includes activity made through the secConsole, secCLI, and AWS SDKs and APIs. CloudTrail is enabled by default for your AWS account.

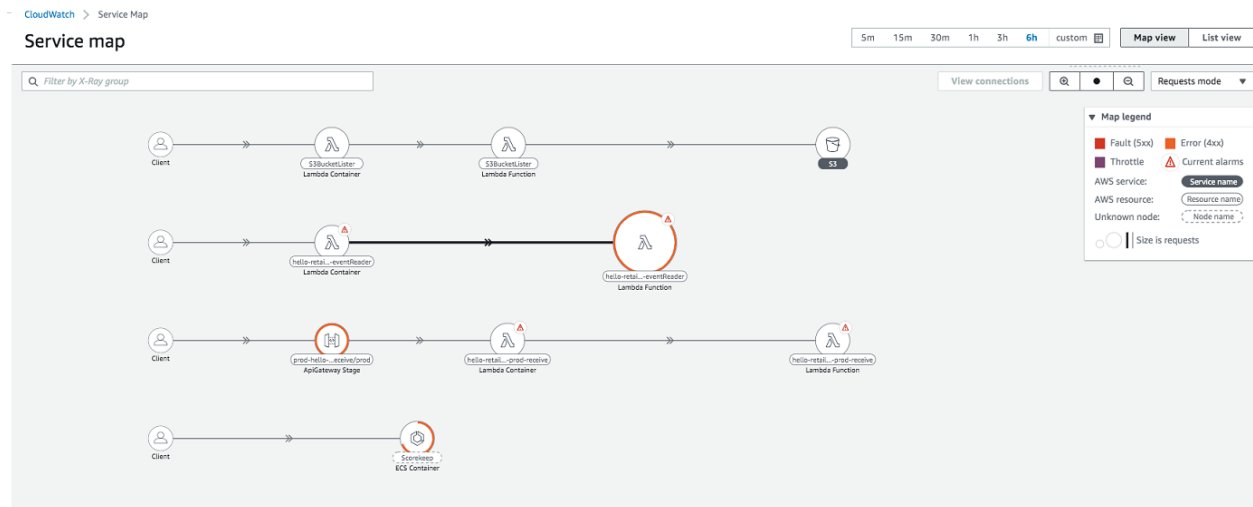


Fig. 1: Service Map

## Service Models

### Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure [2]. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited userspecific application configuration settings.

AWS offers SaaS solutions through [AWS Marketplace](#). AWS Marketplace is a digital catalog with thousands of software listings from independent software vendors that make it easy to find, test, buy, and deploy software that runs on AWS.

### Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider [3]. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

AWS has a PaaS service called [Elastic Beanstalk](#).

### Infrastructure as a Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

Most of AWS services are IaaS.

## Deployment Models

- **Private cloud.** The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
- **Community cloud.** The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.
- **Public cloud.** The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
- **Hybrid cloud.** The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

Most of AWS Services are provisioned using a Public cloud deployment model. However, [AWS Outposts](#) bring native AWS services, infrastructure, and operating models across on premises and the AWS cloud to deliver a truly consistent hybrid experience. AWS Outposts is designed for connected environments and can be used to support workloads that need to remain on-premises due to low latency or local data processing needs.

AWS Outposts come in two variants:

1. VMware Cloud on AWS Outposts allows you to use the same VMware control plane and APIs you use to run your infrastructure.
2. AWS native variant of AWS Outposts allows you to use the same exact APIs and control plane you use to run in the AWS cloud, but on premises.

AWS Outposts infrastructure is fully managed, maintained, and supported by AWS to deliver access to the latest AWS capabilities. Getting started is easy, you simply log into the AWS Management Console to order your Outpost, choosing from a wide catalog of Amazon EC2 instances and capacity and EBS storage options.

## Six advantages of cloud computing

How AWS came to be

Six Advantages of Cloud Computing

The Six Main Benefits of Cloud Computing with Amazon Web Services

## AWS cloud benefits

- No up-front investment
- Low on-going costs
- Focus on innovation
- Flexible innovation
- Speed and agility
- Global reach on demand

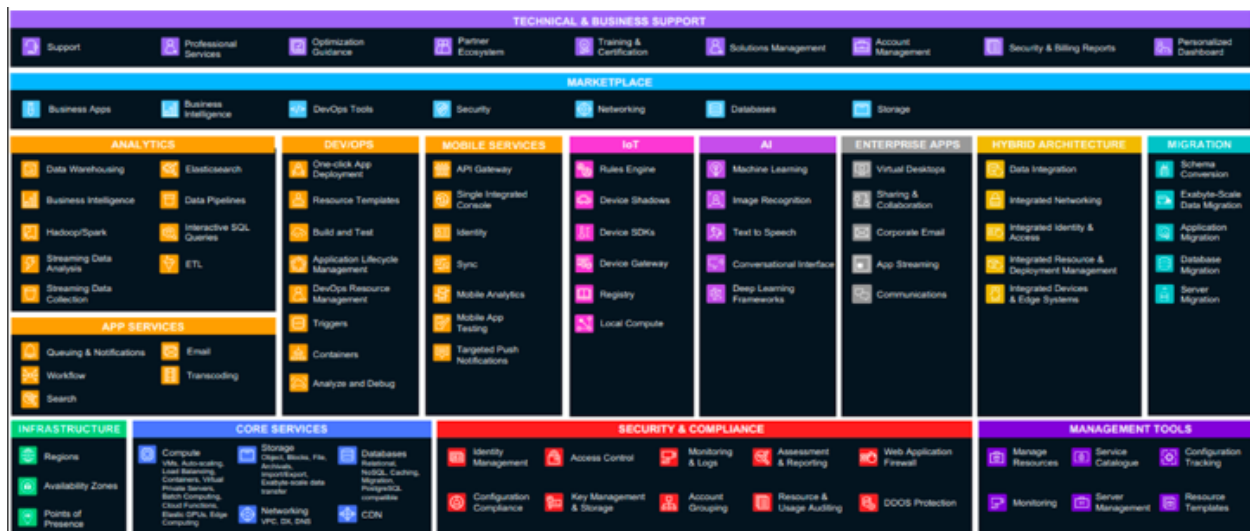
## On-premises considerations

- Large upfront capital expense
- Labor, patches and upgrade cycles
- System administration
- Fixed capacity
- Procurement and setup
- Limited geographic regions

## 1.2.2 Overview of Amazon Web Services

Overview of Amazon Web Services

### AWS products and services



## The management continuum

It is important to consider how much of the infrastructure the customer wants to manage.

1. On one end of the management continuum is a self-managed infrastructure. In this approach, the customer manages the entirety of the infrastructure stack, including the hardware, the software, hypervisor, operating systems, applications, network, patching, upgrades, real state, power, cooling and staffing. This is the approach customers take when they manage their own data centers.
2. The next level is a partially managed service, like Amazon EC2. When a customer chooses EC2, they only need to manage the OS and software applications. In essence, the customer manages their virtual machine and AWS takes care of the rest. The customer can choose the amount of resources to allocate to a virtual machine, including CPU, memory, network, and storage amounts, as well as other configuration options, such as the inclusion of GPUs.

- On the other end of the management continuum, the customer can opt for a fully managed service, such as Amazon RDS. In this case, the customer can focus on the characteristics of the service and not on the infrastructure being used to deliver the service.

As the customer moves across the management continuum, they increasingly shift their focus away from infrastructure and more toward the application.

## Managed services

An architectural consideration unique to cloud solutions is service scope. The scope of any AWS service falls into one of 3 categories: global, regional or zonal.

- An instance of a **global service**, such as Amazon Route 53, spans the entire AWS global infrastructure, including all Regions simultaneously.
- An instance of a **regional service**, such as Amazon Aurora managed database service, is contained in a single AWS Region, but it can span all AZs simultaneously.
- An instance of a **zonal service**, such as Amazon EC2, is contained in a single AZ.

## AWS pricing

### AWS cost fundamentals

There are 3 fundamental characteristics you pay for with AWS: compute capacity, storage and outbound data transfer. The outbound data transfer is aggregated across Amazon EC2, S3, RDS, SimpleDB, SQS, SNS, and VPC and then charged at the outbound data transfer rate. These characteristics vary depending on the AWS product you are using. This charge appears on the monthly statement as AWS Data Transfer Out. There are no charge for inbound data transfer between other services within the same region.

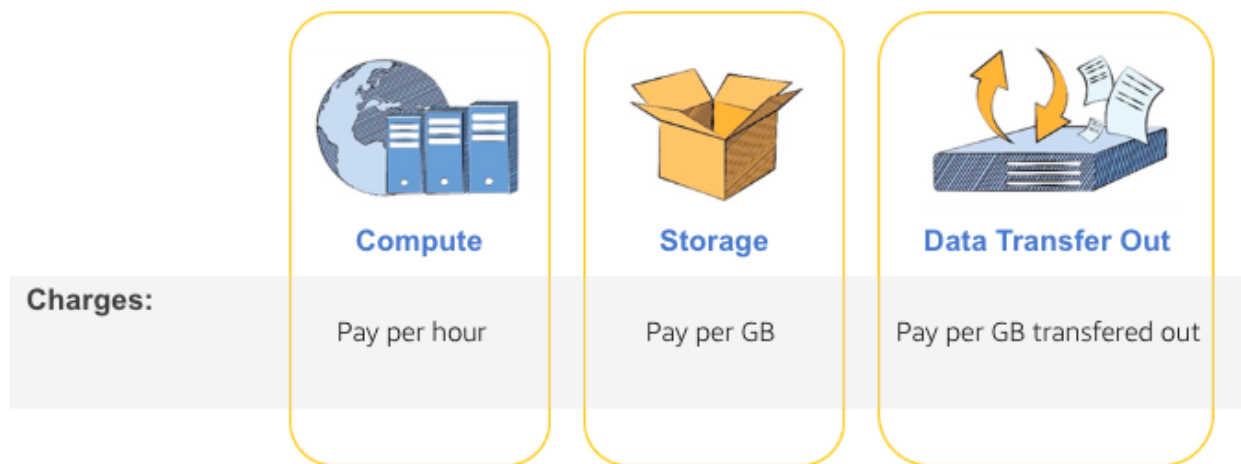


Fig. 2: Pricing characteristics

Prices for many services are tiered, so that you get a volume discount. The more you use it, the cheaper the service you get.

## Consumption-based model

With a consumption-based model, AWS customers pay only for what they use rather than pay upfront for what they think they will need in the future.

When a customer overprovisions, they order more servers than they need, and they end up with unused resources. When a customer underprovisions, they order too few servers and fail to deliver on their SLAs. In both cases, the trade-off is inefficiency and financial loss.

AWS helps their customers avoid inefficiencies and financial losses by enabling their customers to launch the infrastructure they need when they need it. With this model, overprovisioning and underprovisioning are no longer issues. Customers align their infrastructure, services, and costs to usage patterns.

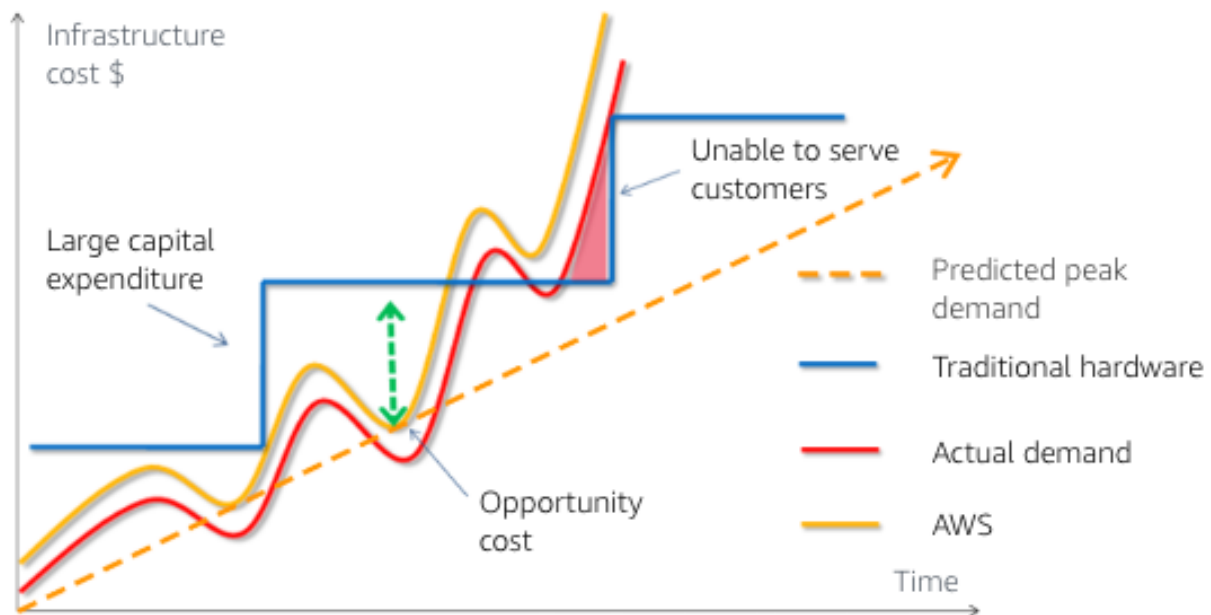


Fig. 3: Provisioning infrastructure

## AWS pricing models

AWS pricing models offer a second way AWS can help their customers lower costs. Customers can buy from traditional vendors using various payment plans, yet in the end, they are stuck with the same set of servers until their next refresh. With AWS, customers can optimize the usage model to the requirements of workloads by maturity and behavior.

By *maturity*, we mean, “Does the customer have enough data to determine a usage baseline so they can use a Reserve Instance?” When we say *behaviour*, we mean, “Is data usage too spiky for Reserved Instances? Will Spot Instances work better?”.

Typically, customers start with On-Demand Instances to gauge their needs, and then switch to Reserved Instances (RIs), once they know their demand baseline. Reserved Instances reduce costs by up to 75% versus On-Demand.

AWS excess capacity sold for a deep discount is called a Spot Instance.

Finally, for customers with regulatory or software licensing constraints who need their own hosts, AWS provides Dedicated Hosts.



How do you pay for AWS?

How AWS Pricing Works

## Tools

1. [Simple monthly calculator](#). It will be replaced by [AWS Pricing Calculator](#)
2. Each AWS service has a detailed pricing page.
3. To programmatically access pricing details, use the [AWS Price List API](#). Customers can [set up notifications](#) to receive alerts when AWS changes prices, such as for adds, updates, and removals.
4. [AWS Cost Explorer](#) has improved trend-based forecasting, based on machine learning, and rules-based models to predict spend across charge types.

## Frequent price reductions

The third way AWS lowers costs is by continually reducing prices. AWS operates on a massive scale, allowing them to increase efficiencies and reduce costs. AWS then passes the savings to their customers through price reductions. Every time AWS reduces its prices, all AWS customers experience the savings immediately.

When price reductions occur, customers automatically receive the lower prices. They never need to call AWS, contact an account manager, renegotiate a contract, or ask for the price reduction.

One exception to price reductions is Reserved Instances. RIs are already heavily discounted with prices accounting for forecasted future discounts. With planning, customers can benefit from both worlds.

## Amazon flywheel

The Amazon flywheel, created by Jeff Bezos, shows the Amazon business model of scale. It illustrates how the Amazon high-volume, low-margin business model works. As you can see, it's a positive feedback loop driven by price reductions.

This is the AWS version of the Amazon flywheel. AWS offers lower prices, which means more customers will use AWS platforms. AWS then increases its infrastructure size, which improves AWS economies of scale. AWS can then lower infrastructure costs and lower prices for customers again. And this cycle continues.

Another aspect of economies of scale is that AWS can offer services that a company might not be able to develop, support, or run in an efficient manner. These services might be relevant for only a limited number of customers, or they might not make sense for customers to develop on their own, yet with AWS scale, the services become inexpensive and easy-to-use. By using AWS, companies can focus more on innovation and less on infrastructure.

## Resources for cost analysis

[TCO calculator](#)

[AWS Economics Center](#)

[Case Studies and Research](#)

**TSO Logic** was acquired December 2018 by Amazon for their ability to quickly deliver an optimized business case for AWS. TSO Logic's analytics software gives customers the answers they need to make sound cloud planning, migration, and modernization decisions. Offered as a service, TSO Logic gives customers an understanding of how much compute they have, how it's used, what it costs to operate, and the projected costs of running on AWS. It also

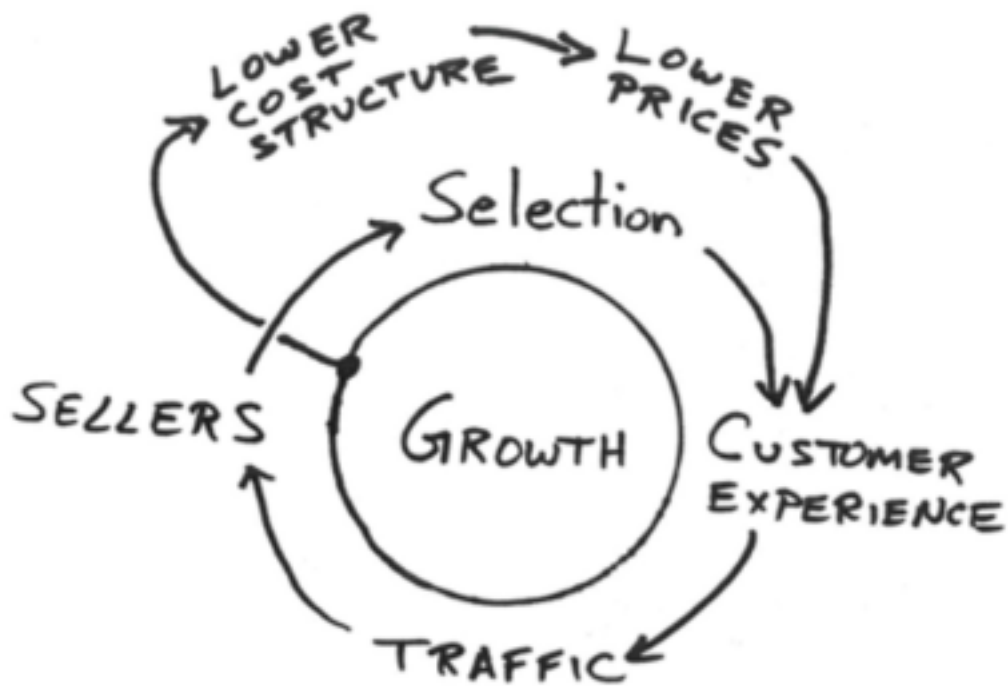


Fig. 4: Amazon flywheel

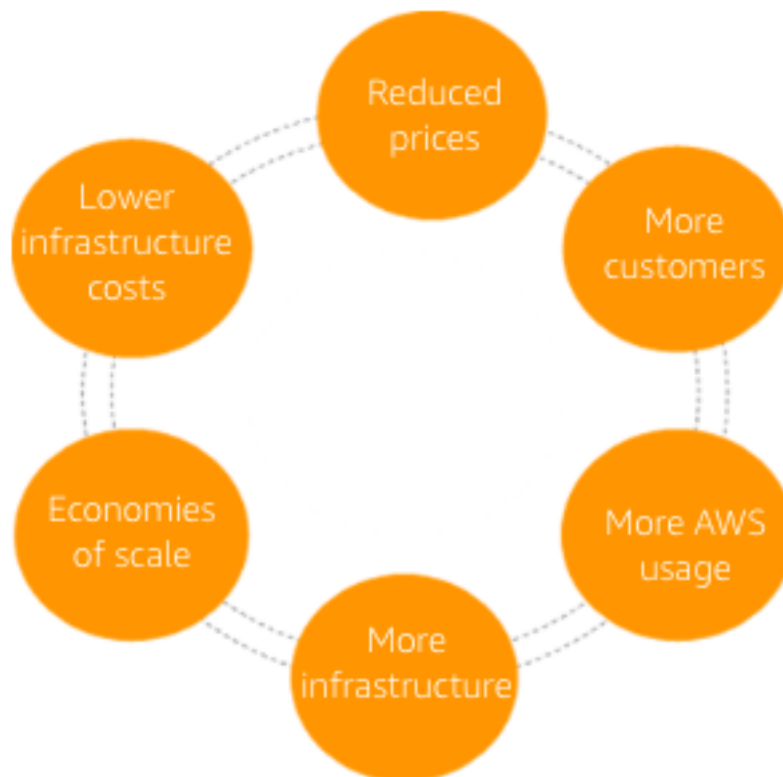


Fig. 5: AWS flywheel

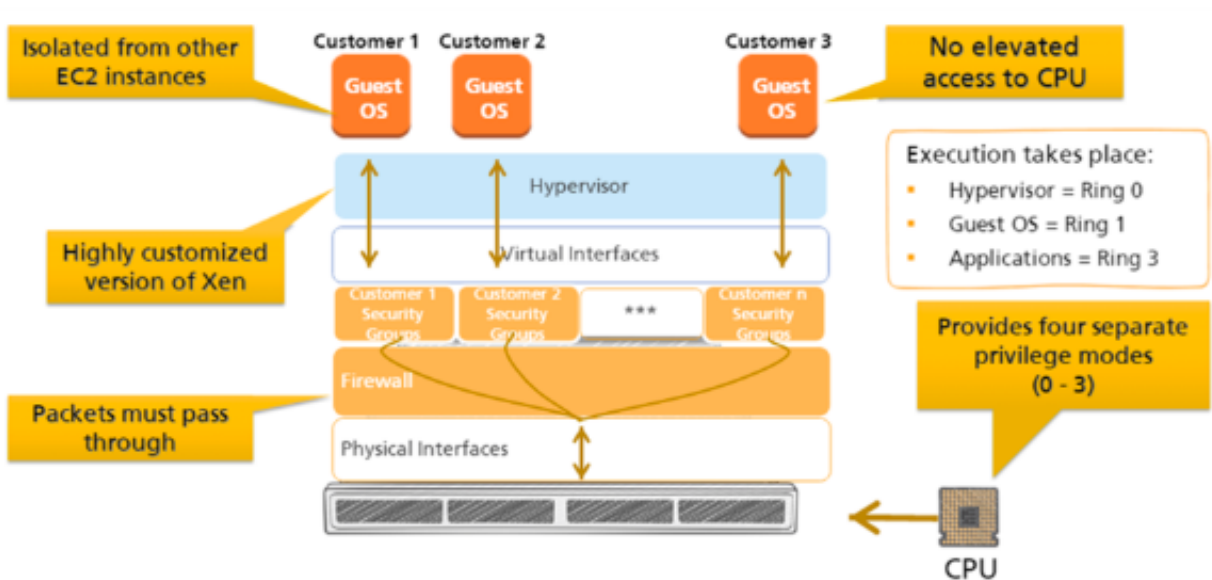
shows where on-premises instances are overprovisioned and where alternate AWS placements can meet or exceed those requirements at a lower cost. TSO Logic is available to AWS APN Partners based on certain qualifications.

**Migration Portfolio Assessment (MPA)** tool include the following key features:

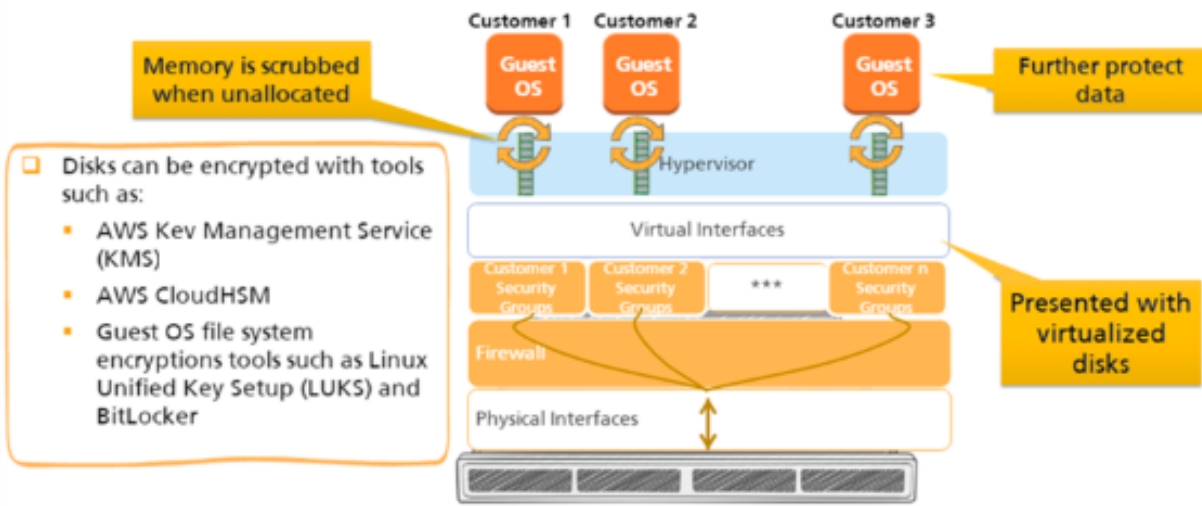
- Guided data ingestion.
- Right-sized EC2 instance and storage recommendations based on performance profiles.
- Run rate cost comparisons between current state and recommended AWS.
- Over 100 variables with pre-populated industry defaults to help dial in current costs, migration costs, and future run rates.
- Migration pattern analysis and recommendations.
- Migration project estimating for resources, timelines, and costs.
- Customizable dashboards to visualize data.
- Secure access for Advanced and Premier tier AWS APN Partners.

## Security of the cloud

### AWS compute security



## AWS memory and storage protection

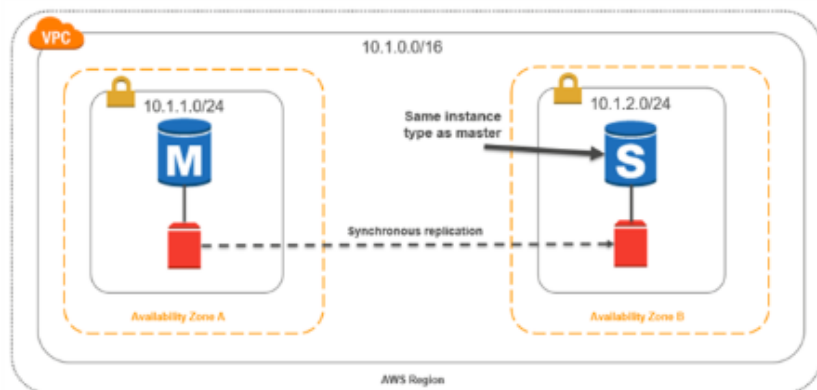


## Database security



Amazon RDS

- DB security groups
- SSL connections
- Snapshots
- Multi-AZ deployments
- VPC deployment
- Encryption
- SSL encrypted endpoints



## AWS Support Plans

AWS support offers 4 support plans:

- Basic support.
- Developer support.
- Business support.
- Enterprise support.

[Compare AWS Support Plans](#)

### 1.2.3 The AWS Well-Architected Framework

AWS Well-Architected Framework

AWS Well-Architected Framework whitepaper

AWS Architecture Center

The AWS Well-Architected Framework covers:

- Strategies and best practices for architecting in the cloud.
- It provides you with a way to measure your architecture against AWS best practices.
- Identifies how to address any shortcomings.

The AWS Well-Architected Framework enables the customer:

- To make informed decisions about their architecture.
- Think in a cloud-natively way.
- Understand the impact of design decisions that are made.

The purposes of AWS Well-Architected Framework are the following:

- Increases awareness of architectural best practices.
- Addresses foundational areas that are often neglected.
- Provides a consistent approach to evaluating architectures

It is composed of Questions, Pillars and Design Principles.

#### General design principles

The AWS Well-Architected Framework includes many best-practices design principles for AWS solutions. These principles may be general, such as the importance of enabling traceability. But they also include pillar-specific design principles, such as maintaining defining security responses based on traced information, an automating responses when possible.

The general design principles in a on-premises environment are:

- *You have to guess the infrastructure needs.* It is often based on high level business requirements and demand and often before a line of code is written.
- *You cannot afford to test at scale.* A complete duplicate of production costs is hard to justify, especially with low utilization. So when you go to production, you normally find a whole new class of issues at high scale.
- *You fear make significant architectural changes* because you do not have no way to test it properly. It would stop you from delivering other features because of your environment is single pipeline.
- *Experiments* were PoC at the start. It is *hard to justify* later due to the effort of getting resources to try different things out. You manually build and customize which is hard to reproduce.
- *Your architecture is frozen in time.* Eventhough everything progresses, such as your user are doing, the requirements and even the business model.

The general design principles in an AWS environment are different. Customers can:

- *Stop guessing capacity needs.*
- *Test systems at production scale.*
- *Automate to make experimentation easier.*

- *Allow architectures to evolve.*
- *Build data-driven architectures.*
- *Improve through game days.*

A **workload** is defined as a collection of interrelated applications, infrastructure, policy, governance, and operations running on AWS that provide business or operational value.

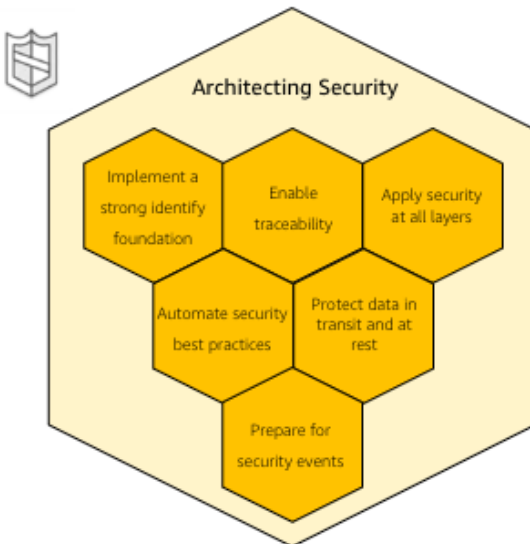
## Security

### Security pillar

Security is the ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies. The focus areas are:

- Identity and access management
- Detective controls
- Infrastructure protection
- Data protection
- Incident response

## Design Principles



### Pillar Design Principles

- Implement a strong identify foundation
- Enable traceability
- Apply security at all layers
- Automate security best practices
- Protect data at rest and in transit
- Prepare for security events with incident simulations

- Implement security at all layers
- Enable traceability: To log and audit all actions and changes to an environment, to automatically respond and take action.
- Apply principle of least privilege: persons (or processes) can perform all activities they need to perform, and no more.
- Focus on securing your system
- Automate responses to security events

## Reliability

### Reliability Pillar

Reliability is the ability of a system to recover from infrastructure or service failures, dynamically acquire computing resources to meet demand, and mitigate disruptions, such as misconfigurations or transient network issues. The focus areas are:

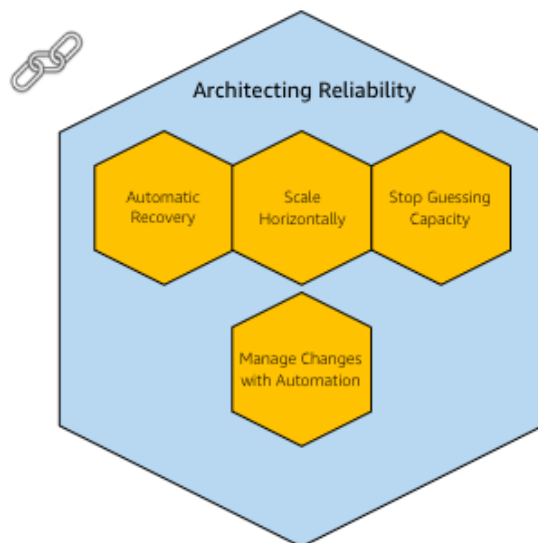
- Foundations
- Change management
- Failure management

**Chaos Monkey** is a Netflix resiliency tool that helps applications tolerate random instance failures.

**Bees with Machine Guns!** is a utility for arming (creating) many bees (micro EC2 instances) to attack (load test) targets (web applications).

La crisis con Amazon AWS.

## Design Principles



### Pillar Design Principles

- Testing recovery procedures
- Trigger automatic responses to failure events
- Scale horizontally and eliminate single points of failure
- Stop guessing capacity and automate addition and removal of resources
- Automate infrastructure changes and only manage automation

- Test recovery procedures
- Automatically recover
- Scale horizontally
- Stop guessing capacity
- Manage change in automation

## Cost Optimization

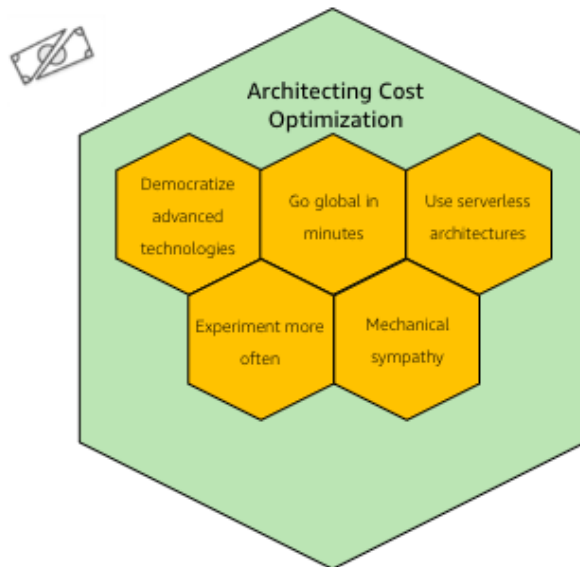
### Cost Optimization Pillar

Cost Optimization is the ability to avoid or eliminate unneeded cost or suboptimal resources. The focus areas are:

- Use cost-effective resources

- Matched supply and demand
- Increase expenditure awareness
- Optimize over time

## Design Principles



### Pillar Design Principles

- Adopt pay as you go consumption model
- Measure overall efficiency
- Analyzing and attributing expenditures
- Measure return on investment

- Adopt a consumption model
- Measure overall efficiency
- Reduce spending on data center operations
- Analyze and attribute expenditure
- Use managed services

## Operational Excellence

### Operational Excellence Pillar

Operational Excellence is the ability to run and monitor systems to deliver business value, and continually improve supporting processes and procedures.

Its key service is **AWS CloudFormation**. You can have IaC and the CloudFormation templates are correct documentation of your environment.

The focus areas and their key tools are:

- **Prepare:**
  - Understand the business goals they are trying to achieve and *set up operational priorities* to support these goals and use these priorities to focus their operational improvement efforts where they have the greatest impact.
  - *Design for operations* is focused on architecting for runtime. This includes making architectural design decisions that would enable deployments of your workloads using techniques that mitigate deployment



risks and allow you to reduce defects and fix or workaround them safely in production. It includes instrumenting your workloads to enable understanding of their operational health and enable useful businesses and technical insights. Ideally, design standards and templates should be established and shared across organizations to simplify development efforts and reduce duplication effort.

- *Operational readiness* is focused on validating the readiness of both the workloads into production and the operational team to support the workload in production. Evaluation of the workload includes consideration of your standards and governance conduct using checklists to ensure consistent evaluation. Review of the operational team including team size and skillset as well as the capture of procedures. Routine operations procedures should be captured in runbooks and processed for issue resolution in playbooks. Ideally, procedures should be scripted to increase consistency and execution, reduce the introduction of human error and to enable automation triggered in response to observed events.

- **Operate:**

- *Understanding the operational health* of your workload. It is defined in the achievement of business and customer outcomes. You must define metrics to measure the behavior of the workload against expected outcomes. Metrics to measure the status of the workloads and their components and metrics to measure the execution of operations activities.
- *Deriving business and technical insights.* By establishing baselines and collecting and analysing these metrics you can evaluate operational health, gain business insights and identify areas for intervention or improvement.
- *Responding to operational events.* Both planned and unplanned operational events should be anticipated. Planned events may include sales promotions, deployments and game days. Unplanned events may include component failures. Runbooks and playbooks should be used to enable consistent response to events and to limit the introduction of human error.

- **Evolve:** It is focused on dedicated work cycles to make continuous incremental improvement to your operations.

- *Learn from experience.* Customers should evaluate opportunities from improvements captured from feedback loops, lessons learned analysis and cross-team reviews.
- \*

Their key tools are:

- **Prepare:** *AWS Config and AWS Config Rules* can be used to create standards for workloads and determine if environment are compliant with standards before putting them into production.
- **Operate:** *Amazon CloudWatch* to monitor the operational health of the workload
- **Evolve:** *Amazon Elasticsearch* to analyze log data and get actionable insights quickly and securely.

## Design Principles



### Pillar Design Principles

- Perform operations as code to reduce human error
- Automate document annotation
- Frequent small reversible changes
- Refine procedures frequently
- Anticipate and test failure scenarios

Operational excellence design in an on-premises environment:

- Most changes are made by human beings following runbooks that are often out of date.
- It is easy to be focused on the technology metrics rather than business outcomes.
- Because making changes are difficult and risky you tend to want them often and therefore tend to batch changes into large releases.
- You rarely simulate failures or events because you do not have the system or human capacity to do this.
- Often things are moving so fast that you move from one reactive situation to the next, with no time to learn from mistakes.
- Due to the rate of change it becomes to keep documentation current.

Operational excellence design in an AWS environment:

- Performs operations by code with business metrics with you can measure against.
- Annotated documentation.
- By automating change and using code, you can make frequent, small and reversible changes.
- Refine operations procedures frequently by making game days and detecting failures in procedures.
- Learn from these and other operational events to improve your responses.
- Anticipate failure. Because infrastructure is code, you can detect when documentation is out of date and even generate documentation and include annotations to enable its use and automation.

## Performance efficiency

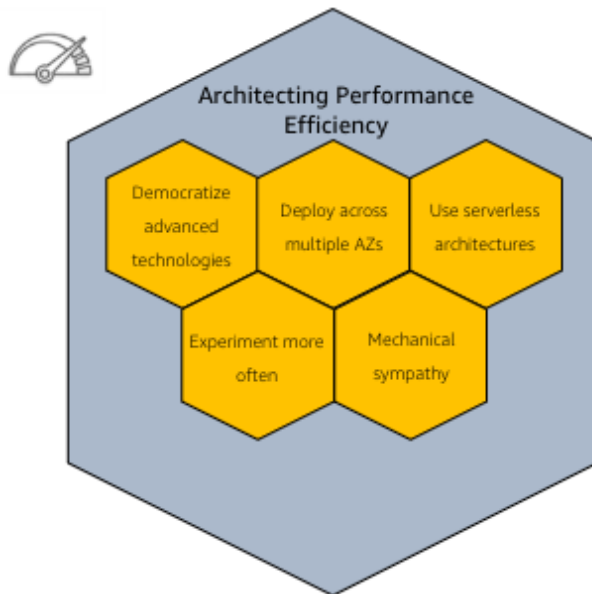
### Performance efficiency Pillar

Performance efficiency is the ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve. The focus areas are:

- Select customizable solutions

- Review to continually innovate
- Monitor AWS services
- Consider the trade-offs

## Design Principles



### Pillar Design Principles

- Democratize advanced technologies
- Deploy across multiple AZs
- Use serverless architectures
- Experiment more often
- Mechanical sympathy

- Democratize advanced technologies
- Go global in minutes
- Use a serverless architecture
- Experiment more often
- Have mechanical sympathy

## AWS Well-Architected Tool

The AWS Well-Architected Tool helps you review your workloads against current AWS best practices and provides guidance on how to improve your cloud architectures. This tool is based on the **AWS Well-Architected Framework**. The AWS Well-Architected Framework has been developed to help cloud architects build secure, high-performing, resilient, and efficient infrastructure for their applications. Based on five pillars — operational excellence, security, reliability, performance efficiency, and cost optimization. — the Framework provides a consistent approach for customers and partners to evaluate architectures, and implement designs that will scale over time.

This is a free tool, available in the [AWS Management Console](#). The process to evaluate a workload consists of 3 steps:

1. **Define a workload** by filling information about the architecture: name, a brief description to document its scope and intended purpose, the environment (production or pre-production), AWS Regions and Non-AWS Regions in which the workload runs, Account IDs (optional), industry type (optional) and industry (optional).
2. **Document the Workload State** by answering a series of questions about your architecture that span the pillars of the AWS Well-Architected Framework: operational excellence, security, reliability, performance efficiency, and cost optimization. After documenting the state of your workload for the first time, you should save a

milestone and generate a workload report. A *milestone* captures the current state of the workload and enables you to measure progress as you make changes based on your improvement plan.

3. **Review the Improvement Plan.** Based on the best practices you selected, AWS WA Tool identifies areas of high and medium risk as measured against the AWS Well-Architected Framework. The Improvement items section shows the recommended improvement items identified in the workload. The questions are ordered based on the pillar priority that is set, with any high risk items listed first followed by any medium risk items.
4. **Make Improvements and Measure Progress.** After deciding what improvement actions to take, update the Improvement status to indicate that improvements are in progress. After making changes, you can return to the Improvement plan and see the effect those changes had on the workload.

## Common uses

- Learn how to build cloud-native architectures.
- Build a backlog.
- Use a gating mechanism before launch.
- Compare maturity of different teams.
- Due-diligence for acquisitions.

## 1.2.4 AWS Global Infrastructure

### AWS Data Centers

[Learn how we secure AWS data centers by design](#)

### AWS Availability Zones

An Availability Zone (AZ) consists of several datacenters, all of them linked via intra-AZ connections and each with with redundant power supplies, networking and connectivity, housed in separated facilities. All AZ are connected among them through inter-AZ connections and to the exterior via Transit Center connections. AZs are represented by a region code followed by a letter identifier.

### AWS Regions

[Global Infrastructure](#)

[AWS Global infrastructure interactive map](#)

[AWS re:Invent 2016: Tuesday Night Live with James Hamilton](#)

[Choosing regions for your architectures:](#)

[AWS & Sustainability](#)

[Save yourself a lot of pain \(and money\) by choosing your AWS Region wisely](#)

### AWS Edge Locations

[Amazon CloudFront Key Features](#)



Fig. 6: Intra-Region connectivity

## 1.3 Storage services

### AWS Storage Services Overview

- In **Block storage**, data is stored in blocks on disks and multiple blocks comprise a file. It has no native metadata and is designed for achieving high performance. It requires that the operating system has direct byte-level access to a storage device. A block device is a storage device that moves data in sequences of bytes or bits. Amazon EBS provides durable, block-level storage volumens that you can attach to a running Amazon EC2 instance.
- In **File storage**, data is stored on a file system. It requires the network file system (NFS) protocol to abstract the operating system from storage devices. Data stored in a file system contains attributes such as permissions, file owner and they are stored as metadata in the file system. You can use an Elastic File System (EFS) file system as a common data source for workloads and applications running on multiple instances.
- **Object storage**. It is a flat system which stores data as objects which encapsulates the attributes, metadata, and other properties. It stores the data, data attributes, metadata, and object IDs as objects. It provides API access to data. It is metadata driven, policy-based, etc. It allows you to store limitless amount of data and scales easily. Content type, permissions and the file owner are stored as metadata in a file system. There are more metadata that can be stored as object. You can create custom metadata named values that can be used for such things as analytics. Amazon S3 is designed to make web-scale computing easier by enabling you to store and retrieve any amount of data, at any time, from within Amazon EC2 or anywhere on the web.

Each storage option has a unique combination of performance, durability, cost and interface. Your applications and workloads will help you determine which type of storage you will need for your implementation.

### 1.3.1 Amazon S3

#### Amazon S3 Overview

##### Introduction to Amazon Simple Storage Service (S3)

It is intentionally built with a minimum feature set focusing on simplicity and robustness: It is a highly durable, highly available, high performant object storage service. It provides you an easy-to-use, scalable object storage service that is payable as you go and integrates with other AWS services and 3rd party solutions. Main characteristics:

- You pay for what you use. You don't need to preprovision the size of your Amazon S3 for your data.

- It is designed for 11 nines of durability.
- It is performant system that is highly scalable.
- It has different storage classes and transitions targets:
  - S3 Standard.
  - S3 Standard - Infrequent Access.
  - S3 One Zone - Infrequent Access.
  - and the possibility to automatically transition data to Amazon Glacier.

Amazon S3 stores the data as objects within buckets. An object consists of data and optionally any metadata that describes that file.

### Object Storage Classes

#### Amazon S3 Storage Classes

Consider the type of data, the resiliency requirements and usage pattern in order to decide which object storage class is best suited to your needs. The typical lifecycle of data is the newer it is, the more frequently it is consumed. Amazon S3 offers a range of storage classes designed for different use cases. These include:

The first and default option is **Amazon S3 Standard** designed for the active data or hot workloads, it provides milliseconds access and has the highest cost of the 3 classes. If you don't know what your access patterns are or you need frequent retrieval, start with S3 Standard. Some common use cases are Big Data analysis, Content distribution and Web site hosting.

The **S3 Intelligent-Tiering storage class** is designed to optimize costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead.

S3 Intelligent-Tiering works by storing objects in 2 access tiers: one tier that is optimized for frequent access and another lower-cost tier that is optimized for infrequent access. For a small monthly monitoring and automation fee per object, Amazon S3 monitors access patterns of the objects in S3 Intelligent-Tiering and move the ones that have not been accessed for 30 consecutive days to the infrequent access tier.

The S3 Intelligent-Tiering charges a per object monitoring fee to monitor and place an object in the optimal access tier. As a result, on a relative basis, the object fee is less when the objects are larger and thus the amount you can save is potentially higher. In the following image, you can see an example of the potential cost savings for objects stored in S3 Standard versus S3 Intelligent-Tiering. For these calculations, we assumed 10 PB of data, in US-East-1 AWS Region, and a minimum object size of 128 KB.

**Average object size in S3 Intelligent-Tiering**

		128 KB	512 KB	1 MB	10 MB	100 MB	1 GB
Percent of data in S3 Intelligent-Tiering IA Tier	10%	-93%	-20%	-8%	3%	4%	4%
	20%	-89%	-16%	-4%	7%	8%	8%
	30%	-85%	-12%	0%	11%	12%	12%
	40%	-81%	-8%	4%	15%	16%	16%
	50%	-77%	-4%	8%	19%	20%	20%
	60%	-73%	0%	12%	23%	24%	24%
	70%	-69%	4%	16%	27%	28%	28%
	80%	-65%	8%	20%	31%	32%	32%
	90%	-61%	12%	24%	35%	36%	37%
	100%	-57%	16%	28%	39%	41%	41%

As your data ages and it's accessed less, you can use **Amazon S3 Standard - IA**. It is designed for colder or less frequently accessed data that requires milliseconds access. For example, you can leverage the Standard IA class to store detailed application logs that you analyze infrequently. It provides the same performance, throughput and low latency as the Standard storage class. It is lower in storage costs but along with storage costs, there is a retrieval cost associated with object access higher than with Amazon S3 Standard. Some common use cases are backup storage, DR, data that doesn't change frequently.

**Amazon S3 One Zone - IA** is ideal for customers who want a lower-cost option for infrequently accessed data, require similar milliseconds access but don't require the same durability and resiliency as the previous classes. It costs less than the previous classes as it stores data in only 1 AZ. It has the same associated retrieval cost as Amazon S3 Standard - IA. It is a good choice for example for storing secondary backup copies of on-premises data, or easily recreated data, or storage use of Amazon S3 Cross Region replication target from another AWS S3 Region.

**Amazon Glacier** and **Amazon S3 Glacier Deep Archive** are suitable for archiving data where data access is infrequent. Archive data is not available for real-time access. You must restore the objects before you can access them. It is storage service designed for long-term archival storage and asynchronous retrieval from minutes to hours. For example you can leverage Amazon Glacier for storing long-term backup archives. The cost of storage for Amazon Glacier is less than the other storage classes and has also an additional cost for data retrieval. There are 3 retrieval options ranging in time from minutes to hours.

#### Coming Soon - S3 Glacier Deep Archive for Long-Term Data Retention

S3 Standard, S3 Standard-IA and Glacier replicates its data across at least to 3 different AZs within a single region. If you do not need this resiliency, S3 One Zone-IA is stored in 1 AZ. If this AZ is destroyed, you will lose your data.

	S3 Standard	S3 Standard - IA	S3 One Zone - IA	Amazon Glacier
Designed for durability	99.999999999%	99.999999999%	99.999999999%*	99.999999999%
Availability Zones	≥ 3	≥ 3	1	≥ 3
Designed for availability	99.99%	99.9%	99.5%	N/A
Minimum capacity charge per object	N/A	128KB	128KB	
Minimum storage duration	N/A	30 days	30 days	90 days
Retrieval fee	N/A	Yes	Yes	Yes
Lifecycle transitions	Yes	Yes	Yes	Yes

\* Because S3 One Zone-IA stores data in a single AWS Availability Zone, data stored in this storage class will be lost in the event of Availability Zone destruction.

Fig. 7: Comparing Object Storage Classes

## Using Amazon S3

### Buckets

Amazon S3 uses buckets to store your data. Before you can transfer data into Amazon S3, you must first create a bucket. The data that is transfer is stored as objects in the bucket. When creating bucket you don't pre-determine size, apy for what you use.

One factor that you must consider when creating a bucket is the region where the bucket will be created. Wherever region the bucket is created in is where your data resides. You consider the location to optimize latency, minimize cost and comply with regulations.

Data in Amazon S3 do not expands regions automatically, although you can replicate your bucket to other regions if needed. This feature is called Cross-Region replication.

When you create a bucket, the bucket is owned by the AWS account that created it and the bucket ownership is not transferable. There is no limit in the number of object that can be stored in a bucket. There is no difference in performance whether you use many or just a few. You can store all of your objects in a single bucket or you can organize them across several buckets. You cannot create a bucket within another bucket. By default, you can create 100 buckets under each of your AWS accounts. If you need additional buckets, you can increase your bucket limit by submitting a service limit increase.

When naming your bucket, there are some rules you need to follow: the name of your bucket must globally unique and to be DNS-compliant. Be aware of uppercase letters in your bucket name, all names should be lowercase. The rules for DNS-compliant names are:

- Bucket names must be 3 and 63 characters long.
- Bucket names can contain lowercase letters, numbers, and hyphens. Each label must start and end with a lowercase letter or a number.
- Bucket names must not be formatted as an IP address.
- It is recommended that you do not use periods in bucket names.



## Objects

The file and metadata that you upload or create are essentially containerized in an object. Knowing the parts that make up an object is useful when you need to find accessed object in your bucket or when you create policies to secure your data.

If we have an object called `mybucket/mydocs/mydocument.doc`. The *key* is the name we assigned to an object, in this example: `mydocs/mydocument.doc`. You will use the object key to retrieve the object. Although you can use any UTF-8 characters in an object name, using the key naming best practices helps ensure maximum compatibility with other applications. The following object key name guidelines will help you comply with DNS, website characters, XML parsers and other APIs:

- Alphanumeric characters: 0-9, a-z, A-Z.
- Special characters: `!, -, _., *, ' , (, )`.

If you utilize any other characters in key names, they may require special handling.

The parts that make up an object are:

- *Version ID* uniquely identify an object. It is the string that AWS generates when you add an object to a bucket. This is utilized when versioning is enabled on your bucket.
- *Value* is the content that you are storing. It can be any sequence of bytes. Objects size can be 0-5 TB.
- *Metadata* is a set of name/value pairs where you can store information regarding the object. Your applications and data analysis may take advantage of your metadata to identify and classify your data. There are 2 kinds of metadata:
  - System-defined metadata. For every object stored in a bucket, Amazon S3 maintains a set of system metadata of the objects for managing them. For example: creation time and date, size, content type, storage class. Some system metadata can be modified, for more details go to [Object Key and Metadata](#)
  - User-defined metadata. You provide this optional information as a name-value pair when you send the request to create an object or update the value when you need. User-defined metadata requires a special prefix: `x-amz-meta-` when uploading via the REST API, otherwise S3 will not set the key-value pair as user-defined. You can only set the value of the metadata at the time when you upload it. After you uploaded the object, you cannot modify existing metadata. The only way to modify existing object metadata is to make a copy of the object and set the new metadata value. There is one exception to this: the use of object tags. Object tags are another form of metadata that help with the organization of the data that can be changed at any time.
- *Access control information*. You can control access to the objects stored in Amazon S3. It supports resource access control such as ACLs, bucket policies, and User-based access control.

Another important aspect about objects is that objects are not partially updated. When you make a change to an object or upload a new copy into the bucket which does not have versioning enabled, a new object is created and overwrites the existing object. If you have versioning enabled in your bucket and upload a new copy, a new version of the object is created.

Amazon S3 is a distributed system. If it receives multiple write requests for the same object simultaneously, it overwrites all but the last object written.

Amazon S3 didn't provide object locking but now it does: [Locking Objects Using Amazon S3 Object Lock](#).

## Accessing your data

There are multiple ways in which you can make your requests to retrieve or add data to your Amazon S3 bucket:

- You can view, upload and download objects through the AWS Console. For large amounts of data this is not the best way to transfer or access the data. The maximum size of a file that you can upload using the console is 78 GB.
- Via AWS CLI.
- Via AWS SDK.

Amazon S3 supports 2 types of URLs to access objects:

- *Path-style URL*. Structure:

```
http://<region-specific endpoint>/<bucket name>/<object name>
```

Example:

```
http://s3-eu-west-1.amazonaws.com/mybucket/sensordata.html
```

- *Virtual-hosted-style*. Structure:

```
http://<bucketname>.s3.amazonaws.com/<object key>
```

Example:

```
http://mybucket.s3.amazonaws.com/sensordata.html
```

It is recommended to use Virtual-hosted-style URLs. It is useful if you are using your S3 bucket to host a static website. You can also publish to the root directory of your bucket virtual server. This ability can be important since many existing applications search for files in this standard location.

You should be aware that when accessing your with HTTP-based URL, if you name your bucket to match your registered domain name such as `www.example.com` and set that DNS name as a CNAME alias for `www.example.com.s3.amazonaws.com` you can access objects with a customized URL such as:

```
http://www.example.com/sensordata.html
```

When accessing your bucket with a HTTP-based URL, if your bucket has a period in your bucket name it can cause certificate exceptions when accessed. To support HTTP access to your bucket, you should avoid using a period in the bucket name.

## How a request is routed

In AWS, S3 does not provide any query facility. To retrieve a specific object the user needs to know the exact bucket / object key. In this case, it is recommended to have an own DB system which manages the S3 metadata and key mapping.

S3 uses DNS to route requests to facilities that can process them. This system works very effectively. However, temporary routing errors can occur. If a request arrives at the wrong Amazon S3 region, S3 responds with a temporary redirect that tells the requester to resend the request to the correct region. If a request is incorrectly formed, S3 uses permanent redirects to provide direction on how to perform the request correctly and S3 will respond with a 400 error.

In the following diagram are the steps of how the DNS request process occurs:

1. The client makes a DNS request to get an object stored on S3.
2. The client receives one or more IP addresses for facilities that can process the request.
3. The client makes a request to S3 regional endpoint.
4. S3 return a copy of the object.

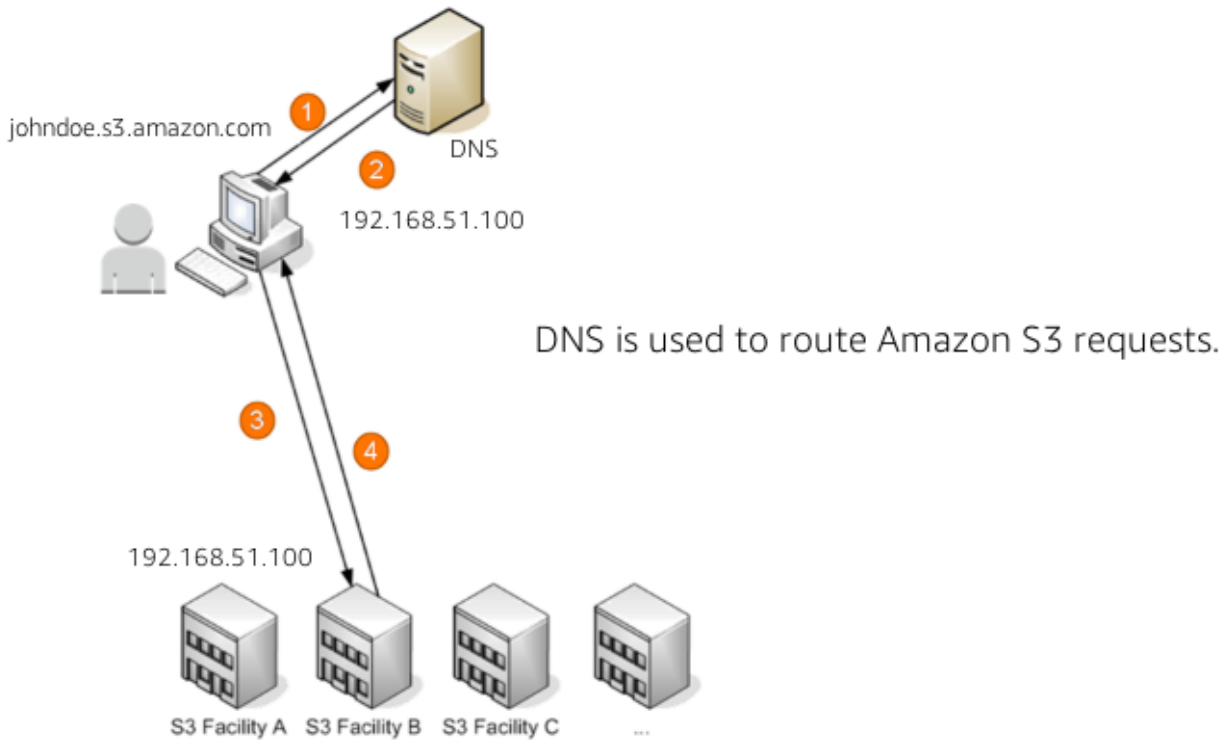


Fig. 8: How a request is routed

## Operations on Objects

### PUT

In order to get an object into a bucket, you will use the PUT operation. You can upload or copy objects of up to 5 GB in a single PUT operation. For larger objects up to 5 TB, you must use the multipart upload API.

If you triggered an S3 API call and got HTTP 200 result code and MD5 checksum, then it is considered as a successful upload. The S3 API will return an error code in case the upload is unsuccessful.

Multipart upload allows you to upload a single object as a set of parts. You can upload each part separately. If one of the parts fails to upload, you can retransmit that particular part without retransmitting the remaining parts. After all the parts of your object are uploaded to the server, you must send a complete multipart upload request that indicates that multipart upload has been completed. S3 then assembles these parts and creates the complete object. You should consider using multipart upload for objects larger than 100 MB. With multipart uploads you can upload parts in parallel to improve throughput, recover quickly from network issues, pause and resume object uploads, and begin an upload before you know the final size of an object.

You can also abort a multipart upload. When you abort an upload, S3 deletes all the parts that were already uploaded and frees up storage. S3 retains all parts on the server until you complete or abort the upload. Make sure to complete or abort an upload to avoid unnecessary storage costs related to incomplete uploads. You can also take advantage of lifecycle rules to clean up incomplete multipart uploads automatically. As a best practice, it is recommended to enable the Clean up incomplete multipart uploads in the lifecycle settings even if you are not sure that you are actually making use of multipart uploads. Some applications will default to the use of multipart uploads when uploading files above a particular, application-dependent, size.

## **COPY**

Once your objects are in the bucket you can use the COPY operation to create copies of an object, rename an object, move it to a different S3 location, or to update its metadata.

## **GET**

Using a GET request you can retrieve a complete object from your bucket. You can also retrieve an object in parts using ranged GETs, by specifying the range of bytes needed. This is useful in scenarios where network connectivity is poor or your application can or must process only subsets of object data.

## **DELETE**

You can delete a single object or delete multiple objects in single delete request. There are 2 things that can occur when you issue a DELETE request, depending if versioning is enabled or disabled on your bucket.

In a bucket that is not versioning-enabled, you can permanently delete an object by specifying the key that you want to delete. Issuing the delete request permanently removes the object and it is not recoverable, there is no recycle bin type feature in buckets when versioning is disabled.

In a bucket that is versioning-enabled, you can permanently delete an object or a delete marker is created by S3 and the object, depending on how the delete request is made:

- If you specify a key only with the delete request, S3 adds a delete marker which becomes the current version of the object. If you try to retrieve an object that has a delete marker, S3 returns a 404 Not Found error. You can recover the object by removing the delete marker from the current version of the object and it will then become available again for retrieval.
- You can also permanently delete individual versions of an object, by invoking a delete request with a key and the version ID. To completely remove the object from your bucket, you must delete each individual version.

## **List Keys**

With object storage such as S3, there is no hierarchy of objects stored in buckets, it is a flat storage system. In order to organize your data you can use prefixes in key names to group similar items. You can use delimiters (any string such as / or \_) in key names to organize your keys and create a logical hierarchy. If you use prefixes and delimiters to organize keys in a bucket, you can retrieve subsets of keys that match certain criteria. You can list keys by prefix. You can also retrieve a set of common key prefixes by specifying a delimiter. This implementation of the GET operation returns some or all (up to 1000) of the objects in a bucket.

In the following example, the bucket named scores contains objects with English and Maths scores of students for the year 2017.

```
aws s3api list-objects --bucket scores --query "Contents[].{Key: Key}"

2017/score/english/john.txt
2017/score/english/sam.txt
2017/score/maths/john.txt
2017/score/maths/sam.txt
2017/score/summary.txt
overallsummary.txt
```

To list keys related to the year 2017 in our scores bucket, specify the prefix of 2017/.

```
aws s3api list-objects --bucket scores --prefix 2017/ --query "Contents[].{Key: Key}"

2017/score/english/john.txt
2017/score/english/sam.txt
2017/score/math/s/john.txt
2017/score/math/s/sam.txt
2017/score/summary.txt
```

To retrieve the key for the 2017 scores summary in the scores bucket, specify a prefix of `2017/score/` and delimiter of `/`. The key `2017/score/summary.txt` is returned because it contains the prefix `2017/score/` and does not contain the delimiter `/` after the prefix.

```
aws s3api list-objects --bucket scores --prefix 2017/score/ --delimiter / --query
  ↪ "Contents[].{Key: Key}"

2017/score/summary.txt
```

To find subjects for which scores are available in our bucket, list the keys by specifying the prefix of `2017/score/` and delimiter of `/` and then you will get a response with the common prefixes.

```
aws s3api list-objects --bucket scores --prefix 2017/score/ --delimiter /

COMMONPREFIXES 2017/score/english/
COMMONPREFIXES 2017/score/math/s/
2017/score/summary.txt
```

## Restricting object access with pre-signed URL

In Amazon S3, all objects are private by default. Only the object owner has permission to access these objects. However, the object owner can optionally share objects with others by creating a pre-signed URL, using their own security credentials, to grant time-limited permission to download the objects.

Pre-signed URLs are useful if you want your user to be able to upload a specific object to your bucket without being required to have AWS security credentials or permissions. When you create a pre-signed URL, you must provide your security credentials, bucket name, an object key, an HTTP method (PUT for uploading objects, GET for retrieving objects), and an expiration date and time. The pre-signed URLs are valid only for the specified duration.

Share the pre-signed URL with users who need to access your S3 bucket to put or retrieve objects. Anyone who receives the pre-signed URL can then access the object. For example, if you have a video in your bucket and both the bucket and the object are private, you can share the video with others by generating a pre-signed URL.

## Cross-Origin Resource Sharing

Cross-Origin Resource Sharing (CORS) defines a way for client web application that are loaded in one domain to interact with resources in a different domain. Consider the following examples:

- You want to host a web font in your S3 bucket. A web page in a different domain may try to use this web font. Before the browser loads this web page, it will perform a CORS check to make sure that the domain from which the page is being loaded is allowed to access resources from your S3 bucket.
- Javascript in one domain's web pages (<http://www.example.com>) wants to use resources from your S3 bucket by using the endpoint `website.s3.amazonaws.com`. The browser will allow such cross-domain access only if CORS is enabled on your bucket.

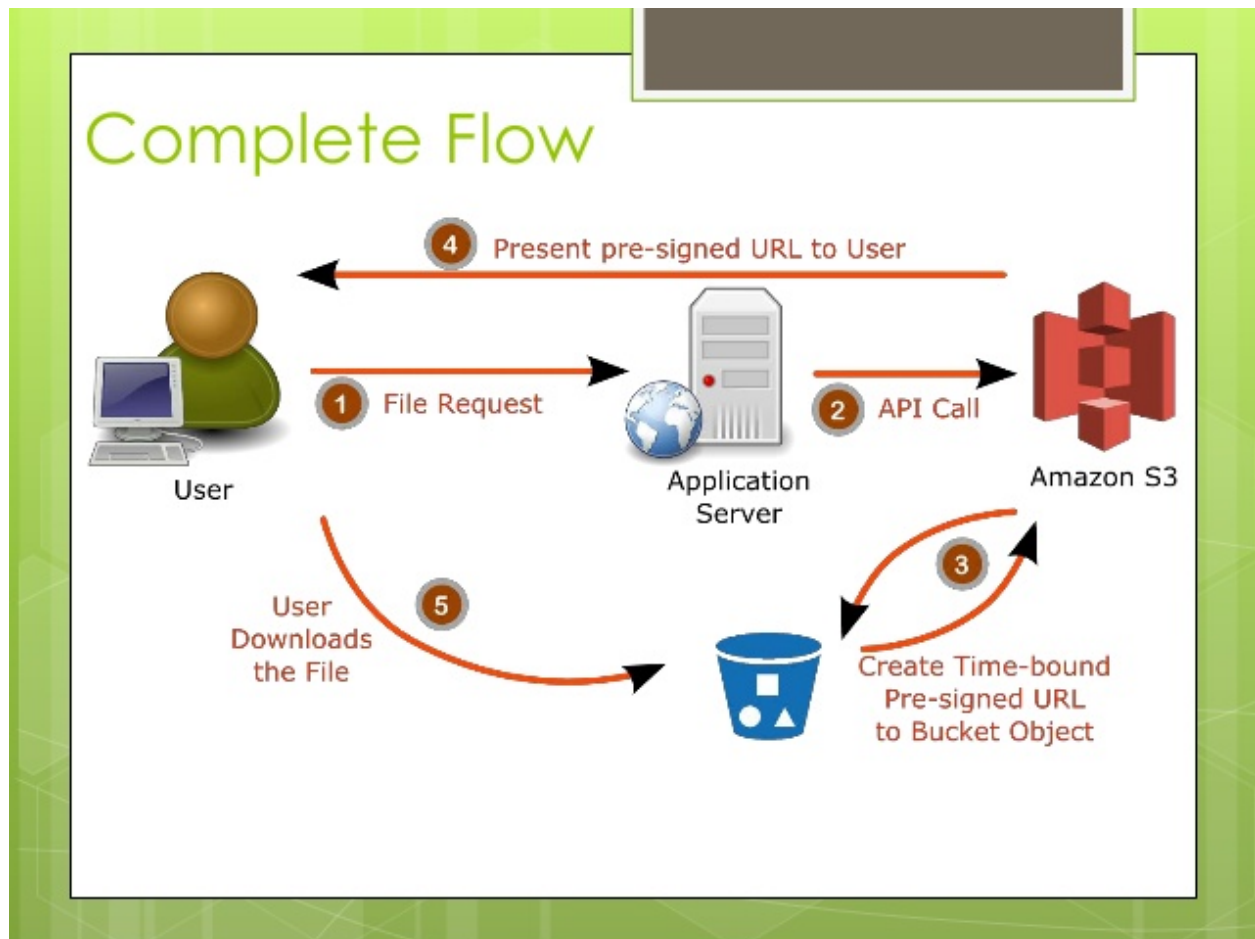


Fig. 9: Pre-signed URL

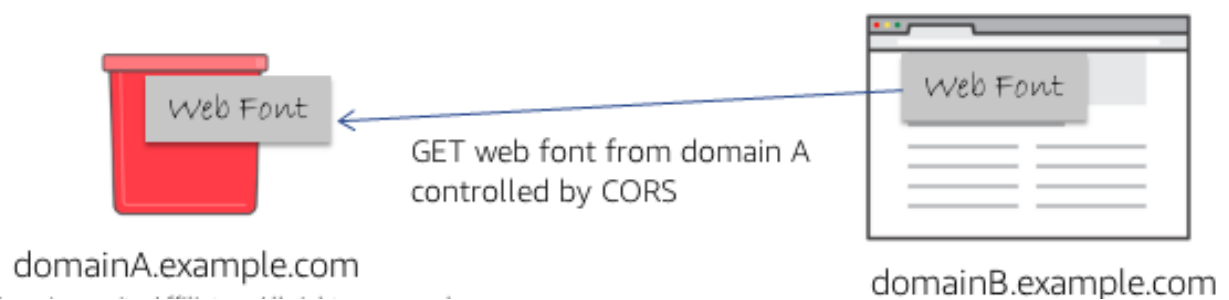


Fig. 10: CORS use case example

With CORS support in S3, you can build web applications with S3 and selectively allow cross-origin access to your S3 resources.

To enable CORS, create a CORS configuration XML file with rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that you will support for each origin, and other operation-specific information. You can add up to 100 rules to the configuration. You can apply the CORS configuration to the S3 bucket by using the AWS SDK.

## Managing access

### Access control in Amazon S3

#### Access policies

By default, all S3 resources (buckets, objects, and related sub-resources) are private, only the resource owner, and AWS account that created it, can access the resource. The resource owner can optionally grant access permissions to others by writing an access policy. By default, any permission that is not granted Allow access is an implicit Deny. There are 2 types of access policies: resource-based and IAM policies.

- IAM policies are assigned to IAM users, groups, or roles. They provide fine grained control over access and can be administered as part of a role based access configuration. These type of policies are applied at the IAM role, user, and group level to control access to S3 and its resources. It answers the question *What can this user do in AWS?*, not only in S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::<bucket_name>/<key_name>",
    }
  ]
}
```

- Access policies which are attached to your resources (buckets and objects) are referred to as resource-based policies. For example: bucket policies and ACLs are resource-based policies. Bucket policies are very similar to IAM policies, but the major difference is you need to define a Principal in the policy and it is embedded in a bucket in S3 versus created in AWS IAM and assigned to a user, group or role. Amazon S3 Bucket policies answer the question *Who can access this S3 bucket?* You can also grant cross account access using bucket policies without having to create IAM roles. You may find that your IAM policies bump up against the size limit (up to 2 kb for users, 5 kb for groups, and 10 kb for roles), and you can then use bucket policies instead. Amazon supports bucket policies of up to 20 kb. Another reason you may want to use bucket policies is that you may just want to keep access policies within Amazon S3 rather than using IAM policies created in the IAM console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
```

(continues on next page)

(continued from previous page)

```

        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::MYEXAMPLEBUCKET",
    "Principal": {
        "AWS": [
            "arn:aws:iam::123456789012:user/testuser"
        ]
    }
}
]
}

```

You may choose to use resource-based policies, user policies, or some combination of these to manage permissions to your S3 resources. Both bucket policies and user policies are written in JSON format and not easily distinguishable by looking at the policy itself, but by looking at what the policy is attached to, it should help you figure out which type of policy it is. The [AWS Policy Generator](#) is a tool that enables you to create policies that control access to AWS products and resources.

Additionally, when trying to understand if the application of your policies will work as expected, AWS has a [Policy Simulator](#) you can use to determine if your policies will work as expected.

How do I configure an S3 bucket policy to Deny all actions unless they meet certain conditions?

## Access Control Lists

As a general rule, it is recommended to use S3 bucket policies or IAM policies for access control. Amazon S3 ACLs is a legacy access control mechanism that predates IAM. A S3 ACL is a sub-resource that's attached to every S3 bucket and object. It defines which AWS accounts or groups are granted access and the type of access. When you create a bucket or an object, Amazon S3 creates a default ACLs that grants the resource owner full control over the resource. ACLs are much more limited in the fact that you can only use ACLs to grant access to other AWS accounts and not IAM users in the same account where the bucket resides.

Be very careful to ensure you do not enable public access unless it is required. If you do have a publicly accessible bucket, the S3 console displays a prominent indicator with a warning showing that Everyone means everyone on the Internet.

S3 has a set of predefined groups that can be used to grant access using ACLs. It is recommended that you do not use the Authenticated Users and All Users in ACLs when granting access permissions to your bucket unless you are sure you want to open your bucket to being publicly accessible.

- **Authenticated Users** group represents all AWS accounts in the world, not just yours. Utilizing this group to grant access could allow any AWS authenticated user in the world access to your data.
- **All users** group is similar to the Authenticated Users group in that it is not limited to just your AWS account. The request can be signed (authenticated) or unsigned (anonymous). Unsigned requests omit the Authentication header in the request. It is highly recommended that you never grant the All Users group `WRITE`, `WRITE_ACP`, or `FULL_CONTROL` permissions. For example, `WRITE` permissions allow anyone to store objects in your bucket, for which you are billed. It also allows others to delete objects that you might want to keep.
- **Log delivery** group. When granted `WRITE` permission to your bucket, it enables the S3 log delivery group to write server access logs.

[Amazon S3 Block Public Access – Another Layer of Protection for Your Accounts and Buckets](#)

[Using Amazon S3 Block Public Access](#)



Access for your AWS account					
Account ⓘ	List objects ⓘ	Write objects ⓘ	Read bucket permissions ⓘ	Write bucket permissions ⓘ	
<input type="radio"/> micgalv	Yes	Yes	Yes	Yes	
Access for other AWS accounts					
<input type="button" value="+ Add account"/> <input type="button" value="Delete"/>					
Account ⓘ	List objects ⓘ	Write objects ⓘ	Read bucket permissions ⓘ	Write bucket permissions ⓘ	
Public access					
Group ⓘ	List objects ⓘ	Write objects ⓘ	Read bucket permissions ⓘ	Write bucket permissions ⓘ	
<input type="radio"/> Everyone	-	-	-	-	
S3 log delivery group					
Group ⓘ	List objects ⓘ	Write objects ⓘ	Read bucket permissions ⓘ	Write bucket permissions ⓘ	
<input type="radio"/> Log Delivery	-	Yes	Yes	-	

Fig. 11: ACL expanded view

### How Do I Block Public Access to S3 Buckets?

## Data Transfer

You may need a variety of tools to move or transfer data in and out the cloud, depending on your data size and time to transfer. These are the options:

- **AWS Direct Connect** is a dedicated network connection from your on-premises data center to AWS for higher throughput and secure data transfer without traversing Internet.
- **AWS Storage Gateway**, either with or without AWS Direct Connect. This is a virtual appliance that lets you connect to your bucket as an NFS mount point.
- **Third-party connectors (ISV connectors)**. Amazon partners can help you move your data to the cloud. The simplest way to do that may be via a connector embedded in your backup software. With this approach, your backup catalog stays consistent, so you maintain visibility and control across jobs that span disk, tape and cloud.
- You can stream data into S3 via **Amazon Kinesis Firehose**, a fully managed streaming service. Because it captures and automatically loads streaming data into S3 and Amazon Redshift, you get near real-time analytics with the business intelligence tools you're already using.
- **Amazon Kinesis Video Streams** makes it easy to securely stream video from connected devices to AWS for analytics, machine learning, and other processing. Kinesis Video Streams automatically provisions and elastically scales all the infrastructure needed to ingest streaming video data from millions of devices. Kinesis Video Streams uses S3 as the underlying data store, which means your data is stored durably and reliably. You can set and control retention periods for data stored in your streams.
- **Amazon Kinesis Data Streams** enables you to build custom applications that process or analyze streaming data for specialized needs. Kinesis Data Streams can continuously capture and store TBs of data per hour from hundreds of thousands of sources such as website clickstreams, financial transactions, social media feeds, IT

logs, and location-tracking events. You can also emit data from Kinesis Data Streams to other AWS services such as S3, Amazon Redshift, EMR, AWS Lambda.

- **Amazon S3 Transfer Acceleration** is used for fast, easy, and secure transfers of files over long distances. It takes advantage of CloudFront's globally distributed edge locations, routing data to S3 over an optimized network path. Transfer Acceleration works well for customers who either transfer data to a central location from all over the world, or who transfer significant amounts of data across continents regularly. It can also help you better utilize your available bandwidth when uploading to S3.
- For large data migrations where transferring over a network would be too time consuming or costly, use **AWS Snowball, Snowball Edge or Snowmobile**. These are for petabyte-scale and exabyte-scale data transport that use secure appliances to transfer large amounts of data into and out of AWS.

### AWS Snowball Edge Overview

#### Using AWS Snowball Edge and AWS DMS for Database Migration

Bear in mind that you can also use these methods for exporting your data. [Cloud Data Migration](#).

Listing 1: Create a bucket and upload data to Amazon S3 using the CLI

```
c:\mydata> aws s3 mb s3://myappbucket6353 --region us-east-1
make_bucket: myappbucket6353

c:\mydata> aws s3 ls
2013-07-11 17:08:50 mybucket
2013-07-24 14:55:44 myappbucket6353

c:\mydata> aws s3 cp c:\mydata s3://myappbucket6353 --recursive
upload: myDir/test1.txt to s3://myappbucket635/myDir/test1.txt
upload: myDir/test2.txt to s3://myappbucket635/myDir/test1.txt
upload: test3.txt to s3://myappbucket635/test3.txt

c:\mydata> aws s3 ls s3://myappbucket6353
                PRE myDir/
2013-07-25 17:06:27          88 test3.txt
```

## Amazon S3 Select

S3 Select is a new S3 capability designed to pull out only the data you need from an object using a SQL expression, dramatically improving the performance and reducing the cost of applications that need to access data in S3. Most applications have to retrieve the entire object and then filter out only the required data for further analysis. S3 Select enables applications to offload the heavy lifting of filtering and accessing data inside objects to the S3 service. By reducing the volume of data that has to be loaded and processed by your applications, S3 Select can improve the performance of most applications that frequently access data from S3 by up to 400%.

Amazon S3 Select works like a GET request as it is an API call. But where Amazon S3 Select is different is we are asking for data within an object that matches a set of criteria, rather than just asking to get an entire object. You can use Amazon S3 Select through the available Presto connector, with AWS Lambda, or from any other application using the S3 Select SDK for Java or Python. In the query, you use a standard SQL expression.

Amazon S3 Select works on objects stored in delimited text (CSV, TSV) or JSON format. It also works with objects that are compressed with GZIP, and server-side encrypted objects. You can specify the format of the results as either delimited text (CSV, TSV) or JSON, and you can determine how the records in the result will be delimited. To retrieve the information you need, you pass SQL expressions to S3 in the request. Amazon S3 Select supports a subset of SQL as listed in table below.

### Selecting Content from Objects

Clauses	Data types	Operators	Functions
Select	String	Conditional	String
From	Integer, Float, Decimal	Math	Cast
Where	Timestamp	Logical	Math
	Boolean	String (Like,   )	Aggregate

Fig. 12: SQL queries with Amazon S3 Select

There are a few ways you can use Amazon S3 Select. You can perform SQL queries using AWS SDKs, the SELECT Object Content REST API, the AWS CLI, or the Amazon S3 console. When using the Amazon S3 console, it limits the amount of data returned to 40 MB.

### Securing your data in Amazon S3

AWS re:Invent 2018: [Repeat] Deep Dive on Amazon S3 Security and Management (STG303-R1)

In the decision process for determining access to your bucket and objects, S3 starts with a default deny to everyone. When you create a bucket, the owner is granted access, and as the owner you can then allow access to other users, groups, roles and resources. When determining the authorization of access to your resource in S3, it is always a union of user policies, resource policies and ACLs. In accordance with the principle of least-privilege decisions default to DENY, and an explicit DENY always trumps an ALLOW.

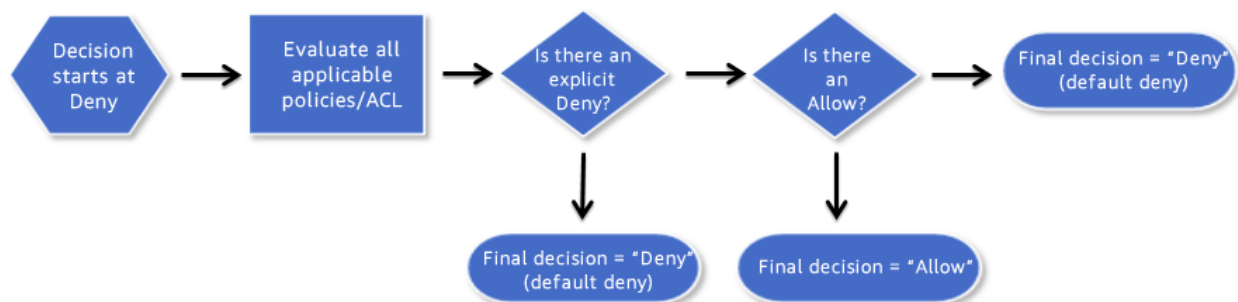


Fig. 13: Access decision process

For example, assume there is an IAM policy that grants a user access to a bucket. Additionally, there is a bucket policy defined with an explicit DENY for the user to the same bucket. When the user tries to access the bucket, the access is denied.

Keep in mind that if no policy or ACLs specifically grants ALLOW access to a resource the entity will be denied access by default. Only if no policy or ACLs specifies a DENY and one or more policies or ACLs specify an ALLOW will the request be allowed.

### Policies

A policy is an entity in AWS that, when attached to an identity or resource, defines the permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the

request is allowed or denied. Policies are stored in AWS as JSON documents attaches to principals as identity-based policies, or to resources as resource-based policies.

The language elements that are used in a policy are the following:

- **Resources.** The Resource element specifies the buckets or objects that the statement covers. Buckets and objects are the S3 resources for which you can allow or deny permissions. In a policy, you use the Amazon Resource Name (ARN) to identify the resource. For example, your resource could be just the bucket or it could be a bucket and objects, a bucket and subset of objects or even a specific object.
- **Actions.** For each resource, S3 support a set of operations. You identify resource operations you want to allow or deny by using action keywords. You specify a value using a namespace that identifies the service, for example s3, followed by the name of the action. The name must match an action that is supported by the service. The prefix and the action name are case insensitive. You can use wilcards \* to allow all operations for a service.
- **Effect.** This is what the effect will be when the user requests the specific action, this can be either allow or deny. If you do not explicitly grant allow access to a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do in order to make sure that a user cannot access it, even if a different policy grants access. For example, you may want to explicitly deny the ability to delete objects in a bucket.
- **Principal.** Use the principal element to specify the user (IAM user, federated user, or assumed-role user), AWS account, AWS service, or other principal entity that is allowed or denied access to a resource. You specify a principal only in a resource policy, for example a bucket policy. It is the user, account, role, service, or other entity who is the recipient of this permission. When using an IAM policy, the user, group or role to which the policy is attached is the implicit principal.
- **Conditions.** You can optionally add a Condition element (or Condition block) to specify conditions for when a policy is in effect. In the Condition element, you build expressions in which you can use condition operators (equal, less than, etc.) to match the condition in the policy against values in the request. Condition values can include date, time, the IP address of the requester, the ARN of the request source, the user name, user ID, and the user agent of the requester. Some services let you specify additional values in conditions; for example S3 lets you write condition using items such as object tags (s3:RequestObjectTag) to grant or deny the appropriate permission to a set of objects.

### Bucket Policy Examples

#### Example IAM Identity-Based Policies

There are some additional elements that can be used in policies: NotPrincipal, NotAction, and NotResource.

You can use the **NotPrincipal** element to specify an exception to a list of principals. For example, you can deny access to all principals except the one named in the NotPrincipal element.

Although you can use the NotPrincipal with an Allow, when you use NotPrincipal in the same policy statement as “Effect”:”Allow”, the permissions specified in the policy statement will be granted to all principals except the one(s) specified, including anonymous (unauthenticated) users. It is recommended not to use NotPrincipal in the same policy statement as “Effect”:”Allow”.

When creating a policy, combining “Deny” and “NotPrincipal” is the only time that the order in which AWS evaluated principals makes a difference. AWS internally validates the principals from the “top down”, meaning that AWS checks the account first and then the user. If an assumed-role user (someone who is using a role rather than an IAM user) is being evaluated, AWS looks at the account first, then the role, and finally the assumed-role user. The assumed-role user is identified by the role session name that is specified when the user assumes the role. Normally, this order does not have any impact on the results of the policy evaluation. However, when you use both “Deny” and “NotPrincipal”, the evaluation order requires you to explicitly include the ARNs for the entities associated with the specified principal. For example, to specify a user, you must explicitly include the ARN for the user’s account. To specify an assumed-role user, you must also include both the ARN for the role and the ARN for the account containing the role.

**NotAction** is an advanced policy element that explicitly matches everything except the specified list of actions and it can be used with both the Allow and Deny effect. Using NotAction can result in a shorter policy by listing only a

few actions that should not match, rather than including a long list of actions that will match. When using `NotAction`, you should keep in mind that actions specified in this element are the only actions that are limited. This means that all of the actions or services that are not listed, are allowed if you use the `Allow` effect, or are denied if you use the `Deny` effect.

You can use the `NotAction` element in a statement with `“Effect”:``“Allow”` to provide access to all of the actions in an AWS service, except for the actions specified in `NotAction`. You can also use it with the `Resource` element to provide access to one or more resources with the exception of the action specified in the `NotAction` element.

Be careful using the `NotAction` and `“Effect”:``“Allow”` in the same statement or in a different statement within a policy. `NotAction` matches all services and actions that are not explicitly listed, and could result in granting users more permissions than you intended.

You can also use the `NotAction` element in a statement with `“Effect”:``“Deny”` to deny access to all of the listed resources except for the actions specified in the `NotAction` element. This combination does not allow the listed items, but instead explicitly denies the actions not listed. You must still allow actions that you want to allow.

**NotResource** is an advanced policy element that explicitly matches everything except the specified list of resources. Using `NotResource` can result in a shorter policy by listing only a few resources that should not match, rather than including a long list of resources that will match. When using `NotResource`, you should keep in mind that resources specified in this element are the only resources that are limited. This, in turn, means that all of the resources, including the resources in all other services, that are not listed, are allowed if you use the `Allow` effect, or are denied if you use the `Deny` effect. Statements must include either the `Resource` or a `NotResource` element that specifies a resource using an ARN.

Be careful using the `NotResource` and `“Effect”:``“Allow”` in the same statement or in a different statement within a policy. `NotResource` allows all services and resources that are not explicitly listed, and could result in granting users more permissions than you intended. Using the `NotResource` element and `“Effect”:``“Deny”` in the same statement denies services and resources that are not explicitly listed.

Normally, to explicitly deny access to a resource you would write a policy that uses `“Effect”:``“Deny”` and that includes a `Resource` element that lists each folder individually.

## Cross account policies

One option you can use is to ensure that account that created the object adds the grant that gives the `bucket-owner-full-control` permission on the object so the bucket owner can set permissions as needed. You can do this by adding a condition in the policy. Additionally, you can deny the ability to upload objects unless that account grants `bucket-owner-full-control` permissions.

In the example below, when Jane uploads an object to the images bucket, she includes the grant `bucket-owner-full-control` permission. If she did not include this grant, the upload would fail. Now when Joe tries to GET the new object uploaded by Jane with the additional permissions, he is successful.

[Identity and Access Management in Amazon S3](#)

## Multiple policies

You can attach more than 1 policy to an entity. If you have multiple permissions to grant to an entity, you can put them in separate policies, or you can put them all in one policy. Generally, each statement in a policy includes information about a single permission. If your policy includes multiple statements, a logical OR is applied across the statements at evaluation time. Similarly, if multiple policies are applicable to a request, a logical OR is applied across the policies at evaluation time.

Users often have multiple policies that apply to them (but aren't necessarily attached to them). For example, an IAM user could have policies attached to them, and other policies attached to the groups of which they are a member. In

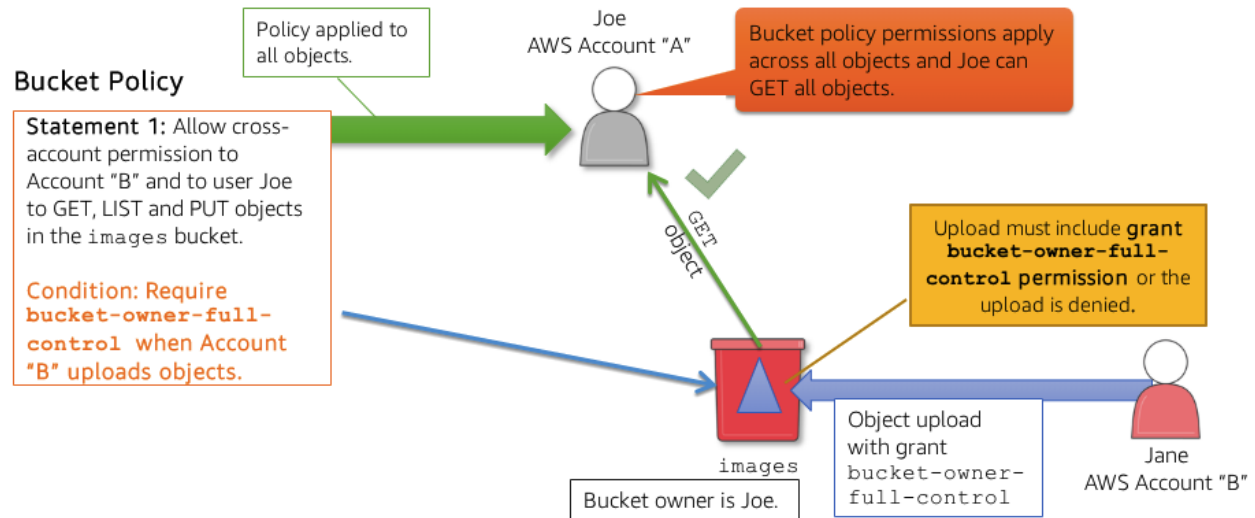


Fig. 14: Access decision process

addition, they might be accessing an S3 bucket that has its own bucket policy (resource-based policy). All applicable policies are evaluated and the result is always that access is either granted or denied.

## Best practices

Some best practices to use in securing your S3 data to follow in your setup are the following:

- Use bucket policies to restrict deletes.
- For additional security, enable MFA delete, which requires additional authentication to:
  - Change the versioning state of your bucket.
  - Permanently delete an object version.

Note that to enable MFA delete with Amazon S3 you will need root credentials. When using MFA you will require an approved AWS authentication device.

## Data at rest encryption

For data at rest protection in S3 you have 2 options: Server-Side Encryption and Client-Side Encryption.

### Server-Side Encryption

When using server-side encryption, your request S3 to encrypt your object saving it on disks in its data centers and decrypt it when you download the object. With server side encryption there are a few ways in which you can choose to implement the encryption. You have 3 server-side encryption options for your S3 objects:

- Amazon S3-Managed Keys (**SSE-S3**). This method uses keys that are managed by S3. Each object is encrypted with a unique key. Additionally a master key, which is rotated regularly, encrypts each unique key. This method uses AES-256 algorithm to encrypt your data. This option can also be used when setting the default encryption option.



- **AWS KMS-Managed keys (SSE-KMS)** is similar to SSE-S3, but with some additional benefits along with some additional charges for using service. SSE-KMS encrypts only the object data, any object metadata is not encrypted. In this model, the AWS Key Management Service (AWS KMS) is utilized to fully manage the keys and encryption and decryption. AWS KMS encrypts your objects similar to the way SSE-S3 does. There is a unique per-object data key, which is encrypted with customer master keys (CMK) in KMS. This scheme is called envelop encryption. You use AWS KMS via the Encryption Keys section in the IAM console or via AWS KMS APIs to centrally create encryption keys, define the policies that control how keys can be used, and audit key usage to prove they are being used correctly. The first time you add an SSE-KMS-encrypted object to a bucket in a region, a default CMK is created for you automatically. This key is used for SSE-KMS-encryption unless you select a CMK that you created separately using AWS KMS. Creating your own CMK gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data. Using SSE-KMS also adds a layer of security in that any user that attempts to access an object that is SSE-KMS encrypted will also require access to the KMS key to decrypt the object. You can configure access to the KMS encryption keys using AWS IAM. This option can also be used when setting the default encryption option.

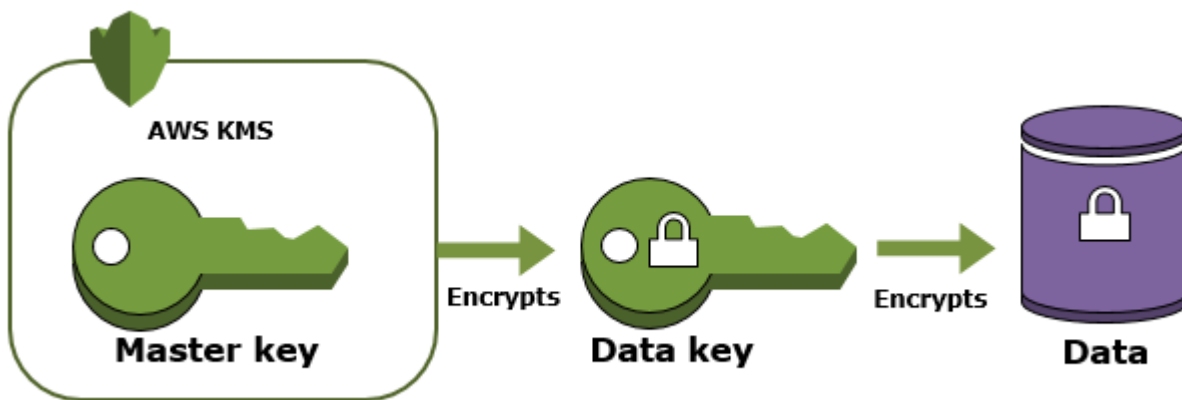


Fig. 15: SSE-KMS

You should be aware that when using AWS KMS there are some limits on requests per second. AWS KMS throttles API requests at different limits depending on the API operation. Throttling means that AWS KMS rejects an otherwise valid request because the request exceeds the limit for the number of requests per second, [AWS KMS Limits](#). When a request is throttled, AWS KMS returns a `ThrottlingException` error.

A customer master key (CMK) is a logical representation of a master key. The CMK includes metadata, such as the key ID, creation date, description, and key state. The CMK also contains the key material used to encrypt and decrypt data. You can use a CMK to encrypt and decrypt up to 4 KB (4096 bytes) of data. Typically, you use CMKs to generate, encrypt, and decrypt the data keys that you use outside of AWS KMS to encrypt your data.

- **Customer provided keys (SSE-C)**. In this model, you manage the encryption keys and S3 manages the encryption, as it writes to disks, and decryption, when you access your objects. Therefore, you don't need to maintain any code to perform data encryption and decryption. The only thing you do is manage the encryption keys you provide. When you upload an object, S3 uses the encryption key you provide to apply AES-256 encryption to your data and then removes the encryption key from memory. When you retrieve an object, you must provide the same encryption key as part of your request, S3 first verifies that the encryption key you provided matches, and then decrypts the object before returning the object data to you.

It is important to note that S3 does not store the encryption key you provide. Instead, AWS stores a randomly salted HMAC value of the encryption key in order to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means that if you lose the encryption key, you lose the object.

## Client-Side Encryption

Client side encryption happens before your data is uploaded into your S3 bucket. In this case, you manage the encryption process, the encryption keys, and related tools. There are 2 options for client-side encryption:

- **AWS KMS managed customer master key (CSE-KMS).** You don't have to worry about providing any encryption keys to S3 encryption client. Instead, you provide only an AWS KMS customer master key ID, and the client does the rest.
- **Customer managed master encryption keys (CSE-C).** You use your own client-side master key. When using your client-side master keys, your client-side master keys and your unencrypted data is never sent to AWS. It is important that you safely manage your encryption keys. If you lose them, you don't be able to decrypt your data.

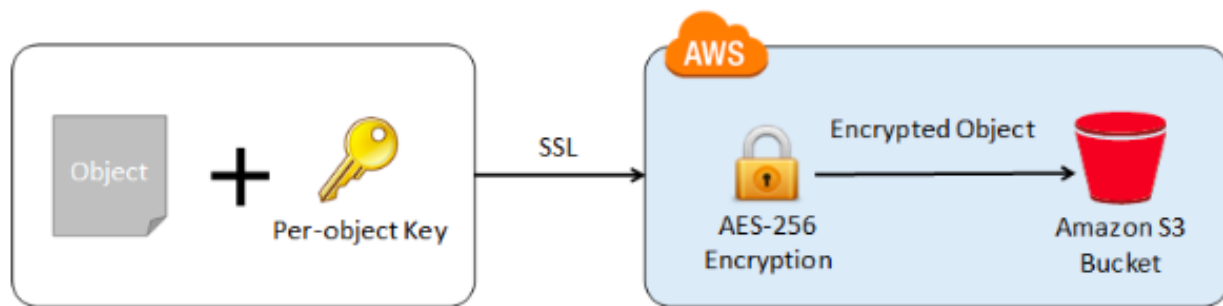


Fig. 16: Access decision process

This is how client-side encryption using client-side master key works:

- When *uploading an object*. You provide a client-side master key to the Amazon S3 encryption client. The client uses the master key only to encrypt the data encryption key that it generates randomly. The process works like this:
  1. The Amazon S3 encryption client generates a one-time-use symmetric key (also known as a data encryption key or data key) locally. It uses the data key to encrypt the data of a single Amazon S3 object. The client generates a separate data key for each object.
  2. The client encrypts the data encryption key using the master key that you provide. The client uploads the encrypted data key and its material description as part of the object metadata. The client uses the material description to determine which client-side master key to use for decryption.
  3. The client uploads the encrypted data to Amazon S3 and saves the encrypted data key as object metadata (x-amz-meta-x-amz-key) in Amazon S3.
- When *downloading an object*. The client downloads the encrypted object from Amazon S3. Using the material description from the object's metadata, the client determines which master key to use to decrypt the data key. The client uses that master key to decrypt the data key and then uses the data key to decrypt the object.

---

**Note:** Default Encryption is an option that allows you to enable automatically encrypt of all new objects written to your Amazon S3 bucket using either SSE-S3 or SSE-KMS. This property does not affect existing objects in your bucket.

---

## AWS Config

Once you have completed AWS Config setup, you can use the AWS Config built in rules for Amazon S3.



- `s3-bucket-logging-enabled`. Checks whether logging is enabled for your S3 buckets.
- `s3-bucket-public-read-prohibited`. Checks that your S3 buckets do not allow public read access. If a S3 bucket policy or bucket ACL allows public read access, the bucket is noncompliant.
- `s3-bucket-public-write-prohibited`. Checks that your S3 buckets do not allow public write access. If a S3 bucket policy or bucket ACL allows public write access, the bucket is noncompliant.
- `s3-bucket-ssl-requests-only`. Checks that your S3 buckets have policies that require requests to use SSL.
- `s3-bucket-versioning-enabled`. Checks whether versioning is enabled for your S3 buckets. Optionally, the rule checks if MFA delete is enabled in your S3 buckets.

## AWS CloudTrail

AWS CloudTrail is the API logging service in AWS that provide fine grained access tracking for your Amazon S3 buckets and objects. For each request, CloudTrail captures and logs who made the API call, when it was made, what resources were affected. By default, CloudTrail logs capture bucket level operations. You can additionally capture object level actions when S3 Data Events are enabled

CloudTrail Logging: Bucket-level actions REST API name			
DELETE Bucket	GET Bucket cors	GET Bucket requestPayment	PUT Bucket policy
DELETE Bucket cors	GET Bucket lifecycle	GET Bucket versioning	PUT Bucket logging
DELETE Bucket lifecycle	GET Bucket policy	GET Bucket website	PUT Bucket notification
DELETE Bucket policy	GET Bucket location	GET Service (List all buckets)	PUT Bucket replication
DELETE Bucket replication	GET Bucket logging	PUT Bucket	PUT Bucket requestPayment
DELETE Bucket tagging	GET Bucket notification	PUT Bucket acl	PUT Bucket tagging
DELETE Bucket website	GET Bucket replication	PUT Bucket cors	PUT Bucket versioning
GET Bucket acl	GET Bucket tagging	PUT Bucket lifecycle	PUT Bucket website

<b>CloudTrail Logging: Object-level actions REST API name</b>	
<b>DeleteObject</b>	<b>PutObjectACL</b>
<b>GetObject</b>	<b>PutObjectCopy</b>
<b>GetObjectACL</b>	<b>InitiateMultipartUpload</b>
<b>GetObjectTorrent</b>	<b>UploadPart</b>
<b>HeadObject</b>	<b>UploadPart - Copy</b>
<b>PostObject</b>	<b>CompleteMultipartUpload</b>
<b>PostObjectRestore</b>	<b>AbortMultipartUpload</b>
<b>PutObject</b>	<b>ListParts</b>

CloudTrail also integrates with CloudWatch and you can utilize CloudWatch alarms to notify you of certain events or to take actions based on your configuration. When utilizing CloudTrail, the Amazon S3 data events are delivered to CloudWatch Events within seconds so you can configure your account to take immediate action on a specified activity to improve your security posture. Additionally, CloudTrail logs are delivered to CloudWatch logs and S3 within 2-5 minutes.

CloudTrail logging can be enabled at the bucket or prefix level. You can filter your logging based on reads or writes or include both.

Additionally, AWS CloudTrail allows you to automatically add your new and existing S3 buckets to S3 Data Events. S3 Data Events allow you to record API actions on S3 objects and receive detailed information such as the AWS account, IAM user role, and IP address of the caller, time of the API call, and other details. Previously, you had to manually add individual S3 buckets in your account to track S3 object-level operations, and repeat the process for each new S3 bucket. Now, you can automatically log Amazon S3 Data Events for all of your new and existing buckets with a few clicks.

When enabling CloudTrail for S3 bucket you will need to make sure your destination bucket has the proper permissions to allow CloudTrail to deliver the log files to your bucket. CloudTrail will automatically attach the required permissions if you create a bucket as part of creating or updating a trail in the CloudTrail console or create a bucket with the AWS CLI `create-subscription` and `update-subscription` commands.

If you specified an existing S3 bucket as the storage location for log file delivery, you must attach a policy to the bucket that allows CloudTrail to write the bucket. [Amazon S3 Bucket Policy for CloudTrail](#). As a best practice, it is recommended to use a dedicated bucket for CloudTrail logs.

## Security inspection

You can verify if you are meeting your security needs with various AWS tools:

- To verify if objects in your bucket are encrypted, you can use *Amazon S3 Inventory*.
- To know if any of your buckets are publicly accessible, there are 2 ways:
  - *AWS TrustedAdvisor*, which can check your S3 bucket permissions and list the buckets that have open access. Set a *CloudWatch* alarm to alert you should any buckets fail the check.
  - Using *AWS TrustedAdvisor* technology, the S3 console now includes a bucket permissions check. A new column, called *Access*, shows any buckets that have public access. If you click on the bucket where it shows public access you can then see which policy is granting public access as well by going to the *Permissions* tab.

For Amazon S3 there are 3 checks you might want to look at: Amazon S3 bucket permissions, bucket logging, and bucket versioning.

## Amazon Macie

Amazon Macie is a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. It will search your Amazon S3 bucket for personally identifiable information (PII), personal health information (PHI), access keys, credit card information and other sensitive data and alert you if you have insecure data. It uses S3 CloudTrail Events to see all of the requests that are sent to your Amazon S3 bucket and uses ML to determine patterns and will alert if there is anything suspicious or if the patterns change.

Amazon Macie can answer the following questions:

- What data do I have in the cloud?
- Where is it located?
- How is data being shared and stored?
- How can I classify data in near-real time?
- What personally identifiable information or personal health information is possibly exposed?
- How do I build workflow remediation for my security and compliance needs?

## Amazon S3 Storage Management

Among the different options that are available to configure on buckets and objects are the following: Versioning, Server access logging, object-level logging, Static website hosting, default encryption, object tags, Transfer Acceleration, events notification, requester pays.

## Versioning

### Using Versioning

By enabling versioning, you can create a data protection mechanism for your Amazon S3 bucket. With versioning enabled on your bucket, you are able to protect your objects from accidental deletion or overwrites. Versioning is applied at the bucket level and all the objects in your bucket will have this feature applied. There is no performance penalty for versioning and it is considered a best practice. Once enabled you have also essentially created a recycle bin for your bucket.

Rather than a hard delete on an object, when versioning is enabled it creates a delete marker. You can then remove this delete marker and you have your original object back. Objects cannot be partially updated, so utilizing versioning still does not allow you to just update a portion of the object.

To efficiently control your storage capacity and keep it to only the proper amount required, you can utilize lifecycle policies to move versions of objects to the appropriate storage class as well as expire old versions if needed, providing you with an automatic cleanup process for your data.

### Server access logging

In order to track requests for access to your bucket, you can enable access logging. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. Access log information can be useful in security and access audits. It can also help you learn about your customer base and understand your Amazon S3 bill. There is no extra charge for enabling server access logging on an Amazon S3 bucket; however, any log files the system delivers to you will accrue the usual charges for storage.

By default, logging is disabled. To enable access logging, you must do the following:

1. Turn on the log delivery by adding logging configuration on the bucket for which you want S3 to deliver access logs.
2. Grant the Amazon S3 Log Delivery Group write permission on the bucket where you want the access logs saved.

If you use the Amazon S3 console to enable logging on a bucket, the console will both enable logging on the source bucket and update the ACL on the target bucket to grant write permission to the Log Delivery Group

### Object-level logging

To help ensure security of your data, you need the ability to audit and monitor access and operations, you can do that by enabling object-level logging with AWS CloudTrail integration. AWS CloudTrail logs capture bucket level and object level requests. For each request, the log includes who made the API call, when it was made, what resources were affected. You can use a CloudTrail log to understand your end users' behavior and tune access policies for tighter access control.

AWS CloudTrail Data Events allows you to log object level activity such as puts, gets, and deletes, the logs include account, IAM user, IP address, and more. This can be configured with CloudWatch Events to take action when changes are made. For example, if any object ACL is changed, you can automatically have the change reverted as needed.

### Static website hosting

Enabling this option allows you to host static websites using just your S3 bucket, no additional servers are required. On a static website, individual webpages include static content. They might also contain client-side scripts. By contrast, a dynamic website relies on server-side processing, including server-side scripts such as PHP, JSP, or ASP.NET. Amazon S3 does not support server-side scripting. To host a static website, you configure an Amazon S3 bucket for website hosting, and then upload your website content to the bucket. The website is then available at the AWS Region-specific website endpoint of the bucket: `<bucket-name>.s3-website-<AWS-region>.amazonaws.com`.

There are several ways you can manage your bucket's website configuration. You can use the AWS Management Console to manage configuration without writing any code or you can programmatically create, update, or delete the website configuration by using the AWS SDKs.

Here are the prerequisites for routing traffic to a website that is hosted in an Amazon S3 Bucket:

- An S3 bucket that is configured to host a static website. The bucket must have the same name as your domain or subdomain. For example, if you want to use the subdomain

<bucket-name>.s3-website-<AWS-region>.amazonaws.com, the name of the bucket must be <bucket-name>.

- A registered domain name. You can use Route 53 as your domain registrar, or you can use a different registrar.
- Route 53 as the DNS service for the domain. If you register your domain name by using Route 53, AWS configure Route 53 as the DNS service for the domain.

## Object tags

You can organize your data by several dimensions:

- *Location*, by bucket and prefixes.
- *Nature of the data*. You can take advantage of object tagging to apply more granular control.

Amazon S3 tags are key-value pairs that can be created with the console, CLI or via APIs. The key name value you create is case sensitive and you can have up to 10 tags assigned to an object. With object tags, you can control access, lower costs with lifecycle policies, analyze your data with storage class analytics, and monitor performance with CloudWatch metrics.

Here is an example of setting access permission using tags. If you want to give a user permission to GET objects in your bucket that have been tagged as Project X, you can use a condition as seen in the example to allow them access to any object or bucket tagged with Project X.

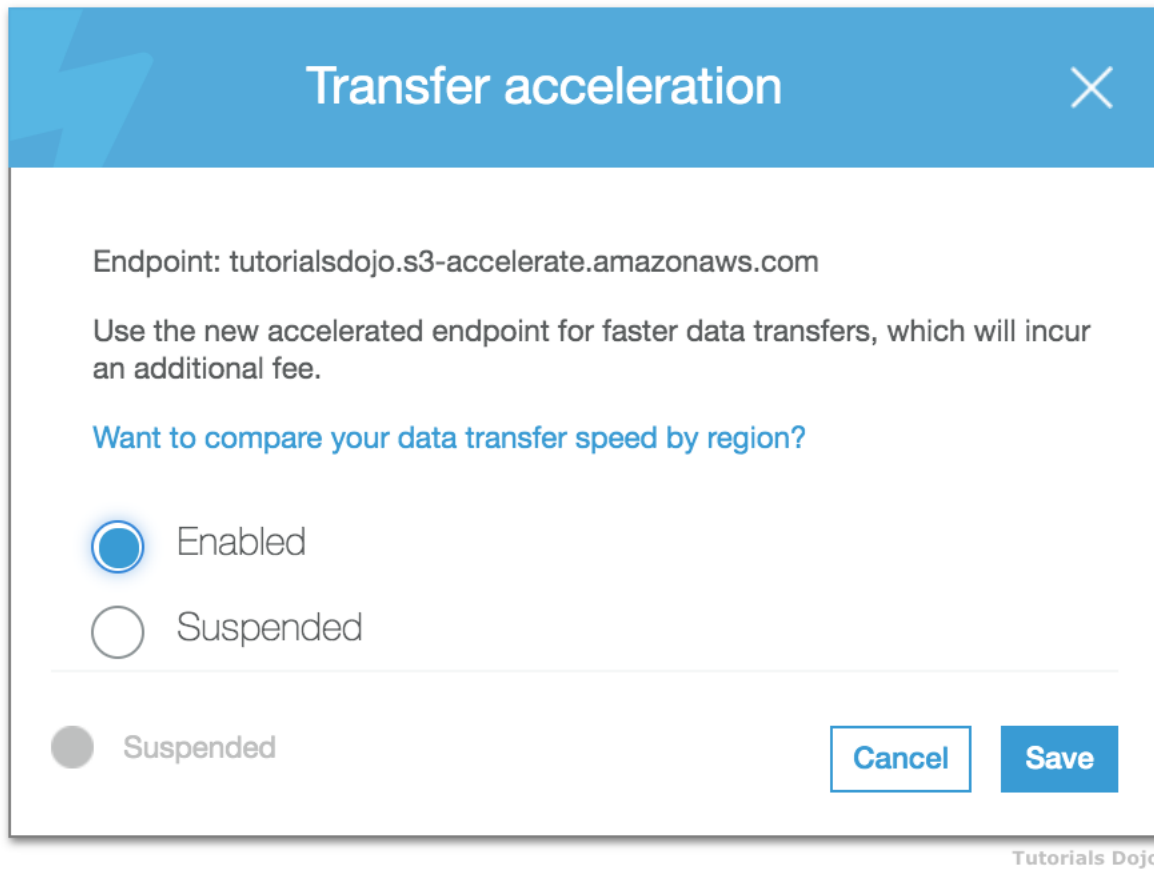
This simplifies some of your security by being able to easily allow and deny users access to specific objects and buckets using policies and tags.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::Project-bucket/*"
      "Condition": {
        "StringEquals": {
          "s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

## Object Tagging

## Transfer Acceleration

Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and your Amazon S3 bucket. Transfer Acceleration leverages Amazon CloudFront's globally distributed AWS Edge Locations. As data arrives at an AWS Edge Location, data is routed to your Amazon S3 bucket over an optimized network path.



Amazon S3 Transfer Acceleration can speed up content transfers to and from Amazon S3 by as much as 50-500% for long-distance transfer of larger objects. Customers who have either web or mobile applications with widespread users or applications hosted far away from their S3 bucket can experience long and variable upload and download speeds over the Internet. S3 Transfer Acceleration (S3TA) reduces the variability in Internet routing, congestion and speeds that can affect transfers, and logically shortens the distance to S3 for remote applications. S3TA improves transfer performance by routing traffic through Amazon CloudFront's globally distributed Edge Locations and over AWS backbone networks, and by using network protocol optimizations. Enabling Transfer Acceleration provides you with a new URL to use with your application.

## Event notifications

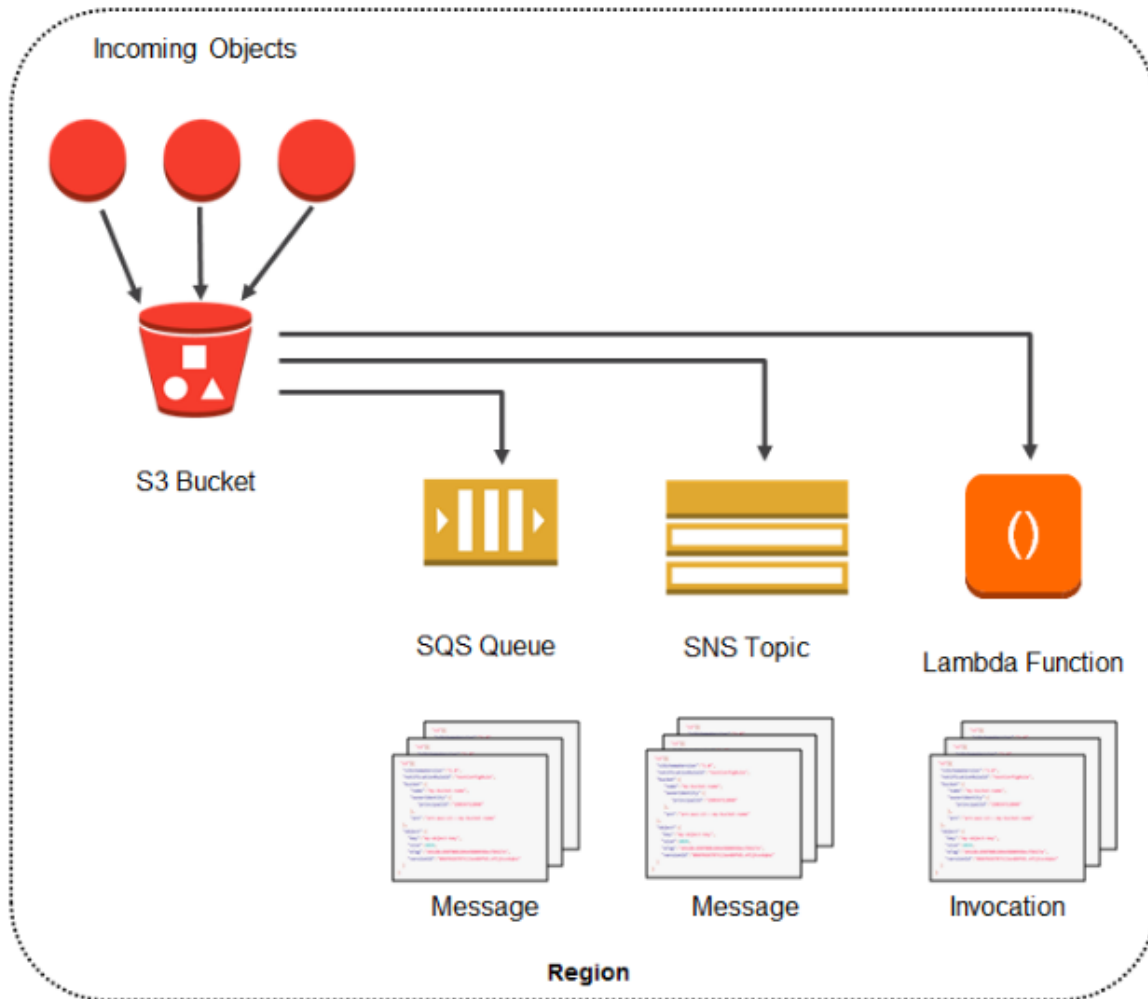
Events will enable you to receive notifications based on events that occur in your bucket. The S3 notification feature enables you to receive notifications when certain events happen in your bucket, for example: you can receive a notification when someone uploads new data to your bucket.

To enable notifications, you must first add a notification configuration identifying the events you want Amazon S3 to publish, and the destinations where you want S3 to send the event notifications. S3 events integrate with SNS, SQS and AWS Lambda to send notifications.

Here's what you need to do in order to start using this event notifications with your application:

1. Create the queue, topic, or Lambda function (which I'll call the target for brevity) if necessary.
2. Grant S3 permission to publish to the target or invoke the Lambda function. For SNS or SQS, you do this by applying an appropriate policy to the topic or the queue. For Lambda, you must create and supply an IAM role, then associate it with the Lambda function.

3. Arrange for your application to be invoked in response to activity on the target. As you will see in a moment, you have several options here.
4. Set the bucket's Notification Configuration to point to the target.



### Requester pays

A bucket owner can configure a bucket to be a Requester Pays bucket. With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket. The bucket owner always pays the cost of storing data. You might, for example, use Requester Pays buckets when making available large data sets, such as zip code directories, reference data, geospatial information, or web crawling data.

### Object Lifecycle policies

#### Object Lifecycle Management

To manage your objects so they are stored cost effectively throughout their lifecycle, you can configure lifecycle rules. A lifecycle configuration or lifecycle policy, is a set of rules that define the actions that S3 applies to a group of objects.

A lifecycle rule can apply to all or a subset of objects in a bucket based on the filter element that you specify in the lifecycle rule. A lifecycle configuration can have up to 1000 rules. These rules also have a status element where it can be either enabled or disabled. If a rule is disabled, S3 doesn't perform any of the actions defined in the rule. Each rule defines an action. The actions can be either a transition of objects to another storage class or an expiration of objects.

### Automate transitions

You can automate the tiering process from one storage class to another. There are some considerations you should be aware of:

- There is no automatic transition of objects less than 128 KB in size to S3 Standard - IA or S3 One Zone - IA.
- Data must remain on its current storage class for at least 30 days before it can be automatically moved to S3 Standard - IA or S3 One Zone - IA.
- Data can be moved from any storage class directly to Amazon Glacier.

### Action types

You can direct S3 to perform specific actions in an object's lifetime by specifying one or more of the following predefined actions in a lifecycle rule. The effect of these actions depends on the versioning state of your bucket.

1. **Transition.** You can tell S3 to transition objects to another S3 storage class. A transition can move objects to the S3 Standard - IA or S3 One Zone - IA or Amazon Glacier storage classes based on the object age you specify.
2. **Expiration.** Expiration deletes objects after the time you specify. When an object reaches the end of its lifetime, S3 queues it for removal and removes it asynchronously.

In addition, S3 provides the following actions that you can use to manage noncurrent object versions in a version-enabled bucket:

- On a versioning-enabled bucket, if the current object version is not a delete marker, S3 adds a delete marker with a unique version ID. This makes the current version noncurrent, and delete marker the current version.
- On a versioning-suspended bucket, the expiration action causes S3 to create a delete marker with null as the version ID. This delete marker replaces any object version with a null version ID in the version hierarchy, which effectively deletes the object.

You can also combine actions for a completely automated lifecycle.

### Parameters

You can set lifecycle configuration rules based on the bucket, the object or object tags.

### Versions

You can configure your lifecycle configuration rules to take an action on a particular version of an object, either the current version or any previous versions.

### Transitioning objects

From S3 Standard you can transition to any of other storage classes (Standard-IA, One Zone-IA and Glacier) using lifecycle configuration rules, but there are some constraints:



- S3 does not support transition of objects less than 128 KB.
- Objects must be stored for at least 30 days before you can transition to S3 Standard-IA or to One Zone-IA. S3 doesn't transition objects within the first 30 days because newer objects are often accessed more frequently or deleted sooner than is suitable for S3 Standard-IA or to S3 One Zone-IA storage.
- If you are transitioning noncurrent objects in version-enabled buckets, for example a particular version of an object, you can transition only objects that are least 30 days noncurrent to S3 Standard-IA or One Zone-IA storage.

From S3 Standard-IA you can transition to S3 One Zone-IA or to Amazon Glacier using lifecycle configuration rules, but there is a constraint:

- Objects must be stored at least 30 days in the S3 Standard-IA storage class before you can transition them to the S3 One Zone-IA class.

You can only transition from S3 One Zone-IA to Glacier using lifecycle configuration rules.

You cannot transition from Glacier to any storage class. When objects are transitioned to Glacier using lifecycle configurations, the objects are visible and available only through S3, not through Glacier. You can access them using the S3 console or the S3 API but not through Glacier console or Glacier API in this scenario.

Amazon S3 supports a waterfall model for transitioning between storage classes, as shown in the following diagram.

[Transitioning Objects Using Amazon S3 Lifecycle](#)

## Amazon S3 inventory

In order to help you manage your data you may need to get a list of objects and their associated metadata. S3 has a LIST API that can provide this function, but a new and less costly alternative is the Amazon S3 Inventory service. You can use it to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. Amazon S3 provides a CSV or ORC file output of your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or a shared prefix.

You can configure what object metadata to include in the inventory, whether to list all object versions or only current versions, where to store the inventory list flat-file output, and whether to generate the inventory on a daily or weekly basis. Amazon S3 inventory costs half of what it costs to run the LIST API, and it's readily available when you need it since it's scheduled. The inventory report objects can also be encrypted using either SSE-S3 or SSE-KMS.

You have object level encryption status field in the report to give you this visibility or audits and reporting for compliance. You can query the S3 inventory report directly from Amazon Athena, Redshift Spectrum, or any Hive tools.

The inventory report can live in the source bucket or can be directed to another destination bucket.

The source bucket contains the objects that are listed in the inventory and contains the configuration for the inventory.

The destination bucket contains the flat file list and the `manifest.json` file that lists all the flat file inventory lists that are stored in the destination bucket. Additionally, the destination bucket for the inventory report must have a bucket policy configured to grant S3 permission to verify ownership of the bucket and permission to write files to the bucket, it must be in the same region as the source bucket it is listing, and it can be the same as the source bucket. Also the destination bucket can be in another AWS account. When creating any filters for your inventory report, it should be noted that tags cannot be used in the filter.

You can set up an Amazon S3 event notification to receive notice when the manifest checksum file is created, which indicates that an inventory list has been added to the destination bucket.

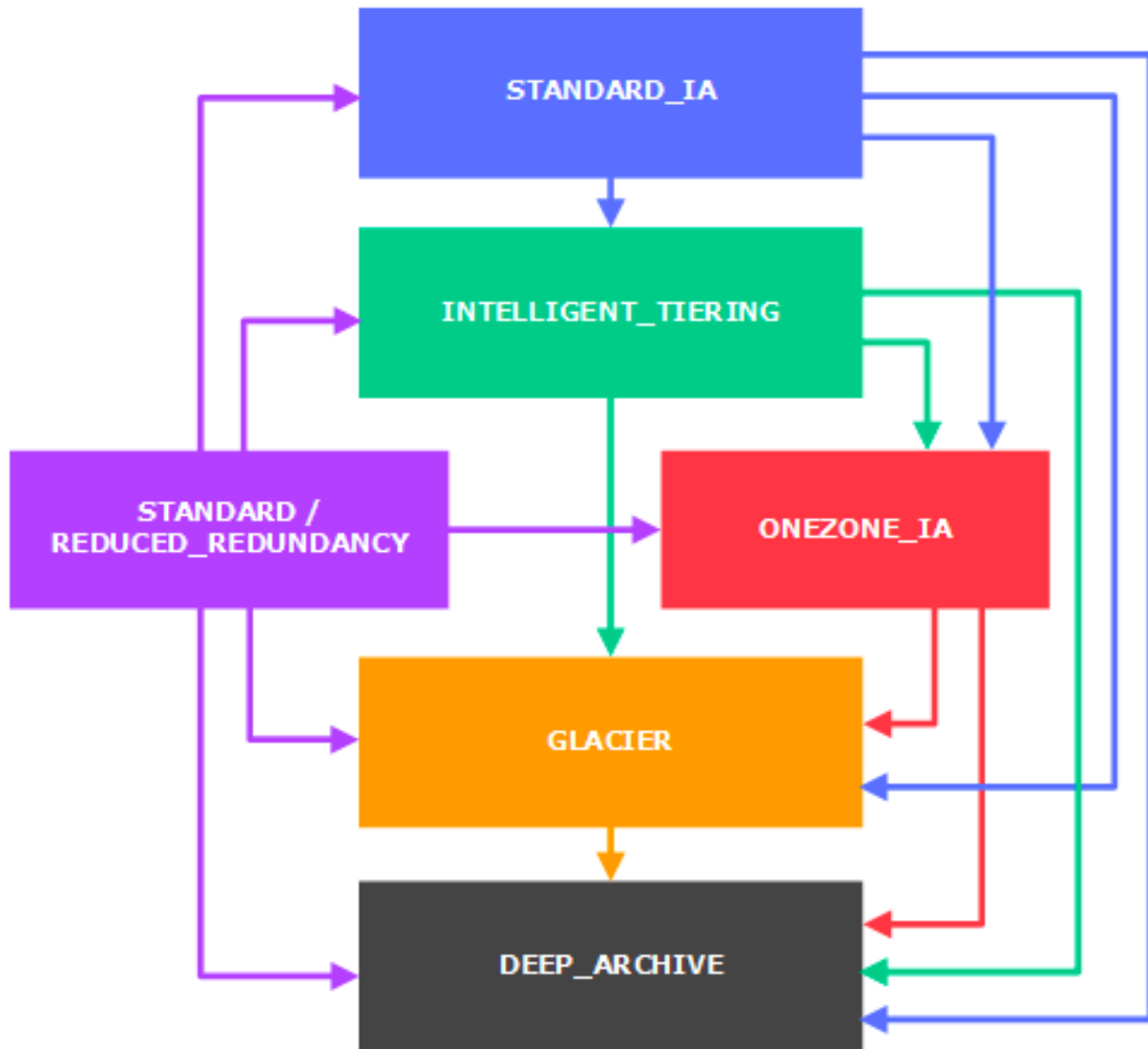


Fig. 17: Lifecycle configuration: Transitioning objects

Report Field	Description	Report Field	Description
Bucket Name	Name of bucket for inventory	ETag	ETag is the MD5 hash in many cases. However, it will not be the MD5 if the object is SSE or uploaded via multipart upload.
Key Name	Object Key name	Storage class	Storage class used for object
Version ID	Object version id (if versioning is enabled)	Multipart upload flag	Set to TRUE if object was uploaded as multipart upload
IsLatest	Set to TRUE if object at current version	Delete marker	Set to TRUE if object is a delete marker
Size	Object size in bytes	Replication status	Set to PENDING, COMPLETED, FAILED or REPLICA
Last modified date	Creation or last modified date	Encryption Status	Set to SSE-S3, SSE-C, SSE-KMS, or NO-SSE

Fig. 18: Fields that are contained in the Inventory report

### Cross-region replication

Cross-region replication (CRR) is a bucket-level feature that enables automatic, asynchronous replication of objects across buckets in different AWS regions. To activate this feature, you add a replication configuration to your source bucket. To configure, you provide information such as the destination bucket where you want objects replicated to. The destination bucket can be in either the same account or another AWS account. Once enabled you will only replicate new PUTs or new object creation. Any existing objects in your bucket will need to be manually copied to the destination.

You can request S3 to replicate all or a subset of objects with specific key name prefixes. Deletes and lifecycle actions are not replicated to the destination. If you delete an object in the source, it will not be deleted in the destination bucket. Additionally, any lifecycle policies you have on the source bucket will only be applied to that bucket. If you wish to enable lifecycle policies on the destination bucket, you will have to do so manually.

To ensure security, S3 encrypts all data in transit across AWS regions using SSL/TLS. In addition to the secure data transmission, CRR can support the replication of server side encrypted data. If you have SSE objects, either SSE-S3 or SSE-KMS, then CRR will replicate these keys to the remote region.

You might configure CRR on the bucket for various reasons. Some common use cases are:

- *Compliance requirements.* Although, by default, S3 stores your data across multiple geographically distant AZs, compliance requirements might dictate that you store data at even further distances. CRR allows you to replicate data between distant AWS regions to satisfy these compliance requirements.
- *Minimize latency.* Your customers might be in 2 geographic locations. To minimize latency in accessing objects, you can maintain object copies in AWS regions that are geographically closer to your users.
- *Operational reasons.* You might have compute clusters in 2 different regions that analyze the same set of objects. You might choose to maintain object copies in those regions.
- *Data protection.* You might have a need to ensure your data is protected, ensuring you have multiple copies of your most important data for quick recovery or business continuity reasons.

There are some requirements you should be aware of for using and configuring CRR:

- The source and destination buckets must have versioning enabled.

- The source and destination buckets must be in different AWS Regions.
- S3 must have the proper permissions to replicate objects from the source bucket to the destination bucket on your behalf.

You can now overwrite ownership when replication to another AWS account. CRR supports SSE-KMS encrypted objects for replication. You can choose a different storage class for your destination bucket. You can replicate to any other AWS region in the world for compliance or business needs or for costs considerations. You can have bi-directional replication. This means you can replicate source to destination and destination back to source. You will have independent lifecycle policies on the source and destination buckets.

If you want to prevent malicious delete of the secondary copy, you can take advantage of the ownership overwrite feature. You can also choose to replicate to another AWS account with CRR. When choosing another AWS account as the destination, you can enable ownership overwrite and S3 will replicate your data and change the ownership of the object to the owner of the destination bucket.

### Trigger-based events

You can automate function based on events. You can use notifications when objects are created via PUT, POST, COPY, DELETE or a multipart upload. You can also filter the event notification on prefixes and suffixes of your objects, so you can ensure you only get the event notification you want and not just on the whole bucket. For example, you can choose to receive notifications on object names that start with “*images/*”. You can then trigger a workflow from an event notification sent to SNS, SQS or Lambda. The benefits of this feature are:

- *Simplicity.* Notifications make it simple to focus on applications by attaching new functionality driven by events. There is no need to manage fleets of EC2 instances to process a queue.
- *Speed.* For example, if you need processing to occur quickly when new objects arrive in your bucket. On average, notifications are sent in less than 1 second.
- *Integration.* Use services to connect storage in S3 with workflows. You can architect an application in a new way, where blocks of code or workflows are invoked by changes in your data.

### Avoid accidental deletion

To avoid accidental deletion in Amazon S3 bucket, you can:

- Enable Versioning.
- Enable MFA (Multi-Factor Authentication) Delete.

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures.

If the MFA (Multi-Factor Authentication) Delete is enabled, it requires additional authentication for either of the following operations:

- Change the versioning state of your bucket.
- Permanently delete an object version.

### S3 consistency

Amazon S3 provides read-after-write consistency for PUTS of new objects in your S3 bucket in all regions with one caveat: if you make a HEAD or GET request to the key name (to find if the object exists) before creating the object,

Amazon S3 provides eventual consistency for read-after-write. Amazon S3 offers eventual consistency for overwrite PUTS and DELETES in all regions.

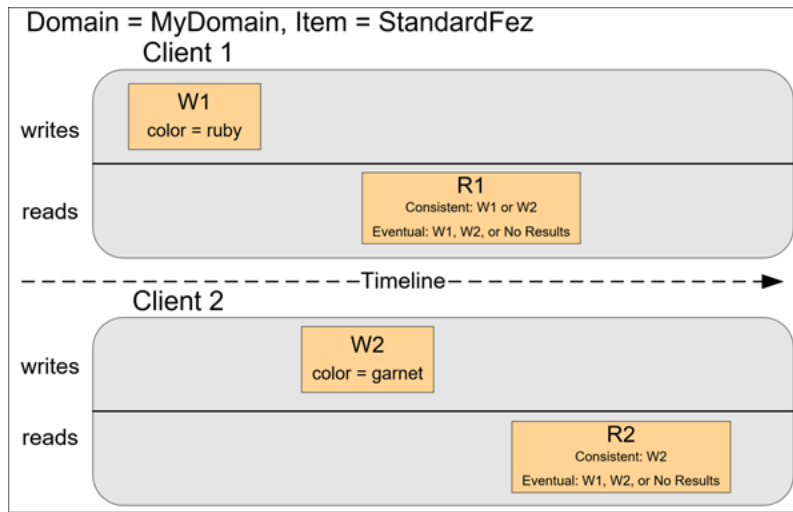


Fig. 19: S3 consistency

Updates to a single key are atomic. For example, if you PUT to an existing key, a subsequent read might return the old data or the updated data, but it will never return corrupted or partial data. This usually happens if your application is using parallel requests on the same object.

Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However, information about the changes must replicate across Amazon S3, which can take some time, and so you might observe the following behaviors:

- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.
- A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 might list the deleted object.

Amazon S3 support for parallel requests means you can scale your S3 performance by the factor of your compute cluster, without making any customizations to your application. Amazon S3 does currently support Object Locking: [Locking Objects Using Amazon S3 Object Lock](#).

Updates are key-based; there is no way to make atomic updates across keys. For example, you cannot make the update of one key dependent on the update of another key unless you design this functionality into your application.

## Monitoring and analyzing Amazon S3

### Storage class analysis

You might ask yourself, which part of my data is cold or hot? What is the right lifecycle policy for my data? Let's look at ways that you can save storage costs by leveraging storage class analysis. Storage class analysis allows you get some intelligence around object access patterns that will give you some guidance around the optimal transition time to a different storage class.

Storage class analysis delivers a daily-updated report of object access patterns in your S3 console that helps you visualize how much of your data is hot, warm, or cold. Then, after about a month of observation, Storage class analysis presents you with recommendations for lifecycle policy settings designed to reduce TCO. It does this by monitoring object access patterns over that period of time, and populates a series of visualizations in your S3 console.

By using Amazon S3 Storage class analysis you can analyze storage access patterns to help you decide how to transition the right data to the right storage class. It will provide a visualization of your data access patterns over time, measure the object age where data is infrequently accessed, and enable you to deep dive by bucket, prefix or object tag. Storage class analysis also provides daily visualizations of your storage usage in the AWS Management console. You can optionally export files that include a daily report of usage, retrieved bytes, and GETs by object age to a S3 bucket to analyze using the BI tools of your choice, such as Amazon QuickSight.

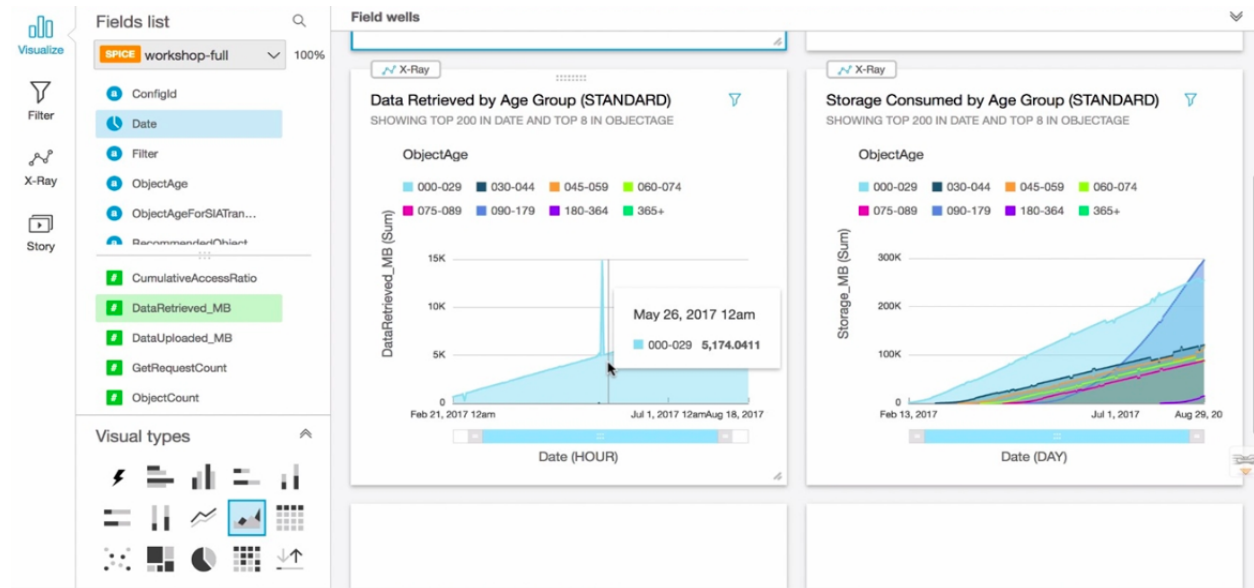


Fig. 20: Storage class analysis QuickSight dashboard

## Amazon CloudWatch metrics

Amazon CloudWatch metrics for Amazon S3 can help you understand and improve the performance of applications that use S3. There are 2 types of CloudWatch metrics for Amazon S3, storage metrics and request metrics.

**Daily storage metrics** for buckets are reported once per day for bucket size and number of objects and metrics, it is included at no additional cost.

Additionally you can enable **Request metrics**. There is an additional cost for these metrics. You can receive 13 metrics that are available at 1-minute intervals. Once enabled, these metrics are reported for all object operations.

## Amazon CloudWatch logs

Amazon CloudWatch logs is a feature of CloudWatch that you can use specifically to monitor log data. Integration with Amazon CloudWatch logs enables CloudTrail to send events containing API activity in your AWS account to a CloudWatch Logs log group. CloudTrail events that are sent to CloudWatch Logs can trigger alarms according to the metric filters you define. You can optionally configure CloudWatch alarms to send notifications or make changes to the resources that you are monitoring based on log stream events that your metric filters extract.

Metric Name	value
BucketSizeBytes	Count
NumberOfObjects	Count

Fig. 21: Daily storage metrics

Metric name	Value
AllRequests	Count
PutRequests	Count
GetRequests	Count
ListRequests	Count
DeleteRequests	Count
HeadRequests	Count
PostRequests	Count

Metric name	Value
BytesDownloaded	MB
BytesUploaded	MB
4xxErrors	Count
5xxErrors	Count
FirstByteLatency	ms
TotalRequestLatency	ms

Fig. 22: Request metrics



Using Amazon CloudWatch Logs, you can also track CloudTrail events alongside events from the OS, applications, or other AWS services that are sent to CloudWatch Logs. For example, you may want to be alerted if someone changes or deletes a bucket policy, lifecycle rule or other configuration.

## **Optimizing performance in Amazon S3**

Amazon S3 now provides increased performance to support at least 3,500 requests per second to add data and 5,500 requests per second to retrieve data, which can save significant processing time for no additional charge. Each S3 prefix can support these request rates, making it simple to increase performance significantly.

Applications running on Amazon S3 today will enjoy this performance improvement with no changes, and customers building new applications on S3 do not have to make any application customizations to achieve this performance. Amazon S3's support for parallel requests means you can scale your S3 performance by the factor of your compute cluster, without making any customizations to your application. Performance scales per prefix, so you can use as many prefixes as you need in parallel to achieve the required throughput. There are no limits to the number of prefixes.

This S3 request rate performance increase removes any previous guidance to randomize object prefixes to achieve faster performance. That means you can now use logical or sequential naming patterns in S3 object naming without any performance implications. This improvement is now available in all AWS Regions.

When using Amazon S3 it is important to consider the following best practices:

- Faster uploads over long distances with Amazon S3 Transfer Acceleration.
- Faster uploads for large objects with Amazon S3 multipart upload.
- Optimize GET performance with Range GET and CloudFront.
- Distribute key name for high RPS workload.
- Optimize list with Amazon S3 inventory.

## **High requests per second**

In some cases, you may not need to be overly concerned about what your key names are, as S3 can handle thousands of requests per second (RPS) per prefix. This is not to say that prefix naming should not be taken into consideration, but rather that you may or may not fall into the category of high request rates of over 1000 RPS on a given prefix in your bucket.

However, if you will regularly be performing over 1000 RPS, then you should take care with your key naming scheme to avoid hot spots which could lead to poor performance. The 2 most common schemes which can lead to hotspotting are daa based keys and monotonically increased numbers. These schemes do not partition well due to the sameness at the begin of the key. Objects in S3 are distributed across S3 infrastructure according to the object's full name (that is, its key). The object keys are stored in S3's indexing layer. S3 splits the indexing layer into partitions and stores keys within those partitions based on the prefix. S3 will also split these partitions automatically to handle an increase in traffic. What is important to understand is that your key name alone does not necessarily show you what partition your object is stored in, as S3 is automatically partitioning objects broadly across its infrastructure for performance. When the automatic partitioning is occurring you may see a temporary increase in 503 - Slow Down responses. These should be handled like any other 5xx response, using exponential backoff retries and jitter. By using exponential backoff retries and jitter, you are able to reduce the requests per second such that they no longer exceed the maximum RPS rate. Once the partitioning is complete, the 503s will no longer be sent and you will be able to go at the higher rate.

With this in mind, you should also ensure that when you have a known expectation of a high RPS event you should request a pre-partitioning of your prefixes to ensure the optimal performance. You can contact AWS support to pre-partition your workload. You will need to provide the key naming scheme and expected requests to PUT, GET, DELETE, LIST and AWS can then partition for you in advance in the event. As this generally takes a few days, it is recommended that you open a case at least 1-2 weeks in advance.



To avoid hot-spotting, avoid using sequential key names when you are expecting greater than 1000 RPS. It is recommended to place a random hash value at the left most part of the key. One of the easiest ways to do this is to take the MD5 or an equivalent hashing scheme like CRC of the original key name, and then take the first 3 to 4 characters of the hash and prepend that to the key. You can see in the following example that this naming scheme makes it harder to list objects that are related to each other.

```
awsexamplebucket/232a-2013-26-05-15-00-00/cust1234234/animation1.jpg
awsexamplebucket/7b54-2013-26-05-15-00-00/cust3857422/video2.jpg
awsexamplebucket/91c-2013-26-05-15-00-00/cust1248473/photo3.jpg
```

To help with that, you can use a small set of prefixes to organize your data. Deciding where the random value such as the hash should be placed can be confusing. A recommendation is to place the hash after the more static portion of your key name, but before values such as dates and monotonically increasing numbers.

```
awsexamplebucket/animations/232a-2013-26-05-15-00-00/cust1234234/photo1.jpg
awsexamplebucket/videos/7b54-2013-26-05-15-00-00/cust3857422/video2.jpg
awsexamplebucket/photos/91c-2013-26-05-15-00-00/cust1248473/photo3.jpg
```

Even with good partitioning, bad traffic that heavily hits one prefix can still create hot partitions, which can be confusing if you have already pre-partitioned the bucket. An example of this could be a bucket that has keys that are UUIDs. In theory, this can be good for bucket partitioning, but if you copy these keys between buckets, or from the bucket to local hosts, it will list the keys alphabetically causing all the requests to hit “000”, then “001”, “002”, etc. potentially creating a hotspot.

## High throughput

The most common design pattern used by customers for performance optimization is parallelization. Parallelization can be achieved in 2 ways: one is to have multiple connections to upload your data and the other is multipart uploads. You should also take into consideration the network throughput of the hosts and devices along the network path when you are uploading data. For example, if you are using EC2, you may want to choose network optimized best network performance when using parallel uploads or downloads.

## Multipart uploads

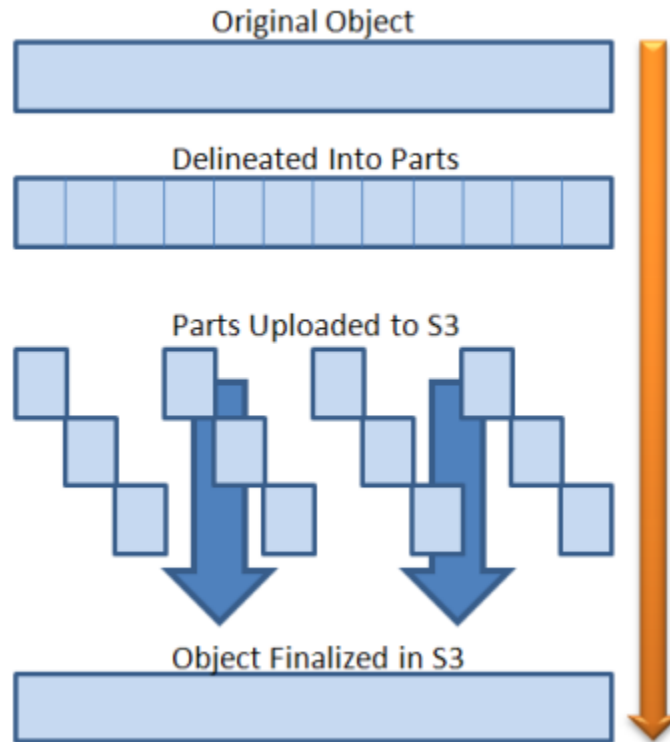
The multipart upload API enables you to upload large objects in a set of parts and you can also upload those parts in parallel. Multipart uploading is a three-step process:

1. You initiate the upload: `InitiateMultipartUpload(partSize) -> uploadId`
2. Upload the object parts: `UploadPart(uploadId, data)`
3. Complete the multipart upload: `CompleteMultipartUpload(uploadId) -> objectId`

Upon receiving the complete multipart upload request, S3 constructs the object from the uploaded parts, and you can then access the object just as you would any other object in your bucket.

In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of a single object upload. Also consider using multipart upload when uploading objects over a spotty network, this way you only need to retry the parts that were interrupted, this increasing the resiliency for your application. Using multipart upload provides a few advantages. The following are some examples:

- **Improved throughput.** You can upload parts in parallel to improve throughput.
- **Quick recovery from any network issues.** A smaller part size minimizes the impact of restarting a failed upload due to a network error.



- **Pause and resume object uploads.** You can upload object parts over time. Once you initiate a multipart upload there is no expiry; you must explicitly complete or abort the multipart upload.
- **Begin an upload before you know the final object size.** You can upload an object as you are creating it.

You should also remember to use `AbortIncompleteMultipartUpload` action in case your upload doesn't complete to avoid unwanted storage costs of abandoned multipart uploads. You can configure the `AbortIncompleteMultipartUpload` in the lifecycle rules configuration of your bucket in the S3 console. This directs S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When a multipart upload is not completed within the time frame, it becomes eligible for an abort operation and S3 aborts the multipart upload and deletes the parts associated with the multipart upload.

There are also some tools that can be used in S3 for multipart uploads such as `TransferManager` which is part of S3 SDK for Java. `TransferManager` provides a simple API for uploading content to S3, and makes extensive use of S3 multipart uploads to achieve enhanced throughput, performance and reliability. When possible, `TransferManager` attempts to use multiple threads to upload multiple parts of a single upload at once. When dealing with large content size and high bandwidth, this can have a significant increase on throughput.

### Multipart Upload Overview

## Amazon CloudFront

You can consider using Amazon CloudFront in conjunction with Amazon S3 to achieve faster downloads.

## Transfer Acceleration

With Transfer Acceleration, as the data arrives at an edge location, data is routed to S3 over an optimized network path. Each time you use Transfer Acceleration to upload an object, AWS will check whether Transfer Acceleration is likely to be faster than a regular S3 transfer. If AWS determines that Transfer Acceleration is not likely to be faster

than a regular S3 transfer of the same object to the same destination AWS region, AWS will not charge for that use of Transfer Acceleration for that transfer, and may bypass Transfer Acceleration for that upload. You can test and see if Amazon S3 Transfer Acceleration will provide you a benefit by going to [Amazon S3 Transfer Acceleration - Speed Comparison](#).

A possible use case is: Let's say you transfer data on a regular basis across continents or have frequent uploads from distributed locations. Transfer Acceleration can route your data to the closest edge location, so it travels a shorter distance on the public internet and majority of the distance on an optimized network on the Amazon backbone. Moreover, Transfer Acceleration is that it uses standard TCP and HTTP/HTTPS so it does not require any FW exceptions or custom software installation.

When using Transfer Acceleration, additional transfer charges may apply.

## Optimize List with Inventory

You can help optimize getting this list by using Amazon S3 inventory. If using the LIST API, this may take some time to parse through all the objects and add time to running the process for your application. By using Amazon S3 inventory, you can have your application parse through the flat file which inventory has produced helping decrease the amount of time required to list your objects.

## Amazon S3 Cost and Billing

### Amazon S3 charges

AWS always bills the owner of the S3 bucket for Amazon S3 fees, unless the bucket was created as a Request Pays bucket. To estimate the cost of using S3, you need to consider the following:

- **Storage** (Gbs per month). You pay for storing objects in your Amazon S3 buckets. The rate you're charged depends on your object's size, how long you stored the objects during the month and the storage class. You can reduce the costs by storing less frequently accessed data at slightly lower levels of redundancy than the Amazon S3 standard storage. It is important to note that each class has different rates.
- **Requests**. You pay for the number and type of requests. GET requests incur charges at different rates than other requests, such as PUT and COPY requests.
- **Management**. You pay for the storage management features. For example: S3 inventory, analytics, and object tagging, that are enabled on your account's buckets.
- **Data transfer**. You pay for the amount of data transferred in and out of the Amazon S3 region, except for the following:
  - Data transfer into Amazon S3 from the Internet.
  - Data transfer out to an Amazon EC2 instance, when the instance is in the same AWS Region as the S3 bucket.
  - Data transfer out to Amazon CloudFront.

You also pay a fee for any data transferred using Amazon S3 Transfer Acceleration.

When using the S3 Standard-IA, S3 One Zone-IA or Amazon Glacier, there are some additional charges that you can incur for retrieval. These storage classes have a minimum storage time commitment to avoid additional charges. You pay for deleting an object stored before the minimum storage commitment has passed.

- Amazon S3 Standard-IA and S3 One Zone-IA have a 30 day minimum storage requirement.
- Amazon Glacier has a 90 day minimum storage requirement.

## **Bills**

In your AWS console in the Bills section you can see some detailed cost information for S3 for each region where you have data stored.

From the AWS console dashboard, you will be able to easily see the monthly cost per service.

## **Cost explorer**

You can enable cost explorer from the AWS console dashboards. Once enabled, it will take 24 hours for the information to populate, and you can have the graphically view of your monthly costs per service.

New - AWS Transfer for SFTP - Fully Managed SFTP Service for Amazon S3

## **1.3.2 Amazon S3 Glacier**

### **Amazon Glacier overview**

#### **Definition**

At its core, Amazon Glacier is an economical, highly durable storage service optimized for infrequently used or cold data. It is widely used for workloads such as backup, preservation archival, regulatory compliance, or as a tier for historical data in a data-lake architecture.

With Amazon Glacier, customers can store their data cost-effectively for months, years, or even decades. It enables customers to offload the administrative burdens of operating and scaling storage to AWS, so they don't have to worry about capacity planning, HW provisioning, data replication, HW failure detection and recovery, or time-consuming HW and tape-media migrations.

Amazon Glacier is designed to deliver 11 9s of durability, and provides comprehensive security and compliance capabilities that help meet the most stringent regulatory requirements such as SEC-17A4. For data-lake architectures, Amazon Glacier provides data filtering functionality, allowing you to run powerful analytics directly on your archive data at rest. Amazon Glacier provides several data retrieval options that allow access to archives in as little as a few minutes to several hours.

### **Amazon Glacier data model**

The Amazon Glacier data model is composed of vaults and archives. The concept is similar to Amazon S3's bucket and object model. An archive can consist of any data such as photos, videos, or documents. You can upload a single file as an archive or aggregate multiple files into a TAR or ZIP file, and upload it as one archive, via either the Amazon S3 or the Amazon Glacier native API.

When storing data via Amazon Glacier-native API, a single archive can be as large as 40 TB, and an unlimited number of archives can be stored in Amazon Glacier. Each archive is assigned a unique archive ID at the time of creation, and the contents are immutable and cannot be modified. Amazon Glacier archives support only upload, download, and delete operations. Unlike Amazon S3 objects, the ability to overwrite or modify an existing archive is not supported via the Amazon Glacier-native API.

A vault is a container for storing archives. When creating a vault in Amazon Glacier, you specify a name and AWS region for your vault, and that generates a unique address for each vault resource. The general form is `http://<region-specific endpoint>/<account-id>/vaults/<vault-name>`. For example:

```
https://glacier.us-west-2.amazonaws.com/123456789/vaults/myvault
```

An AWS account can create vaults in any of the supported regions, and you can store an unlimited number of archives in a vault. You can create up to 1000 vaults per account, per region.

### Amazon Glacier entry points

Currently, there are 3 methods for sending data to Amazon Glacier:

1. You can run commands in the AWS CLI by using the Amazon Glacier native API, or automate your uploads via the Amazon Glacier SDK.
2. You can transfer data directly by using direct AWS data ingestion tools or 3rd party software (for instance Commvault or NatApp).
3. You can upload an object to an Amazon S3 bucket and use Amazon S3 lifecycle policies to transition your data to Amazon Glacier when specified conditions are met.

To directly transfer data into Amazon Glacier, AWS Direct Connect, AWS Storage Gateway, and the AWS Snow Family are some of the options available that allow movement of data into and out of Amazon Glacier. Uploads can be performed using REST-based SDKs, the AWS management console, the Amazon Glacier API, and the AWS CLI.

[AWS re:Invent 2018: \[REPEAT 2\] Best Practices for Amazon S3 and Amazon Glacier \(STG203-R2\)](#)

### Benefits

Core benefits for customers who use Amazon Glacier are its unmatched durability (11 9s), availability and scalability (data is automatically distributed across a minimum of 3 physical facilities that are geographically separated), comprehensive security and compliance capabilities, query and analytics features, flexible management capabilities, and a large software ecosystem.

### Security and compliance

Amazon Glacier's comprehensive security capabilities begin with AES 256-bit server-side encryption with built-in key management and key protection. Customers also have the option of managing their own keys by encrypting their data before uploading to Amazon Glacier, but AES256 server-side encryption is always on and cannot be disabled.

Amazon Glacier supports many security standards and compliance certifications including PCI-DSS, HIPAA/HITECH, FedRAMP, SEC Rule 17-a-4, EU Data Protection Directive, and FISMA, helping to satisfy compliance requirements for virtually every regulatory agency around the globe. Amazon Glacier integrates with AWS CloudTrail to log, monitor, and retain storage API call activities for auditing.

Amazon Glacier encrypts your data at rest by default and supports secure data transit with SSL.

### Query in place

Anyone who knows SQL can use Amazon Athena to analyze vast amounts of unstructured data in Amazon S3 on-demand. You can use Amazon Glacier Select to conduct filter in place operations for data-lake analytics.

## Flexible management

Amazon Glacier and Amazon S3 offer a flexible set of storage management and administration capabilities. If you are a storage administrator, you can classify, report, and visualize data usage trends to reduce costs and improve service levels.

Via the Amazon S3 API, objects can be tagged with unique, customizable metadata so that customers can add key-value tags that support data-management functions including lifecycle policies. The Amazon S3 inventory tool delivers daily reports about objects and their metadata for maintenance, compliance, or analytics operations. Amazon S3 can also analyze object access patterns to build lifecycle policies that automate tiering, deletion, and retention.

As Amazon S3 works with AWS Lambda, you can log activities, define alerts, and automate workflows, all without managing any additional infrastructure.

## Large ecosystem

In addition to integration with most AWS services, the Amazon S3 and Amazon Glacier ecosystem includes tens of thousands of consulting, systems integrators, and Independent Software Vendors (ISV) partners. This means that you can easily use Amazon S3 and Amazon Glacier with popular backup, restore, and archiving solutions (including Commvault, Veritas, Dell EMC, and IBM), popular DR solutions (including Zerto, CloudEndure, and CloudRanger), and popular on-premises primary storage environments (including NetApp, Dell EMC, Rubrik, and others)

## Amazon Glacier versus Tape solutions

Amazon Glacier is architected to deliver a tape-like customer experience, with features, performance, and a cost model that is similar in many ways to large-scale on-premises tape solutions, while eliminating the common problems of media handling, complex technology refreshes, mass migrations, and capacity planning.

## Amazon Glacier features

Amazon Glacier comes with many built-in features that help customers effectively secure, manage, and gain insights from their vaults.

Amazon Glacier supports **IAM permissions** that give you fine-grain controls over which users or resources have access to data stored in Amazon Glacier. In addition to this, you can attach **Vault Access Policies** to Amazon Glacier vaults that specify access and actions that can be performed by a particular resource. Vault access policies can also be used to grant read-only access to a 3rd party using a different AWS account.

To satisfy compliance needs, Amazon Glacier's **Vault Lock** feature allows you to easily deploy and enforce WORM (immutability) on individual vaults via a lockable policy, per regulatory requirements such as SEC-17a4. Unlike normal access control policies, when locked, the Vault Lock policy becomes immutable for the life of the Vault. Vault Lock specifies archive retention time and provides the option for legal hold on archives for an indefinite period.

For retrievals, Amazon Glacier allows you to retrieve up to 5% of your data daily each month free of charge. Via the Amazon Glacier-native API, **Ranged Retrievals** make it easier to remain within the 5% threshold. Using the `RetrievalByteRange` parameter, you can fetch only the data you need from a larger file, or spread the retrieval of a larger archive over a period of time. This feature allows you to avoid retrieving an entire archive unnecessarily.

## Data lifecycle management

## Object lifecycle management

Object lifecycle management is a core feature of Amazon S3 that allows you to set rules objects in an Amazon S3 bucket (see *Object Lifecycle policies*). Files that are uploaded via the Amazon S3 API may be transitioned to Amazon Glacier in as little as 0-days. These Amazon S3 objects are then moved to an Amazon Glacier Vault but continue to be managed by the parent Amazon S3 bucket and addressed via the originally specified Amazon S3 key name. In fact, the Amazon Glacier Vault used for lifecycle transitions is privately held by the Amazon S3 parent bucket and directly accessible by customers. This is why certain use-cases, such as utilizing the Amazon Glacier Vault Lock capability, still require usage of the Amazon Glacier-native API.

In this example policy, objects that are older than 30 days are move to Amazon S3 Standard-IA, and objects that are older than 90 days are migrated to Amazon Glacier.

```
{
  "Rules": [
    {
      "Filter": {
        "Prefix": "logs/"
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
        {
          "Days": 90,
          "StorageClass": "GLACIER"
        }
      ],
      "Expiration": {
        "Days": 365
      },
      "ID": "example-id"
    }
  ]
}
```

## Classifying workloads

To optimize data accessibility and storage costs, it is important to properly classify workloads before developing life-cycle policies.

## Selecting the right storage class

When choosing the correct storage class for your workloads, ask several questions before indentifying the most appropriate storage class for your data.

1. *How frequently is the data being accessed?* For example, you don't want to access data moved to Standard-IA more frequently than once a month, or you may not realize the financial benefits of storing data in this class.
2. *How long will you store data?* For example, data stored in Standard-IA is required to be stored for 30 days minimum. Customers who delete objects prior to 30 days will be charged for 30 days capacity.

When determining whether your data should be moved to Amazon Glacier, the questions to ask are:

1. *Do I need millisecond access to my data?* If the answer is yes, for example, images that are retrieved for hosting on a live website, then Amazon Glacier is not a good fit.
2. *How many retrieval requests are made?* Amazon Glacier has 3 different retrieval options:
  - **Expedited** retrieval times are from 1 to 5 minutes.
  - **Standard** ranges from 3 to 5 hours.
  - **Bulk** is from 5 to 12 hours.

Amazon Glacier is a good option if the access frequency for Expedited retrievals is less than 3 per year, for Standard retrievals is less than 1 per month, and for Bulk retrievals is less than 4 per month.

3. *How long will I store the data?* Amazon Glacier is best suited for objects that will be stored periods greater than 90 days. Glacier has a 90-day minimum for the objects you store.

## Standard class analysis

To assist customers with determining which datasets are best suited for Standard or Standard-IA, the storage class analysis feature is available as a part of Amazon S3 (see [Storage class analysis](#))

## Differentiating Glacier and S3

One of the key differences between Amazon S3 and Amazon Glacier is the functionality of read operations. In Amazon S3, reads are “synchronous”, meaning an HTTP GET operation will immediately return an object with millisecond latency. In Amazon Glacier, read operations are “asynchronous”, meaning there are 2 steps to retrieve an object (or archive):

1. Restore command is issued (where one of the previously mentioned retrieval methods is specified)
2. An HTTP GET operation will only succeed after Glacier has restored the object to an online state.

## Lifecycle policy structure

You can generate lifecycle policies from the Amazon S3 console or write them manually in XML or JSON. An XML file of a lifecycle policy consists of a set of rules with predefined actions that Amazon S3 can perform on objects in your bucket. Each rule consists of the following elements:

- A **filter** that identifies a subset of object the rule applies to, such as a key name prefix, object tag, or a combination.
- A **status** of whether the rule is in effect.
- One or more lifecycle transitions or expirations actions to perform on filtered objects

```
<LifecycleConfiguration>
  <Rule>
    <ID>example-id</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

(continues on next page)



(continued from previous page)

```

<Transition>
  <Days>90</Days>
  <StorageClass>GLACIER</StorageClass>
</Transition>
<Expiration>
  <Days>365</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>

```

You can apply lifecycle policies to a S3 bucket by using the AWS CLI, but you must first convert your XML code to JSON. Lifecycle policies also support specifying multiple rules, overlapping filters, tiering down storage classes, and version-enabled buckets.

### Zero day lifecycle policies

Lifecycle policies are commonly used across an entire Amazon S3 bucket to transition objects based on age, but there may be scenarios in which you might want to transition single or multiple objects to Glacier based on workflow operations or other status changes. Currently, an API call does not exist that moves a single object of a certain age to Amazon Glacier, but you can achieve this by using tags.

You can write a zero-day lifecycle policy with a tag filter that moves only a single object or a subset of objects to Glacier based on the tag value in the lifecycle policy. Only the objects that match the tag filter in your policy are moved on that day.

```

<LifecycleConfiguration>
  <Rule>
    <Filter>
      <Tag>
        <Key>flaming</Key>
        <Value>cars</Value>
      </Tag>
    </Filter>
    <Transition>
      <Days></Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>

```

### Creating a Lifecycle policy

The simplest method for creating a lifecycle policy is to use the Amazon S3 console. Use this console to:

- Configure your prefix or tag filter.
- Configure the transition of data to S3-Standard-IA or Amazon Glacier.
- Define when objects within Amazon S3 expire.

After saving the rule, it is applied to the objects specified in the lifecycle policy. You also can set lifecycle configuration on a bucket programmatically by using AWS SDKs or the AWS CLI.

## **Glacier backup SW integration**

A widely adopted use-case for Glacier storage is backup. Several popular backup SW solutions and storage gateway products have built-in integration with S3 and Glacier and their lifecycle capabilities. For example, Commvault, an enterprise-grade backup SW solution, can be featured in a hybrid model in which files are archived and analyzed onsite. However, the metadata from those files are stored in S3. Commvault natively talks to S3, which simplifies moving data into AWS. From there, lifecycle policies move data into Amazon Glacier. Commvault also supports the Amazon Glacier-native API, which could be enabled for customers wishing to utilize Vault Lock compliance capabilities.

In addition to Commvault, there are several ISVs that support integration with Glacier including Veritas NetBackup, Arq Backup, Cloudberry, Synology, Qnap, Duplicati, Ingenious, Rubrik, Cohesity, Vembu, and Retrospect.

## **Amazon Glacier durability and security**

### **Durability with traditional storage**

In a common storage architecture, we can have a single data center with RAID arrays. This setup provides a minimum level of data protection but leaves your data in a highly vulnerable position. The lack of geographic redundancy exposes your data to concurrent failures that could be caused by natural or man-made catastrophic events.

To rectify this scenario, an additional DR site is necessary in a different geographic region, which would double your data storage costs and the effort to maintain the environment. An additional DR site does provide additional durability, but the data may not be synchronously redundant.

AWS uses the Markov model to calculate the 11 9s of durability that Glacier and S3 are designed to provide. In general, a well-administered single-datacenter solution that supports multiple concurrent failures using a single copy of tape can deliver 4 9s of durability. Furthermore, a multiple-data center solution with 2 copies of tape in separate geographic regions can deliver 5-6 9s of durability.

### **Amazon Glacier durability**

With a single upload to Glacier, your objects are reliably stored across multiple AZs. This scenario provides tolerance against concurrent failures across disks, nodes, racks, networks, and WAN providers. In addition to Amazon's default 3-AZ durability model, customers may optionally duplicate their data in another geographic region.

## **Amazon Glacier security**

### **Encryption features**

Data stored in Glacier is encrypted by default, and only vault owners have access to resources created on Glacier. Access to resources can be controlled via IAM policies or vault access policies.

Glacier automatically encrypts data at rest by using AES 256-bit encryption and supports SSL data transfers. Furthermore, if data that is uploaded into Glacier is already encrypted, that data is re-encrypted at rest automatically by default.

Objects that transition from S3 to Glacier also supports SSE-SE. Users can upload objects to SE with SSE and later transition them to Glacier. With SSE enabled, encryption is applied to objects in S3, objects transitioned to Glacier, and objects restored from Glacier to S3. If SSE is not enabled during upload, objects transitioned to Glacier are encrypted, but objects stored in S3 will not be encrypted.

## Auditing and logging

With AWS CloudTrail logging enabled, API calls made to Amazon Glacier are tracked in log files that are stored to a specified S3 bucket. Using the information collected by CloudTrail, you can determine requests made to Glacier, the source IP of the request, the time of the request, and who made the request. By default, CloudTrail log files are encrypted, and you can define lifecycle rules to archive or delete log files automatically. You can also configure SNS notifications to alert you when new log files are generated if you must act quickly.

## Vault Lock and Vault access policies

A Vault Lock policy specifies a retention requirement in days from the date of archive creation. Until the retention date has passed, the archive is locked against changes or deletion that functions like a WORM storage device. Vault Lock supports a Legal Hold, which is intended to support the extension of WORM for archives subject to an ongoing legal event. Locking a policy means that the policy becomes immutable or cannot be changed, and Glacier enforces the compliance controls specified in the policy for the specified vault.

Amazon Vault Lock eliminates the need for customers to purchase expensive WORM storage drives that require periodic refreshes for record retention. Customers can now easily set up a Vault Lock policy with 7 years of record retention. Glacier enforces the retention control that ensures that archives stored in the vault cannot be modified or deleted until the 7-year period expires.

## Differentiating Vault Lock and Vault access policies

A Vault Lock policy is not the same as a Vault access policy. While both policies govern access to your vault, vault lock policies cannot be changed once locked, and that provides strong enforcement for compliance controls. In addition, only one Vault Lock policy may be created for a vault, and it lasts for the life of a vault.

For example, a vault lock policy can deny deleting archives from a vault for a time period to ensure records are retained in accordance with a law or regulation. In contrast, a vault access policy would be used to grant access to archives that are not compliance related and that are subject to frequent modification. Vault lock policies and vault access policies can be used together. For example, a vault lock policy could be created to deny deletes inside of a vault, while a vault access policy could allow read only access to an auditor.

## Vault Lock example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "deny-based-on-archive-age",
      "Principal": "*",
      "Effect": "Deny",
      "Action": "glacier:DeleteArchive",
      "Resource": [
        "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
      ],
      "Condition": {
        "NumericLessThan": {
          "glacier:ArchiveAgeInDays": "365"
        }
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    ]
}

```

Shown here is an example if a Vault policy that is being initiated on a vault named BusinessCritical. The “Effect” for the policy is set to “Deny”, which will affect any actions specified in the policy. The “Principal” designated the user or group that this policy applies to, which in this case is set to “Everyone”, which is denoted by the star symbol in quotes.

The DeleteArchive action is specified and a date condition specifies a range of 365 days or fewer for archives in the BusinessCritical vault. This means that the vault policy denies anyone who attempts to delete archives that are less than 365 days old. Glacier compares the archive creation date and the current date to determine the archive age. If an archive has a calculated age of less than 1 year, any delete request attempt will be declined. Note that if you are uploading older datasets, they will adopt a new date-created when they are uploaded. Therefore, your Vault Lock retention policy should be adjusted accordingly.

## Two-step process

Because locking a vault is immutable and cannot be altered once a vault is locked, it is important to understand the two-step locking process. To lock a vault, follow these steps:

1. Initiate the lock by attaching a Vault Lock policy to your vault, which sets the lock to an in progress state and generates a lock ID.
2. Use the lock ID to complete the process before the lock ID’s 24-hour expiration.

When you select the `InitiateVaultLock` operation, the vault locking process begins and allows you to work in test mode. The test mode enables you to test the effectiveness of your vault lock policy. The unique lock ID that is generated expires after 24 hours, which gives ample time for testing. If you are not satisfied with the policy, you can use “`AbortVaultLock`” operation to delete an in progress policy or modify your policy before locking it down.

After you have thoroughly tested your Vault Lock policy, you can complete the vault lock by using the `CompleteVaultLock` operation. To perform the final commit on the vault, supply the most recent lock ID. After the `CompleteVaultLock` operation is run, no changes can be made to the vault lock policy.

## Vault tags

You can assign tags to your Glacier vaults for easier cost and resource management. Tags are key-value pairs that you associate with your vaults. By assigning tags, you can manage vault resources and organize data, which is useful when constructing AWS reports to determine usage and costs. Create a tag by defining a key and value that can be customized to meet your needs. For example, you can create a set of tags that defines the owner, purpose, and environment for the vaults you create.

The maximum number of tags that you can assign to a vault is 50. Keys and values are case-sensitive. Note that these tags are applied to the Amazon Glacier Vault resource, not to individual archives like Amazon S3 object tags.

You can also use a Legal Hold tag as a condition in Vault Lock policy statements. Suppose that you have a time-based retention rule that an archive can only be deleted if it is less than a 1 year old. At the same time, suppose that you need to place a legal hold on your archives to prevent deletion or modification for an indefinite duration during a legal investigation. In this case, the legal hold takes precedence over the time-based retention rule specified in the Vault Lock policy. To accommodate this scenario, the following example policy has 1 statement with 2 conditions: The first condition will DENY deletion of archives that are less than 365 days old and the second condition will also DENY deletion of archives as long as the vault level hold tag is set to true.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "no-one-can-delete-any-archive-from-vault",
      "Principal": "*",
      "Effect": "Deny",
      "Action": [
        "glacier:DeleteArchive"
      ],
      "Resource": [
        "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
      ],
      "Condition": {
        "NumericLessThan": {
          "glacier:ArchiveAgeInDays": "365"
        },
        "StringLike": {
          "glacier:ResourceTag/LegalHold": [
            "true",
            ""
          ]
        }
      }
    }
  ]
}
```

A vault lock policy written in this manner prevents deletion or modification of archives in a vault for up to 1 year or until the legal hold tag value is set to false.

## Optimizing costs on Amazon Glacier

### Amazon Glacier pricing model

Amazon Glacier's best feature is its ultra-low pricing, its key feature is the ability to store more data for less. In addition to storage costs, Amazon Glacier pricing is segmented into several pricing categories:

- **Retrieval pricing.** Volume-based and transaction-based fees that are based on expedited, standard, or bulk retrieval methods.
- **Request pricing,** which is based on the number of archive upload requests or lifecycle transitions from S3.
- **Data transfer pricing,** which is based on data transferred out of Amazon Glacier.
- **Amazon Glacier Select pricing,** which is charged separately by Amazon and is based on the amount of data scanned and number of Glacier Select requests.

Amazon S3 Glacier pricing (Glacier API only)

### Object targeting best practices

When transitioning to Amazon Glacier, it is advised to target average object sizes because there is a 32-KB overhead charge for each object stored on Glacier regardless of object size. In practice, this means that if the objects in your Glacier vault have an average object size of 32 KB, you would in effect be paying twice: once for the 32-KB overhead charge and also for the standard storage charge. Therefore, it is good practice to target objects that are 1 MB and above

for transitioning to Glacier. If your workloads tend to store small files, it is recommended to containerize small files using TAR, ZIP, or similar tools.

Glacier is designed for long-lived data and imposes a 90-day minimum retention requirement as well as upload fees that are 10 times those of S3. These fees and limitations are typically not material costs for large-scale long-term archival. But customers with very active small-file workloads may find S3 or S3-IA is a more appropriate storage solution.

### Ingesting data into Amazon Glacier

Unlike S3, the Amazon Glacier console does not provide an interface to upload archives directly from the AWS console. Use either lifecycle policies or the Amazon Glacier API, or direct uploads through 3rd party tools.

### Network ingestion options

In addition to uploading data via the Amazon Glacier API, data can be moved into Glacier by implementing a SSL tunnel over the public Internet using several 3rd party options. You can also transfer data using AWS Direct Connect. AWS Direct Connect is a high-performance dedicated bandwidth solution that connects on-premises sites directly to AWS. AWS Direct Connect is particularly useful in hybrid environments like on-premises video editing suites where I/O traffic is relatively high and there is a strong sensitivity to having consistent retrieval latency.

### AWS Snow family

The AWS Snow family is a collection of data transfer appliances that accelerates the transfer of large amounts of data in and out of AWS without using the Internet. Each Snowball appliance can transfer data faster than Internet speeds. The transfer is done by shipping the appliance directly to customers by using a regional carrier. The appliances are rugged shipping containers with tamper-proof electronic ink labels.

With **AWS Snowball**, you can move batches of data between your on-premises data storage locations and S3. AWS Snowball has an 80-TB model available in all regions, and a 50-TB model available only in US. Snowball devices are encrypted at rest and are physically secured while in transit. Data transfers are performed via the downloadable AWS Snowball client, or programmatically using the Amazon S3 REST API. It has 10G network interfaces (RJ45 and SFP+, fiber/copper).

**Snowball Edge** is a more advanced appliance in the Snow family that comes with 100 TB of local storage with built-in local compute that is equivalent to an *m4.4xlarge* instance. Snowball Edge appliances can run AWS Lambda functions or perform local processing on the data while being shuttled between locations. AWS Snowball Edge has multiple interfaces that support S3, HDFS, and NFS endpoints that simplify moving your data to AWS. It has 10GBase-T, 10/25Gb SFP28, and 40Gb QSFP+ network interfaces.

Although an AWS Snowball device costs less than AWS Snowball Edge, it cannot store 80 TB of data in one device. Take note that the storage capacity is different from the usable capacity for Snowball and Snowball Edge. Remember that an 80 TB Snowball appliance and 100 TB Snowball Edge appliance only have 72 TB and 83 TB of usable capacity respectively. Hence, it would be costly if you use two Snowball devices compared to using just one AWS Snowball Edge device.

Snowball Edge devices have three options for device configurations – storage optimized, compute optimized, and with GPU. When this guide refers to Snowball Edge devices, it's referring to all options of the device. Whenever specific information applies only to one or more optional configurations of devices, like how the Snowball Edge with GPU has an on-board GPU, it will be called out.

If your data migration needs are on an exabyte-scale, **AWS Snowmobile** is available as the transfer device for the large amounts of data. AWS Snowmobile is a 45-foot long rugged shipping container that has an internal capacity of 100

PB and 1 TB/s networking. AWS Snowmobile is designed to shuttle 100 PB of data in under a month and connects to NFS endpoints.

Snow family appliances have many uses cases, including: Cloud migration, Disaster Recovery, Data Center decommissioning, and content distribution.

## **AWS Storage Gateway**

AWS Storage Gateway is a hybrid storage service that enables your on-premises applications to seamlessly use AWS Cloud storage through conventional interfaces such as NFS, iSCSI or VTL. You can use this service for backup and archiving, DR, cloud bursting, storage tiering, and migration. Your applications connect to the service through a gateway appliance by using standard storage protocols, such as NFS and iSCSI. The Storage Gateway connects to S3 and Glacier on the back-end, providing storage for files, volumes, and virtual tapes in AWS. The service includes a highly optimized data transfer mechanism, with bandwidth management, automated network resilience, and efficient data transfer. It also provides a local cache for low-latency on-premises access to your most active data.

AWS Storage Gateway integrates seamlessly with Glacier through several pathways. Data can be written to an NFS mount point in S3, which can be tiered to Glacier or kept in S3 Standard or Standard-IA. For backup applications that are not already integrated with the Amazon S3 API, use the Virtual Tape Library Gateway mode, which mimics an Oracle/StorageTk L700 LTO tape library. The virtual tape shelf feature, will 'eject' a virtual tape from S3 into Glacier storage.

For Volume and Tape Gateway configurations, the customers' files are stored in a gateway-proprietary format, thus egress must always be back through the Gateway. For customers wishing to use AWS Storage Gateway as a migration tool, where files can subsequently be accessed via the S3 API in native format, the AWS File Gateway solution supports this capability.

## **Migrating with AWS Storage Gateway and Snowball**

AWS Storage Gateway can plug into an existing on-premises environment and combine with AWS Snowball to perform data migrations. Suppose you have a customer who is archiving data to an LTO tape and accessing the data using NAS devices. The customer can migrate their data from the NAS devices to AWS by connecting them to multiple AWS Snowball devices.

When the data transfer is complete, the AWS Snowball devices are shipped to AWS where the data is copied in S3 Standard. AWS Lambda functions can be run against the data to verify that the data that was transferred to each AWS Snowball device matches the data that was transferred from on-premises. After verification, an AWS Storage Gateway is deployed at the data center. It is mounted to the S3 bucket via AWS Direct Connect through NFS and data is written back to the on-premise file servers.

Migrating to AWS enables real-time access because all data is accessible via AWS Storage Gateway or S3 Standard. You can then transition the data to Standard-IA or Glacier using lifecycle policies.

There are 10 steps to replace tape backup with AWS Storage Gateway:

1. Go to AWS console to select type of Storage Gateway and hypervisor.
2. Deploy new Tape Gateway appliance in your environment.
3. Provision local disk for cache and upload buffer. The minimum required space for these 2 disks is 150 GiB.
4. Connect and activate your Tape Gateway appliance.
5. Create virtual tapes in seconds.
6. Connect Tape Gateway to backup application.
7. Import new tapes into backup application.



8. Begin backup jobs.
9. Archive tapes for storage on Amazon Glacier. In the backup applications you need to eject (also called export by some backup applications).
10. Retrieve tape on gateway for recovery.

## Multipart uploads

For large archives, you can use the multipart upload in Glacier-Native API to increase reliability in throughput. It allows uploading or copying objects that are 5 GB in size or greater.

With multipart uploads, a single object becomes a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Glacier assembles these parts and creates the object. In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation. Multipart uploading is a three-step process:

1. You initiate the upload: `InitiateMultipartUpload(partSize) -> uploadId`
2. Upload the object parts: `UploadPart(uploadId, data)`
3. Complete the multipart upload: `CompleteMultipartUpload(uploadId) -> archiveId`

## Data access on Amazon Glacier

### Amazon Glacier retrieval

#### Features

Amazon Glacier provides 3 retrieval methods that may be specified when retrieving Glacier archives.

**\*\*Expedited\*** retrievals are often used in situations when access to an archive is urgent. You can retrieve a single file from an archive from 1 to 5 minutes.

Expedited retrievals allow you to quickly access your data when occasional urgent requests for a subset of archives are required. For all but the largest archives (250 MB+), data accessed using Expedited retrievals are typically made available within 1?5 minutes. Provisioned Capacity ensures that retrieval capacity for Expedited retrievals is available when you need it.

**Standard** retrievals grants access to archives in about 3-5 hours, which is a good fit for DR situations or when there is not an immediate need for an archive.

**Bulk** retrievals grant access to archives from 5 to 12 hours. This tier is primarily for batch processing workloads, such as log files analysis, video transcoding, or large data lakes that require background processing. You can use bulk retrievals to comb through data slowly and pull back PBs of data at a low cost that can be processed by elastic compute or spot instances.

To make an Expedited, Standard, or Bulk retrieval, set the Tier parameter in the Initiate Job (POST jobs) REST API request to the option you want, or the equivalent in the AWS CLI or AWS SDKs. If you have purchased provisioned capacity, then all expedited retrievals are automatically served through your provisioned capacity.

#### Performance

Glacier reads are asynchronous: first, the application issues a restore job request and specifies one of 3 retrieval methods to be used, then after the specified time period has passed and file is restored, then the application can issue a GET to



retrieve the file.

For expedited retrievals, the 1-5-minute restore time is guidance for objects that are up to 250 MB in size. However, larger objects can take longer to restore. For example, a 1-GB object can be restored in 10 minutes or less, given optimal network conditions. Like S3, Glacier's HTTP transaction layer is designed to push back in the event of high loading. If your workload requires guaranteed responsiveness and HA for expedited retrievals, Amazon Glacier's Provisioned Capacity Units are an optional solution.

For Standard retrievals, restore times will be 3-5 hours, even for a PB of data or more. And the bulk retrievals, this option can be used to restore multiple PBs of data in less than 12 hours.

### Amazon Glacier restore options

When an archive is restored through the Glacier-Native API, you are given 24 hours to get the archive from staging storage before you must issue a new restore job. However, via S3-restore, an arbitrary TTL may be specified in days, where a temporary copy of the asset is retained in the S3 bucket in the Reduced Redundancy Storage class. In the case of S3 restores, customers pay for the incremental S3 storage.

If you must restore a PB-scale workload, contact your AWS account team to explore some helpful options. Scripts are available to spread bulk requests over a specified duration to ensure that you are using bulk retrievals in the most efficient manner. Bulk retrievals are often used in datasets that will be processed, so it is important to ensure that your data is being restored at the right speed for the application that will be processing the data.

### Provisioned capacity units

Provisioned capacity units (PCUs) are a way to reserve I/O capacity for Glacier expedited retrievals and guarantees that restore requests will not be throttled. Performance guidance for PCUs is that 1 PCU can deliver up to 3 expedited retrievals and up to 250 MB in 5 minutes, and provide up to 150 MB/s of retrieval throughput for a best-case sequential read. Note that there is no maximum file size restriction for Glacier Expedited retrievals. A large file, however, will take somewhat longer to restore. For example, most 25 GB archive retrievals should complete in under 10 minutes.

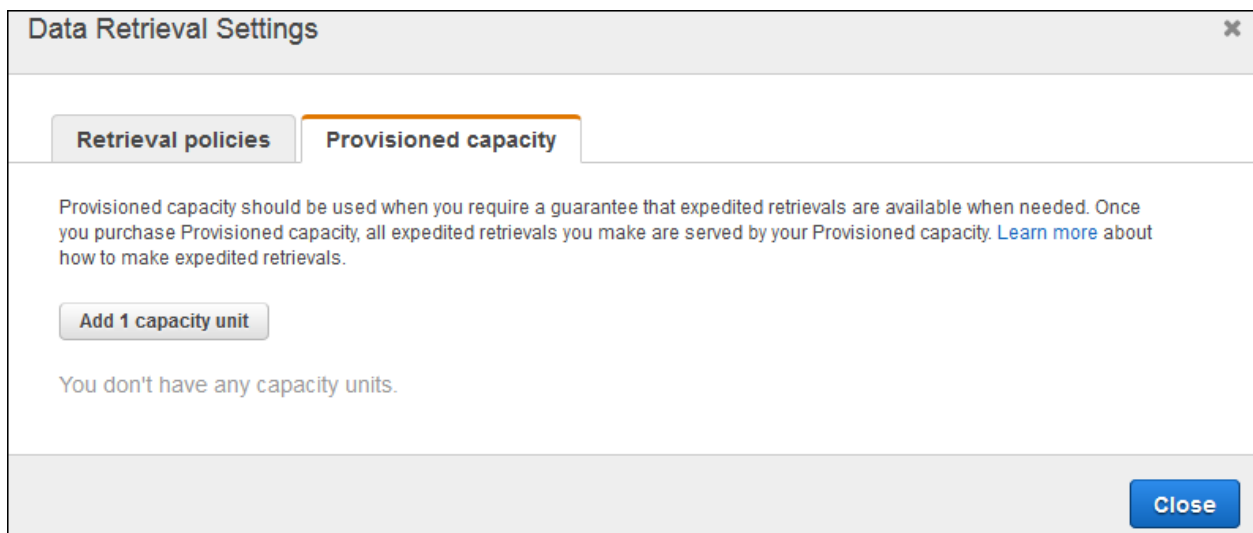


Fig. 23: Data retrieval settings

Without PCUs, expedited retrieval requests are accepted only if capacity is available at the time the request is made. It's possible that your workload will see performance variability or API-level pushback. PCUs are not guaranteed to improve Glacier performance, but PCUs will reduce variability, and may improve performance if the system is

under heavy load. It's important to note, however, that when you enable provisioned capacity units in Glacier, all I/O retrieval requests are routed through provisioned units. None of your Glacier I/O will go through the traditional I/O pool. Consequently you must ensure that you have provisioned enough units to handle your workload peaks.

Each provisioned capacity unit can generally be modeled as providing the equivalent to two LTO6 tape drives. If you are using Glacier as part of an active workflow and you expect deterministic retrieval performance, consider using provisioned capacity units to meet your performance needs.

## Glacier Select

Glacier Select allows you to directly filter delimited Glacier objects using simple SQL expressions without SQL having to retrieve the full object. Glacier Select allows you to accelerate bulk ad hoc analytics while reducing overall cost.

Glacier Select operates like a GET request and integrates with the Amazon Glacier SDK and AWS CLI. Glacier Select is also available as an added option in the S3 restore API command.

## Use cases

Glacier Select is a suitable solution for pattern matching, auditing, big data integration, and many other use cases.

To use Amazon Glacier Select, archive objects that are queried must be formatted as uncompressed delimited files. You must have a S3 bucket with write permissions that resides in the same region as your Glacier vault. Finally, you must have permissions to call the Get Job Output command from a command line.

## Query structure

An Amazon Glacier Select request is similar to a restore request. The key difference is that there are `SELECT` statements inside the expression inside the expression, and a derivative object produced. The restore request must specify an Glacier Select request and also choose the restore option of standard, expedited, or bulk. The query returns results based on the timeframe of the specified retrieval tier.

You can add a description and specify parameters. If your query has header information, you can also specify the input serialization of the objects. Use the expression `taf` to define your `SELECT` statement and the `OutputLocation` tag to specify the bucket location for the results. For example, the following JSON file runs the query `SELECT * FROM object`. Then, it sends the query results to the S3 location `awsexamplebucket/outputJob`:

```
{
  "Type": "SELECT",
  "Tier": "Expedited",
  "Description": "This is a description",
  "SelectParameters": {
    "InputSerialization": {
      "CSV": {
        "FileHeaderInfo": "USE"
      }
    },
    "ExpressionType": "SQL",
    "Expression": "SELECT * FROM object",
    "OutputSerialization": {
      "CSV": {}
    }
  },
  "OutputLocation": {
    "S3": {
```

(continues on next page)

(continued from previous page)

```

    "BucketName": "awsexamplebucket",
    "Prefix": "outputJob",
    "StorageClass": "STANDARD"
  }
}

```

## SQL expressions supported by Amazon Glacier

With Glacier Select, you can specify all items in an expression with `SELECT *`, or specify columns using a positional header and the `WHERE` filter. You can also specify name headers as well. Use these SQL filters to retrieve only a portion of an object.

Row-level SQL expressions are support such as `+`, `</>`, `LIKE`, `AND/OR`, `ISNULL`, `STRING`, `CASE`, `COUNT`, `MAX`. For full SQL capabilities including joins and windowing, tools such as Amazon Athena should be subsequently employed.

Clauses	Data types	Operators	Functions
Select	String	Conditional	String
From	Integer, Float,Decimal	Math	Cast
Where	Timestamp	Logical	
	Boolean	String (Like,   )	

Fig. 24: SQL expressions supported by Amazon Glacier

[SQL Reference for Amazon S3 Select and S3 Glacier Select](#)

[Amazon S3 Glacier API Permissions: Actions, Resources, and Conditions Reference](#)

## How Amazon Glacier Select works

Glacier Select works when initiate a request job with the SQL parameters you specify. When the job request is received, Glacier returns an acknowledgment ID 200 to indicate that the request was received successfully. Glacier then processes the request and writes the output to a S3 bucket with the specified prefix in the job request. SNS sends a notification to alert you when the job is complete.

## Pricing model

Glacier Select pricing depends on the retrieval option you specify in your retrieval request job. You can specify standard, expedited, or bulk in your expression. The pricing for Glacier Select falls into 3 categories for each retrieval

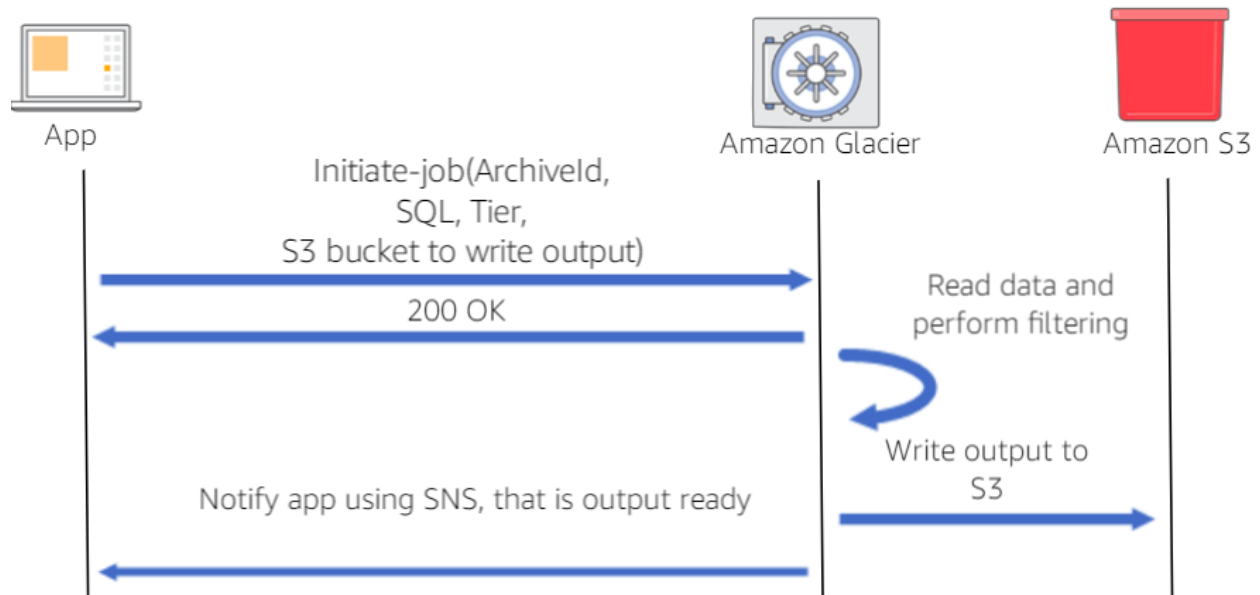


Fig. 25: How Amazon Glacier Select works

option: Data Scanned, Data Returned, and Request.

Data Scanned is the size of the object that the SQL statement is run against. Data Returned is the data that returns as the result of your SQL query. Request is based on the initiation of the retrieval request.

### 1.3.3 Amazon EBS

#### Overview

EBS is block storage as a service. With an API call, you create an EBS volume, which is a configurable amount of raw block storage with configurable performance characteristics. With another API call, you can attach a volume to an EC2 instance when you need access to that data. Access to the volume is over the network. Once attached, you can create a file system on top of a volume, run a database on it, or use it in any other way you would use block storage.

An EBS volume is not a single physical disk. It is a logical volume. EBS is a distributed system and each EBS volume is made up of multiple, physical devices. By distributing a volume across many devices EBS provides added performance and durability that you can't get from a single disk device.

EBS volumes are created in a specific AZ, and can then be attached to any instances in that same AZ. To make a volume available outside of the AZ, you can create a snapshot and restore that snapshot to a new volume anywhere in that region.

Data on an EBS volume persists independently from the life of an EC2 instance. As a result, you can detach your volume from one instance and attach it to another instance in the same AZ as the volume. If an EC2 instance fails, EBS storage persists independently from the EC2 instance. Because the system has failed, and has not been terminated by calling an API, the volume is still available. In this case, you can re-attach the EBS volume to a new EC2 instance.

An EBS volume can be attached to only one instance at a time, but many volumes can attach to a single instance. For example, you might separate boot volume from application-specific data volumes.

EBS is designed to give a high level of volume availability and durability. Each volume is designed for five 9s of service availability. That's an average down time of about 5 minutes per year. For durability, the Annualized Failure Rate (AFR) is between 0.1% and 0.2%, nearly 20 times more reliable than a typical HDD at 4% AFR. To recover from

a volume failure, you can use EBS to create snapshots of volumes, which are point-in-time copies of your data that is stored in S3 buckets.

## Block storage offerings

AWS addresses the block storage market in 3 categories to align with different customer workloads:

- EC2 instance store provides SSD and HDD-based local instance store that targets high IOPS workloads that require low microsecond latency.
- EBS SSD storage offers General Purpose (gp2) and Provisioned IOPS (io1) volume types.
- EBS HDD storage offers Throughput Optimized (st1) and Cold HDD (sc1) volume types.

An instance store provides temporary block-level storage for your instance. This storage is located on disks that are physically attached to the host computer. Instance store is ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, an other temporary content, or for data that is replicated across a fleet of instances, such as load-balanced pool of web servers.

An instance store consists of one or more instance store volumes exposed as block devices. The size of an instance store and the number of devices available varies by instance type. Although an instance store is dedicated to a particular instance, the disk subsystem is shared among instances on a host computer.

The EC2 instance and EBS have some similarities:

- Both are presented as block storage to EC2.
- Both are available as either SSD or HDD, depending on the EC2 instance type.

Their differences are:

- EC2 instance store is ephemeral. Data stored on the instance store does not persist beyond the life of the EC2 instance, consider it as being temporary storage.
- By default, data on EC2 instance store volumes is not replicated as a part of the EC2 service. However, you can set up replication yourself by setting up a RAID volume on the instance or replicating that data to another EC2 instance.
- Snapshots of an EC2 instance store volume are not supported, so you would have to implement your own mechanism to back up data stored on an instance store volume.

## Use Cases

**Relational Databases** such as Oracle, Microsoft SQL Server, MySQL, and PostgreSQL are widely deployed on EBS.

EBS meets the diverse needs of reliable block storage to run **mission-critical applications** such as Oracle, SAP, Microsoft Exchange, and Microsoft SharePoint.

EBS enables your organization to be more agile and responsive to customer needs. Provision, duplicate, scale, or archive your **development, test, and production** environment with a few clicks.

EBS volumes provide the consistent and low-latency performance your application needs when running **NoSQL databases**.

**Business Continuity.** Minimize data loss and recovery time by regularly backing up your data and log files across different geographic regions. Copy AMIs and EBS snapshots to deploy applications in new AWS Regions.

AWS re:Invent 2018: [REPEAT 1] Deep Dive on Amazon Elastic Block Storage (Amazon EBS) (STG310-R1)

Introduction to Amazon Elastic Block Store (EBS)

Amazon Elastic Block Store (Amazon EBS)

## Types of Amazon EBS volumes

### Comparison of EBS volume types

EBS provides the following volume types, which differ in performance characteristics and price, so that you can tailor your storage performance and cost to the needs of your applications. The volume types fall into 2 categories: Solid state drives and Hard disk drives.

Solid state drive, or SSD-backed volumes are optimized for transactional workloads that involve frequent read/write operations with small size and random I/O. For SSD-backed volumes, the dominant performance attribute is I/O operations per second, or IOPS. These volumes provide low latency.

Hard disk drive, or HDD-backed volumes are optimized for large streaming workloads where throughput (measured in MiB/s) is a better performance measure than IOPS. These volumes work best when the workload is sequential. For example, back up operations and writing SQL transaction log files. The more sequential a workload is, the less time spent in read operations. Therefore, the faster disk response leads to higher throughput rates.

There are 2 types of SSD-backed volumes: General Purpose (gp2) and Provisioned IOPS (io1). There are 2 types of HDD-backed volumes: Throughput Optimized (st1) and Cold HDD (sc1). This table describes and compares the 4 volume types.

Volume Type	Solid-State Drives (SSD)		Hard disk Drives (HDD)	
	General Purpose gp2	Provisioned IOPS io1	Throughput Optimized st1	Cold sc1
Description	General purpose SSD volume that balances price and performance	Highest-performance SSD volume designed for mission-critical applications	Low cost HDD volume designed for throughput-intensive workloads	Lowest cost HDD volume designed for infrequent workloads
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max Throughput/Volume	160 MB/s	500 MB/s	500 MB/s	250 MB/s
Max IOPS/Volume	10,000	32,000	500	250
Max Throughput/Instance	1,750 MB/s	1,750 MB/s	1,750 MB/s	1,750 MB/s
Max IOPS/Instance	80,000	80,000	80,000	80,000
Performance Attribute	IOPS	IOPS	MiB/s	MiB/s

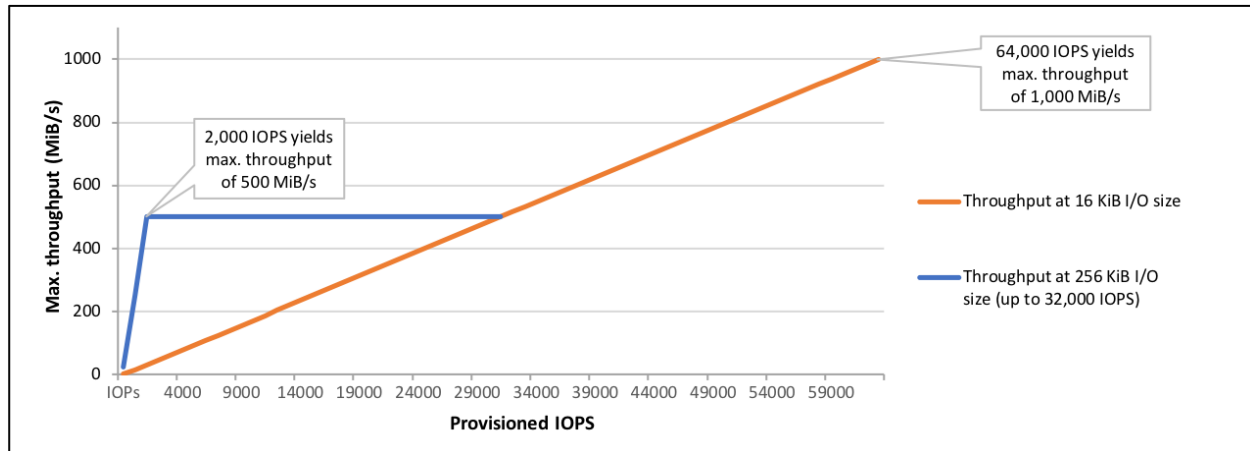
The General Purpose SSD volume can handle most workloads, such as boot volumes, virtual machines, interactive applications, and development environments. The Provisioned IOPS SSD volume can handle critical business applications that require nearly continuous IOPS performance. This type of volume is used for large database workloads.

An `io1` volume can range in size from 4 GiB to 16 TiB. You can provision from 100 IOPS up to 64,000 IOPS per volume on Nitro system instance families and up to 32,000 on other instance families. The maximum ratio of provisioned IOPS to requested volume size (in GiB) is 50:1.



For example, a 100 GiB volume can be provisioned with up to 5,000 IOPS. On a supported instance type, any volume 1,280 GiB in size or greater allows provisioning up to the 64,000 IOPS maximum ( $50 \times 1,280 \text{ GiB} = 64,000$ ).

An `io1` volume provisioned with up to 32,000 IOPS supports a maximum I/O size of 256 KiB and yields as much as 500 MiB/s of throughput. With the I/O size at the maximum, peak throughput is reached at 2,000 IOPS. A volume provisioned with more than 32,000 IOPS (up to the cap of 64,000 IOPS) supports a maximum I/O size of 16 KiB and yields as much as 1,000 MiB/s of throughput.

Therefore, for instance, a 10 GiB volume can be provisioned with up to 500 IOPS. Any volume 640 GiB in size or greater allows provisioning up to a maximum of 32,000 IOPS ( $50 \times 640 \text{ GiB} = 32,000$ ).



The Throughput Optimized HDD volume can handle streaming workloads that require consistent and fast throughput at a low price. For example, big data, data warehouses, and log processing. The Colod HDD volume can handle throughput-oriented storage for large amounts of data that are infrequently accessed. Use this type of volume for scenarios in which achieving the lowest storage cost is important.

FEATURES	SSD Solid State Drive	HDD Hard Disk Drive
Best for workloads with:	<i>small, random</i> I/O operations	<i>large, sequential</i> I/O operations
Can be used as a bootable volume?	Yes	No
Suitable Use Cases	<ul style="list-style-type: none"> <li>- Best for <b>transactional workloads</b></li> <li>- Critical business applications that require sustained IOPS performance</li> <li>- Large database workloads such as MongoDB, Oracle, Microsoft SQL Server and many others...</li> </ul>	<ul style="list-style-type: none"> <li>- Best for <b>large streaming workloads</b> requiring consistent, fast throughput at a low price</li> <li>- Big data, Data warehouses, Log processing</li> <li>- Throughput-oriented storage for large volumes of data that is <b>infrequently</b> accessed</li> </ul>
Cost	moderate / high 	low 
Dominant Performance Attribute	IOPS	Throughput (MiB/s)

Amazon EBS features

Amazon EBS Volume Types



## Choosing an EBS volume type

How do you determine which volume type to use? Begin by asking: Which storage best aligns with the performance and cost requirements of my applications? Each has benefits and limitations that work with or against certain workloads.

### What is more important IOPS or Throughput?

- If IOPS is more important, **how many IOPS do you need?**
  - If the answer is more than 80000, choose an EC2 instance store.
  - If you need 80000 or fewer, **what are your latency requirements?**
    - \* If your latency is in microseconds, the choice is an EC2 instance store. EC2 instance stores provide the lowest latency that you can achieve.
    - \* If your latency is in the single-digit millisecond category, decide **what's more important? Cost or performance?**
      - If cost is more important, choose General purpose (gp2).
      - If performance is the main driver for your workload, choose SSD volume type, Provisioned IOPS (io1). Compared to gp2, io1 provides more consistent latency (less jitter) and an order of magnitude more of IOPS consistency.
- If Throughput is the defining characteristic of your workload, **What type of I/O do you have? Small, random I/O or large, sequential I/O?**
  - If you have small random I/O, re-evaluate the SSD categories.
  - If you have large sequential I/O, **what is your aggregate throughput?**
    - \* If you need more than 1750-MB/s throughput, consider D2 or H1 optimized instance types. D2 is dense storage instance type, which enables you to take advantage of the low cost, high disk throughput, and high sequential I/O access rates. D2 has up to 48 TB of local spinning hard disk, and it can do upwards of 3 GB/s of sequential throughput. H1 instances are storage-optimized instances, which are designed for applications that require low-cost, high-disk throughput, and high sequential disk I/O access to large datasets.
    - \* If your throughput requirement is less than 1750 MB/s, **Which is more important? Cost or performance?**
      - If it's performance, consider the HDD volume type throughput optimized HDD (st1).
      - If the most important factor is cost, choose the Cold HDD (sc1) volume.

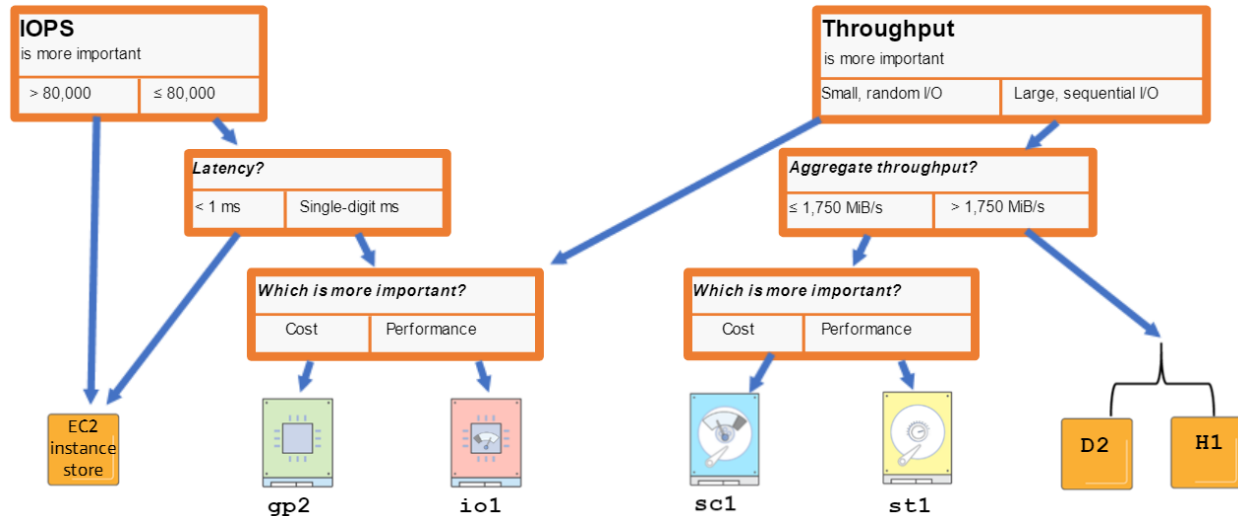
When you are not sure what your workload is, choose General Purpose SSD (gp2) as it satisfies almost all workloads. However, this volume type works well for boot volumes, low-latency applications, and the bursty databases where the data transmission is interrupted at intervals.

## EBS-Optimized instances

A non-EBS optimized instance has a shared network pipe. As a result, Amazon EBS traffic is on the same pipe as the network traffic to other EC2 instances, AWS services, such as Amazon S3, and the Internet. A shared network pipe is used because EBS storage is network-attached storage and not attached directly to the EBS instance. The sharing of network traffic can cause increased I/O latency to your EBS volumes.

EBS-optimized instances have dedicated network bandwidth for Amazon EBS I/O that is not shared. This optimization provides the best performance for your EBS volumes by minimizing contention between EBS I/O and other traffic.





from your instance. The size of an EBS-optimized instance, for example 2xlarge or 16xlarge, defines the amount of dedicated network bandwidth that is allocated to your instance for the attached EBS volumes.

It is important to choose the right size EC2 that can support the bandwidth to your EBS volume. Choosing the right size helps you achieve the required IOPS and throughput.

EBS-optimized instances deliver dedicated bandwidth to EBS, with options between 425 Mbps and 14000 Mbps, depending on the instance type that you use. When attached to an EBS-optimized instance, General Purpose SSD (gp2) volumes are designed to deliver within 10% of their baseline and burst performance 99.9% of the time in a given year. Both Throughput Optimized HDD (st1) and Cold HDD (sc1) are designed for performance consistency of 90% of burst throughput 99% of the time.

### EBS-Optimized burst

C5 instance types provide EBS-optimized burst. The C5 instance types can support maximum performance for 30 minutes at least once every 24 hours. For example, c5.large instances can deliver 275 MB/s for 30 minutes at least once every 24 hours. If you have a workload that requires sustained maximum performance for longer than 30 minutes, choose an instance type based on the baseline performance listed for each C5 instance.

[Amazon EBS-Optimized Instances](#)

### Managing Amazon EBS snapshots

An EBS snapshot is a point-in-time backup copy of an EBS volume that is stored in S3. S3 is a regional service that is not tied to a specific AZ. It is designed for 11 9s of durability, which is several orders of magnitude greater than the volume itself.

Snapshots are incremental backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved. The incremental backups minimize the time required to create the snapshot and save on storage costs by not duplicating data. When you delete a snapshot, only the data that's unique to that snapshot is removed. Each snapshot contains all the information needed to restore your data (from the moment when the snapshot was taken) to a new EBS volume.

When you create an EBS volume from a snapshot, the new volume begins as an exact replica of the original volume that was used to create the snapshot. The replicated volume loads data lazily, that is, loads data when it's needed, in the background so that you can begin using it immediately.

EBS snapshots occur asynchronously. This means that the point-in-time snapshot is created immediately, but the status of the snapshot is `pending` until the snapshot is complete (when all of the modified blocks have been transferred to Amazon S3), which can take several hours for large initial snapshots or subsequent snapshots where many blocks have changed. While it is completing, an in-progress snapshot is not affected by ongoing reads and writes to the volume hence, you can still use the volume.

Snapshots with AWS Marketplace product codes can't be made public.

### How does a snapshot work?

When you execute the first snapshot, all blocks on the volume that have been modified (let's call them A,B,C) are marked as a snapshot copy to S3. Empty blocks are not part of the snapshot. The volume is useable again as soon as the `CreateSnapshot` API returns successfully, usually within a few seconds. You do not have to wait for the actual data transfer to complete before using the volume.

Snapshots are incremental backups so the second snapshot contains only new blocks or blocks that were modified since the first snapshot. For snapshot 2, block C has been modified to C1. A second snapshot is taken. Because the snapshot is incremental, C1 is the only block saved to the second snapshot. None of the data from the snapshot 1 is included in the new snapshot. Snapshot 2 contains only references to snapshot 1 for any unchanged data.

Before snapshot 3 is taken, two new blocks, D and E, are added, and the file system has deleted the file that contained block B. From a block perspective, block B was modified. Snapshot 3 contains any new or changed data since snapshot 2, but the new snapshot only references the blocks in snapshots 1 and 2 that are unchanged.

If you delete the first 2 snapshots, the deletion process is designed so that you retain only the most recent snapshot to restore the volume. The state of the original B block is now gone and A is still restorable from the latest snapshot even though it was not part of the original change set.

Active snapshots contain all the necessary information to restore the volume to the instant at which that snapshot was taken. When you create a new volume from a snapshot, data is lazily loaded in the background so that the volume is immediately available. You don't have to wait for all the data to be transferred to the new volume. If you access a piece of data that hasn't been loaded to the volume yet, that piece immediately moves to the front of the queue of data that flows from the snapshot data in S3 to your new volume.

### Creating snapshots

1. You can use snapshots as an easy way to back up and share data on EBS volumes among multiple AZs within the same region.
2. Another possible scenario is that you want to use snapshots in a different region or share with a different account. You can use `copy-snapshot` API operation, or the snapshot copy action from the AWS management console to copy snapshots to different regions. You can share snapshots by modifying the permissions of the snapshot and granting different AWS accounts permission to use that snapshot. From that snapshot, others can create EBS volumes and get access to a copy of the snapshot data. The copy of snapshots to a different regions can be part of your disaster recovery strategy.

### Snapshots and RAID

When the instance is using a RAID configuration, the snapshot process is different. You should stop all I/O activity of the volumes before creating a snapshot:

1. Stop all applications from writing to the RAID array.
2. Flush all caches to the disk.

3. Confirm that the associated EC2 instance is no longer writing to the RAID array by taking actions such as freezing the file system, unmounting the RAID array, or even shutting down the EC2 instance.
4. After taking steps to halt all disk-related activity to the RAID array, take a snapshot of each EBS volume in the array.

When you take a snapshot of an attached Amazon EBS volume that is in use, the snapshot excludes data cached by applications or the operating system. For a single EBS volume, this is often not a problem. However, when cached data is excluded from snapshots of multiple EBS volumes in a RAID array, restoring the volumes from the snapshots can degrade the integrity of the array.

When creating snapshots of EBS volumes that are configured in a RAID array, it is critical that there is no data I/O to or from the volumes when the snapshots are created. RAID arrays introduce data interdependencies and a level of complexity not present in a single EBS volume configuration.

### Amazon Data Lifecycle Manager (DLM)

When you must back up a large number of volumes or manage a large number of snapshots, you can use Amazon Data Lifecycle Manager (DLM). DLM automates the backup of EBS volumes and the retention of those backups for as long as needed. This feature uses policies to define backup schedules that specify when you want a snapshot to be taken and how often DLM creates the snapshots when you define the policies.

You also specify how many snapshots you want to retain, and Amazon DLM automatically manages the expiration of snapshots. This is an important feature if you need to retain snapshots for a certain period of time, or need to retain a certain number of snapshots for compliance and auditing reasons. Retention rules also help you to keep snapshot costs under control by removing snapshots that are no longer needed.

When you define a DLM policy, you specify volume tags to identify which EBS volumes should be backed up. This allows a single policy to be applied to multiple volumes. You can also use IAM to fully control access to DLM policies. There's no additional cost to using DLM. You pay only for the cost of the snapshots.

An example of DLM policy that satisfies the customer requirement: *All EC2 instance root volumes must be backed up once per day, and that backups should be retained for 7 days.* It is necessary to define a data lifecycle policy by specifying the EBS volume type to use. In this example, the tag name is *voltype* with a value of *root*. As we want to take one snapshot per day, so you specify to take a snapshot every 24 hours at 0700 UTC. Then, you retain the latest 7 snapshots.

DLM enables you to take snapshots every 12 hours or every 24 hours. If you need to take snapshots more often than this, you can use multiple tags on the same volume. For example, suppose that you want to create a snapshot every 8 hours and retain 7 days of snapshots. You add 3 tags to your volume for backup. The tag key-value pairs can be anything that makes sense for your organization. you then define 3 separate policies, one for each tag, running once per day for each policy. You stagger the start time so that it is offset by 8 hours from the other policies.

Here are some considerations to take into account when using DLM for EBS snapshots:

- When specifying multiple tags in a policy, the policy applies to any of the tags specified. A snapshot is generated for each volume with a matching tag.
- You can use a volume tag in only one DLM policy. You cannot create a policy with a volume tag that has been already assigned to an existing policy.
- Snapshots are taken within one hour of the specified start time. The snapshot may not generally be taken exactly at the time specified in the policy.
- To improve the manageability of snapshots, DLM automatically applies AWS tags to each snapshot that is generates. Alternatively, you can specify custom tags to be applied to the snapshot.

## Tagging EBS snapshots

AWS supports tagging both EBS volumes and snapshots during creation. Tagging snapshots during creation is an atomic operation, that is, the snapshot must be created and the tags must be applied for the successful creation of snapshots. This functionality facilitates the proper tracking of snapshots from the moment that they are created. It also means that any IAM policies that apply to snapshots tags are enforced immediately.

You can add up to 50 tags to your snapshot when it's created. AWS provides resource-level permissions to control access to EBS snapshots through IAM policies. You can use tags to enforce tighter security policies. The `CreateSnapshot`, `DeleteSnapshot`, and `ModifySnapshotAttribute` are API operations that support IAM resource-level permissions.

The IAM policies give you precise control over access to resources. For example, you can require that a certain set of tags is applied when the snapshot is created, or you can restrict which IAM users can take snapshots on specific volumes. You can also control who can also control who can delete snapshots.

[Amazon EBS Snapshots](#)

[Automating the Amazon EBS Snapshot Lifecycle](#)

## Managing Amazon EBS volumes

### EBS volume management actions

The actions that you can perform on a volume are the following:

- **Create:** New EBS volumes are created from large amounts of available space. They can have a size from 1 GiB to TiB.
- **Attach:** An EBS volume can be attached to an instance. After attachment, it becomes visible to the operating system as a regular block device. Each EBS volume can be a single EC2 instance at a time.
- **Create snapshot.** Snapshots can be created from a volume at any time while the volume is in the in-use state.
- **Detach.** When the operating system no longer uses the volume, it can be detached from the instance. Data remains stored on the EBS volume, and the volume remains available for attachment to any other instance within the same AZ.
- **Delete.** When the volume and its contents are no longer needed, the EBS can be deleted.

## Creating a volume

You can create an EBS volume and attach it to any EC2 instance within the same AZ. You can create an encrypted EBS volume, but you can attach an encrypted volume only to supported EC2 instance types. When a volume is created, its state is *available*. When the volume is attached to an instance, its state changes to *in-use*.

To create an encrypted volume with the AWS Management console, you must select the *Encrypted* box, and choose the master key you want to use when encrypting the volume. You can choose the default master key for your account, or use AWS KMS to choose any customer master (CMK) that you have previously created.

If you create the volume from a snapshot and do not specify a volume size, the default size is the snapshot size.

The states of a volume are Creating, Deleting, Available, and In-use. A volume in the *available* state can be attached to an EC2 instance.

[Creating an Amazon EBS Volume](#)

## Attaching an EBS volume

To attach a volume to an EC2 instance, the volume must be in the *available* state. After the volume is attached, you must connect to the EC2 instance and make the volume available. You can view the attachment information on the volume's Description tab in the console. The information provides an instance ID and an instance device, which are required to make the volume available on its EC2 instance.

To attach an EBS volume to a running or stopped instance, use the `attach-volume` command. After you attach an EBS volume to your instance, it is exposed as a block device. You can format the volume with any file system and then mount it. After you make the EBS volume available for use, you can access it in the same ways that you access any other volume. Any data written to this file system is written to the EBS volume and is transparent to applications that use the device.

After connecting to an EC2 instance, the steps to make the volume available for use in Linux are the following:

1. List volumes.

```
$ lsblk
NAME      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
xvda      202:0    0    8G  0 disk
-xvda1    202:1    0    8G  0 part /
xvdf      202:80   0   10G  0 disk
```

2. Check that there is no file system on the device.

```
$ sudo file -s /dev/svdf
/dev/svdf: data
```

3. Create and ext4 (a journaling file system that the Linux kernel commonly uses) file system on the new volume.

```
$ sudo mkfs -t ext4 /dev/xvdf
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 261888 4k blocks and 65536 inodes
Filesystem UUID: 15d8146f-bcb3-414c-864f-5100bb4b0bf8
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

4. Create a directory for mounting the new volume.

```
$ sudo mkdir /mnt/data-store
```

5. Mount the new volume.

```
$ sudo mount /dev/xvdf /mnt/data-store
```

6. Check that the volume is mounted.

```
$ df -hT
Filesystem      Type      Size  Used Avail Use% Mounted on
/dev/xvdf       ext4      197G   61M  187G   1% /data-store
```

[Making an Amazon EBS Volume Available for Use on Linux](#)

[Making an Amazon EBS Volume Available for Use on Windows](#)

## Detaching an EBS volume

You can detach an EBS volume from an EC2 instance explicitly or by terminating the instance. If the instance is running, you must first unmount the volume from the instance. If an EBS volume is the root device of an instance, you must stop the EC2 instance before you can detach the volume.

The command to unmount the `/dev/xvdf` device on a Linux instance is

```
sudo umount -d /dev/xvdf
```

## Deleting an EBS volume

After volume deletion, the data is physically gone, and the volume cannot be attached to any instance. However, before deletion, it is good practice to create a snapshot of the volume, which you can use to re-create the volume later if needed. To delete a volume, it must be in the available state (that is, not attached to an instance).

If you want to retain the data on Amazon EBS, then you should disable `DeleteOnTermination` for the Amazon EBS volume.

## Modifying Amazon EBS volumes

You can modify your volumes as the needs of your applications change. You can perform certain dynamic operations on existing volumes with no downtime or negative effects on performance:

- Increase capacity.
- Change the volume type.
- Increase or decrease provisioned IOPS.

You can easily right-size your deployment and adapt to performance changes. Use Amazon CloudWatch with AWS Lambda to automate volume changes to meet the changing needs of your applications, without establishing planning cycles to implement the changes.

In general, you must follow 3 major steps when modifying an EBS volume:

1. Use AWS Management console or the AWS CLI to issue `modify` volume command.
2. Use AWS Management console or the AWS CLI to monitor the progress of the modification.
3. If the size of the volume was modified, extend the volume's file system to take advantage of the increased storage capacity, by using OS commands for your EC2 instances.

[Amazon EBS Update – New Elastic Volumes Change Everything](#)

## Modify a volume

You can modify volumes that are in the *in-use* state (attached to an EC2 instance) or the available state (not attached to an EC2 instance). Minimum and maximum values for size depend on the volume type. For example, for a General Purpose SSD volume, the minimum size is 1 GiB, and the maximum size is 16384 GiB. Though the size of a volume can be changed, it can only be increased, not decreased.

You can change the IOPS value only for the Provisioned IOPS SSD volume type. This setting is the requested number of I/O operations per second that the volume can support. For provisioned IOPS volumes, you can provision up to 50 IOPS per GiB depending on what level of performance your applications require.

## Monitor the progress

During modifications, you can view the status of the volume in the AWS Management Console. An EBS volume that is being modified moves through a sequence of progress states: `Modifying`, `Optimizing`, and `Complete`. You can resize your file system as soon as the volume enters the `Optimizing` state. Size changes usually take a few seconds to complete and take effect after a volume is in the `Optimizing` state. The `Optimizing` state requires the most time to complete.

## Extend the file system

To take advantage of the larger EBS volume, you must extend the file system on your local operating system. Use a file system-specific command to resize the file system to the larger size of the volume. These commands work even if the volume to extend is the root volume. For `ext2`, `ext3`, and `ext4` file systems, the command is `resize2fs`. XFS file systems, the command is `xfs_growfs`.

Extending a Linux File System After Resizing a Volume

## Securing Amazon EBS

Access to EBS volumes is integrated with AWS IAM. IAM enables access control to your EBS volumes and snapshots. For example, you can use an IAM policy to allow specified EC2 instances to attach and detach volumes. You can also encrypt data that is written to EBS volumes.

## EBS encryption

Amazon EBS encryption offers a simple encryption solution for EBS volumes without the need for you to build, maintain, and secure your own key management infrastructure. When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:

- Data at rest inside the volume.
- All data moving between the volume and the instance.
- All snapshots that were created from the volume.

Encryption is supported by all EBS volume types (General Purpose SSD, Provisioned IOPS SSD, Throughput Optimized HDD, Cold HDD and Magnetic[standard]). You can expect the same IOPS performance on encrypted volumes as you would with unencrypted volumes, with only a minimal effect on latency.

You can access encrypted volumes the same way that you access unencrypted volumes. Encryption and decryption are handled transparently, and they require no additional action from the user, the EC2 instances, or the applications.

EBS encryption uses a customer master key (CMK) from the AWS KMS when creating encrypted volumes. The CMK is also used for any snapshots that are created from the volumes. The first time you create an encrypted volume in a region, a default CMK is created automatically. This key is used for EBS encryption unless you select a CMK that you created separately with AWS KMS.

## Encrypting a new volume

If you are not using a snapshot to create a volume, you can choose whether to encrypt the volume. Use the default key or select your own custom key. Creating your own CMK gives you more flexibility, including the ability to create, rotate, and disable keys. By using your own CMK, it is easier to define access controls, and to audit the encryption keys that are used to protect your data.



It is a best practice to create your own master key. By creating your own master key, you gain flexibility over the key. For example, you can have one team of users control encryption and another team of users control decryption. You can have separate master keys per application, or per data classification level. You can:

- Define the key rotation policy.
- Enable AWS CloudTrail auditing.
- Control who can use the key.
- Control who can administer the key.

EBS encryption is integrated into the AWS KMS, and AWS KMS is integrated with the IAM console. You have to access IAM console in order to create a key. Configuration options include:

- Adding tags.
- Defining key administrative permissions.
- Defining key usage permissions.

As part of the process for launching a new EC2 instance, you can use custom keys to encrypt EBS volumes at launch time by using the EC2 console or by executing the `run-instances` command.

## How EBS encryption works

EBS applies a strategy called envelope encryption that uses a two-tier hierarchy of keys. The customer master key (CMK) is at the top of the hierarchy and encrypts and decrypts data keys that are used outside of AWS KMS to encrypt data. CMKs never leave the AWS KMS.

Each newly created EBS volume is encrypted with a unique 256-bit data key. The CMK is called to wrap the data key with a second layer, or second tier, of encryption. EBS stores the encrypted data key with the volume metadata. Any snapshots created from this volume will also be encrypted with this data key.

When a volume is attached to an EC2 instance, the CMK unlocks the volume data key and stores it in the memory of the server that hosts your instance. When the data key is stored in memory, it can decrypt and encrypt data to and from the volume. If the volume is detached, the data key is purged from memory. The data key is never saved to disk. The data key is stored in memory because:

1. If one resource is compromised, exposure is limited. Only one data key is exposed (and not all volumes).
2. The encryption key is in the memory of the system doing the encryption, thus limiting any performance degradation. You are not required to bulk-load large amounts of EBS data over the network to encrypt it.
3. Small number of master keys are used to protect potentially a large number of data keys.

There is no direct way to encrypt an existing unencrypted volume, or to remove encryption from an encrypted volume. Similarly, you cannot remove encryption from an encrypted snapshot.

However, you can migrate data between encrypted and unencrypted volumes. You can also apply a new encryption status while copying a snapshot. While copying an unencrypted snapshot of an unencrypted volume, you can encrypt the copy. Volumes restored from this encrypted copy are also encrypted.

While copying an encrypted snapshot of an encrypted volume, you can re-encrypt the copy by using a different CMK. Volumes that are restored from the encrypted copy are accessible only by using the newly applied CMK.

## Encrypted snapshots

Snapshots that are created from encrypted volumes are automatically encrypted. Volumes that are created from encrypted snapshots are also automatically encrypted. EBS encryption is available only on certain EC2 instance types. You can attach both encrypted and unencrypted volumes to a supported instance type.



The first snapshot copy to another region is generally a full copy. Each subsequent snapshot copy is incremental (which makes the copy process faster), meaning that only the blocks in the snapshot that have changed since your last snapshot copy to the same destination are transferred.

Support for incremental snapshots is specific to a cross-region pair, where a previous complete snapshot copy of the source volume is already available in the destination region. For example, if you copy an unencrypted snapshot from the US East (Ohio) region to the US West (Oregon) region, the first snapshot copy of the volume is a full copy. Subsequent snapshot copies of the same volume transferred between the same regions are incremental.

Similarly, the first encrypted snapshot copied to another region is always a full copy, and each subsequent snapshot copy is incremental. Support for incremental snapshots is specific to a cross-region pair, whereby a previous complete snapshot copy of the source volume is already available in the destination region.

When you copy a snapshot, if the original snapshot was not encrypted, you can choose to encrypt the copy. However, changing the encryption status usually results in a full (not incremental) copy, which may incur greater data transfer and storage charges.

Suppose that an encrypted snapshot is shared with you. It is recommended that you create a copy of the shared snapshot using a different CMK that you control. This protects your access to the volume if the original CMK is compromised, or if the owner revokes the CMK for any reason. You can specify a CMK that is different from the original one, and the resulting copied snapshot uses the new CMK. However, using a custom EBS CMK during a copy operation always results in a full (not incremental) copy, which may incur increased data transfer and storage charges.

By modifying the permissions of the snapshot, you can share your unencrypted snapshots with others in the AWS community by selecting *Private* and providing a user's account number. When you share an unencrypted snapshot, you give another account permission to both copy the snapshot and create a volume from it.

You can share an encrypted snapshot with specific AWS accounts, but you cannot make it public. Be aware that sharing the snapshot provides other accounts access to all the data. For others to use the snapshot, you must also share the custom CMK key used to encrypt it. Users with access can copy your snapshot and create their own EBS volumes based on your snapshot while your original snapshot remains unaffected. Cross-account permissions can be applied to a CMK either when it is created or later.

To create a copy of an encrypted EBS snapshot in another account, complete these 4 steps:

1. Share the custom key associated with the snapshots with the target account. You can do it from the IAM console:
  - 1.1. Under *External Account*, click *Add and External account*.
  - 1.2. Specify which external accounts can use this key to encrypt and decrypt data. Administrators of the specified accounts are responsible for managing the permissions that allow their IAM users and roles to use this key.
2. Share the encrypted EBS snapshot with the target account.
  - 2.1. Select the snapshot.
  - 2.2. From the actions menu, click *Modify Permissions*.
3. From the target account, locate the shared snapshot and create a copy of it.
  - 3.1. Select the shared snapshot.
  - 3.2. Expand the Actions menu.
  - 3.3. Click *Copy*. To select the master key:
    - 3.3.1. In the Destination Region box, select the target region.
    - 3.3.2. In the Description box, enter a description that is used to easily identify the snapshot.
    - 3.3.3. In the Master Key box, select your master key, and click *Copy*.
4. Verify your copy of the snapshot.

## EBS performance and monitoring

Many factors can affect the performance of EBS, including:

- The I/O characteristics of your applications.
- The type and configuration of your EC2 instances.
- The type and configuration of your volumes.

## EBS performance tips

These tips represent best practices for getting optimal performance from your EBS volumes in various user scenarios:

- *Understand how performance is calculated.* Know which units of measure are involved in calculating the performance metrics. For example, kilobytes per second for bandwidth, input/output operations per second (IOPS) for throughput, milliseconds per operation for latency, etc.
- *Understand your workload.* Know the relationship between the maximum performance of your EBS volumes, the size and number of I/O operations, and the time for each action to complete.
- *Use EBS-optimized instances.* Network traffic can contend with traffic between your EC2 instance and your EBS volumes. On EBS-optimized instances, the two types of traffic are kept separate.
- *Certain factors can degrade HDD performance.* For example, when you create a snapshot of a Throughput Optimized HDD (st1) or Cold HDD (sc1) volume, performance can drop to the volume's baseline value while the snapshot is in progress. Also, driving more throughput than the instance can support can limit performance. Performance also degrades when you initialize volumes that are restored from a snapshot\*. Your performance can also be affected if your application does not send enough I/O requests.
- *Be aware of performance behavior when initializing volumes from snapshots.* You can experience a significant increase in latency when you first access blocks of data on a new EBS volume that was restored from a snapshot. This behavior is a tradeoff since, in this case, EBS volumes were designed for availability over performance. You can avoid this effect on performance by accessing each block before moving the volume into production.
- *Use a modern Linux kernel with support for indirect descriptors.* By using indirect descriptors, you can queue more buffers. Indirect descriptors support block devices by making sure that large requests do not exhaust the ring buffer and increase the queue depth. Linux kernel 3.11 and later includes support for indirect descriptors, as do many current-generation EC2 instances.
- *Increase read-ahead for high-throughput, read heavy workloads on Throughput Optimized HDD and Cold HDD volumes.* Some read-heavy workloads access the block device through the operating system page cache. To achieve the maximum throughput, configure the read-ahead setting to 1 MiB.

---

**Note:** Linux performance tuning.

Some workloads are read-heavy and access the block device through the operating system page cache (for example, from a file system). In this case, to achieve the maximum throughput, the default buffer size for Amazon Linux AMI is 128 KB (256 sectors). AWS recommends that you configure the read-ahead setting to 1 MiB. Apply this per-block-device setting only to your HDD volumes. For example: `sudo blockdev --setra 2048 /dev/xvdf`

---

## Initializing EBS volumes

New EBS volumes receive their maximum performance the moment that they are available and do not require *initialization* (formerly known as pre-warming). However, storage blocks or volumes that were restored from snapshots

must be initialized before you can access the block. Initialization involves pulling down data from S3 and writing it to the EBS volume.

This preliminary action takes time and can cause a significant increase in the latency of an I/O operation the first time each block is accessed. For most applications, amortizing this cost over the lifetime of the volume is acceptable. Performance is restored after the data is accessed once.

You can avoid this performance degradation in a production environment by reading from all of the blocks on your volume before you use it; this process is called *initialization*. For a new volume created from a snapshot, read all the blocks that have data before using the volume.

## Deciding when to use RAID

You can join multiple EBS volumes, in a redundant array of independent disks (RAID) 0 striping configuration to use the available bandwidth for these instances. With RAID 0 striping of multiple volumes, IOPS is distributed among the volumes of a stripe. As a result, you gain faster I/O performance. RAID is a data storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy, performance improvement, or both. Use RAID 0 when:

- Your storage requirement is greater than 16 TB.
- Your throughput requirement is greater than 500 MB/s.
- Your IOPS requirement is greater than 32000 IOPS at 16 K.

Avoid RAID for redundancy because:

- Amazon EBS data is already replicated.
- RAID 1 cuts available EBS bandwidth in half.
- RAID 5 and 6 lose 20% to 30% of usable I/O to parity.

## Monitoring performance using Amazon CloudWatch

Amazon CloudWatch monitors in real time your AWS resources and the applications that you run. CloudWatch metrics are statistical data that you can use to view, analyze, and set alarms on the operational behavior of your volumes. Types of monitoring data that are available for your Amazon EBS volumes:

- Basic data is available automatically in 5-minute periods. Data for the root device volumes of EBS-backed instances is also provided.
- Detailed data is provided by Provisioned IOPS SSD volumes that automatically send one-minute metrics to CloudWatch.

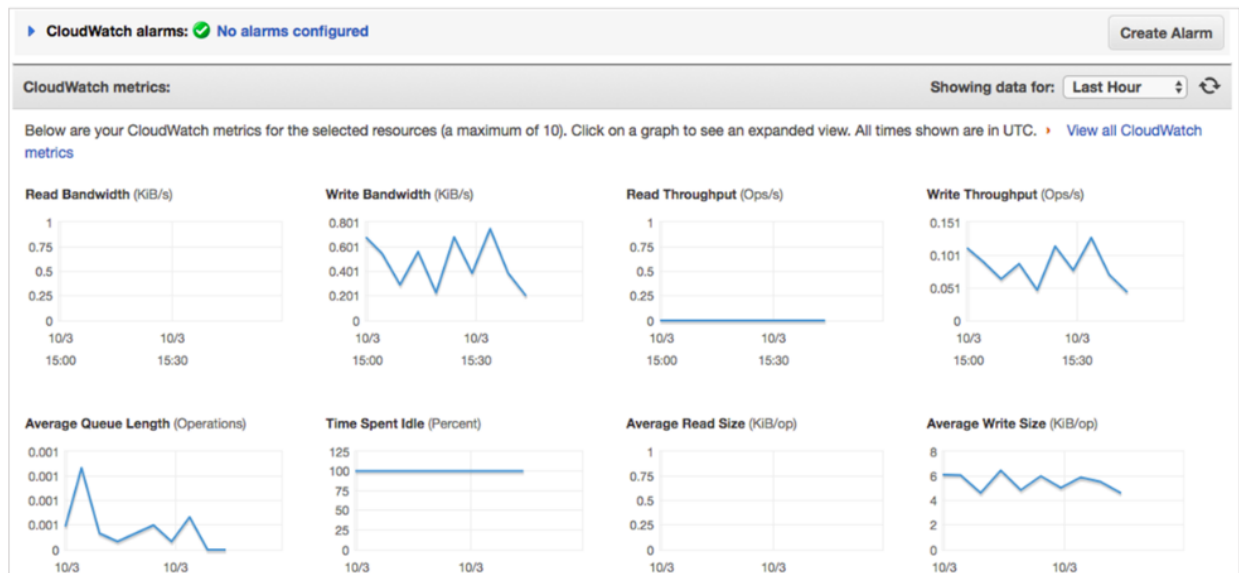
### Monitoring the Status of Your Volumes

A number of performance metrics are available for Amazon EBS. The CloudWatch window in the Amazon EC2 console shows some of the graphs that are created from various EBS metrics. For example:

- The *Write Bandwidth* graph shows the sum of the *VolumeWriteBytes* metric per second. The *VolumeWriteBytes* metric measures the number of bytes that are written to the volume.
- The *Write Throughput* graph shows the *VolumeWriteOps* metric. This metric measures the total number of write operations in a specified period of time.
- The *Average Queue Length* graph shows the *VolumeQueueLength* metric. This metric measures the number of read and write operation requests waiting to be completed in a specified period of time.

Amazon EBS Metric	Description
VolumeReadBytes VolumeWriteBytes	Provide information on the I/O operations in a specified period of time. The Sum statistic reports the total number of bytes transferred during the period.
VolumeReadOps VolumeWriteOps	The total number of I/O operations in a specified period of time.
VolumeQueueLength	The number of read and write operation requests waiting to be completed in a specified period of time.

- The *Average Write* graph uses the *VolumeWriteBytes* metric to measure the average size of each write operation during the specified period.



You can retrieve data from CloudWatch by using either the CloudWatch API or the Amazon EC2 console. Use the `get-metric-data` API to retrieve as many as 100 different metrics in a single request and `get-metric-statistics` for getting statistics for the specified metric. The console uses the raw data that the CloudWatch API return to display a series of graphs that represent the data. Data is reported to CloudWatch only when the volume is active. If the volume is not attached to an EC2 instance, no data is reported.

Volume status checks are automated tests that run every 5 minutes and return a pass or fail status:

- If all checks pass, the status of the volume is `ok`.
- If a check fails, the status of the volume is `impaired`.
- If the status is `insufficient-data`, the checks may still be in progress on the volume.

## Tracking Amazon EBS usage and costs

### EBS pricing model

To estimate the cost of using EBS, you need to consider the following:

- **Volumes.** Volumes storage for all EBS volume types is charged by the amount you provision in GB per month, until you release the storage.
- **Provisioned IOPS SSD volumes.** The throughput amount billed in a month is based on the average provisioned throughput for the month. Your throughput is measured in MB/s-month, which are added up at the end of the month to generate your monthly charges.
- **Snapshots.** Snapshot storage is based on the amount of space your data consumes in S3. Because EBS does not save empty blocks, it is likely that the snapshot size is considerably less than your volume size. Copying EBS snapshots is charged based on the volume of data transferred across regions. For the first snapshot of a volume, EBS saves a full copy of your data to S3. For each incremental snapshot, only the changed part of your EBS volume is saved. After the snapshot is copied, standard EBS snapshot charges apply for storage in the destination region.
- Outbound **Data transfer** is tiered and inbound data is free.

### Tracking costs using tags

Tagging is an important feature that enables you to better manage your AWS resources. Use tags to assign simple key-value pair to resources, like EC2 instances, EBS volumes, and EBS snapshots. With tagging, you can logically group resources to help visualize, analyze, and manage resources. By activating user-defined tags for cost allocation, AWS makes the associated cost data for these tags available throughout the billing pipeline.

#### Using Cost Allocation Tags

You can activate tags as *cost allocation* tags in the Amazon S3 Billing dashboard. By activating cost allocation tags, you can generate reports on cost and usage, broken down by tag values, such as dev, test, or backup. The same dataset that is used to generate AWS Cost and Usage reports is visible in the Cost Explorer for additional visualization. In Cost Explorer, you can view patterns in how much you spend on EBS snapshots. Cost Explorer is enabled from the Billing and Cost Management console in the AWS Management Console.

### Amazon EBS important facts

Here is a list of important information about EBS Volumes:

- When you create an EBS volume in an Availability Zone, it is automatically replicated within that zone to prevent data loss due to a failure of any single hardware component.
- An EBS volume can only be attached to one EC2 instance at a time.
- After you create a volume, you can attach it to any EC2 instance in the same Availability Zone.
- An EBS volume is off-instance storage that can persist independently from the life of an instance. You can specify not to terminate the EBS volume when you terminate the EC2 instance during instance creation.
- EBS volumes support live configuration changes while in production which means that you can modify the volume type, volume size, and IOPS capacity without service interruptions.
- Amazon EBS encryption uses 256-bit Advanced Encryption Standard algorithms (AES-256).
- EBS Volumes offer 99.999% SLA.

### 1.3.4 Amazon EFS

#### Overview

Amazon EFS has a simple web service interface that allows you to create and configure file systems quickly and easily. EFS manages all the file storage infrastructure for you which allows you to avoid the complexity of deploying, patching, and maintaining complex file system deployments. EFS provides simple, scalable file storage for use with Amazon EC2.

EFS file systems store data and metadata across multiple AZs to prevent data loss from the failure of any single component. With EFS, storage capacity is elastic, growing and shrinking automatically as you add or remove files, so your applications have the storage they need, when they need it. Since capacity is elastic, there is no provisioning necessary and you will only be billed for the capacity used.

Choose EFS as a solution for the following scenarios:

- Applications running on EC2 instances that require a file system.
- On-premises file systems that require multi-host attachments.
- High throughput application requirements.
- Applications that require multiple AZs for HA and durability.

It is important to understand your file workload requirements to determine whether EFS is a proper fit for your storage needs.

Amazon FSx provides fully managed third-party file systems. Amazon FSx provides you with the native compatibility of third-party file systems with feature sets for workloads such as Windows-based storage, high-performance computing (HPC), machine learning, and electronic design automation (EDA). You don't have to worry about managing file servers and storage, as Amazon FSx automates the time-consuming administration tasks such as hardware provisioning, software configuration, patching, and backups. Amazon FSx integrates the file systems with cloud-native AWS services, making them even more useful for a broader set of workloads.

Amazon FSx provides you with two file systems to choose from: Amazon FSx for Windows File Server for Windows-based applications and Amazon FSx for Lustre for compute-intensive workloads.

For Windows-based applications, Amazon FSx provides fully managed Windows file servers with features and performance optimized for “lift-and-shift” business-critical application workloads including home directories (user shares), media workflows, and ERP applications. It is accessible from Windows and Linux instances via the SMB protocol. If you have Linux-based applications, Amazon EFS is a cloud-native fully managed file system that provides simple, scalable, elastic file storage accessible from Linux instances via the NFS protocol.

For compute-intensive and fast processing workloads, like high-performance computing (HPC), machine learning, EDA, and media processing, Amazon FSx for Lustre, provides a file system that's optimized for performance, with input and output stored on Amazon S3.

Amazon FSx for Lustre provides a high-performance file system optimized for fast processing of workloads such as machine learning, high performance computing (HPC), video processing, financial modeling, and electronic design automation (EDA). These workloads commonly require data to be presented via a fast and scalable file system interface, and typically have data sets stored on long-term data stores like Amazon S3.

Operating high-performance file systems typically require specialized expertise and administrative overhead, requiring you to provision storage servers and tune complex performance parameters. With Amazon FSx, you can launch and run a file system that provides sub-millisecond access to your data and allows you to read and write data at speeds of up to hundreds of gigabytes per second of throughput and millions of IOPS.

Amazon FSx for Lustre works natively with Amazon S3, making it easy for you to process cloud data sets with high-performance file systems. When linked to an S3 bucket, an FSx for Lustre file system transparently presents S3 objects as files and allows you to write results back to S3. You can also use FSx for Lustre as a standalone high-performance file system to burst your workloads from on-premises to the cloud. By copying on-premises data to an FSx for Lustre

file system, you can make that data available for fast processing by compute instances running on AWS. With Amazon FSx, you pay for only the resources you use. There are no minimum commitments, upfront hardware or software costs, or additional fees.

## Use cases

Some common EFS use cases are:

- *Web serving.* The need for shared file storage for web-serving applications can be a challenge when integrating backend applications. Typically, multiple web servers deliver a website's content, with each web server needing access to the same set of files.
- *Big data analytics.* Big data requires storage that can handle large amounts of data while scaling, to keep up with growth, and providing the performance necessary to deliver data to analytics tools. Many analytics workloads interact with data by means of a file interface, rely on file semantics, such as file locks, and require the ability to write to portions of a file. EFS provides the scale and performance required for big data applications that require high throughput to compute nodes coupled with read-after-write consistency and low-latency file operations. EFS can provide storage for organizations that have many users that need to access and share common datasets.
- *Media and entertainment.* Digital media and entertainment workflows access data by using network file protocols, such as NFS. These workflows require flexible, consistent, and secure access to data from off-the-shelf, custom-built, and partner solutions.
- *Container storage.* Docker containers are ideal for building microservices because they're quick to provision, easily portable, and provide process isolation. A container that needs access to the original data each time it starts may require a shared file system that it can connect to regardless of which instance they're running on.
- *Database backups.* Backing up data by using existing mechanisms, software, and semantics can create a isolated recovery scenario with little locational flexibility for recovery.
- *Content management.* You can use EFS to create a file system accessible to people across an organization and establish permissions for users and groups at the file or directory level.

Customers are using EFS for *Home directories* and *software development tools*.

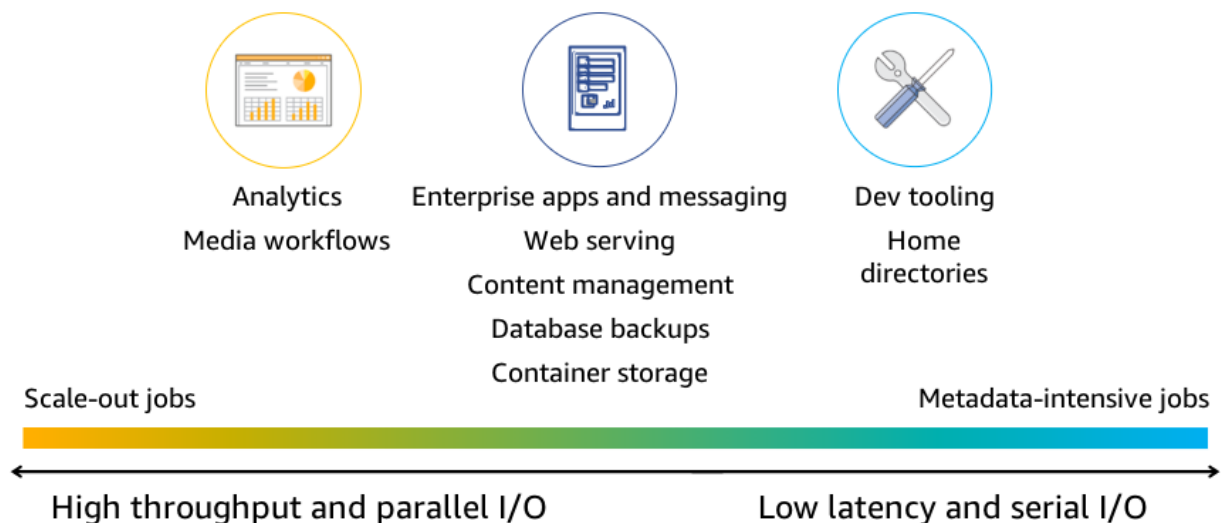


Fig. 26: Amazon EFS usage

Amazon Elastic File System - Scalable, Elastic, Cloud-Native File System for Linux



## Amazon EFS architecture

There are 3 deployment models for NFS storage: On-premises NFS storage, Cloud-based NFS storage in a “do it yourself model”, and Amazon EFS, which is a fully-managed NFS storage service.

An on-premises NFS storage system is designed asking the following questions: “How much storage you need today? How do I ensure that it is sized for the performance needs of my application? What is my expected data growth over the next year, three years, and five years? Do I need to design this system for HA?”. In general, you might not have all the answers to these questions, which can have you speculating on the best approach.

You might also consider building your own NFS storage solution in the cloud through a combination of compute instances (EC2 instances), storage volumes (EBS volumes attached to EC2 instances), and NFS software. The total costs of an NFS setup become expensive to build and manage.

## EFS architecture

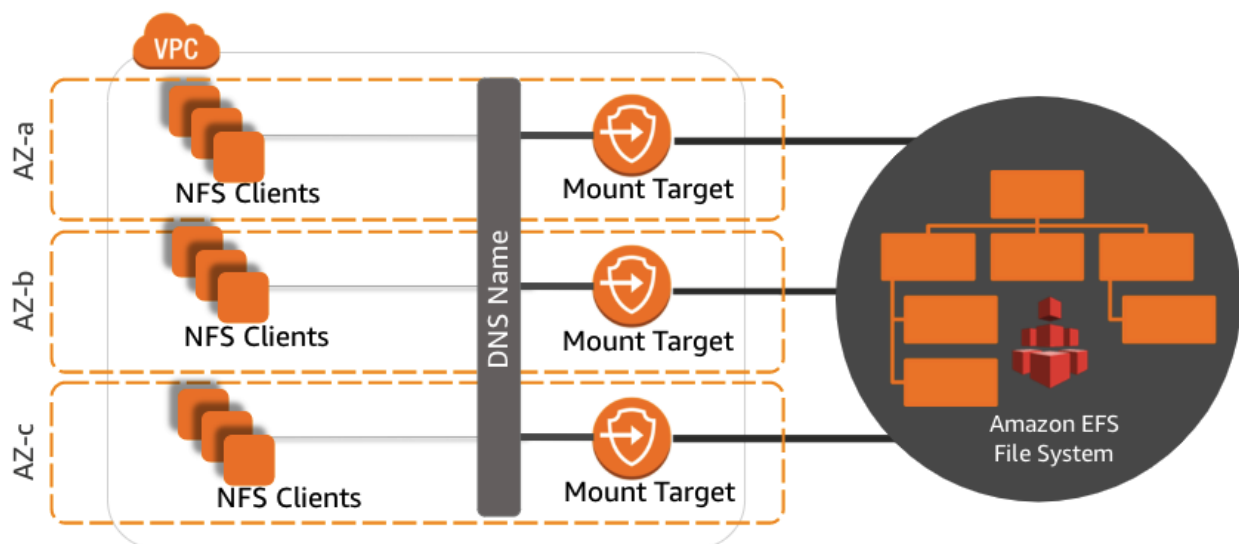
When you choose Amazon EFS, you need to only create mount targets in each AZ where your EC2 instances reside to connect to the Amazon EFS share. Amazon EFS does all the heavy lifting by providing a common DNS namespace in a high durable, scalable, NFS system that your EC2 instances can connect to. EFS supports NFS versions 4.0 and 4.1, so the applications and tools that use this protocols work with EFS. Multiple EC2 instances can access an EFS file system at the same time, providing a common data source for workloads and applications running on more than one instance. EFS charges by gigabyte of data stored and no provisioning of storage is required.

---

**Note:** Provisioned Throughput mode.

Provisioned Throughput mode adds a second price dimension. You are charged for the amount of storage you use and the throughput you provision.

---





## EFS file system

The **EFS file system** is the primary resource for EFS where files and directories are stored. EFS provides strong data consistency and file-locking features, so using EFS is just like using any of the popular NFS file systems. EFS is based on a regional construct, so you create an EFS file system in your chosen supported region. For most regions, EFS limits the number of file systems that you can create per account to 125.

When creating your file system, you can choose from 2 performance modes: General Purpose and Max I/O. **General Purpose** mode is recommended as the best choice for most workloads, and provides the lowest latency for file operations. Consider the **Max I/O** mode for data-heavy applications that scale out significantly over time.

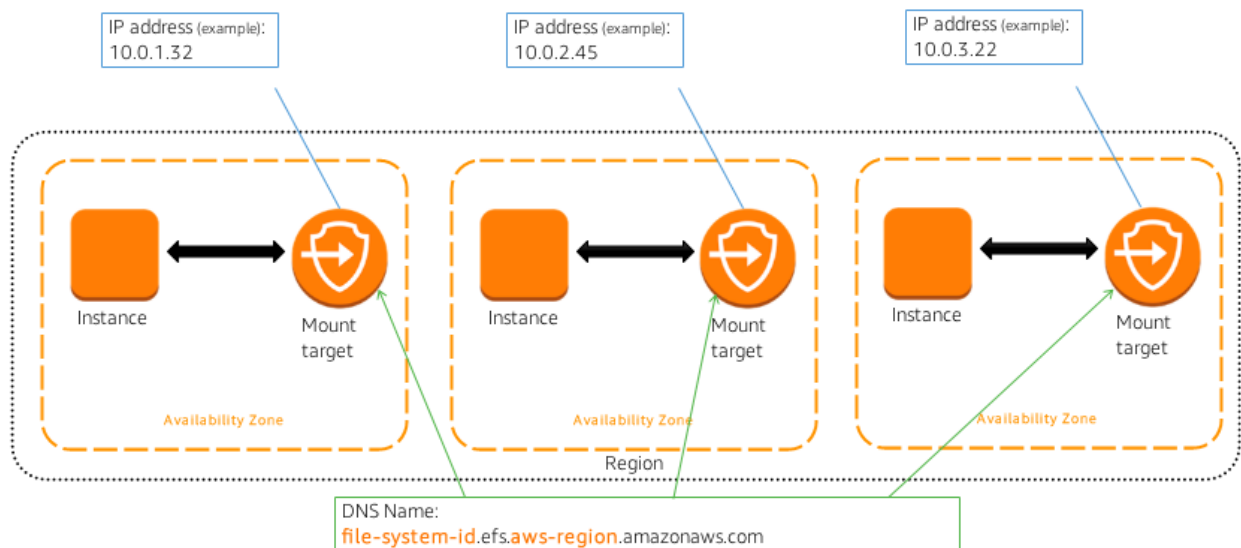
You can choose from 2 throughput modes: Bursting and Provisioned. **Bursting** Throughput mode is recommended for most file systems. Use the **Provisioned** Throughput mode for applications that require more throughput than allowed by Bursting Throughput.

After you have created your file system, it is instantly accessible from EC2 instances within your VPC and can be accessed by multiple instances simultaneously. Additionally, you can connect to EFS from your on-premises clients by using AWS Direct Connect. After you have mounted your EFS file system, you can control access by using Portable Operating System Interface (POSIX) permissions.

To help you easily identify or organize the multiple file systems you create, you can add tags to your file systems. These are key and value pairs that you create during or after your file system creation. You can create up to 50 tags per file system.

## Mount target

A **mount target** is the resource created to enable access to your EFS file system from an EC2 instance or on-premises clients. A single mount target exists in each AZ and is the only way in which you can access your EFS file system. A mount target is analogous to an NFS endpoint. A mount target allows resources that reside in a VPC to access the EFS file system. Mount targets are automatically created during the EFS file system creation process. The mount target has an IP address and a DNS name that you use to mount your EFS file system to your EC2 instances.



The DNS name contains the file system ID and the AWS Region where the file system was created. By default, mount targets are designed to be highly available. You can also configure access to a mount target by using security groups.

Only one mount target is in each AZ, but if your AZ contains multiple subnets, you can choose which subnet ID to assign to your EFS file system, then mount your EC2 instances. After this is done, you can create one mount target in

each AZ. The mount target obtains an IP address from the subnet dynamically, or you can assign a static IP address. The subnet IP address of the mount target does not change.

## EFS requirements

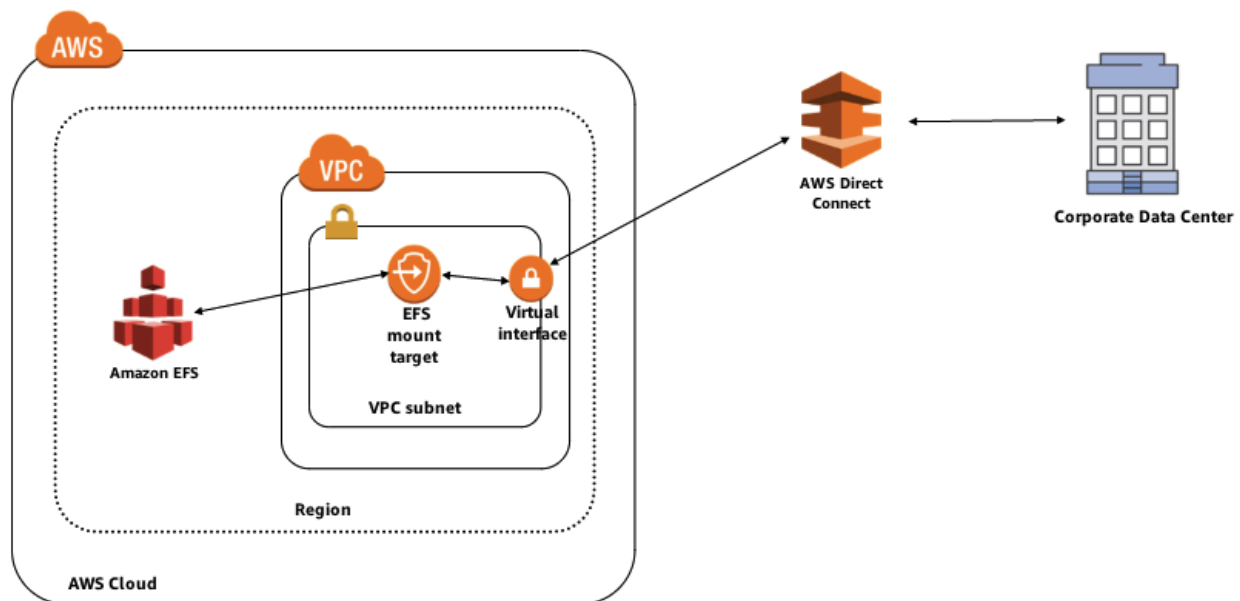
Before setting up your file system in EFS, you must take into account certain requirements:

1. You can mount an EFS file system to instances in only one VPC at a time. As you design and configure your applications, remember to keep your instances, which need access to the same file system, in a single VPC.
2. Your EFS file system and your VPC must reside in the same region. Because EFS is available only in certain regions, consider the locations where you are deploying resources in AWS.

## Accessing EFS file systems

You can access EFS file systems in 3 ways: from EC2 instances, from on-premises hosts that mount EFS file systems over AWS Direct Connect, and from hosts running in the VMware Cloud on AWS.

From on-premises hosts, you mount your file systems through mount targets as you would with EC2, but instead of using a DNS name, AWS recommends that you use an IP address in the mount command. If you choose to mount your file system on your on-premises server through AWS Direct Connect with a DNS name, you must integrate your DNS in your VPC with your on-premises DNS domains. Specifically, to forward the DNS requests for EFS mount targets to a DNS server in the VPC over the AWS Direct Connect connection, you must update your on-premises DNS server.



Some of the common use cases for using AWS Direct Connect are the migration of data, backup/disaster recovery, and bursting. Bursting is copying on-premises data to EFS, analyzing the data at high speed, and sending the data back to the on-premises servers.

## Amazon EFS File Sync

Amazon EFS File Sync copies existing on-premises or cloud file systems to an EFS file system. Data copy operations are 5 times faster than standard Linux copy tools, and you can set up and manage your copy job easily from the AWS Management Console.

## What Is Amazon Elastic File System?

### Amazon EFS File System Management

#### Setting up and managing a file system

With EFS, you can set up and manage your file system by using 3 distinct interfaces: the AWS Management Console, AWS CLI and AWS SDK. The process of creating an EFS file system from the AWS Management Console has the following steps:

1. Choose the region where you want to create your file system.
2. Choose a VPC that has EC2 instances that will connect to your new file system.
3. Choose the appropriate subnets for your EFS where your EC2 instances reside.
4. Assign a security group for your file system. If you have not created a security group, you can create it later.
5. Choose the performance mode for your cloud storage workloads. Choose the throughput mode for your file system.
6. You can enable encryption for your file system, if it is needed.

#### Mounting a file system

After creating a file system, you can view details about it in the AWS Management Console, such as its file system ID, performance mode, and throughput mode. The instructions for mounting the file system on an EC2 instance provide steps for installing the EFS mount helper and the NFS client. Steps to mount the file system with either the NFS client or the EFS mount helper are also provided. Each mount command requires the file system ID as an argument.

To mount your EFS file systems, use the EFS mount helper included in the *amazon-efs-utils* package. It is an open-source collection of Amazon EFS tools and there's no additional cost to use them. This package includes a mount helper and tooling that make it easier to perform encryption of data in transit for EFS. A mount helper is a program that you use when you mount a specific type of file system.

To mount a file system with the EFS mount helper, follow these steps:

1. For an Amazon Linux EC2 instance, install the Amazon EFS utilities

```
sudo yum install -y amazon-efs-utils
```

By default, when using the EFS mount helper with Transport Layer Security (TLS), the mount helper enforces the use of the Online Certificate Status Protocol (OCSP) and certificate hostname checking.

2. Create a new directory on your EC2 instance for the mount point, such as `efs`.

```
sudo mkdir efs
```

3. To mount a file system whose ID is `fs-4fad7b36`, without encryption using the EFS mount helper:

```
sudo mount -t efs fs-4fad7b36:/ efs
```

If you require encryption of data in transit, include the TLS mount option, as shown:

```
sudo mount -t efs -o tls fs-4fad7b36:/ efs
```

4. With your file system created, you can now mount your file system to your EC2 instances. However, before you can begin the mount process, you must install the NFS client onto the operating system of your EC2 instance and create a directory for the mount point.

```
sudo mount -t nfs4 -o nfsvers=4.1, rsize=1048576, wsize=1048576, hard, timeo=600,  
↪retrtrans=2 file-system-id.ef.aws-region.amazonaws.com:/ efs
```

Obtain the file system ID and AWS REgion from the EFS console. The `nfsvers` parameter sets the protocol version for NFS. In this case, it is NFS protocol 4.1. The `rsize` and `wsize` parameters are the I/O parameters for read and write data block size. For EFS, the recommended initial size is 1 MB, which is represented in bytes. The next parameter is the mount type. `Hard` is the recommended default value; although `soft` can be used with some additional considerations for `timeo`. `Timeo` is the timeout setting, with 600 deciseconds (one-tenth of a second) as the recommended default value for EFS. If the timeout value is changed from the recommended 60 seconds set it to a minimum of 15 seconds. The `retrtrans=2` parameter specifies that 2 minor timeouts will occur before a major timeout occurs.

Additionally, you can automatically mount your instance to your EFS file system upon startup. You can do this by using a script or configuring an `fstab` file when launching and starting your EC2 instance.

[Mounting from an Amazon EC2 Instance](#)

[Mounting Your Amazon EFS File System Automatically](#)

[Mounting File Systems Without the EFS Mount Helper](#)

[Using Microsoft Windows File Shares](#)

## Securing your data in Amazon EFS

### Security controls

You can control network traffic to and from file system by using VPC security groups and network access control lists to secure network access to your mount targets. To manage file and directory access, use standard Linux directory and file-level permissions to secure access to your data like any other NFS share. To control administrative access to EFS, use AWS IAM to control who can create, change, and manage EFS file systems. If you require encryption, you can encrypt your data at rest in EFS by integrating with the AWS KMS. You can also encrypt data in transit by enabling TLS when you mount your file system.

### Security groups

For EFS, you can configure up to 5 security groups per mount target. To obtain access to an NFS mount target in AWS, TCP port 2049 must be open for inbound traffic to the mount target and outbound traffic to clients must be connected to the EFS file system.

### Amazon EFS POSIX permissions

Amazon EFS supports the standard permission model you use when working with local file systems in Linux or Windows. These permission settings enforce user and group rights at the file and directory level. In a Linux instance, you can use `chmod` and specify read (r), write (w), execute (x) permissions.

## Integrating with AWS IAM

Use IAM policies to control access to EFS. AWS IAM has 2 default policies for managing access: `AmazonElasticFileSystemFullAccess` and `AmazonElasticFileSystemReadOnlyAccess`. These policies grant full administrative access and read-only access to EFS.

For scenarios in which you want to grant the ability to create new file systems but prohibit the deletion of file systems, create a custom IAM policy for those types of granular permissions. Additionally, you can allow or deny mount target creation and deletion, security group modification, and tag creation.

## Encryption at rest

If your workload requires encrypting data at rest, EFS integrates with the AWS KMS to provide encryption key management. Choosing the encryption option when you are creating your file system encrypts all data and metadata in your file system. After you have chosen encryption and created the file system, you cannot change these settings.

If you have an existing EFS file system that contains data that you want to encrypt, you must copy that data to a new encrypted EFS file system. You can create encrypted file systems through the AWS Management console, AWS CLI and AWS SDK.

AWS KMS handles the encryption and decryption of files, which makes the process transparent. Data and metadata are encrypted before being written to a disk and decrypted before being accessed by a client. Additionally, because the encryption process is transparent, it does not affect an application's access to data or metadata, so modifications to your application are not required.

The process of enabling encryption is simple: select the Enable encryption check box and choose your encryption master key. You must select the encryption option when creating the EFS file system.

## Encrypting data in transit

You can encrypt data in transit for EFS file system by enabling TLS when you mount your file system by using the EFS mount helper. When encryption of data in transit is declared as a mount option for your EFS file system, the mount helper initializes a client stunnel (ssl-tunnel) process. Stunnel is an open source multipurpose network relay. The client stunnel process listens on a local port for inbound traffic, and the mount helper redirects NFS client traffic to this local port. The mount helper uses TLS version 1.2 to communicate with your file system.

Data Encryption in EFS

## Performance and optimization

Amazon EFS is a distributed file system in that it is distributed across an unconstrained number of servers. This configuration enables EFS to avoid bottlenecks and constraints of traditional file servers. Additionally, this distributed design enables a high level of aggregate IOPS and throughput for your applications. Data is also distributed across AZs, which enables the high level of durability and availability that EFS provides. Mount targets are placed in each AZ, and data is distributed across AZs.

## Performance modes

The default general purpose mode, is suitable for most workloads, general purpose mode provides the lowest latency operations, but has a limit of 7000 operations per second. Max I/O mode is suitable for workloads that require more than 7000 operations per second, but there are some tradeoffs to consider. The increase in operations per second often results in higher latency times for file operations. If a change in performance mode is needed after creation, you must create a new file system, select the correct mode, and then copy your data to the new resource.

Mode	What it is for	Advantages	Trade-offs	When to use
General purpose (default)	Latency-sensitive applications and general-purpose workloads	Lowest latencies for file operations	Limit of 7K ops/sec	Best choice for most workloads
Max I/O	Large-scale and data-heavy applications	Virtually unlimited ability to scale out throughput/IOPS	Slightly higher latencies	Consider for large scale-out workloads

Fig. 27: Amazon EFS performance modes

### Throughput in relation to size

With EFS, the throughput performance you can achieve is directly related to the size of the file system. Throughput levels on EFS scale as a file system grows. Because file-based workloads are typically spiky, which drive high levels of throughput for short periods of time and low level of throughput the rest of the time, EFS is designed to burst to high throughput levels when needed.

All file systems, regardless of size can burst to 100 MiB/s of throughput, and those over 1-TiB large can burst to 100 MiB/s per TiB of data stored in the file system. For example, a 3-TiB file system can burst 300 MiB per second of throughput. The portion of time a file system can burst is determined by its size, and the bursting model is designed so that typical file system workloads will be able to burst when needed.

EFS uses a credit system to determine when file systems can burst. Each file system earns credits over time at a baseline rate that is determined by the size of the file system, and uses credits whenever it reads or writes data. If you find that you need more throughput than EFS is providing based on your data size, you can enable Provisioned Throughput mode to provide additional throughput to support your application requirements.

### Throughput modes

The throughput mode of the file system helps determine the overall throughput a file system is able to achieve. You can select the throughput mode at any time (subject to daily limits). Changing the throughput mode is a non-disruptive operation and can be run while clients continuously access the file system. You can choose between 2 throughput modes: Bursting or Provisioned.

**Bursting Throughput** is the default mode and is recommended for a majority of uses cases and workloads. Throughput scales as your file system grows, and you are billed for only the amount of data stored on the file system in GB-Month. Because file (based workloads are typically spiky-driving high levels of throughput for short periods of time and low level of throughput the rest of the time) using Bursting Throughput mode allow for high throughput levels for a limited period of time.

You can monitor and alert on your file system's burst credit balance by using the `BurstCreditBalance` file system metric in Amazon CloudWatch. File systems earn burst credits at the baseline throughput rate of 50 MiB/s per TiB of data stored and can accumulate burst credits up to the maximum size of 2.1 TiB per TiB of data stored. A newly created file systems begin with an initial credit balance of 2.1 TiB. As a result, larger file systems can accumulate and store more burst credits, which allows them to burst to longer periods.

**Provisioned Throughput** is available for applications that require a higher throughput to storage ratio than those

allowed by the Bursting Throughput mode. In this mode, you can provision the file system's throughput independent of the amount of data stored in the file system. Provisioned Throughput allows you to optimize your file system's throughput performance to match your application's needs, and your application can drive up to the provisioned throughput continuously.

When file systems are running in Provisioned Throughput mode, you are billed for the storage you use in GB-Month and for the throughput provisioned in MiB/s-Month. The storage charge for both Bursting and Provisioned Throughput modes includes throughput of the file system in the price storage.

You can increase Provisioned Throughput as often as you need. You can decrease Provisioned Throughput or switch throughput modes as long as it's been more than 24 hours since the last decrease or throughput mode change.

Mode	What it is for	Advantages	Trade-offs	When to use
Bursting Throughput (default)	Varying throughput workloads	Auto-scaling throughput	Fixed throughput to storage ratio	Best choice for most workloads
Provisioned Throughput	Higher-consistent throughput workloads	User-defined throughput	Separate throughput charge	Ingest or higher throughput to storage ratio

Fig. 28: Amazon EFS throughput modes

Whitepaper: Choosing Between the Different Throughput & Performance Modes

Amazon EFS Performance

## Maximizing Amazon EFS throughput

### Parallelizing file operations

When looking at your workload, consider the type of Amazon EC2 instance that is connecting to your file system. Review the virtual CPU, memory, and the network performance capabilities before deciding on Amazon EFS. These attributes determine the performance level of applications that use EFS.

To achieve higher throughput levels, EFS takes advantage of parallelizing your operations owing to EFS' distributed architecture. Parallelization means using more instances and more threads per instance. If your application can use multiple threads to perform file operations, you can take advantage of parallelization to increase performance.

An effective use case could be a data migration or large file copy operation that use multiple threads to shorten the time frame of the data move.

### I/O size - serial operations

The I/O size impact throughput for EFS is when performing serialized operations. As the amount of data you are reading or writing in an operation increases, the throughput increases as well. This is because of the distributed storage design of Amazon EFS.



This distributed architecture results in a small latency overhead for each file operation. As the result of this per-operation latency, overall throughput generally increases as the average I/O size increases because the overhead is amortized over a larger amount of data.

Testing your workloads and understanding how your application is reading or writing data help you determine which applications are eligible for high throughput data transfers with EFS.

### Amazon EFS CloudWatch metrics

Performance metrics are displayed for each file system created. CloudWatch captures the following metrics:

- `BurstCreditBalance` determines the number of burst credits you have available.
- `ClientConnections` displays the number of client connections to a file system. When using a standard client, there is 1 connection per mounted EC2 instance.
- `DataReadIOBytes` is the number of bytes for each file system read operation.
- `DataWriteIOBytes` is the number of bytes for each file system write operation.
- `MetadataIOBytes` is the number of bytes for each metadata operation.
- `PercentIOLimit` is an important metric to look at when using the General Purpose mode. This metric is available only on the General Purpose mode and helps you determine whether you are reaching the limits of this mode, and if you need to switch to the Max I/O mode.
- `PermittedThroughput` shows the amount of throughput the file system is allowed. For file systems in the Provisioned Throughput mode, this value is the same as the provisioned throughput. For the file systems in the Bursting Throughput mode, this value is a function of the file system size and `BurstCreditBalance`.
- `TotalIOBytes` is the number of bytes for each file system operation, including data read, data write, and metadata operations.

The 2 most important metrics for CloudWatch monitoring are `PercentIOLimit` and `BurstCreditBalance`. `PercentIOLimit` because you will use this metric for determining when to use the General Purpose mode or Max I/O mode. `BurstCreditBalance` because it can show you how you are using the available burst credits you have and how often you are using burst capacity. File systems by nature are spiky and will most likely use some bursting, but if a performance issue surfaces, you must also determine how often you might be draining your burst credits.

Metric math enables you to query multiple CloudWatch metrics and use math expressions to create a new time series based on these metrics. You can visualize the resulting time series in the CloudWatch console and add them to dashboards.

### Monitoring EFS with Amazon CloudWatch

### Best practices using Amazon EFS

In summary, here are some key recommendations to consider when using Amazon EFS:

- Before moving your applications to EFS, set up test environments in AWS to understand how your applications behave while using EFS. In your test environment, you can obtain data on performance levels, throughput levels, and latency before deciding on EFS. The General Purpose setting is suitable for the majority of most workloads and provides the lowest latency.
- Start testing with Bursting Throughput. It is the recommended throughput mode for most file systems.
- For optimal performance and to avoid a variety of known NFS client defects, work with a recent Linux kernel, version 4.0 or later.



- To mount your EFS file system on your EC2 instance, use the EFS mount helper. You can use NFS version 4.1 when creating your file system mount points to ensure optimum compatibility. EFS supports NFS versions 4.0 and 4.1 protocols when mounting your file systems on EC2 instances. However, NFS v4.1 provides better performance.
- Improve EFS performance by aggregating I/O operations, parallelizing, and managing burst credits. Check your burst credit earn/spend rate when testing to ensure a sufficient amount of storage.
- Where possible, leverage the use of multiple threads, EC2 instances, and directories.
- Monitor EFS metrics with CloudWatch to maintain the reliability, availability, and performance of EFS.

## Amazon EFS cost and billing

### EFS pricing model

Generally, Amazon EFS pricing varies by region. With EFS, you have 3 pricing dimensions:

- **Storage:** amount of file system storage you use per month.
- **Throughput.** In the default Bursting Throughput mode, no charges are incurred for bandwidth or requests, and you get a baseline rate of 50 KB/s per GB of throughput included with the price of storage. You can choose the Provisioned Throughput mode and provision the throughput of your file system independent of the amount of data stored and pay separately for storage and throughput. Like the default Bursting Throughput mode, the Provisioned Throughput mode also includes 50 KB/s per GB (or 1 MB/s per 20 GB) of throughput in the price of storage. You are billed only for the throughput provisions above what you are provided based on data you have stored. When using the Provisioned Throughput mode, you pay for the throughput you provision per month. No minimum fee and no setup charges are incurred.
- **EFS File Sync.** You pay per-GB for data copied to EFS. You can use CloudWatch to track the amount of data synced with another file system, whether it's on the AWS Cloud or on premises. If your source file system is in AWS, you are billed at standard EC2 rates for the instance on which the EFS File Sync agent runs.

## Amazon EFS versus DIY file storage

EFS pricing is based on paying for what you use. If Provisioned Throughput mode is enabled, there are no minimum commitments or upfront fees, and you are charged per GB stored and throughput consumed. You are not required to provision storage in advance, and you do not incur additional charges or billing dimensions when using EFS. With do-it-yourself file storage configurations, you must consider many costs, such as EC2 instance costs, EBS volume costs, and Inter-Availability Zone transfer costs.

[Amazon EFS now Supports Access Across Accounts and VPCs](#)

[Mounting EFS File Systems from Another Account or VPC](#)

### 1.3.5 Amazon storage comparison

### 1.3.6 AWS Storage Gateway

All data transferred between any type of gateway appliance and AWS storage is encrypted using SSL. By default, all data stored by AWS Storage Gateway in S3 is encrypted server-side with Amazon S3-Managed Encryption Keys (SSE-S3). Also, when using the file gateway, you can optionally configure each file share to have your objects encrypted with AWS KMS-Managed Keys using SSE-KMS.

Storage Need	Solution	AWS Services
<b>Temporary storage</b>	Consider using local instance store volumes for needs such as scratch disks, buffers, queues, and caches.	<a href="#">Amazon Local Instance Store</a>
<b>Multi-instance storage</b>	Amazon EBS volumes can only be attached to one EC2 instance at a time. If you need multiple EC2 instances accessing volume data at the same time, consider using Amazon EFS as a file system.	<a href="#">Amazon EFS</a>
<b>Highly durable storage</b>	If you need very highly durable storage, use S3 or Amazon EFS. Amazon S3 Standard storage is designed for 99.999999999 percent (11 nines) annual durability per object. You can even decide to take a snapshot of the EBS volumes. Such a snapshot then gets saved in Amazon S3, thus providing you the durability of Amazon S3. For more information on EBS durability, see the <a href="#">Durability and Availability</a> section. EFS is designed for high durability and high availability, with data stored in multiple Availability Zones within an AWS Region.	<a href="#">Amazon S3</a> <a href="#">Amazon EFS</a>
<b>Static data or web content</b>	If your data doesn't change that often, Amazon S3 might represent a more cost-effective and scalable solution for storing this fixed information. Also, web content served out of Amazon EBS requires a web server running on Amazon EC2; in contrast, you can deliver web content directly out of Amazon S3 or from multiple EC2 instances using Amazon EFS.	<a href="#">Amazon S3</a> <a href="#">Amazon EFS</a>

Fig. 29: Amazon storage comparison

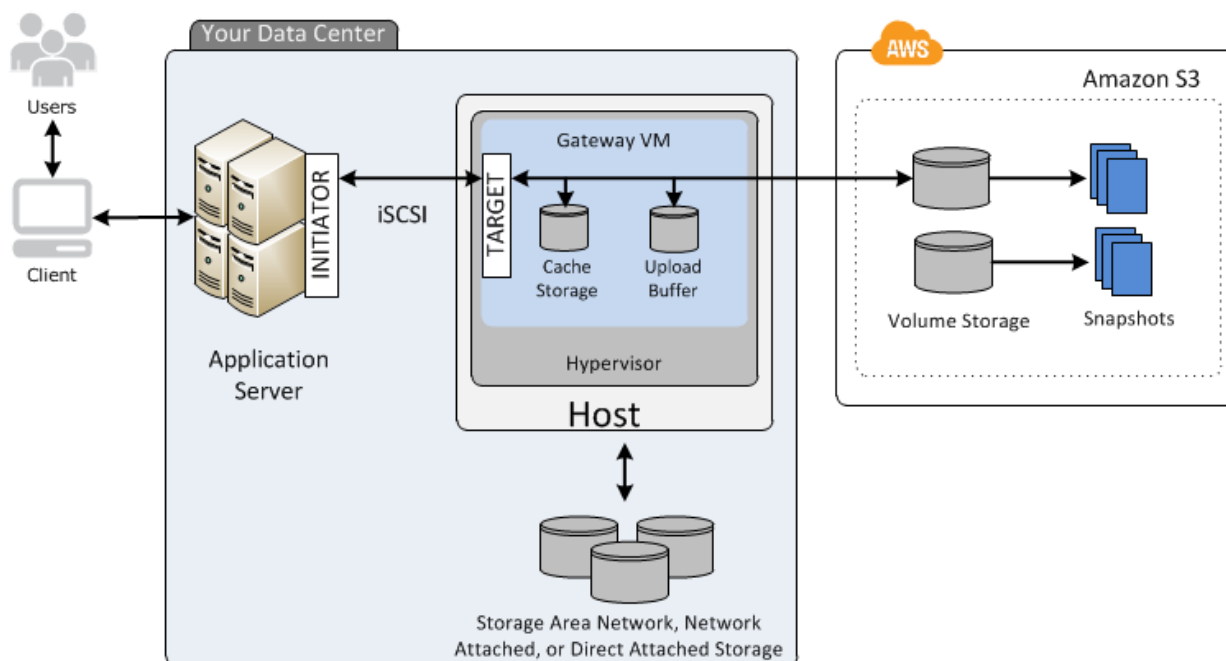


Fig. 30: AWS Storage Gateway

Volume gateway provides an iSCSI target, which enables you to create block storage volumes and mount them as iSCSI devices from your on-premises or EC2 application servers. The volume gateway runs in either a cached or stored mode.

- In the cached mode, your primary data is written to S3, while retaining your frequently accessed data locally in a cache for low-latency access.
- In the stored mode, your primary data is stored locally and your entire dataset is available for low-latency access while asynchronously backed up to AWS.

In either mode, you can take point-in-time snapshots of your volumes, which are stored as Amazon EBS Snapshots in AWS, enabling you to make space-efficient versioned copies of your volumes for data protection, recovery, migration and various other copy data needs.

By using cached volumes, you can use Amazon S3 as your primary data storage, while retaining frequently accessed data locally in your storage gateway. Cached volumes minimize the need to scale your on-premises storage infrastructure, while still providing your applications with low-latency access to frequently accessed data. You can create storage volumes up to 32 TiB in size and afterwards, attach these volumes as iSCSI devices to your on-premises application servers. When you write to these volumes, your gateway stores the data in Amazon S3. It retains the recently read data in your on-premises storage gateway's cache and uploads buffer storage.

Cached volumes can range from 1 GiB to 32 TiB in size and must be rounded to the nearest GiB. Each gateway configured for cached volumes can support up to 32 volumes for a total maximum storage volume of 1,024 TiB (1 PiB).

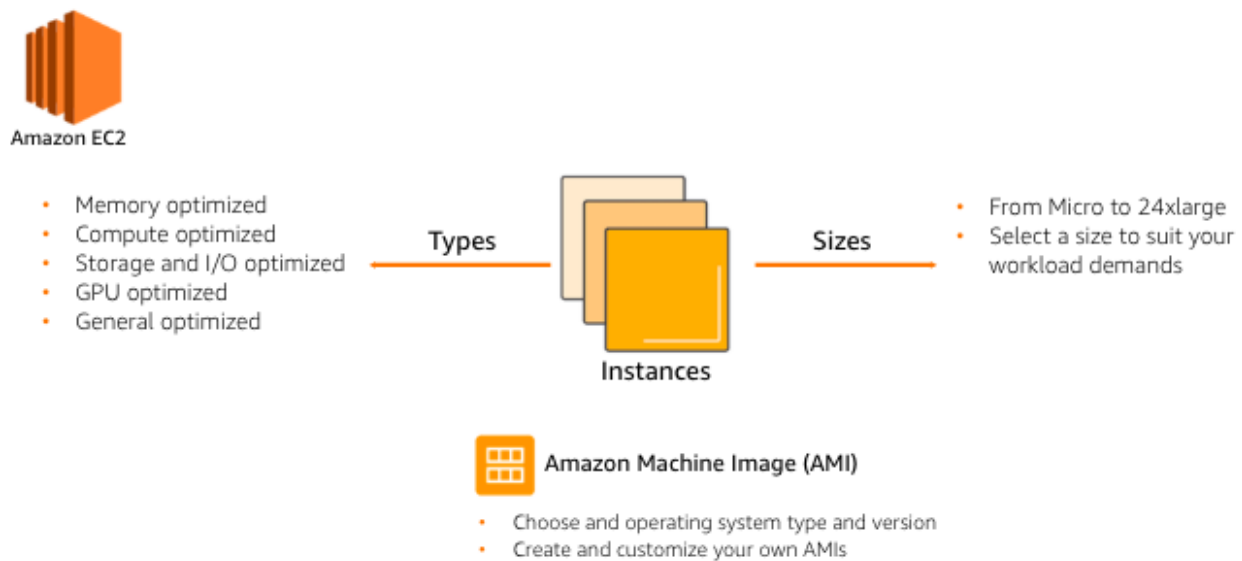
In the cached volumes solution, AWS Storage Gateway stores all your on-premises application data in a storage volume in Amazon S3.

## 1.4 Amazon EC2

### 1.4.1 Resources

#### Instances

Inside AWS physical servers, there are hypervisors on which guest virtual machines live, called Elastic Compute Cloud (EC2) instances. AWS provides various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*. In 2007, AWS provide only one instance type called M1, since then a lot of instances types have been launched by AWS (175 instances type since 2007 to 2018). This continued rapid pace of innovation has been possible thanks to the [AWS Nitro System](#).



#### Amazon EC2 instance types

An example of an API name of an EC2 instance type is `M5d.xlarge`. The first character (in this example M) identifies the instance family. Each instance family is designed for an specific workload needs in terms of CPU, memory, storage and network performance. Amazon EC2 families define different ratios among CPU, memory, storage and network performance. Below you can find the current instance families together with the letters associated with them, and the meaning of each of the letters:

- **General Purpose:** A : ARM, T: Turbo(Burstable), M: Most Scenarios.
- **Compute Optimized:** C: Compute.
- **Memory Optimized:** R: Random-access memory, X: Extra-large memory (approx 4 TB DRAM), U: High memory (purpose built to run large in-memory databases), Z: High frequency.
- **Accelerated Computing:** P: GPU optimized for machine learning, G: GPU optimized for graphics, F: FPGA, Inf: Inferentia chips.
- **Storage Optimized:** I: I/O (NVMe local), D: Dense storage (48 TB local), H: HDD (16 TB local).

The second character represents the instance generation (in this example 5). Sometimes, there is an additional character between the instance generation and the dot. This character can be one of the following:

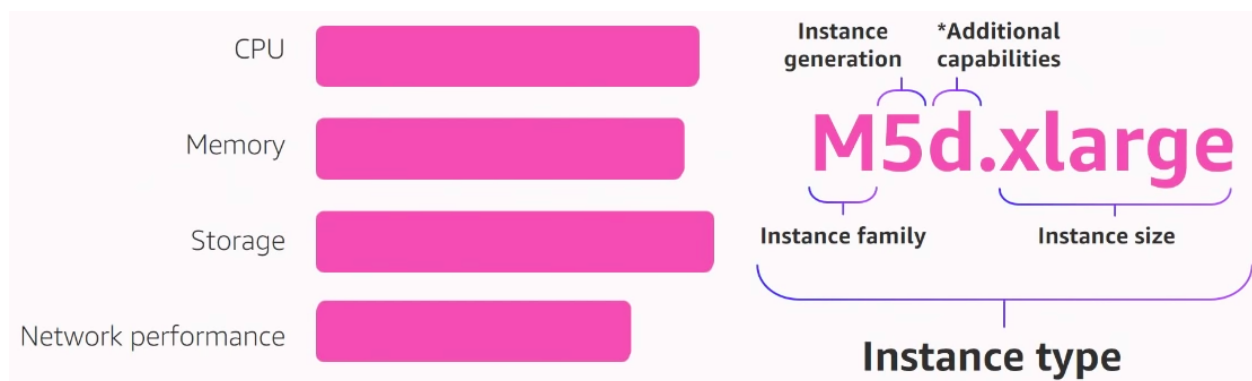
- **a:** AMD CPUs.

- **e**: Extra Capacity (Storage or RAM).
- **n**: Network optimized.
- **d**: Direct-attached instance storage (NVMe).
- **g**: ARM-based AWS Graviton CPUs.

After the dot, you have the instance size (in this example `xlarge`). From nano to large sizes the number of vCPUs is 2 and the number of GiB in memory starts from 0.5 and doubles in the immediate higher instance size. For instance sizes greater than `large`, you should multiply the CPU and memory by the number that is before `x`. If there is no number, 1 is considered. The instance size called `metal` corresponds with Bare metals instance and usually has the same amount of vCPU and memory as the biggest virtual instance size. The following table illustrates instance sizes characteristics:

Table 1: Instance sizes characteristics.

Instance Size	vCPU	Memory (GiB)
nano	2	0.5
micro	2	1
small	2	2
medium	2	4
large	2	8
xlarge	4	16
2xlarge	8	32
...	...	...
<i>y</i> xlarge	$y \times 2$	$y \times 8$



[Amazon EC2 instance types](#)

[Introducing Elastic Fabric Adapter](#)

[EC2 Instance Types & Pricing](#)

[EC2Instances.info](#)

## Amazon Machine Images (AMIs)

AMI is the software required to launch an EC2 instance. There are 3 different types:

- **Amazon maintained** AMIs which offer a broad set of Linux and Windows images.
- **Marketplace maintained** AMIs managed and maintained by AWS Marketplace partners.

- **Your machine images** are AMIs you have created from Amazon EC2 instances. You can maintain them as private in your account, shared with other accounts and even made them public to the community.

You can copy an Amazon Machine Image (AMI) within or across an AWS region using the AWS Management Console, the AWS command line tools or SDKs, or the Amazon EC2 API, all of which support the `CopyImage` action. You can copy both Amazon EBS-backed AMIs and instance store-backed AMIs. You can copy encrypted AMIs and AMIs with encrypted snapshots.

Amazon EC2 uses public-key cryptography to encrypt and decrypt login information. Public-key cryptography uses a public key to encrypt a piece of data, such as a password, then the recipient uses the private key to decrypt the data. The public and private keys are known as a key pair.

To log in to your instance, you must create a key pair, specify the name of the key pair when you launch the instance, and provide the private key when you connect to the instance. On a Linux instance, the public key content is placed in an entry within `~/.ssh/authorized_keys`. This is done at boot time and enables you to securely access your instance using the private key instead of a password.

The appropriate user names for connecting to a newly created Amazon EC2 instance are as follows:

- For an Amazon Linux AMI, the user name is `ec2-user`.
- For a RHEL AMI, the user name is `ec2-user` or `root`.
- For an Ubuntu AMI, the user name is `ubuntu` or `root`.
- For a Centos AMI, the user name is `centos`.
- For a Debian AMI, the user name is `admin` or `root`.
- For a Fedora AMI, the user name is `ec2-user`.
- For a SUSE AMI, the user name is `ec2-user` or `root`.

### Amazon Machine Images (AMI)

How do I create an Amazon Machine Image (AMI) from my EBS-backed EC2 instance?

## Processors and architectures

There are mainly 3 types of processors:

- **Intel** Xeon processors.
- **AMD** EPYC processors.
- AWS **Graviton** processor based on 64-bit Arm architecture.

Additionally, there are multiple GPUs and FPGAs for compute acceleration.

## Storage

### Instance store

The data in an instance store persists only during the lifetime of its associated instance. If an instance reboots (intentionally or unintentionally), data in the instance store persists. However, data in the instance store is lost under any of the following circumstances:

- The underlying disk drive fails
- The instance stops
- The instance terminates

The data is not replicated by default and no snapshot is supported. There are SSD or HDD disks configurations.

The virtual devices for instance store volumes are named as `ephemeral[0-23]`. Instance types that support one instance store volume have `ephemeral0`. Instance types that support two instance store volumes have `ephemeral0` and `ephemeral1`, and so on until `ephemeral23`.

If you create an AMI from an instance, the data on its instance store volumes aren't preserved and aren't present on the instance store volumes of the instances that you launch from the AMI. You can specify instance store volumes for an instance only when you launch it. You can't detach an instance store volume from one instance and attach it to a different instance.

[Amazon EC2 Instance Store](#)

## Amazon EBS

See section [Amazon EBS](#).

## Networking

**Virtual Private Cloud (VPC)** provision a logically isolated cloud where you can launch AWS resources into a virtual network. More information in [Amazon Virtual Private Cloud \(VPC\)](#).

You can use **security groups** and **ACLs** to restrict inbound and outbound traffic. **NAT Gateways** to allow an instance within a private subnet to talk to Internet. You can enable Flow Logs in any of the network interfaces, and allow you to monitor the traffic in and out these interfaces.

Within a VPN, you can add VPC endpoints to provide private and secure connectivity to S3 and DynamoDB.

Shared VPC allows multiple accounts to launch applications into a VPC.

AWS privatelink allows you the ability to have an endpoint from any VPC to share services privately to any VPC and on-premises networks. You can also use AWS privatelinks to exchange data between a VPC and a SaaS solution (for instance: Salesforce, Heroku) using AWS Direct Connect.

When you have many VPCs in your application, you can simplify the network with AWS Transit Gateway. It provides hub and spoke for managing VPCs. You essentially connect each of your VPCs to the AWS Transit Gateway, as well as the AWS Direct connect gateway and the customer gateway, all talking to each other via the AWS Transit Gateway.

If the instance is stopped, its **Elastic IP address** is disassociated from the instance if it is an EC2-Classic instance. Otherwise, if it is an EC2-VPC instance, the Elastic IP address remains associated. By default, all AWS accounts are limited to 5 Elastic IP addresses per region for each AWS account, because public (IPv4) Internet addresses are a scarce public resources. AWS strongly encourages you to use EIP primarily for load balancing use cases, and use DNS hostnames for all other inter-node communication.

If you feel your architecture warrants additional EIPs, you would need to complete the Amazon EC2 Elastic IP Address Request Form and give reasons as to your need for additional addresses.

When you launch an EC2 instance into a default VPC, AWS provides it with public and private DNS hostnames that correspond to the public IPv4 and private IPv4 addresses for the instance.

However, when you launch an instance into a non-default VPC, AWS provides the instance with a private DNS hostname only. New instances will only be provided with public DNS hostname depending on these two DNS attributes: the DNS resolution and DNS hostnames, that you have specified for your VPC, and if your instance has a public IPv4 address.

## 1.4.2 Availability

### Regions and AZs

AWS global infrastructure provides an SLA of 99.99% availability on EC2. See [secGlobalInfrastructure](#)

### Placement groups

Placement groups enable you to influence AWS selection of capacity for member instances, optimizing the experience for a workload. The selection could be to make the instances fall together or fall apart.

**Cluster** placement groups. EC2 places instances closely in order to optimize the performance of inter-instance network traffic. The use case is when you want to minimize the latency among instances.

**Spread** placement groups. EC2 places instances on distinct HW in order to help reduce correlated failures. A use case could be when deploying a NoSQL database cluster in EC2, spread placement will ensure the instances in your cluster are on distinct HW, helping to insulate a single HW failure to a single node.

It is recommended that you launch the number of instances that you need in the placement group in a single launch request and that you use the same instance type for all instances in the placement group. If you try to add more instances to the placement group later, or if you try to launch more than one instance type in the placement group, you increase your chances of getting an insufficient capacity error.

If you stop an instance in a placement group and then start it again, it still runs in the placement group. However, the start fails if there isn't enough capacity for the instance.

If you receive a capacity error when launching an instance in a placement group that already has running instances, stop and start all of the instances in the placement group, and try the launch again. Restarting the instances may migrate them to hardware that has capacity for all the requested instances.

### Elastic Load Balancing

A Load Balancer is used to route incoming requests to multiple Amazon EC2 instances, containers, or IP addresses in your VPC. Elastic Load Balancing provides HA by utilizing multiple AZs.

### Auto Scaling

Amazon EC2 Auto scaling dynamically react to changing demans, optimizing cost.

### Fleet management

A common use case is to put the EC2 instances in an auto scaling group that allows to have a health check. If one of the health checks fail, it automatically brings up a new instance to replace it.

### Dynamic scaling

Another common use case is via an scaling policy that is monitoring a parameter (such as CPU utilization). If it detects a spike, it brings additional instances onboard and it will terminate those when that spike subsides.

Predictive scaling looks at the patterns of application cycles on your application and the set of applications that run on AWS and uses machine learning techniques to predict when you are going to need to scale ahead of demanda and when you need to scale down ahead of seeing drops in demand.



### 1.4.3 Management

#### Deployment

##### Launch templates

When you launch an instance you can specify a lot of parameters: Instance type, EBS volume, AMI ID, Network interface, tags, user data, block device mapping, placement. Some of them are mandatory and others are not.

You can encapsulate all these parameters in a template, called **Launch template**. These templates can be useful to ensure a *consistent experience* in the organization. You can define *simple permissions*: the EC2 instances you want to launch, what are the AMIs I want them to use, what are the subnets and security group rules, and you can prevent to launch anything outside this template. Launch templates provides you with the ability to define *governance and best practices*, for instance, you can choose what can be overridden. These templates increase productivity, a common use case is using it in conjunction with Auto Scaling groups.

##### Launching Amazon EC2 instances with user data

Instance metadata is the data about your instance that you can use to configure or manage the running instance. You can get the instance ID, public keys, public IP address and many other information from the instance metadata by firing a URL command in your instance to this URL:

<http://169.254.169.254/latest/meta-data/>

Instance Metadata and User Data

#### Administration

##### AWS Systems Manager

AWS Systems Manager allows you to operate cloud and on-premises Linux and Windows workloads safely at scale in the following way:

- Stay patch and configuration compliant.
- Automate across accounts and regions.
- Connect to EC2 instances via browser and CLI.
- Track SW inventory across accounts.
- Install agents safely across instances with rate control.

You can use Run Command from the console to configure instances without having to login to each instance. AWS Systems Manager Run Command lets you remotely and securely manage the configuration of your managed instances. A managed instance is any Amazon EC2 instance or on-premises machine in your hybrid environment that has been configured for Systems Manager. Run Command enables you to automate common administrative tasks and perform ad hoc configuration changes at scale. You can use Run Command from the AWS console, the AWS Command Line Interface, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.

##### AWS Resource Access Manager

AWS Resource Access Manager allows you to securely share AWS resources with other accounts or AWS organizations. It offers the following features:

- Reduces need to provision duplicate resources.
- Efficiently uses resources across different departments.
- AWS Identity and Access Management policies govern consumption of shared resources.
- Integration with Amazon CloudWatch and AWS CloudTrail.
- Supports resource sharing for License Manager Configs, Route 53 Resolver Rules, Subnets, and Transit Gateways.

### AWS License Manager

AWS License Manager offers a simplified license management for on premises and cloud (even if it is AWS). It offers the following features:

- More easily manage licenses from software vendors (SAP, Windows, Oracle).
- Define licensing rules, discover usage, manage access.
- Gain single view of license across AWS and on-premises.
- Discover non-compliant software and help prevent misuse.
- Seamless integration with AWS Systems Manager and AWS Organizations.
- Free service for all customers.

### Monitoring

### Troubleshooting

#### Troubleshooting Instance Launch Issues

##### Instance Terminates Immediately

**Description:** Your instance goes from the pending state to the terminated state immediately after restarting it.

**Cause:** The following are a few reasons why an instance might immediately terminate:

- You've reached your EBS volume limit.
- An EBS snapshot is corrupt.
- The root EBS volume is encrypted and you do not have permissions to access the KMS key for decryption.
- The instance store-backed AMI that you used to launch the instance is missing a required part (an image.part.xx file).

### Connecting to Your Instance

#### Error connecting to your instance: Connection timed out

If you try to connect to your instance and get an error message `Network error: Connection timed out` or `Error connecting to [instance], reason: -> Connection timed out: connect`, try the following:

- Check your security group rules. You need a security group rule that allows inbound traffic from your public IPv4 address on the proper port.
- Check the route table for the subnet. You need a route that sends all traffic destined outside the VPC to the internet gateway for the VPC.
- Check the network access control list (ACL) for the subnet. The network ACLs must allow inbound and outbound traffic from your local IP address on the proper port. The default network ACL allows all inbound and outbound traffic.
- If your computer is on a corporate network, ask your network administrator whether the internal firewall allows inbound and outbound traffic from your computer on port 22 (for Linux instances) or port 3389 (for Windows instances).
- Check that your instance has a public IPv4 address. If not, you can associate an Elastic IP address with your instance.
- Check the CPU load on your instance; the server may be overloaded. AWS automatically provides data such as Amazon CloudWatch metrics and instance status, which you can use to see how much CPU load is on your instance and, if necessary, adjust how your loads are handled.

To connect to your instance using an IPv6 address, check the following:

- Your subnet must be associated with a route table that has a route for IPv6 traffic (::/0) to an internet gateway.
- Your security group rules must allow inbound traffic from your local IPv6 address on the proper port (22 for Linux and 3389 for Windows).
- Your network ACL rules must allow inbound and outbound IPv6 traffic.
- If you launched your instance from an older AMI, it may not be configured for DHCPv6 (IPv6 addresses are not automatically recognized on the network interface).
- Your local computer must have an IPv6 address, and must be configured to use IPv6.

### Error: unable to load key ... Expecting: ANY PRIVATE KEY

If you try to connect to your instance and get the error message, `unable to load key ... Expecting: ANY PRIVATE KEY`, the file in which the private key is stored is incorrectly configured. If the private key file ends in `.pem`, it might still be incorrectly configured. A possible cause for an incorrectly configured private key file is a missing certificate.

### Error: User key not recognized by server

If you use PuTTY to connect to your instance:

- Verify that your private key (`.pem`) file has been converted to the format recognized by PuTTY (`.ppk`).
- Verify that you are connecting with the appropriate user name for your AMI. See section [Amazon Machine Images \(AMIs\)](#).
- Verify that you have an inbound security group rule to allow inbound traffic to the appropriate port.

### Error: Host key not found, Permission denied (publickey), or Authentication failed, permission denied

If you connect to your instance using SSH and get any of the following errors, `Host key not found in [directory]`, `Permission denied (publickey)`, or `Authentication failed, permission denied`

denied, verify that you are connecting with the appropriate user name for your AMI and that you have specified the proper private key (.pem) file for your instance. See section [Amazon Machine Images \(AMIs\)](#).

### Error: Unprotected Private Key File

Your private key file must be protected from read and write operations from any other users. If your private key can be read or written to by anyone but you, then SSH ignores your key and you see the following warning message below.

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0777 for '.ssh/my_private_key.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
bad permissions: ignore key: .ssh/my_private_key.pem
Permission denied (publickey).

```

If you see a similar message when you try to log in to your instance, examine the first line of the error message to verify that you are using the correct public key for your instance. The above example uses the private key `.ssh/my_private_key.pem` with file permissions of `0777`, which allow anyone to read or write to this file. This permission level is very insecure, and so SSH ignores this key. To fix the error, execute the following command, substituting the path for your private key file.

```
[ec2-user ~]$ chmod 0400 .ssh/my_private_key.pem
```

### Error: Private key must begin with “—BEGIN RSA PRIVATE KEY—” and end with “—END RSA PRIVATE KEY—”

If you use a third-party tool, such as `ssh-keygen`, to create an RSA key pair, it generates the private key in the OpenSSH key format. When you connect to your instance, if you use the private key in the OpenSSH format to decrypt the password, you'll get the error Private key must begin with `"-----BEGIN RSA PRIVATE KEY-----"` and end with `"-----END RSA PRIVATE KEY-----"`.

To resolve the error, the private key must be in the PEM format. Use the following command to create the private key in the PEM format:

```
ssh-keygen -m PEM
```

### Error: Server refused our key or No supported authentication methods available

If you use PuTTY to connect to your instance and get either of the following errors, `Error: Server refused our key` or `Error: No supported authentication methods available`, verify that you are connecting with the appropriate user name for your AMI. See section [Amazon Machine Images \(AMIs\)](#).

You should also verify that your private key (.pem) file has been correctly converted to the format recognized by PuTTY (.ppk).

## 1.4.4 Pricing options

AWS offers 3 core purchasing options: On-Demand, Reserved Instances, Spot Instances. Each purchasing model launches the same underlying EC2 instances.

Using **On-demand Instances** is often where customers begin their Amazon EC2 journey, because they need to define their needs and support spiky workloads.

Then, once customers have identified what is steady state and what is predictable, **Reserved Instances** come into play. Reserved Instances are instances that require a 1 to 3-year commitment, and in exchange, customers get a significant discount off of On-Demand prices. This is ideal for customers' committed and more predictable, steady state use.

**Spot instances** are the most inexpensive and flexible way to access Amazon EC2 instances.

With all these pricing models, the key is striking a balance. Use RIs for known, steady-state, predictable or always-on workloads. On-Demand, for unknown spiky workloads. Scale using Spot Instances for fault-tolerant, flexible, stateless workloads.

## Cost Factors

To estimate the cost of using EC2, you need to consider the following:

- **Clock seconds/hours of server Time.** Resources incur charges when they are running. For example, from the time EC2 instances are launched until they are terminated, or from the time elastic IPs are allocated until the time they are de-allocated.
- **Instance configuration.** Consider the physical capacity of the EC2 instance you choose. Instance pricing varies with the AWS region, OS, instance type and instance size.
- **Number of instances.** You can provision multiple instances to handle peak loads.
- **Load balancing.** An elastic load balancer can be used to distribute traffic among EC2 instances. The number of hours the ELB runs and the amount of data it processes contribute to the monthly cost.
- **Data transfer.** Data transferred between Amazon S3, Amazon Glacier, Amazon DynamoDB, Amazon SES, Amazon SQS, Amazon Kinesis, Amazon ECR, Amazon SNS or Amazon SimpleDB and Amazon EC2 instances in the same AWS Region is free. AWS Services accessed via PrivateLink endpoints will incur standard PrivateLink charges as explained here.

The following illustration represents the transitions between instance states.

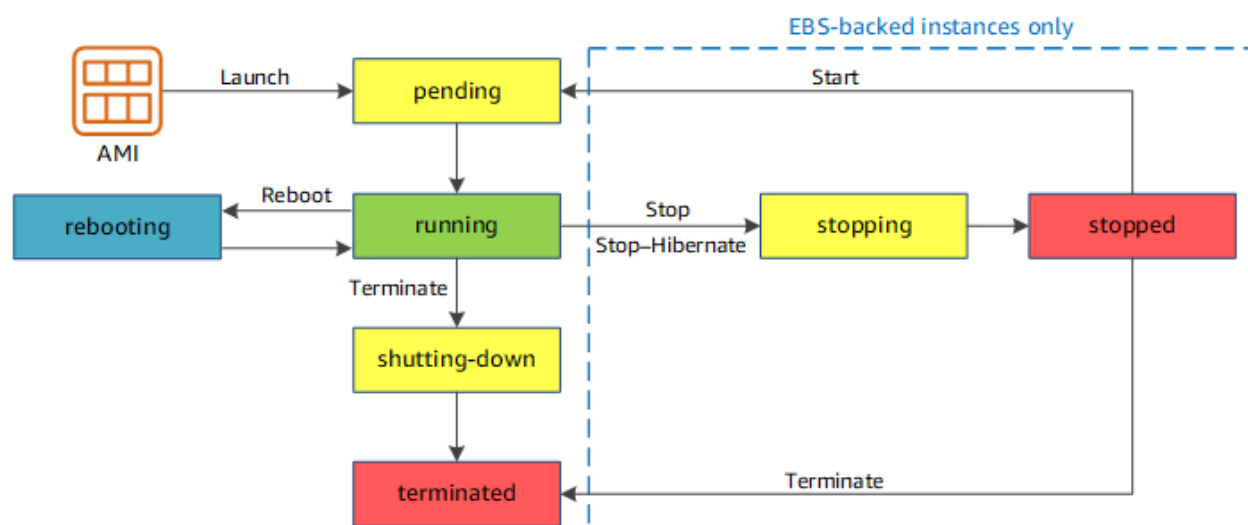


Fig. 31: EC2 instance lifecycle

Below are the valid EC2 lifecycle instance states:

- **pending.** The instance is preparing to enter the running state. An instance enters the pending state when it launches for the first time, or when it is restarted after being in the stopped state.
- **running.** The instance is running and ready for use.
- **stopping.** The instance is preparing to be stopped. Take note that you will not be billed if it is preparing to stop however, you will still be billed if it is just preparing to hibernate.
- **stopped.** The instance is shut down and cannot be used. The instance can be restarted at any time.
- **shutting-down.** The instance is preparing to be terminated.
- **terminated.** The instance has been permanently deleted and cannot be restarted. Take note that Reserved Instances that applied to terminated instances are still billed until the end of their term according to their payment option.

### Instance Lifecycle

The product options are the following:

- **Detailed monitoring.** You can use Amazon CloudWatch to monitor your EC2 instances. By default, basic monitoring is enabled and available at no additional cost. However, for a fixed monthly rate, you can opt for detailed monitoring, which includes 7 preselected metrics recorded once a minute. Partial months are charged on an hourly prorated basis, at a per instance-hour rate.
- **Auto scaling** automatically adjusts the number of EC2 instances in your deployment according to conditions you define. This service is available at no additional charge beyond CloudWatch fees.
- **Elastic IP addresses.** An Elastic IP address doesn't incur charges as long as the following conditions are true:
  - The Elastic IP address is associated with an Amazon EC2 instance.
  - The instance associated with the Elastic IP address is running.
  - The instance has only one Elastic IP address attached to it.

If you've stopped or terminated an EC2 instance with an associated Elastic IP address and you don't need that Elastic IP address anymore, consider disassociating or releasing the Elastic IP address.

Operating systems and Software packages:

- **Operating system** prices are included in the instance prices.
- **Software packages.** AWS has partnerships with Microsoft, IBM, etc. to simplify running certain commercial software packages on your EC2 instances, for example: MS SQL Server on Windows. For commercial software packages that AWS does not provide, such as nonstandard OS, Oracle applications, Windows Server applications such as MS SharePoint and MS Exchange, you need to obtain a license from the vendors. You can bring your existing license to the cloud through specific vendor programs such as Microsoft License Mobility through Software Assurance Program.

### How AWS Pricing Works

#### AWS Free Tier

## Reserved Instances

### Introduction to Amazon EC2 Reserved Instances

#### Amazon EC2 Reserved Instances and Other Reservation Models

Using Reserved Instances can have a significant impact on savings compared to on-demand, in some cases up to 75%. Typically, Reserved Instances are used for workloads that need to run most or all of the time, such as production environments. The commitment level could be 1 year or 3 years. AWS services offering RIs are: Amazon EC2, ECS, RDS, DynamoDB, Redshift, ElastiCache, Reserved Transcode Slots and Reserved Queues (AWS Elemental

MediaConvert). It offers payment flexibility with 3 upfront payment options (all, partial, none). RI types are Standard, Convertible and Scheduled.

While using RIs, in certain cases, customers can take advantage of regional benefits. Regional benefits can simplify reserved instance optimization by allowing a reserved instance to be applied for the whole AWS Region, rather than just a specific Availability Zone, which can simplify capacity planning.

Characteristic	Availability Zone	Region
<b>Capacity reservation</b> Capacity reservation in a specific AZ.	Yes	No
<b>Availability Zone flexibility</b> The Reserved Instance discount applies to instance usage in any Availability Zone in a Region.	No	Yes
<b>Instance size flexibility</b> The Reserved Instance discount applies to instance usage regardless of size, within the instance family. Only supported on Linux/Unix Reserved Instances with default tenancy.	No	Yes

Fig. 32: Regional RIs simplify optimization

AWS Cost Explorer generates RI recommendations for AWS services including Amazon EC2, RDS, ElastiCache and Elasticsearch. You can use the *Recommendations* feature to perform “what-if” scenarios comparing costs and savings related to different RI types (standard versus convertible RIs), and RIs term lengths (1 versus 3 years).

Customers can combine regional RIs with on-demand capacity reservations to benefit from billing discounts. On-demand capacity reservations means:

- Reserving capacity for Amazon EC2 instances in a specific Availability Zone for any duration. This ensures access to EC2 capacity when needed, for as long as needed.
- Capacity reservations can be created at any time, without entering into a 1-year or 3-year term commitment, and the capacity is available immediately.
- Capacity reservations can be cancelled at anytime to stop incurring charges.

Capacity reservation is charged the equivalent on-demand rate, regardless of whether the instances are run. Customers can combine regional RIs with capacity reservations to get billing discounts. If customers do not use a reservation, it is shown as an unused reservation on the customer’s EC2 bill.

Zonal RI billing discounts do not apply to capacity reservations. Capacity reservations can’t be created in placement groups. Capacity reservations can’t be used with dedicated hosts.

## Convertible Reserved Instances

Convertible RIs give customers the ability to modify reservations across families, sizes, operating system, and tenancy. The only aspect customer cannot modify is the Region. So, as long as the customer stays in the same Region, they can continue to modify the RIs. Convertibles give customers the opportunity to maximize flexibility and increase savings.

The only time customers cannot convert RIs is between the time the request to exchange is submitted and the time the request to exchange is fulfilled. Typically requests take only a matter of hours to fulfill but could take a up to 2 days.

Characteristic	Standard	Convertible
Terms (average discount off On-Demand)	1 year (40%) 3 year (60%)	1 year (31%) 3 year (54%)
Change Availability Zone, instance size (for Linux OS), networking type	Yes	Yes
Change instance families, operating system, tenancy, and payment option	No	Yes
Benefits from price reduction		Yes
Sellable on the Reserved Instance Marketplace	Yes	No

Fig. 33: Standard and convertible RI payments

Some guidelines for exchanging convertible RIs are the following:

- Customers can exchange to the same value or higher of convertible RIs.
- Converted RIs retain the expiration data of the original RIs.
- Converted RIs have the same term as the original RIs.
- When exchanging a group of convertible RIs:
  - Converted RIs have the latest expiration data of the whole group.
  - In the case of multiple terms, converted RIs will be a 3-year RIs.

For complete set of conversion rules, see [Exchanging Convertible Reserved Instances](#).

## Scheduled Reserved Instances

Scheduled Reserved Instances (Scheduled Instances) enable you to purchase capacity reservations that recur on a daily, weekly, or monthly basis, with a specified start time and duration, for a one-year term. You reserve the capacity in advance, so that you know it is available when you need it. You pay for the time that the instances are scheduled, even if you do not use them.

Scheduled Instances are a good choice for workloads that do not run continuously, but do run on a regular schedule. For example, you can use Scheduled Instances for an application that runs during business hours or for batch processing that runs at the end of the week.

## Spot Instances

Spot is spare, on-demand capacity that is available for discounts of up to 90% off On-Demand prices. Some of the differences with Spot compared to Reserved Instances and On-Demand Instances is the deep discount, no commitment requirement, and customers can pay for Linux instances by the second and Windows instances by the hour. One last key difference with Spot is spare, on-demand capacity. If AWS has a spike in requests in the on-demand space, AWS reclaims Spot instances with a 2-minute notification. The best workloads for Spot instances are fault-tolerant, flexible, and stateless. With Amazon EC2 instances, there are 3 simple rules to remember:



**Scheduled Instances Reservation Wizard**

Step 1: Find a schedule  
Step 2: Review and purchase

**Find available schedules**

Scheduled instances allow you to purchase recurring Amazon EC2 compute capacity by the hour for a one-year term. To find a Scheduled instance offering, specify the date, duration, frequency, and compute capacity requirements for your application. Click "Find schedules" to view a list of available schedules, and choose one or more schedules to purchase by adding them to your cart.

**Create a schedule**

Starting on: Mon, 1/4/2016 16:00 UTC ☐ +/- 2 hours

for duration: 8 hours

Recurring: Weekly

on days: Monday, Wednesday, Friday

**Instance details**

Platform: Linux/UNIX (Amazon VPC)

Instance type: c3.4xlarge

Availability Zone: Any

**Find schedules**

Fig. 34: Scheduled Reserved Instances

1. **Spot infrastructure**, or Spot Instances, are the exact same instances that customers would purchase with on-demand and RIs. The only difference in terms of the price points and the fact that it can be reclaimed by AWS. But otherwise, it functions the exact same way as on-demand instances.
2. **Spot pricing** is set based on long-term trends and supply and demand. This is typically an average discount of 70-90% off the on-demand price point. AWS eliminated the bidding model in 2017 in order to simplify the access model for customers and not require them to worry about pricing strategy anymore. This change has made things much simpler for the customer. To get Spot instances, customers simply request them, and if they're available, they will pay the current market rate, and they will hold on to them unless AWS needs to reclaim them for capacity reasons. There is no need to stress over situations where other customers can reclaim them because they were willing to bid or pay more for the instances. The price point is a lot smoother, so customers no longer have lots of fluctuation throughout the day. Prices can fluctuate slowly over time, but customers can take a look at the 90-day price history API and see that the price points are very stable and much more predictable.
3. For customers to **diversify** their instance fleet, is especially important when it comes to overall Spot capacity availability. Diversifying is having the flexibility to use multiple instance types and Availability Zones for their workloads. The importance of flexibility is that Spot is spare on-demand capacity; so there may be times when there is a spike in demand, for particular instance type, and those instances may become unavailable on Spot. But if the customer has flexibility and have specified additional capacity pools, then it just increases the total pool of available Spot capacity that is available for their requests. This increases the likelihood that the requested capacity will be fulfilled. If there is a spike in demand for a particular instance, and AWS has to reclaim some of those instances, it minimizes the overall impact of losing some of those instances.

Interruptions are important to understand when it comes to Spot, because Spot is an interruptible product. Over 95% of the instances were not interrupted in the last 3 months. The workloads on Spot should be stateless, fault tolerant, loose coupled and flexible. Any application that can have part or all the work paused and resumed or restarted can use Spot. Anything containerized is generally a good target workload for Spot. But more specifically, other areas where there is a lot of adoption is big data analysis, CI/CD, web services, and HPC.

What happens when AWS needs to reclaim an instance is that they will give you a 2-minute warning, either through a CloudWatch event, or customers can pull the metadata on the local instance and then they will have 2 minutes to take

action and gracefully move off of the instance. There are different strategies that can be taken, for instance:

- Implementing a check-pointing strategy so that if an instance is interrupted, customers won't have to start the job over from scratch.
- AWS can provide example scripts triggering a Lambda function when the CloudWatch event is received, to automatically bring the workload up on another instance in their fleet. You can see [AWS Instance Scheduler](#) for more information.
- AWS also has capabilities called stop-start and hibernate. Stop-start means customers would be able to persist an EBS volume if an instance is interrupted and when that instance becomes available again, it will re-attach to that EBS volume and continue on with the work where the customer left off. Hibernate takes that a step further and allows customers to flush in-state memory to disk.
- Spot blocks, which allows you reserve spot instances up to 6 hours in the spot market.

In 2018, AWS announced the integration of EC2 fleet with EC2 Auto Scaling. This means customers can now launch a single auto scaling group. This includes a mix of all the Spot instances that will work for customers across all of these, plus their on-demand instances and RIs in a single auto scaling group. Customers can set different target capacities for what their requirements are and it will scale amongst that.

With the integration of EC2 fleet, customers also get all the benefits of fleet, such as being able to automatically replace a Spot instance, if it is interrupted, with another instance in the fleet, or taking advantage of different strategies within the fleet, such as launching in the cheapest capacity pools or diversifying across all the Spot instances that they have specified.

The integration of EC2 Auto Scaling and EC2 fleet helps customers to drive down costs, optimize performance, and eliminate operational overhead.

You can choose to have your Spot instances terminated, stopped, or hibernated upon interruption. Stop and hibernate options are available for persistent Spot requests and Spot Fleets with the maintain option enabled. By default, your one-time Spot instances are terminated.

Amazon EC2 Spot instances integrate natively with a number of other AWS services, such as: AWS Batch, Data Pipeline and CloudFormation, Amazon EMR, ECS and EKS.

To use Spot Instances, you create a Spot Instance request that includes the number of instances, the instance type, the Availability Zone, and the maximum price that you are willing to pay per instance hour. If your maximum price exceeds the current Spot price, Amazon EC2 fulfills your request immediately if capacity is available. Otherwise, Amazon EC2 waits until your request can be fulfilled or until you cancel the request.

You can specify whether Amazon EC2 should hibernate, stop, or terminate Spot Instances when they are interrupted. You can choose the interruption behavior that meets your needs.

Take note that there is no "bid price" anymore for Spot EC2 instances since March 2018. You simply have to set your maximum price instead.

If your Spot instance is terminated or stopped by Amazon EC2 in the first instance hour, you will not be charged for that usage. However, if you terminate the instance yourself, you will be charged to the nearest second. If the Spot instance is terminated or stopped by Amazon EC2 in any subsequent hour, you will be charged for your usage to the nearest second. If you are running on Windows and you terminate the instance yourself, you will be charged for an entire hour.

[Spot Instance Advisor](#)

[Amazon EC2 Spot Instances Pricing](#)

[Amazon EC2 Spot Instances workshops website](#)

[New -? Hibernate Your EC2 Instances](#)

[AWS ANZ Webinar Series - Spot Instances: Benefits and Best Practices Explained](#)

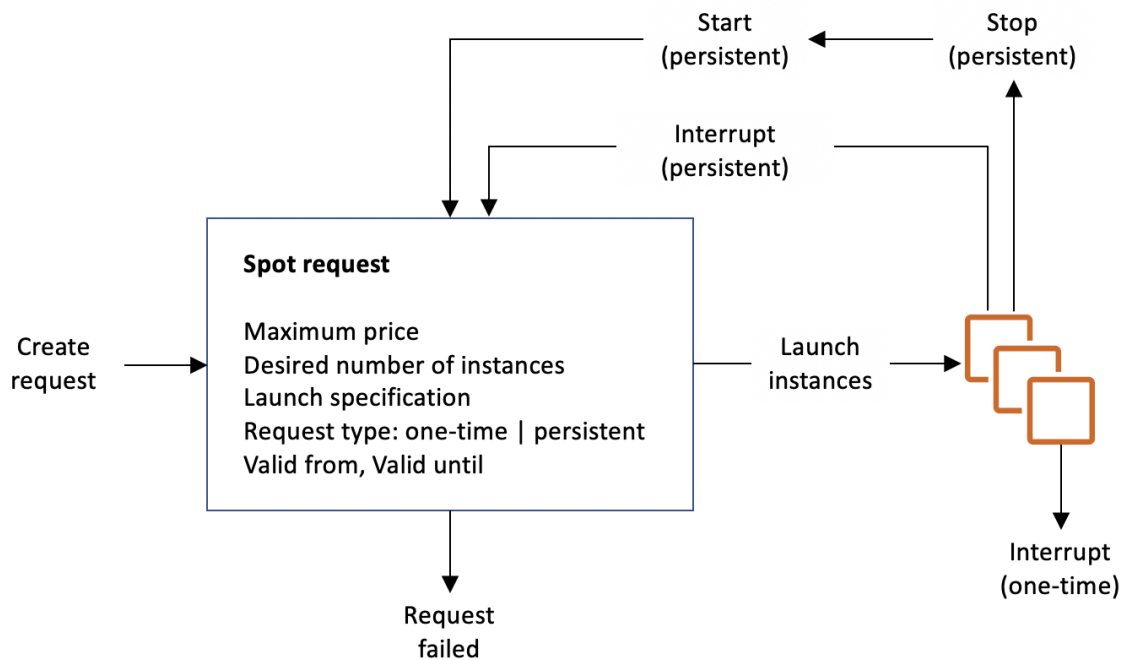


Fig. 35: Spot instance lifecycle

## Amazon EC2 fleet

Amazon EC2 Fleet is a new feature that simplifies the provisioning of Amazon EC2 capacity across different Amazon EC2 instance types, Availability Zones and across On-Demand, Amazon EC2 Reserved Instances (RI) and Amazon EC2 Spot purchase models. With a single API call, you can provision capacity across EC2 instance types and across purchase models to achieve desired scale, performance and cost.

It uses all 3 purchase options to optimize costs. It is integrated with Amazon EC2 Auto Scaling, Amazon ECS, Amazon EKS, and AWS Batch.

## Amazon EC2 dedicated options

Amazon EC2 Dedicated Hosts

Introducing AWS License Manager

Changing the Tenancy of an Instance

## AWS tagging strategies

AWS Tagging Strategies

- **Cost Allocation Tags** only eases the organization of your resource costs on your cost allocation report, to make it easier for you to categorize and track your AWS costs.

AWS re:Invent 2018: [REPEAT 1] Amazon EC2 Foundations (CMP208-R1)

## 1.4.5 Considerations

Amazon EC2 has a soft limit of 20 instances per region, which can be easily resolved by completing the Amazon EC2 instance request form where your use case and your instance increase will be considered. Limit increases are tied to the region they were requested for.

### Resource Locations

For all new AWS accounts, 20 instances are allowed per region. However, you can increase this limit by requesting it via AWS support.

Instances within a VPC with a public address have that address released when it is stopped and are reassigned a new IP when restarted.

All EC2 instances in the default VPC have both a public and private IP address.

Table 2: EC2-Classic vs Default VPC vs Nondefault VPC

Characteristic	EC2-Classic	Default VPC	Nondefault VPC
Public IPv4 address (from Amazon's public IP address pool)	Your instance receives a public IPv4 address from the EC2-Classic public IPv4 address pool.	Your instance launched in a default subnet receives a public IPv4 address by default, unless you specify otherwise during launch, or you modify the subnet's public IPv4 address attribute.	Your instance doesn't receive a public IPv4 address by default, unless you specify otherwise during launch, or you modify the subnet's public IPv4 address attribute.
Private IPv4 address	Your instance receives a private IPv4 address from the EC2-Classic range each time it's started.	Your instance receives a static private IPv4 address from the address range of your default VPC.	Your instance receives a static private IPv4 address from the address range of your VPC.
Multiple private IPv4 addresses	We select a single private IP address for your instance; multiple IP addresses are not supported.	You can assign multiple private IPv4 addresses to your instance.	You can assign multiple private IPv4 addresses to your instance.
Elastic IP address (IPv4)	An Elastic IP is disassociated from your instance when you stop it.	An Elastic IP remains associated with your instance when you stop it.	An Elastic IP remains associated with your instance when you stop it.

## 1.5 Database services

### 1.5.1 Amazon RDS

Amazon Relational Database Service (RDS) is a managed relational database with a choice of 6 popular database engines: Amazon Aurora, MySQL, PostgreSQL, MariaDB, Microsoft SQL Server and Oracle. This service has the following characteristics:

- **Easy to administer.** As Amazon RDS is a managed service, you manage application optimization tasks (schema design, query construction, query optimization) and AWS manages infrastructure tasks (power, HVAC, network, rack & stack, installation, routine maintenance, automated patching, scaling and monitoring, industry compliance, isolation & security, backup & recovery, High Availability).
- **Available and durable.** It provides automatic Multi-AZ data replication, automated backup, snapshots, and failover.
- **Highly scalable.** It scales DB compute and storage with a few clicks with no application downtime.

- **Fast and secure.** It provides SSD storage and guaranteed provisioned I/O; data encryption at rest and in transit.

### Amazon Relational Database Service (Amazon RDS)

#### Use cases

Amazon RDS is ideal for:

- **Web and mobile applications** that need a DB with high throughput, massive storage scalability, and HA. Since RDS does not have any licensing constraints, it perfectly fits the variable usage pattern of these applications.
- For **E-commerce applications**, RDS provides a flexible, secured, and low-cost DB solution for online sales and retailing.
- **Mobile and online games** require a DB platform with high throughput and HA. RDS manages the DB infrastructure so game developers don't have to worry about provisioning, scaling or monitoring DB servers.

#### Get started with RDS

#### DB instances

The basic building block of Amazon RDS is the DB instance. A DB instance is an isolated DB environment that can contain multiple user-created DBs and can be accessed by using the same tools and applications that you use with an standalone DB instance. The resources found in a DB instance are determined by its DB instance class, and the type of storage is dictated by the type of disks. DB instances and storage differ in performance characteristics and price, allowing you to tailor your performance and cost to the needs of your DB. When you choose to create a DB instance, you first have to specify which **DB engine** to run. Amazon RDS currently supports 2 commercial engines (MS SQL Server and Oracle), 3 open source (MySQL, PostgreSQL, and MariaDB) and 1 cloud native (Amazon Aurora).

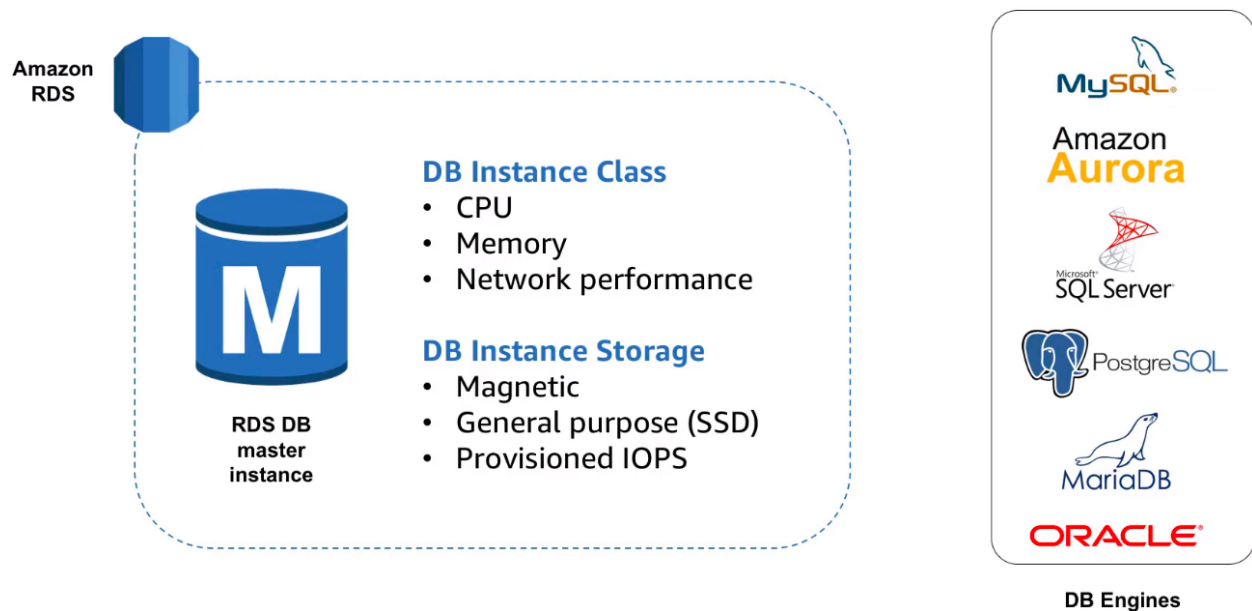


Fig. 36: Amazon RDS DB instance

DB instances with commercial and open source DB engines use Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage. Depending on the amount of storage requested, Amazon RDS automatically stripes

across multiple Amazon EBS volumes to enhance performance. Amazon Aurora, which is MySQL and PostgreSQL compatible, uses purpose-built distributed, log-structured storage system.

If you are developing a new application, then it is recommended to go with the newest DB engine version. If you are migrating an existing application, then it is necessary to check compatibility requirements.

You can run Amazon RDS on VMware through the Amazon RDS connector. It packages technologies at the core of Amazon RDS in AWS and deploys them on premises. You need connectivity to AWS via a dedicated VPN tunnel enables database management entirely within your private data center.

You manage your DB engine configuration through the use of parameters in a DB parameter group. DB parameter groups act as a container for engine configuration values that are applied to one or more DB instances.

### Instance types

Amazon RDS provides a selection of instance types optimized to fit different relational database use cases. Each instance type includes several instance sizes, allowing you to scale your database to the requirements of your target workload. Not every instance type is supported for every database engine, version, edition or region. The current Amazon RDS Instance Types are:

- General purpose: T3, T2, M5, M4.
- Memory optimized: R5, R4, X1e, X1, Z1d.

You can scale compute and memory vertically up or down. The new host is attaches to existing storage with minimal downtime. The endpoint will be the same, but you will have to reconnect to the new instance.

### Storage types

Amazon RDS provides three storage types: General Purpose SSD (also known as gp2), Provisioned IOPS SSD (also known as io1), and magnetic. The following list briefly describes the three storage types:

- **General Purpose SSD.** General Purpose SSD volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time. Baseline performance for these volumes is determined by the volume's size.
- **Provisioned IOPS.** Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that require low I/O latency and consistent I/O throughput.
- **Magnetic.** Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose SSD or Provisioned IOPS for any new storage needs. The maximum amount of storage allowed for DB instances on magnetic storage is less than that of the other storage types.

General Purpose SSD is a great choice, but you have to be aware of burst credits on volumes less than 1 TB. Hitting credit-depletion results in IOPS drop, latency and queue depth metrics will spike until credits are replenished. You have to monitor `BurstBalance` to see the percent of burst-bucket I/O credits available. You have to monitor read/write IOPS to see if average IOPS is greater than the baseline. You can think General Purpose SSD burst rate and PIOPS stated rate as maximum I/O rates.

You can scale up Amazon EBS storage with no downtime. Initial scaling operation may take longer, because storage is reconfigured on older instances. You can re-provision IOPS on the fly without any disruption in the storage. You are limited to scale up to once every 6 hours.

### Managing High Availability, read replicas, and backups

## High Availability with Multi-AZ

Multi-AZ provides enterprise-grade fault-tolerance solution for production databases. Once configured, Amazon RDS automatically generates a standby copy of the primary instance in another AZ within the same Amazon VPC. Each DB instance manages a set of Amazon EBS volumes with a full copy of the data. After seeding the DB copy, transactions are synchronously replicated to the standby copy. Instances are monitored by an external observer to maintain consensus over quorum.

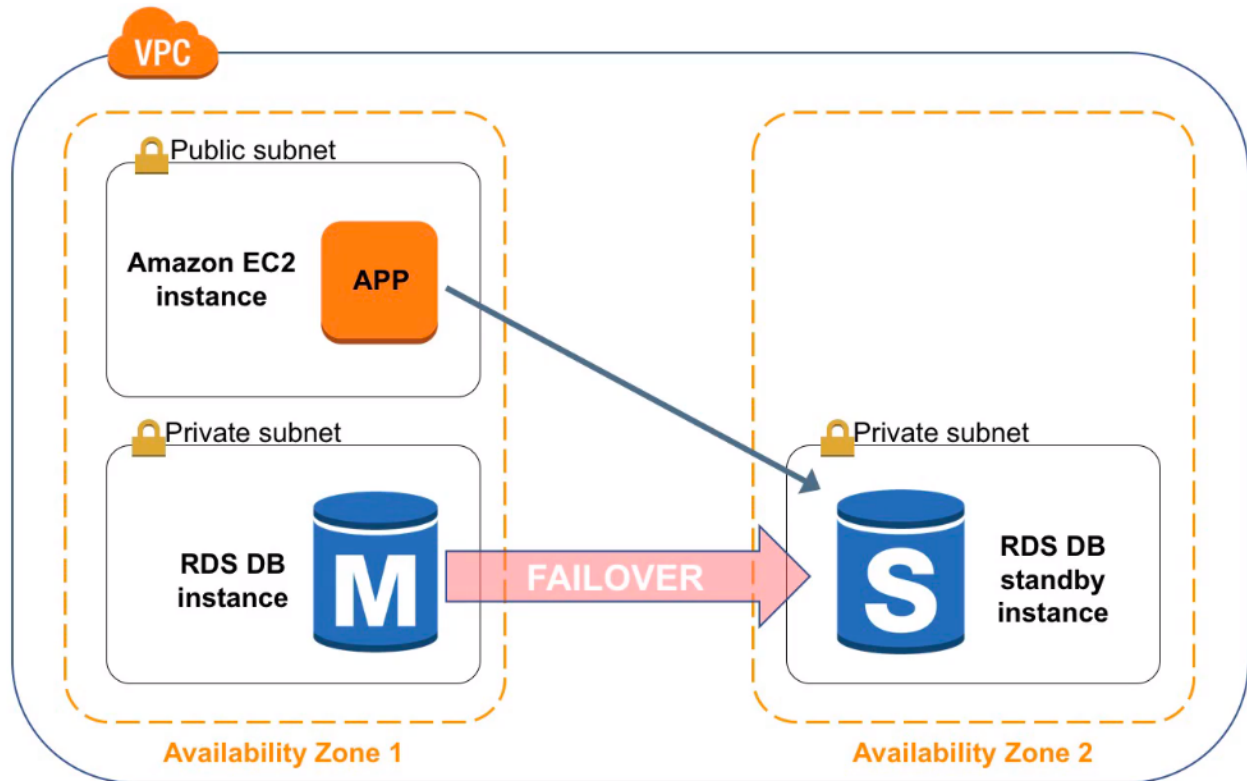


Fig. 37: High Availability with Multi-AZ

In Amazon RDS, failover is automatically handled so that you can resume database operations as quickly as possible without administrative intervention in the event that your primary database instance went down. When failing over, Amazon RDS simply flips the canonical name record (CNAME) for your DB instance to point at the standby, which is in turn promoted to become the new primary.

Running a DB instance with Multi-AZ can enhance availability during planned system maintenance and help protect your DBs against DB instance failure and AZ disruption. If the master DB instance fails, Amazon RDS automatically brings the standby DB online as the new primary. Failover can be initiated by automation or through the Amazon RDS API. Because of the synchronous replication, there should be no data loss. As your applications reference the DB by name using RDS DNS endpoint, you don't need to change anything in your application code to use the standby copy for failover. A new DB instance in the AZ where it was located the failed previous primary DB instance is provisioned as the new secondary DB instance.

Amazon RDS automatically performs a failover in the event of any of the following:

- Loss of availability in primary Availability Zone.
- Loss of network connectivity to primary.
- Compute unit failure on primary.



- Storage failure on primary.

**Note:** It is important to have in mind that it detects infrastructure issues, not database engine problems.

**Important:** The standby instance will not perform any read and write operations while the primary instance is running.

In case of a Single-AZ database failure, a user-initiated point-in-time-restore operation will be required. This operation can take several hours to complete, and any data updates that occurred after the latest restorable time (typically within the last five minutes) will not be available.

### Read scalability with Amazon RDS Read Replicas

Amazon RDS can gain read scalability with the creation of read replicas for MySQL, MariaDB, and PostgreSQL. Updates made to the source DB instance are asynchronously copied to the read replica instance. You can reduce the load on your source DB instance by routing read queries from your applications to the read replica. Using read replicas, you can also scale out beyond the capacity constraints of a single DB instance for read-heavy DB workloads. It brings data close to your applications in different regions.

You can create up to 5 replicas per source database. You can monitor replication lag in Amazon CloudWatch or Amazon RDS console. Read replicas can be created in a different region than the master DB. This feature can help satisfy DR requirements or cutting down on latency by directing reads to a read replica closer to the user. Single-region read replicas is supported for Oracle EE and is coming soon for SQL Server.

Read replicas can also be promoted to become the master DB instance, but due to the asynchronous replication, this requires manual action. You can do it for faster recovery in the event of a disaster. You can upgrade a read replica to a new engine version.

Amazon RDS Read Replicas provide enhanced performance and durability for database (DB) instances. This feature makes it easy to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. You can create one or more replicas of a given source DB Instance and serve high-volume application read traffic from multiple copies of your data, thereby increasing aggregate read throughput.

Table 3: Multi-AZ vs Read Replicas

Multi-AZ	Read Replicas
Synchronous replication: highly durable	Asynchronous replication: highly scalable
Only primary instance is active at any point in time	All replicas are active and can be used for read scaling
Backups can be taken from secondary	No backups configured by default
Always in 2 AZs within a region	It can be within an AZ, cross-AZ, or cross-region
Database engine version upgrades happen on primary	Database engine version upgrades independently from source instance
Automatic failover when a problem is detected	It can be manually promoted to a standalone database

How do I make my RDS MySQL read replica writable?

### Plan for DR

You can plan for disaster recovery by using automated backups and manual snapshots. You can use a cross-region read replica as a standby database for recovery in the event of a disaster. Read replicas can be configured for Multi-AZ to reduce recovery time.



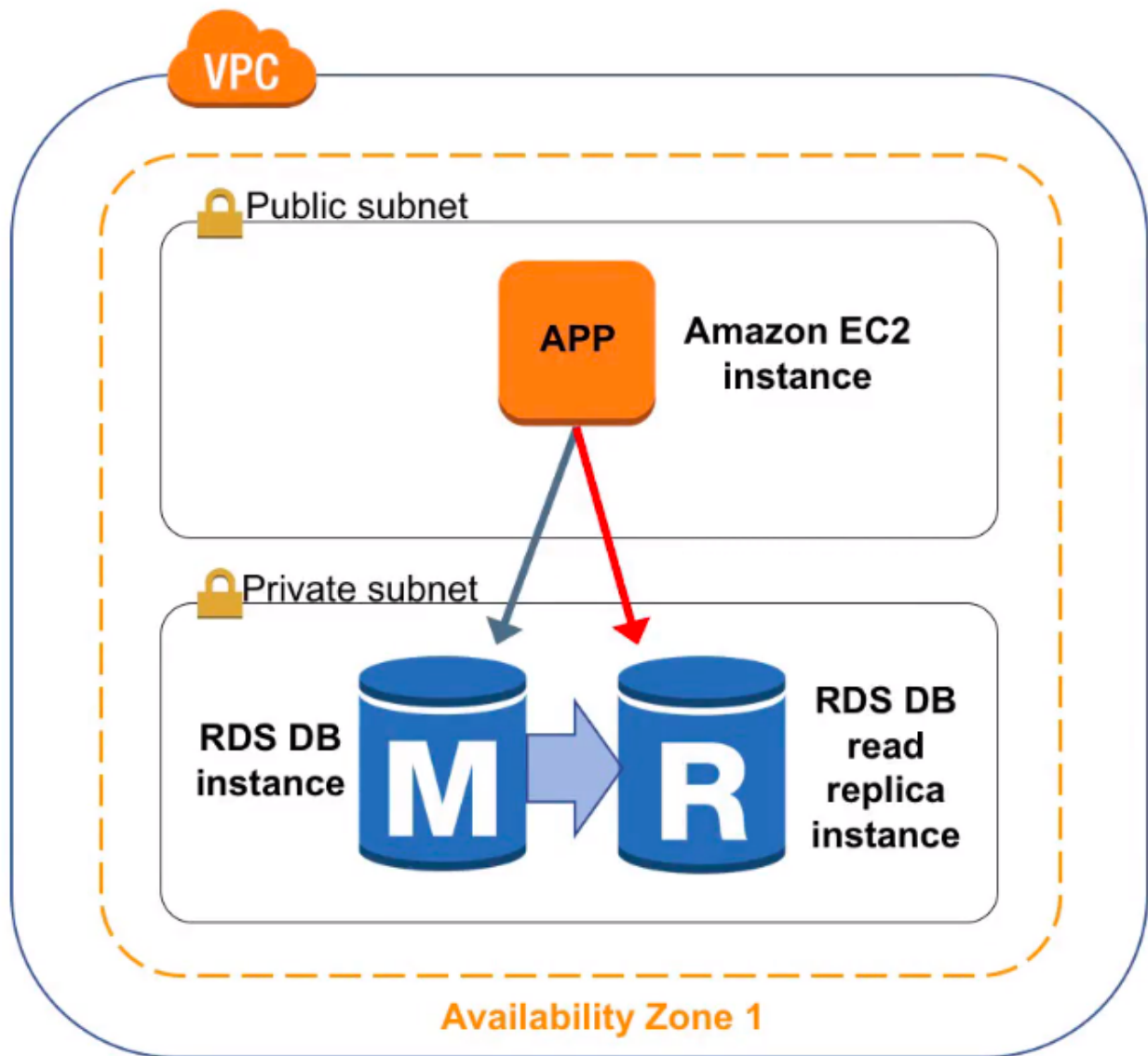


Fig. 38: Amazon RDS read replicas

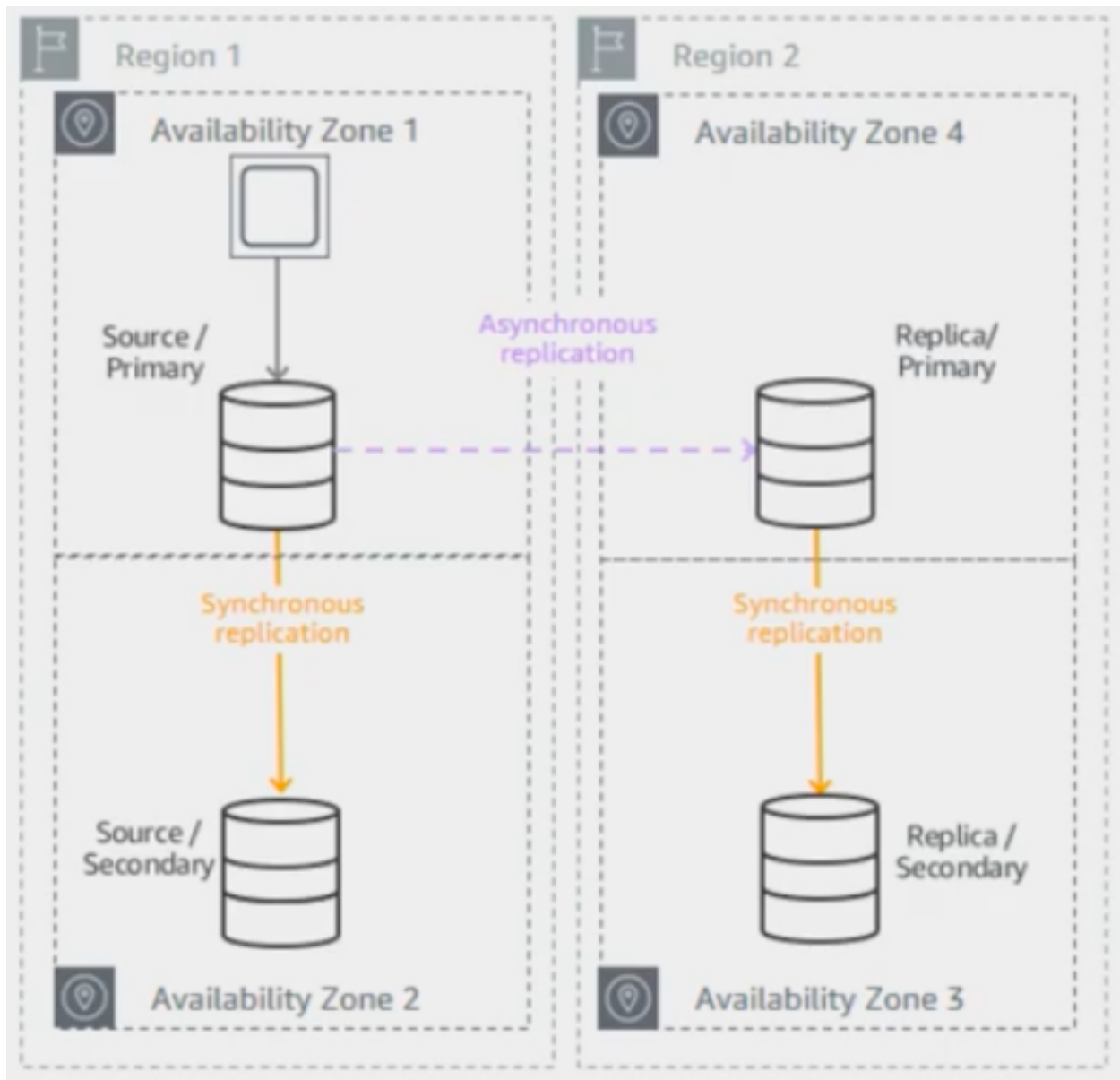


Fig. 39: Disaster recovery with Multi-AZ and read replicas

This architecture is supported for MySQL, MariaDB and PostgreSQL. For Oracle and SQL Server, you can use cross-region backup copies.

**Note:** You can use delayed replication for MySQL to protect from self-inflicted disasters.

## Backups

Amazon RDS can manage backups using one of these two options:

- You can do **manual snapshots**. These are storage level snapshots with no performance penalty for backups in multi-AZ configurations and only a brief pause in you I/O for single-AZ configurations. It leverages EBS snapshots stored in Amazon S3.
- **Automated backups** gives you a point-in-time restore capability. AWS will take snapshots once a day and capture transactions logs and store them every 5 minutes in S3.

Snapshots can be copied across regions or shared with other accounts.

Table 4: Automated backups vs Manual snapshots

Automated backups	Manual snapshots
Specify backup retention window per instance (7-day default)	Manually created through AWS console, AWS CLI, or Amazon RDS
Kept until outside of window (35-day maximum) or instance is deleted	Kept until you delete them
Supports PITR	Restores to saved snapshot
Good for DR	Use for checkpoint before making large changes, non-production/test environments, final copy before deleting a database

When you restore a backup, you are creating a entirely new DB instance. In this process, it is defined the instance configuration just like a new instance. It will get the default parameters, security, and option groups.

Restoration can get a long period of time because new volumes are hydrated from Amazon S3. While the volume is usable immediately, full performance requires the volume to be initialized until fully instantiated. One way to mitigate the length of the restoration process is to migrate to a DB instance class with high I/O capacity and later downsizes it. You should maximize I/O during restore process.

## Security controls

Amazon RDS is designed to be secure by default. Network isolation is provided with Amazon VPC. AWS IAM based resource-level permission controls are supported. It provides encryption at rest using AWS KMS (for all engines) or Oracle/Microsoft Transparent Data Encryption (TDE). SSL protection for data in transit is used.

## Identity and Access Management

You can use IAM to control who can perform actions on RDS resources. It is recommended not to use AWS root credentials to manage Amazon RDS resources, you should create an IAM user for everyone, including the administrator. You can use AWS MFA to provide extra level of protection.

## IAM Database Authentication for MySQL and PostgreSQL

You can authenticate to your DB instance using AWS Identity and Access Management (IAM) database authentication. IAM database authentication works with MySQL and PostgreSQL. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token.

An *authentication token* is a unique string of characters that Amazon RDS generates on request. Authentication tokens are generated using AWS Signature Version 4. Each token has a lifetime of 15 minutes. You don't need to store user credentials in the database, because authentication is managed externally using IAM. You can also still use standard database authentication.

**Database options**

**Database port**  
Specify the TCP/IP port that the DB instance will use for application connections. The connection string of any application connecting to the DB instance must specify the port number of the DB instance. Both the security group applied to the DB instance and your company's firewalls must allow connections to the port. [Learn More](#)

3306

**DB parameter group**  
Database parameter group to associate with this DB instance

default.mysql5.7

**Option group**  
Name of an option group that contains options (e.g. Memcached, Oracle Enterprise Manager) you want attached to this DB instance. If there aren't any option groups compatible with the selected engine, a default option group will be created at launch.

default:mysql-5-7

**IAM DB authentication** [Info](#)

☐ Enable IAM DB authentication  
Manage your database user credentials through AWS IAM users and roles.

☒ Disable

Fig. 40: Database options

For enabling RDS IAM authentication, you should check the “Enable IAM DB authentication” option on RDS modify or create phase.

After this step, you should create a user for your database account and use “FLUSH PRIVILEGES;” command in MySQL. The DB user account has to use same name with your IAM account.

### Listing 2: Create user phase for MySQL

```
mysql > CREATE USER myuser IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
Query OK, 0 rows affected (0.18 sec)
```

### Listing 3: Create user phase for PostgreSQL

```
# psql --host postgres-sample-instance.cbr4qtvbvyrz.us-east-2.rds.amazonaws.com --
->username-postgres
Password for user postgres:
```

(continues on next page)

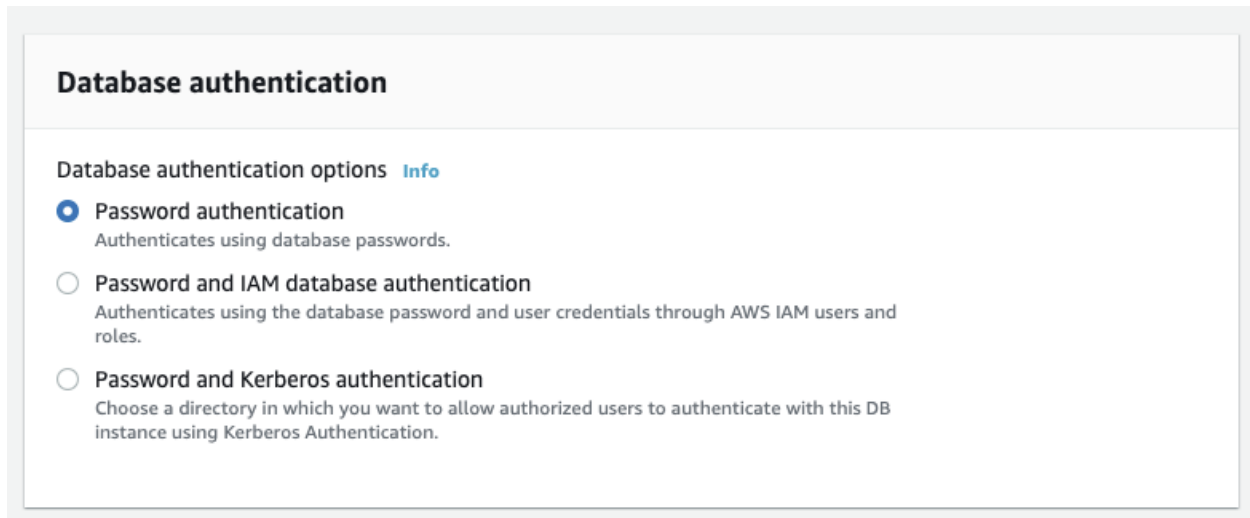


Fig. 41: Database authentication

(continued from previous page)

```
psql (9.5.19, server 11.5)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
compression: off)
Type "help" for help.

postgres-> CREATE USER myuser WITH LOGIN;
CREATE ROLE
postgres-> GRANT rds_iam TO myuser;
GRANT ROLE
```

After this command, you have to add an IAM role to your IAM user for creating a relation between your IAM account and RDS DB user.

IAM database authentication provides the following benefits:

1. Network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).
2. You can use IAM to centrally manage access to your database resources, instead of managing access individually on each DB instance.
3. For applications running on Amazon EC2, you can use profile credentials specific to your EC2 instance to access your database instead of a password, for greater security

[How To Connect an AWS RDS Instance with IAM User Authentication](#)

## Encryption

You can use AWS KMS-based encryption in the AWS console. There is no performance penalty for encrypting data and it is performed at the volume level. It provides you with a centralized access and audit of key activity. It uses two-tier encryption with the customer master key provided by you and each individual instance has its data key, which is used to encrypt the data.

Best practices for encryption are follow with RDS:

- Encryption cannot be removed from DB instances.
- If source is encrypted, Read Replicas must be encrypted.



Fig. 42: Database encryption

- Add encryption to an unencrypted DB instance by encrypting a snapshot copy.

## Microsoft SQL Server

You can use Secure Sockets Layer (SSL) to encrypt connections between your client applications and your Amazon RDS DB instances running Microsoft SQL Server. SSL support is available in all AWS regions for all supported SQL Server editions.

When you create a SQL Server DB instance, Amazon RDS creates an SSL certificate for it. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. There are 2 ways to use SSL to connect to your SQL Server DB instance:

- Force SSL for all connections, this happens transparently to the client, and the client doesn't have to do any work to use SSL.
- Encrypt specific connections, this sets up an SSL connection from a specific client computer, and you must do work on the client to encrypt connections.

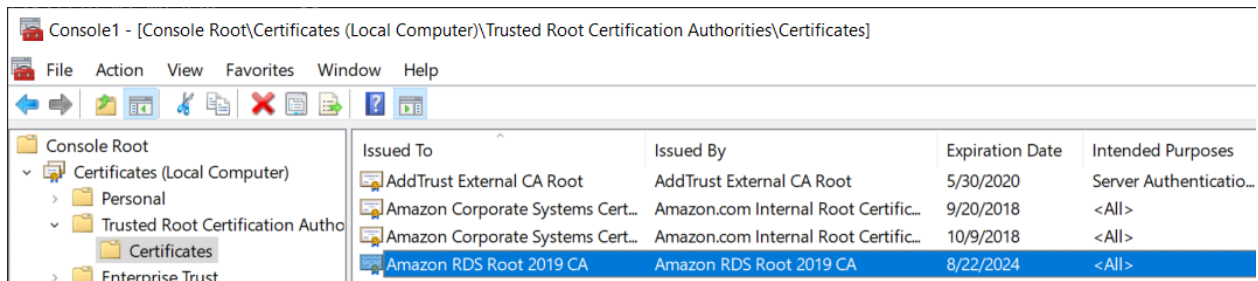


Fig. 43: SSL certificate

You can force all connections to your DB instance to use SSL, or you can encrypt connections from specific client computers only. To use SSL from a specific client, you must obtain certificates for the client computer, import certificates on the client computer, and then encrypt the connections from the client computer.

If you want to force SSL, use the `rds.force_ssl` parameter. By default, the `rds.force_ssl` parameter is set to `false`. Set the `rds.force_ssl` parameter to `true` to force connections to use SSL. The `rds.force_ssl` parameter is static, so after you change the value, you must reboot your DB instance for the change to take effect.

## Monitoring

### Monitor

Amazon RDS comes with comprehensive monitoring built-in:

- **Amazon CloudWatch metrics and alarms.** It allows you to monitor core metrics:
  - CPU/Storage/Memory
  - Swap usage
  - I/O (read and write)
  - Latency (read and write)
  - Throughput (read and write)
  - Replica lag

The monitoring interval is usually down to 1 minute. You can configure alarms on these metrics.

- **Amazon CloudWatch logs.** It allows publishing DB logs (errors, audit, slow queries) to a centralized log store (except SQL Server). You can access logs directly from RDS console and API for all engines.
- **Enhanced monitoring.** It is an agent-based monitoring system that allows you to have access to over 50 CPU, memory, file system, database engine, and disk I/O metrics. It is configurable to monitor up to 1 second intervals. They are automatically published to Amazon CloudWatch logs on your behalf.

Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output from CloudWatch Logs in a monitoring system of your choice. By default, Enhanced Monitoring metrics are stored in the CloudWatch Logs for 30 days. To modify the amount of time the metrics are stored in the CloudWatch Logs, change the retention for the `RDSOSMetrics` log group in the CloudWatch console.

Take note that there are certain differences between CloudWatch and Enhanced Monitoring Metrics. CloudWatch gathers metrics about CPU utilization from the hypervisor for a DB instance, and Enhanced Monitoring gathers its metrics from an agent on the instance. As a result, you might find differences between the measurements, because the hypervisor layer performs a small amount of work.

The differences can be greater if your DB instances use smaller instance classes, because then there are likely more virtual machines (VMs) that are managed by the hypervisor layer on a single physical instance. Enhanced Monitoring metrics are useful when you want to see how different processes or threads on a DB instance use the CPU and memory.

NAME	VIRT	RES	CPU%	MEM%	VMLIMIT
▼ postgres [3181]†	283.55 MB	17.11 MB	0.02	1.72	
postgres:					
rdsadmin	384.7 MB	9.51 MB	0.02	0.95	
rdsadmin					
localhost(40156)					
idle [2953]†					

Fig. 44: Use of CPU and memory of processes or threads on a DB instance

In RDS, the Enhanced Monitoring metrics shown in the Process List view are organized as follows:

- **RDS child processes.** Shows a summary of the RDS processes that support the DB instance, for example `aurora` for Amazon Aurora DB clusters and `mysqld` for MySQL DB instances. Process threads appear nested beneath the parent process. Process threads show CPU utilization only as other metrics are the same for all threads for the process. The console displays a maximum of 100 processes and threads. The results are a combination of the top CPU consuming and memory consuming processes and threads. If there are more than 50 processes and more than 50 threads, the console displays the top 50 consumers in each category. This display helps you identify which processes are having the greatest impact on performance.
- **RDS processes.** Shows a summary of the resources used by the RDS management agent, diagnostics monitoring processes, and other AWS processes that are required to support RDS DB instances.
- **OS processes.** Shows a summary of the kernel and system processes, which generally have minimal impact on performance.



- **Performance Insights** uses lightweight data collection methods that don't impact the performance of your applications, and makes it easy to see which SQL statements are causing the load, and why. It requires no configuration or maintenance, and is currently available for Amazon Aurora (PostgreSQL- and MySQL-compatible editions), Amazon RDS for PostgreSQL, MySQL, MariaDB, SQL Server and Oracle. It provides an easy and powerful dashboard showing load on your database. It helps you identify source of bottlenecks: top SQL queries, wait statistics. It has an adjustable time frame (hour, day week, month). With 7 days of free performance history retention, it's easy to track down and solve a wide variety of issues. If you need longer-term retention, you can choose to pay for up to two years of performance history retention.

## Events

Amazon RDS event notifications let you know when important things happen. You can leverage built-in notifications via Amazon SNS. Events are published to Amazon CloudWatch Events, where you can create rules to respond to the events. It supports cross-account event delivery. There are 6 different source types: DB instance, DB parameter group, DB security group, DB snapshot, DB cluster, DB cluster snapshot. There are 17 different event categories, such as availability, backup, deletion, configuration change, etc.

## Maintenance and billing

### Maintenance

Any maintenance that causes downtime (typically only a few times per year) will be scheduled in your maintenance window. Operating system or Amazon RDS software patches are usually performed without restarting databases. Database engine upgrades require downtime:

- Minor version upgrades: automatic or manually applied.
- Major version upgrades: manually applied because there can be application compatibility issues.
- Version deprecation: 3-6-month notification before scheduled upgrades.

You can view upcoming maintenance events in your AWS Personal Health Dashboard.

Event	Status	Region	Start time	Last update time	Affected entities	Event category
RDS operations	Closed	us-east-1	October 13, 2017	October 13, 2017	-	Issue
RDS operations	-	-	September 13, 2017	September 13, 2017	1 entity	Notification
RDS operations	-	us-east-1	August 20, 2017	September 13, 2017	1 entity	Notification

Fig. 45: AWS Personal Health Dashboard

## Billing

To estimate the cost of using RDS, you need to consider the following factors:

- **Database instance** (instance hours), from the time you launch a DB instance until you terminate it. It depends on Database characteristics: a combination of region, instance type, DB engine, and license (optional).

- **Database storage** (GB-mo). It can be either provisioned (Amazon EBS) or consumed (Amazon Aurora). If you are using provisioned IOPS for `io1` storage type in IOPS-Mo. You are charged for the number of database input and output requests for Amazon Aurora and Amazon EBS magnetic-storage types. If your purchase options is on-demand DB, then instances are charged by the hour. Reserved DB instances require upfront payment for DB instances reserved.
- **Backup storage** (GB-mo). Size of backups and snapshots stored in Amazon S3. There is no additional charge for backup storage of up to 100% of your provisioned DB storage for an active DB instance. After the DB instance is terminated, backup storage is billed per GB/month.
- **Number of database instances**, to handle peak loads.
- **Deployment type**. You can deploy the DB to a single AZ or multiple AZs.
- **Outbound data transfer** is tiered and inbound data transfer is free.

In order to reduce costs you can stop and start a DB instance from the console, AWS CLIs and SDKs. You can stop and start a DB instance whether it is configured for a single Availability Zone or for Multi-AZ, for database engines that support Multi-AZ deployments. You can't stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration.

When you stop a DB instance, the DB instance performs a normal shutdown and stops running. The status of the DB instance changes to `stopping` and then `stopped`. Any storage volumes remain attached to the DB instance, and their data is kept. Any data stored in the RAM of the DB instance is deleted.

Stopping a DB instance removes pending actions, except for pending actions for the DB instance's option group or DB parameter group. While DB instance is stopped, you only pay for storage. The backup retention window is maintained while stopped.

---

**Important:** You can stop a DB instance for up to 7 days. If you don't manually start your DB instance after 7 days, your DB instance is automatically started so that it doesn't fall behind any required maintenance updates. You cannot stop DB instances that have read replicas or are read replicas. If you want to stop a DB instance for more than 7 days, a possible strategy is to take an snapshot.

---

You can also save money by using Reserved Instances (RIs) that provide a discount over on-demand prices. You have to match region, instance family and engine of on-demand usage to apply benefit. There is size flexibility available for open source and Oracle BYOL engines. By default, RIs are shared among all accounts in consolidated billing. You can use the RI utilization and coverage reports to determine how your RIs are being used. Amazon RDS RI recommendations report uses historical data to recommend which RIs to buy.

## Best practices

### MySQL

- Ensure that automated backups are enabled for the RDS.
- Use InnoDB as the storage engine for MySQL.
- Partition your large tables so that file sizes does not exceed the 16 TB limit.

## 1.5.2 Amazon Aurora

## Introduction

Amazon Aurora is a fully managed, relational DBaaS that combines the speed and reliability of high-end commercial DBs, with the simplicity and cost-effectiveness of open source databases. It is designed to be compatible with MySQL and PostgreSQL, so existing applications and tools, can run against it without modification. It follows a simple pay as you go pricing model.

It is part of Amazon RDS and it tightly integrated with an SSD-backend virtualized storage layer purposefully built to accelerate DB workloads. It delivers up to 5 times the throughput of standard MySQL and up to 3 times the throughput of standard PostgreSQL.

It can scale automatically in a non-disruptive way, expanding up to 64 TB and rebalancing storage I/O to provide consistent performance, without the need for over-provisioning. Aurora storage is also fault-tolerant and self-healing, so any storage failures are repaired transparently.

It automatically replicates your storage six ways, across three Availability Zones. Amazon Aurora storage is fault-tolerant, transparently handling the loss of up to two copies of data without affecting database write availability and up to three copies without affecting read availability. Aurora always replicates your data across three Availability Zones, regardless of whether your database uses read replicas.

It is a regional service that offers greater than 99.99% availability. The service is designed to automatically detect DB crashes and restart the DB without the need for crash recovery or DB rebuilds. If the entire instance fails, Amazon Aurora automatically fails over to one of up to 15 read replicas.

AWS re:Invent 2018: [REPEAT 1] Deep Dive on Amazon Aurora with MySQL Compatibility (DAT304-R1)

## Performance

Aurora scale out, distributed architecture made two main contributions:

1. Push log applicator to storage. That allows us to construct pages from the logs themselves. It is not necessary to write full pages anymore. Unlike traditional databases that have to write logs and pages, Aurora only writes logs. This means that there is significantly less I/O and warm up. There is no checkpointing, you don't have to worry about cache additions,...
2. Instead of using heavy consistency protocols, Aurora uses 4/6 write quorum with local tracking.

The benefits of this are:

- Better write performance.
- Read scale out because the replicas are sharing the storage with the master.
- AZ+1 failure tolerance. Aurora stores 6 copies: 2 copies per AZ.
- Instant database redo recovery, even if an entire AZ goes down.

Amazon Aurora typically involves a cluster of DB instances instead of a single instance. Each connection is handled by a specific DB instance. When you connect to an Aurora cluster, the host name and port that you specify point to an intermediate handler called an endpoint. Aurora uses the endpoint mechanism to abstract these connections. Thus, you don't have to hardcode all the hostnames or write your own logic for load-balancing and rerouting connections when some DB instances aren't available.

For certain Aurora tasks, different instances or groups of instances perform different roles. For example, the primary instance handles all data definition language (DDL) and data manipulation language (DML) statements. Up to 15 Aurora Replicas handle read-only query traffic.

Using endpoints, you can map each connection to the appropriate instance or group of instances based on your use case. For example, to perform DDL statements you can connect to whichever instance is the primary instance. To perform queries, you can connect to the reader endpoint, with Aurora automatically performing load-balancing among all the Aurora Replicas. For clusters with DB instances of different capacities or configurations, you can connect to

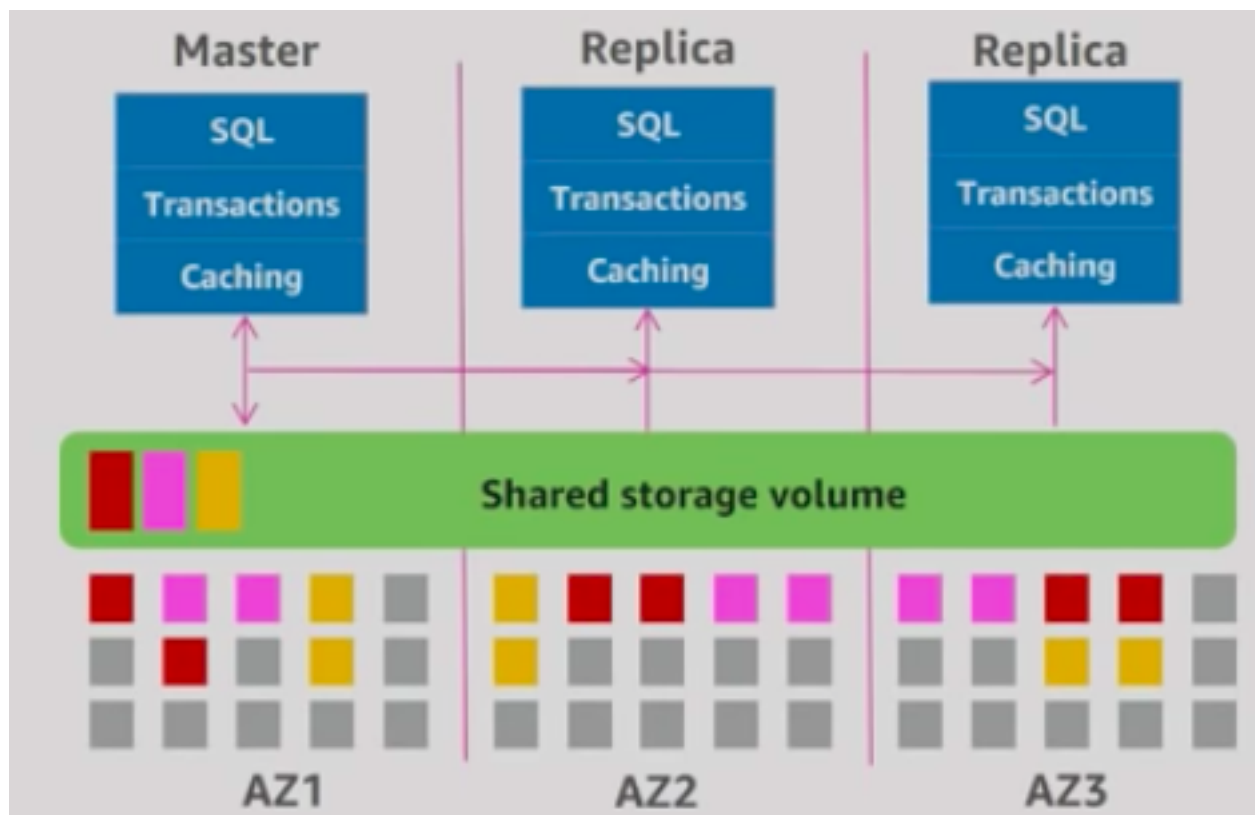


Fig. 46: Aurora architecture

custom endpoints associated with different subsets of DB instances. For diagnosis or tuning, you can connect to a specific instance endpoint to examine details about a specific DB instance.

The custom endpoint provides load-balanced database connections based on criteria other than the read-only or read-write capability of the DB instances. For example, you might define a custom endpoint to connect to instances that use a particular AWS instance class or a particular DB parameter group. Then you might tell particular groups of users about this custom endpoint. For example, you might direct internal users to low-capacity instances for report generation or ad hoc (one-time) querying, and direct production traffic to high-capacity instances.

## Amazon Aurora Replicas

Amazon Aurora MySQL and Amazon Aurora PostgreSQL support Amazon Aurora Replicas, which share the same underlying volume as the primary instance. Updates made by the primary are visible to all Amazon Aurora Replicas. With Amazon Aurora MySQL, you can also create MySQL Read Replicas based on MySQL's binlog-based replication engine. In MySQL Read Replicas, data from your primary instance is replayed on your replica as transactions. For most use cases, including read scaling and high availability, it is recommended using Amazon Aurora Replicas.

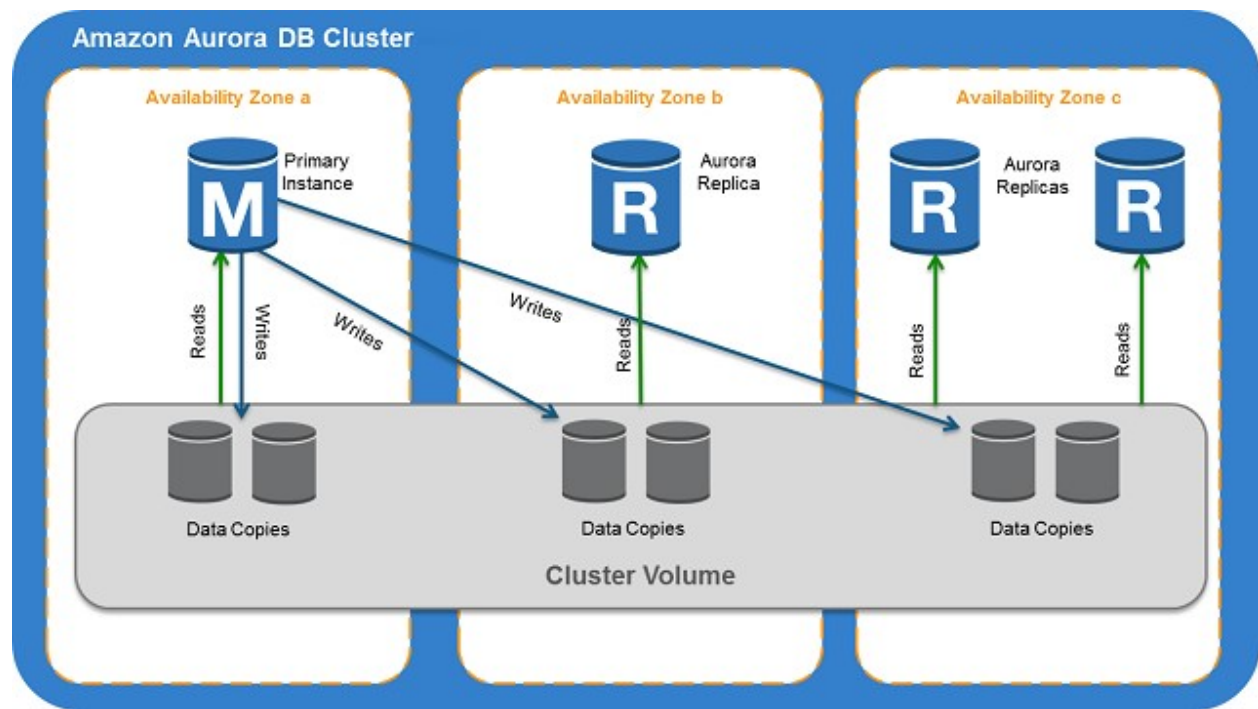


Fig. 47: Aurora failover

Read Replicas are primarily used for improving the read performance of the application. The most suitable solution in this scenario is to use Multi-AZ deployments instead but since this option is not available, you can still set up Read Replicas which you can promote as your primary stand-alone DB cluster in the event of an outage.

## Failover

Failover is automatically handled by Amazon Aurora so that your applications can resume database operations as quickly as possible without manual administrative intervention.

If you have an Amazon Aurora Replica in the same or a different Availability Zone, when failing over, Amazon Aurora flips the canonical name record (CNAME) for your DB Instance to point at the healthy replica, which in turn is promoted to become the new primary. Start-to-finish, failover typically completes within 30 seconds.

If you are running Aurora Serverless and the DB instance or AZ become unavailable, Aurora will automatically recreate the DB instance in a different AZ.

If you do not have an Amazon Aurora Replica (i.e. single instance) and are not running Aurora Serverless, Aurora will attempt to create a new DB Instance in the same Availability Zone as the original instance. This replacement of the original instance is done on a best-effort basis and may not succeed, for example, if there is an issue that is broadly affecting the Availability Zone.

You can use the reader endpoint to provide high availability for your read-only queries from your DB cluster by placing multiple Aurora Replicas in different Availability Zones and then connecting to the read-only endpoint for your read workload. The reader endpoint also load-balances connections to the Aurora Replicas in a DB cluster.

### 1.5.3 Amazon DynamoDB

#### Introduction

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models. Its flexible data model, reliable performance, and automatic scaling of throughput capacity makes it a great fit for mobile, web, gaming, ad tech, IoT, and many other applications. It is fast and consistent because it has a fully distributed request router. It provides fine-grained access control for accessing, for instance: the tables, the attributes within an item, etc. It allows event driven programming.

Table 5: SQL vs NoSQL

SQL	NoSQL
Optimized for storage	Optimized for compute
Normalized/relational	Denormalized/hierarchical
Ad hoc queries	Instantiated views
Scale vertically	Scale horizontally
Good for OLAP	Built for OLTP at scale

The following are the basic DynamoDB components:

- **Tables.** Similar to other database systems, DynamoDB stores data in tables. A table is a collection of data. For example, see the example table called People that you could use to store personal contact information about friends, family, or anyone else of interest. You could also have a Cars table to store information about vehicles that people drive.
- **Items.** Each table contains zero or more items. An item is a group of attributes that is uniquely identifiable among all of the other items. In a People table, each item represents a person. For a Cars table, each item represents one vehicle. Items in DynamoDB are similar in many ways to rows, records, or tuples in other database systems. In DynamoDB, there is no limit to the number of items you can store in a table.
- **Attributes.** Each item is composed of one or more attributes. An attribute is a fundamental data element, something that does not need to be broken down any further. For example, an item in a People table contains attributes called PersonID, LastName, FirstName, and so on. For a Department table, an item might have attributes such as DepartmentID, Name, Manager, and so on. Attributes in DynamoDB are similar in many ways to fields or columns in other database systems.

When you create a table, in addition to the table name, you must specify the primary key of the table. The primary key uniquely identifies each item in the table, so that no two items can have the same key.

DynamoDB supports two different kinds of primary keys:

- **Partition key** A simple primary key, composed of one attribute known as the partition key. DynamoDB uses the partition key's value as input to an internal hash function. The output from the hash function determines the

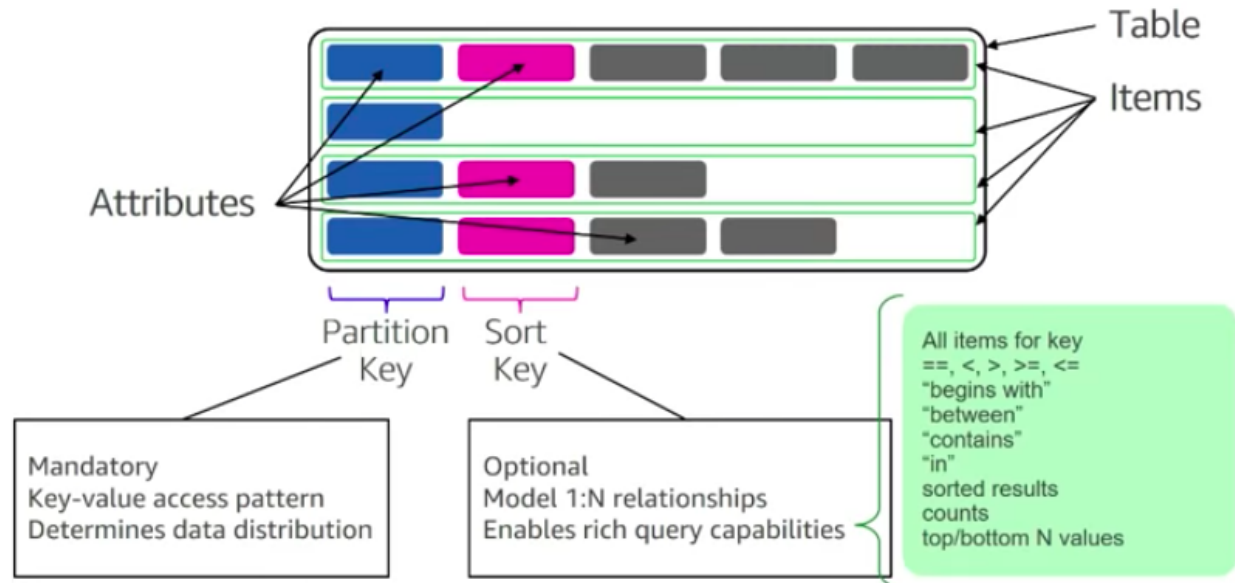


Fig. 48: DynamoDB tables, items and attributes

partition (physical storage internal to DynamoDB) in which the item will be stored. In a table that has only a partition key, no two items can have the same partition key value. It allows table to be partitioned for scale.

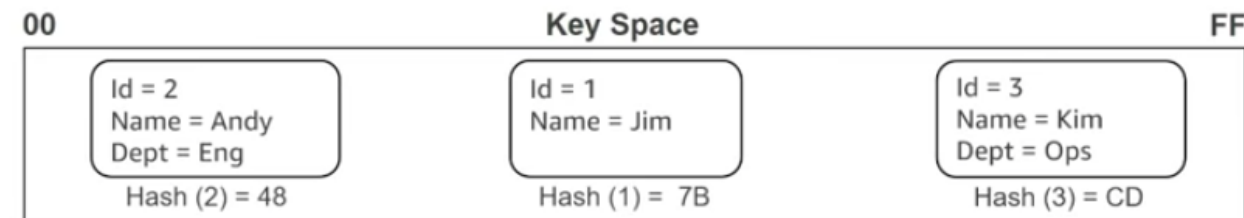


Fig. 49: Partition keys

- **Partition key and sort key.** Referred to as a composite primary key, this type of key is composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key. All items with the same partition key value are stored together, in sorted order by sort key value. In a table that has a partition key and a sort key, it's possible for two items to have the same partition key value. However, those two items must have different sort key values. There is no limit on the number of items per partition key, except if you have local secondary indexes.

Partitions are three-way replicated. In DynamoDB, you get an acknowledge when two of these replicas has been done.

The partition key portion of a table's primary key determines the logical partitions in which a table's data is stored. This in turn affects the underlying physical partitions. Provisioned I/O capacity for the table is divided evenly among these physical partitions. Therefore a partition key design that doesn't distribute I/O requests evenly can create "hot" partitions that result in throttling and use your provisioned I/O capacity inefficiently.

The optimal usage of a table's provisioned throughput depends not only on the workload patterns of individual items, but also on the partition-key design. This doesn't mean that you must access all partition key values to achieve an efficient throughput level, or even that the percentage of accessed partition key values must be high. It does mean that the more distinct partition key values that your workload accesses, the more those requests will be spread across the partitioned space. In general, you will use your provisioned throughput more efficiently as the ratio of partition key values accessed to the total number of partition key values increases.



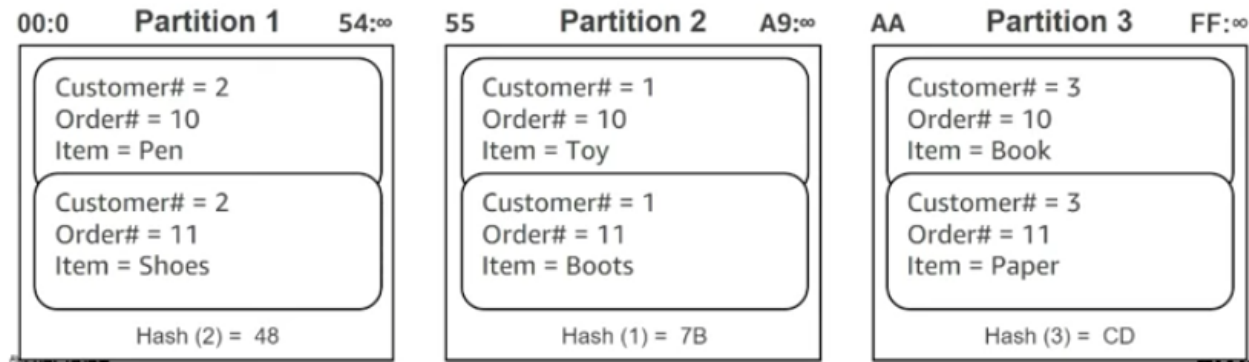


Fig. 50: Partition: Sort key

One example for this is the use of partition keys with high-cardinality attributes, which have a large number of distinct values for each item.

DynamoDB supports in-place atomic updates. It supports both increment and decrement atomic operations.

[AWS re:Invent 2018: Amazon DynamoDB Deep Dive: Advanced Design Patterns for DynamoDB \(DAT401\)](#)

## DynamoDB Streams

A DynamoDB stream is an ordered flow of information about changes to items in an Amazon DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.

Whenever an application creates, updates, or deletes items in the table, DynamoDB Streams writes a stream record with the primary key attribute(s) of the items that were modified. A stream record contains information about a data modification to a single item in a DynamoDB table. You can configure the stream so that the stream records capture additional information, such as the “before” and “after” images of modified items.

Amazon DynamoDB is integrated with AWS Lambda so that you can create triggers—pieces of code that automatically respond to events in DynamoDB Streams. With triggers, you can build applications that react to data modifications in DynamoDB tables.

If you enable DynamoDB Streams on a table, you can associate the stream ARN with a Lambda function that you write. Immediately after an item in the table is modified, a new record appears in the table’s stream. AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records.

You can create a Lambda function which can perform a specific action that you specify, such as sending a notification or initiating a workflow. For instance, you can set up a Lambda function to simply copy each stream record to persistent storage, such as EFS or S3, to create a permanent audit trail of write activity in your table.

## DynamoDB auto scaling

DynamoDB auto scaling uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns. This enables a table or a global secondary index to increase its provisioned read and write capacity to handle sudden increases in traffic, without throttling. When the workload decreases, Application Auto Scaling decreases the throughput so that you don’t pay for unused provisioned capacity.



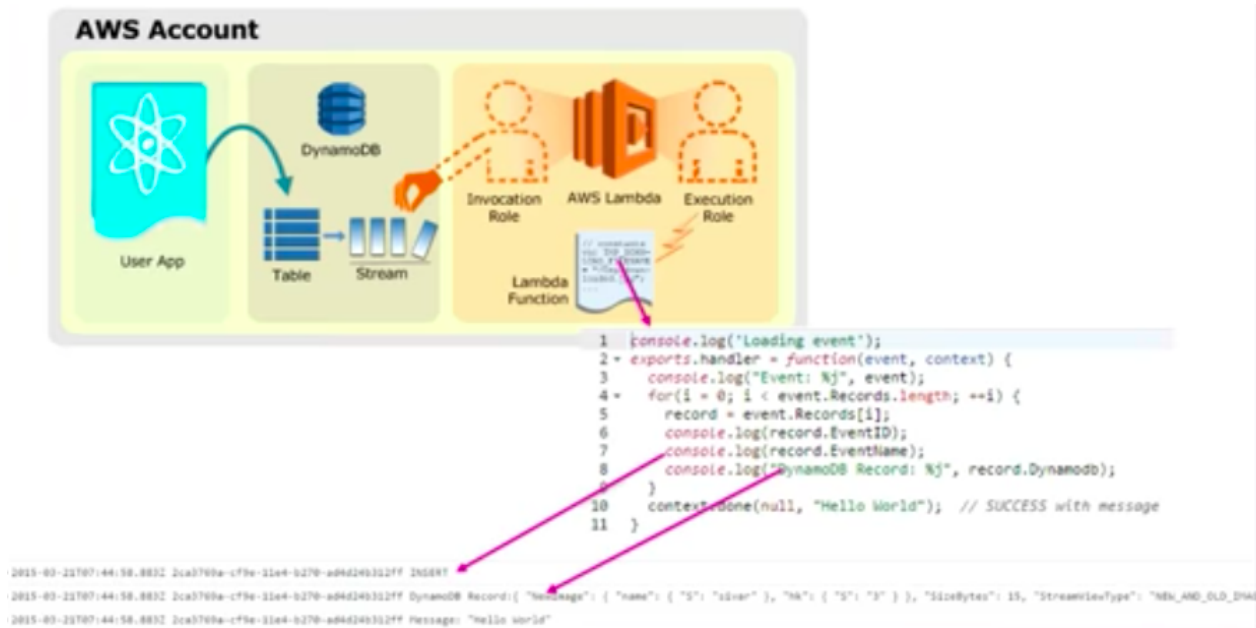


Fig. 51: DynamoDB Streams and AWS Lambda

## Use cases

- *Managing web sessions.* DynamoDB Time-to-Live (TTL) mechanism enables you to manage web sessions of your application easily. It lets you set a specific timestamp to delete expired items from your tables. Once the timestamp expires, the corresponding item is marked as expired and is subsequently deleted from the table. By using this functionality, you do not have to track expired data and delete it manually. TTL can help you reduce storage usage and reduce the cost of storing data that is no longer relevant.
- *Storing metadata for Amazon S3 objects.* Amazon DynamoDB stores structured data indexed by primary key and allow low latency read and write access to items ranging from 1 byte up to 400KB. Amazon S3 stores unstructured blobs and is suited for storing large objects up to 5 TB. In order to optimize your costs across AWS services, large objects or infrequently accessed data sets should be stored in Amazon S3, while smaller data elements or file pointers (possibly to Amazon S3 objects) are best saved in Amazon DynamoDB.

To speed up access to relevant data, you can pair Amazon S3 with a search engine such as Amazon CloudSearch or a database such as Amazon DynamoDB or Amazon RDS. In these scenarios, Amazon S3 stores the actual information, and the search engine or database serves as the repository for associated metadata such as the object name, size, keywords, and so on. Metadata in the database can easily be indexed and queried, making it very efficient to locate an object's reference by using a search engine or a database query. This result can be used to pinpoint and retrieve the object itself from Amazon S3.

## Amazon DynamoDB Accelerator (DAX)

Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement ? from milliseconds to microseconds ? even at millions of requests per second. DAX does all the heavy lifting required to add in-memory acceleration to your DynamoDB tables, without requiring developers to manage cache invalidation, data population, or cluster management.

DAX is a DynamoDB-compatible caching service that enables you to benefit from fast in-memory performance for demanding applications. DAX addresses three core scenarios:

**Movies** [Close](#)

OverviewItemsMetricsAlarms**Capacity**IndexesTriggersAccess controlTags

► Scaling activities

Provisioned capacity

Read capacity units

Table

Write capacity units

❗ Consumed read capacity >= 4 for 5 minutes

Estimated cost \$2.91 / month ([Capacity calculator](#))

Auto Scaling

☒ Read capacity

☐ Write capacity

Target utilization

%

Minimum provisioned capacity

units

Maximum provisioned capacity

units

☒ Apply same settings to global secondary indexes

**IAM Role** I authorize DynamoDB to scale capacity using the following role:

☒ New role: DynamoDBAutoscaleRole

☐ Existing role with pre-defined policies [\[Instructions\]](#)

Role Name\*

Save

Cancel

Fig. 52: Amazon DynamoDB Accelerator (DAX) configuration

1. As an in-memory cache, DAX reduces the response times of eventually consistent read workloads by an order of magnitude, from single-digit milliseconds to microseconds.
2. DAX reduces operational and application complexity by providing a managed service that is API-compatible with DynamoDB. Therefore, it requires only minimal functional changes to use with an existing application.
3. For read-heavy or bursty workloads, DAX provides increased throughput and potential operational cost savings by reducing the need to overprovision read capacity units. This is especially beneficial for applications that require repeated reads for individual keys.

DAX supports server-side encryption. With encryption at rest, the data persisted by DAX on disk will be encrypted. DAX writes data to disk as part of propagating changes from the primary node to read replicas. The following diagram shows a high-level overview of DAX.

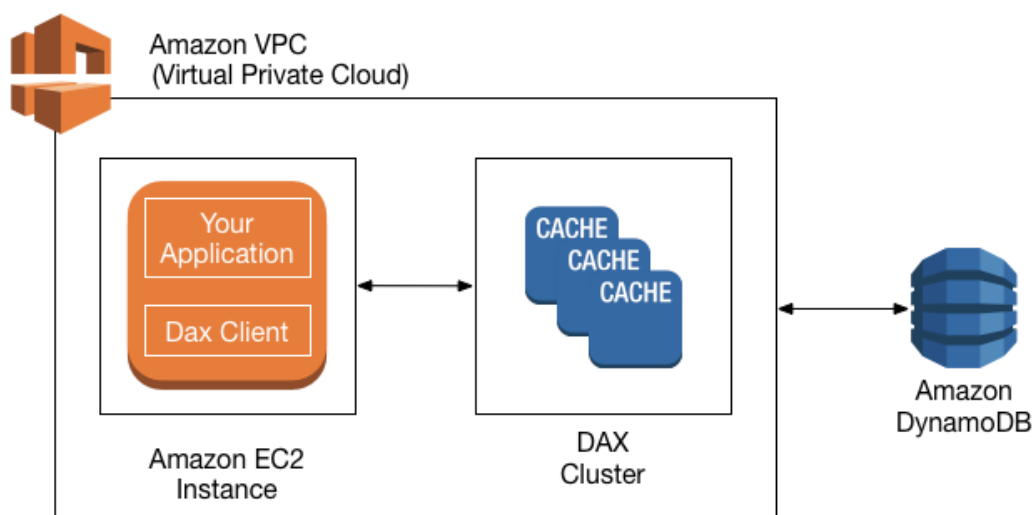


Fig. 53: High-level overview of DAX

## 1.5.4 Amazon Redshift

### Architecture and concepts

Amazon Redshift is a fast, scalable data warehouse that makes it simple and cost-effective to analyze all your data across your data warehouse and data lake. Redshift delivers ten times faster performance than other data warehouses by using machine learning, massively parallel query execution, and columnar storage on high-performance disk. Redshift is a fully-managed service which is the result of rewriting PostgreSQL to:

- Become a columnar database.
- Provide MPP (massive parallel processing) that allows to scale out up to a several petabytes database.
- Provide analytics functions to work as an OLAP service.

It is also integrated with the rest of the AWS ecosystem: S3, KMS, Route 53, etc.

The maintenance window occurs weekly, and DB instances can receive upgrades to the operating system (OS) or to the DB engine. AWS requires at least a 30-minute window in your instance's weekly schedule to confirm that all instances have the latest patches and upgrades. During the maintenance window, tasks are performed on clusters and instances. For the security and stability of your data, maintenance can cause instances to be unavailable.

The maintenance window defines when the deployment or operation begins, but maintenance itself can take longer to complete. As a result, the time used for some operations can exceed the duration of the maintenance window.

## Architecture

The elements of the Amazon Redshift data warehouse architecture is shown in the following figure.

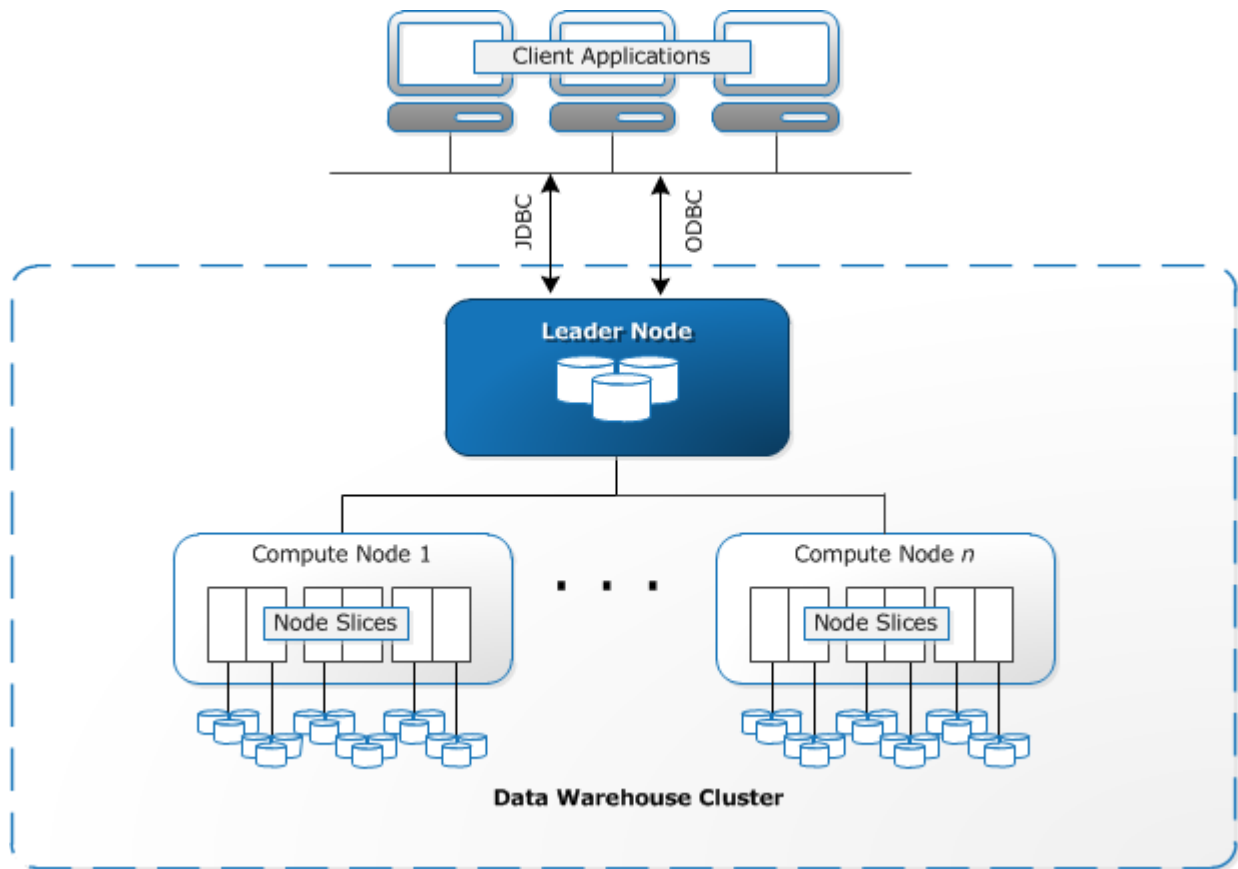


Fig. 54: Amazon Redshift architecture

## Client applications

Amazon Redshift integrates with various data loading and ETL (extract, transform, and load) tools and business intelligence (BI) reporting, data mining, and analytics tools. Amazon Redshift is based on industry-standard PostgreSQL, so most existing SQL client applications will work with only minimal changes. Amazon Redshift communicates with client applications by using industry-standard JDBC and ODBC drivers for PostgreSQL.

## **Leader node**

The leader node manages communications with client programs and all communication with compute nodes. It parses and develops execution plans to carry out database operations, in particular, the series of steps necessary to obtain results for complex queries. Based on the execution plan, the leader node compiles code, distributes the compiled code to the compute nodes, and assigns a portion of the data to each compute node.

The leader node distributes SQL statements to the compute nodes only when a query references tables that are stored on the compute nodes. All other queries run exclusively on the leader node. Amazon Redshift is designed to implement certain SQL functions only on the leader node. A query that uses any of these functions will return an error if it references tables that reside on the compute nodes.

## **Clusters**

The core infrastructure component of an Amazon Redshift data warehouse is a cluster. A cluster is composed of one or more compute nodes. If a cluster is provisioned with two or more compute nodes, an additional leader node coordinates the compute nodes and handles external communication. Your client application interacts directly only with the leader node. The compute nodes are transparent to external applications.

## **Compute nodes**

The leader node compiles code for individual elements of the execution plan and assigns the code to individual compute nodes. The compute nodes execute the compiled code and send intermediate results back to the leader node for final aggregation.

Each compute node has its own dedicated CPU, memory, and attached disk storage, which are determined by the node type. As your workload grows, you can increase the compute capacity and storage capacity of a cluster by increasing the number of nodes, upgrading the node type, or both.

Amazon Redshift provides two node types; dense storage nodes and dense compute nodes. Each node provides two storage choices. You can start with a single 160 GB node and scale up to multiple 16 TB nodes to support a petabyte of data or more.

Hopefully all the data is spread across the compute nodes. The SQL queries are executed in parallel and that's why it is called a massively parallel, shared nothing columnar architecture.

## **Node slices**

A compute node is partitioned into slices. Each slice is allocated a portion of the node's memory and disk space, where it processes a portion of the workload assigned to the node. The leader node manages distributing data to the slices and apportions the workload for any queries or other database operations to the slices. The slices then work in parallel to complete the operation. The number of slices per node is determined by the node size of the cluster.

When you create a table, you can optionally specify one column as the distribution key. When the table is loaded with data, the rows are distributed to the node slices according to the distribution key that is defined for a table. Choosing a good distribution key enables Amazon Redshift to use parallel processing to load data and execute queries efficiently.

## **Internal network**

Amazon Redshift takes advantage of high-bandwidth connections, close proximity, and custom communication protocols to provide private, very high-speed network communication between the leader node and compute nodes. The compute nodes run on a separate, isolated network that client applications never access directly.

## Databases

A cluster contains one or more databases. User data is stored on the compute nodes. Your SQL client communicates with the leader node, which in turn coordinates query execution with the compute nodes.

Amazon Redshift is a relational database management system (RDBMS), so it is compatible with other RDBMS applications. Although it provides the same functionality as a typical RDBMS, including online transaction processing (OLTP) functions such as inserting and deleting data, Amazon Redshift is optimized for high-performance analysis and reporting of very large datasets.

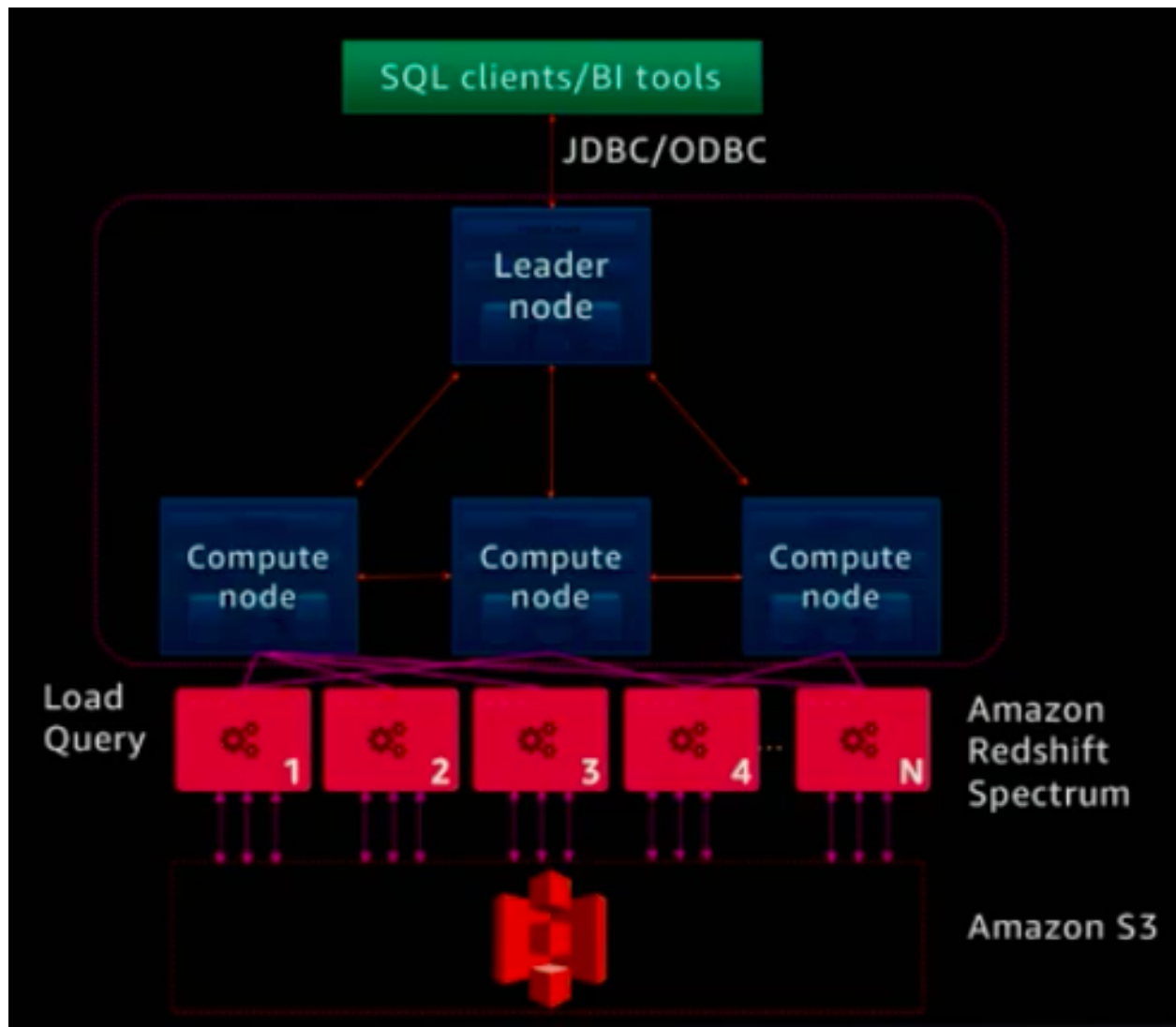


Fig. 55: Amazon Redshift Spectrum architecture

The Load, unload, backup and restore of data is performed on Amazon S3. Amazon Redshift Spectrum is an extension of Amazon Redshift in which the nodes load data from Amazon S3 and execute queries directly against Amazon S3.

## Terminology

## Columnar

Amazon Redshift uses a columnar architecture for storing data on disk column by column rather than row by row like a traditional database. The reason for doing this is that the types of queries that you execute in an analytics database usually query a subset of the columns, so we are able to reduce the amount of IO needed to be done for analytics queries.

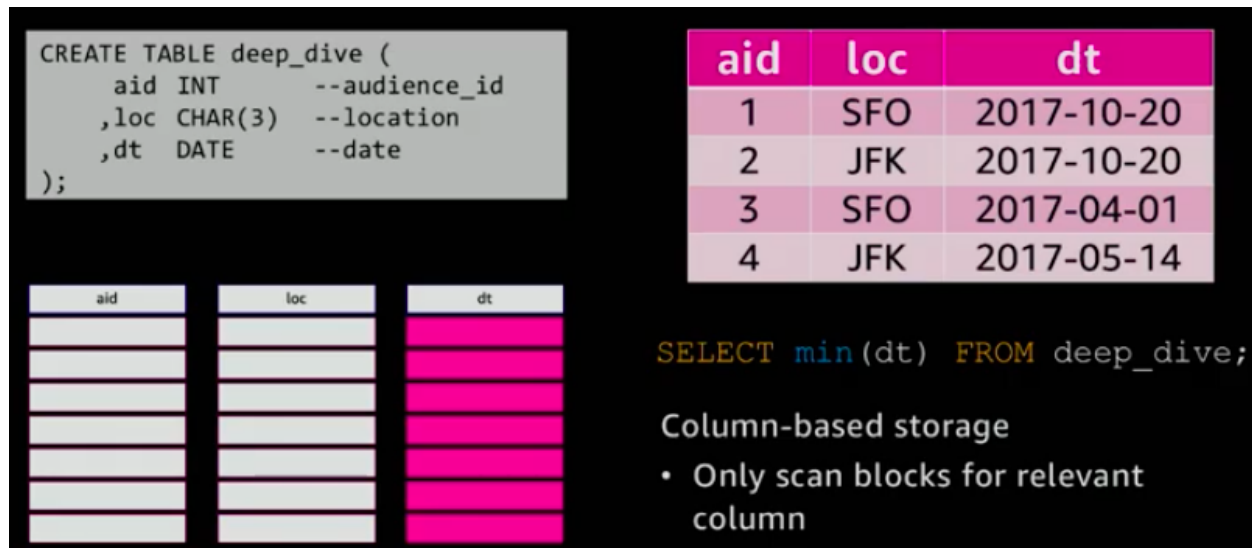


Fig. 56: Columnar architecture: Example

In a columnar architecture, the data is stored physically on disk by column rather than by row. It only reads the column data that is required.

## Compression

The goals of compression (sometimes called encoding) is to allow more data to be stored within an Amazon Redshift cluster and improve query performance by decreasing I/O. As a consequence, it allows several times more data to be stored within the cluster and also improves performance. The reason of this performance improvement is because it reduces the amount of I/O needed to do off disk.

By default, `copy` automatically analyzes and compresses data on first load into an empty table. The `analyze compression` is a built-in command that will find the optimal compression for each column on an existing table.

The best practices are:

- To apply compression to all tables.
- Use `analyze compression` command to find the optimal compression. If it returns a encoding type of RAW, it means that there is no compression, it happens for sparse columns and small tables.
- Changing column encoding requires a table rebuild: <https://github.com/awslabs/amazon-redshift-utils/tree/master/src/ColumnEncodingUtility>. To verify if columns are compressed:
- Verify that columns are compressed:

```
SELECT "column", type, encoding FROM pg_table_def WHERE tablename = 'deep_dive'
```

column	type	encoding
--------	------	----------

(continues on next page)

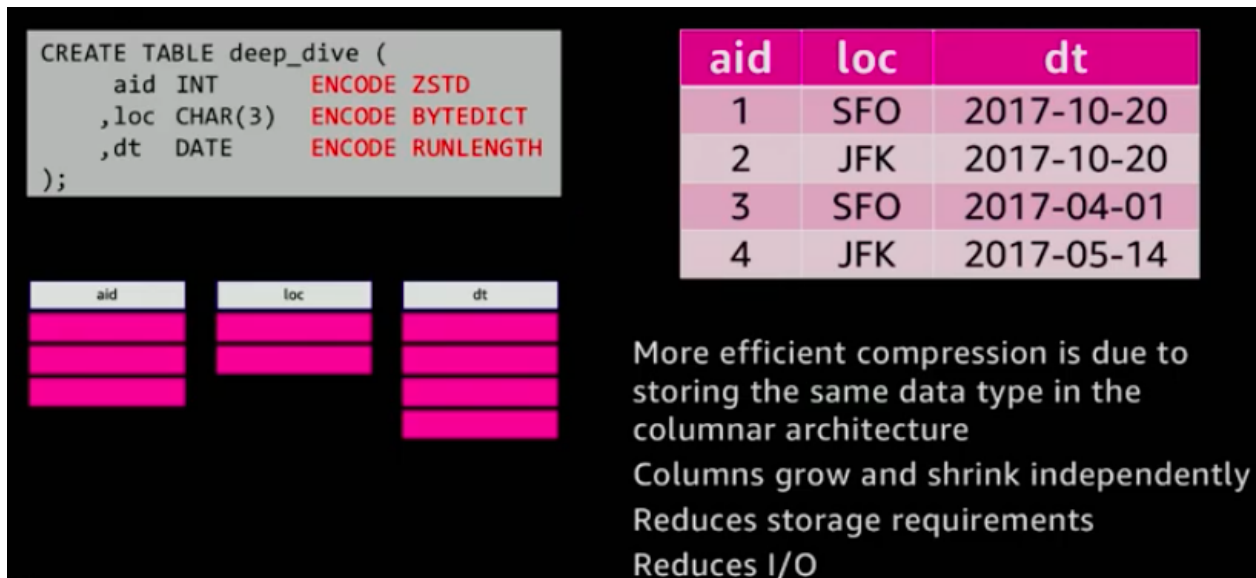


Fig. 57: Compression: Example

(continued from previous page)

```

-----+-----+-----
aid    | integer | zstd
loc    | character(3) | bytedict
dt     | date    | runlength

```

## Blocks

Column data is persisted to 1 MB immutable blocks. Blocks are individually encoded with 1 of 12 encodings. A full block can contain millions of values.

## Zone maps

Zone maps are in-memory block metadata that contains per-block min and max values. All blocks automatically have zone maps. They effectively prune blocks that cannot contain data for a given query. Their goal is to eliminate unnecessary I/O.

## Data storage, ingestion, and ELT

### Workload management and query monitoring

#### Workload management (WLM)

Workload management (WLM) allows for separation of different query workloads. Their main goals are prioritize important queries and throttle/abort less important queries. It allows us to control concurrent number of executing of queries, divide cluster memory and set query timeouts to abort long running queries.

Every single query in Redshift will execute in one queue. Queues are assigned a percentage of cluster memory. SQL queries execute in queue based on user group (which groups the user belongs to) and query group session level



variable. WLM allows us to define the number of query queues that are available and how queries are routed to those queues for processing.

When you create a parameter group, the default WLM configuration contains one queue that can run up to five queries concurrently. You can add additional queues and configure WLM properties in each of them if you want more control over query processing. Each queue that you add has the same default WLM configuration until you configure its properties. When you add additional queues, the last queue in the configuration is the default queue. Unless a query is routed to another queue based on criteria in the WLM configuration, it is processed by the default queue. You cannot specify user groups or query groups for the default queue.

As with other parameters, you cannot modify the WLM configuration in the default parameter group. Clusters associated with the default parameter group always use the default WLM configuration. If you want to modify the WLM configuration, you must create a parameter group and then associate that parameter group with any clusters that require your custom WLM configuration.

Short query acceleration (SQA) allows to automatically detect short running queries and run them within the short query queue if queuing occurs.

## Enhanced VPC Routing

When you use Amazon Redshift Enhanced VPC Routing, Amazon Redshift forces all COPY and UNLOAD traffic between your cluster and your data repositories through your Amazon VPC. By using Enhanced VPC Routing, you can use standard VPC features, such as VPC security groups, network access control lists (ACLs), VPC endpoints, VPC endpoint policies, internet gateways, and Domain Name System (DNS) servers. Hence, Option 2 is the correct answer.

You use these features to tightly manage the flow of data between your Amazon Redshift cluster and other resources. When you use Enhanced VPC Routing to route traffic through your VPC, you can also use VPC flow logs to monitor COPY and UNLOAD traffic. If Enhanced VPC Routing is not enabled, Amazon Redshift routes traffic through the Internet, including traffic to other services within the AWS network.

The screenshot shows the 'Configure Networking Options' dialog box for Amazon Redshift. It contains the following fields and options:

- Choose a VPC:** A dropdown menu showing 'vpc-81acdae4'. To the right, it says 'The identifier of the VPC in which you want to create your cluster'.
- Cluster Subnet Group:** A dropdown menu showing 'gute-11-05-ae4'. To the right, it says 'Selected Cluster Subnet Group may limit the choice of Availability Zones'.
- Publicly Accessible:** Radio buttons for 'Yes' (selected) and 'No'. To the right, it says 'Select Yes if you want the cluster to have a public IP address that can be accessed from the public internet, select No if you want the cluster to have a private IP addressed that can only be accessed from within the VPC.'
- Choose a Public IP Address:** Radio buttons for 'Yes' and 'No' (selected). To the right, it says 'Select Yes if you want to select an elastic IP (EIP) address that you already have configured. Otherwise, select No to have Amazon Redshift create an EIP for your instance.'
- Enhanced VPC Routing:** Radio buttons for 'Yes' (selected) and 'No'. To the right, there is a warning icon and text: 'Additional VPC configuration might be required. [Learn more.](#)'
- Availability Zone:** A dropdown menu showing 'No Preference'. To the right, it says 'The EC2 Availability Zone that the cluster will be created in.'

Fig. 58: Configure enhanced VPC routing

## Cluster sizing and resizing

## Additional resources

You can configure Amazon Redshift to copy snapshots for a cluster to another region. To configure cross-region snapshot copy, you need to enable this copy feature for each cluster and configure where to copy snapshots and how long to keep copied automated snapshots in the destination region. When cross-region copy is enabled for a cluster, all new manual and automatic snapshots are copied to the specified region.

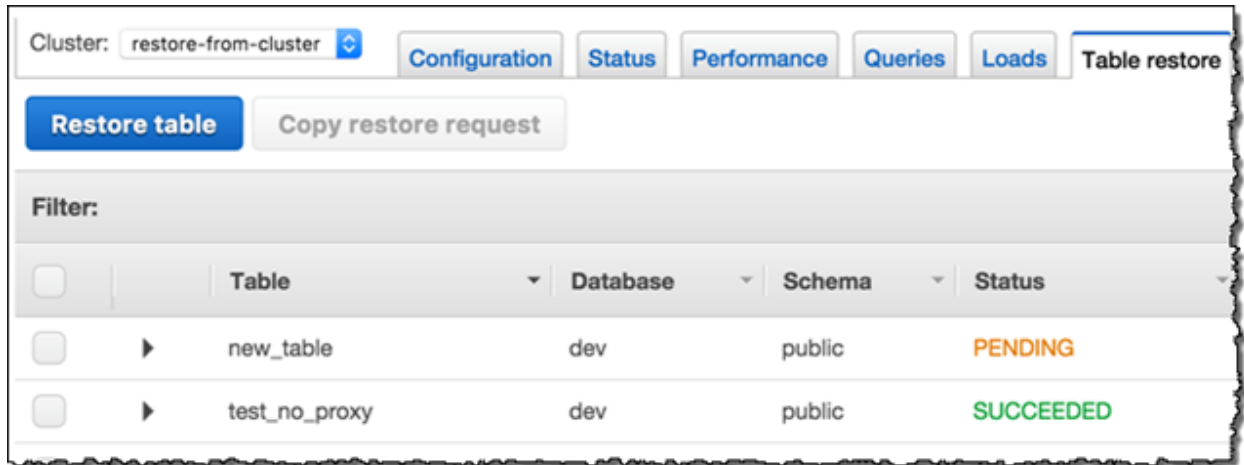


Fig. 59: Restore table from snapshot on Redshift

AWS re:Invent 2018: [REPEAT 1] Deep Dive and Best Practices for Amazon Redshift (ANT401-R1)

## Amazon Redshift Spectrum

Amazon Redshift also includes Redshift Spectrum, allowing you to directly run SQL queries against exabytes of unstructured data in Amazon S3. No loading or transformation is required, and you can use open data formats, including Avro, CSV, Grok, ORC, Parquet, RCFile, RegexSerDe, SequenceFile, TextFile, and TSV. Redshift Spectrum automatically scales query compute capacity based on the data being retrieved, so queries against Amazon S3 run fast, regardless of data set size.

### 1.5.5 Relational vs NoSQL databases

Table 6: Relational databases vs NoSQL databases

Optimal workloads	They are designed for transactional and strongly consistent OLTP applications and are good for OLAP.	NoSQL key-value, document, graph, and in-memory databases are designed for OLTP for a number of data access patterns that include low-latency applications. NoSQL, search databases are designed for analytics and semi-structured data.
Data model	The relational model normalizes data into tables that are composed of rows and columns. A schema strictly defines the tables, rows, columns, indexes, relationships between tables, and other database elements. The database enforces the referential integrity in relationships between tables.	NoSQL databases provide a variety of data models that includes document, graph, key-value, in-memory, and search.
ACID properties	They provide atomicity, consistency, isolation, and durability (ACID) properties: Atomicity requires a transaction to execute completely or not at all. Consistency requires that when a transaction has been committed, the data must conform to the database schema. Isolation requires that concurrent transactions execute separately for each other. Durability requires the ability to recover from an unexpected system failure or power outage to the last known state.	They often make tradeoffs by relaxing some of the ACID properties of relational databases for a more flexible data model that can scale horizontally. This makes NoSQL databases an excellent choice for high throughput, low-latency use cases that need to scale horizontally beyond the limitations of a single instance.
Performance	Performance is generally dependent on the disk subsystem. The organization of the queries, indexes, and table structure is often required to achieve peak performance.	Performance is generally a function of the underlying HW cluster size, network latency, and the calling application.
Scale	They typically scale up by increasing the compute capabilities of the HW or scale-out by adding replicas for read-only workloads.	NoSQL databases typically are partitionable because key-value access patterns and able to scale out by using a distributed architecture to increase throughput that provides consistent performance at near boundless scale.
API	Requests to store and retrieve data are communicated using queries that conform to a structured query language (SQL). These queries are parsed and executed by the relational database.	Object-based APIs allow app developers to easily store and retrieve in-memory data structures. Partition keys let apps look up key-value pairs, column sets, or semistructured documents that contain serialized app objects and attributes.

### 1.5.6 Migrating data into your AWS databases

AWS Database Migration Service helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate your data to and from most widely used commercial and open-source databases.

AWS Database Migration Service can migrate your data to and from most of the widely used commercial and open

source databases. It supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle to Amazon Aurora. Migrations can be from on-premises databases to Amazon RDS or Amazon EC2, databases running on EC2 to RDS, or vice versa, as well as from one RDS database to another RDS database. It can also move data between SQL, NoSQL, and text based targets.

In heterogeneous database migrations the source and target databases engines are different, like in the case of Oracle to Amazon Aurora, Oracle to PostgreSQL, or Microsoft SQL Server to MySQL migrations. In this case, the schema structure, data types, and database code of source and target databases can be quite different, requiring a schema and code transformation before the data migration starts. That makes heterogeneous migrations a two step process. First use the AWS Schema Conversion Tool to convert the source schema and code to match that of the target database, and then use the AWS Database Migration Service to migrate data from the source database to the target database. All the required data type conversions will automatically be done by the AWS Database Migration Service during the migration. The source database can be located in your own premises outside of AWS, running on an Amazon EC2 instance, or it can be an Amazon RDS database. The target can be a database in Amazon EC2 or Amazon RDS.

## 1.6 Networking in AWS

### 1.6.1 Design and implement AWS networks

#### Amazon Virtual Private Cloud (VPC)

A VPC can span multiple AZs inside of a region. The subnets that you create within the VPC are going to bound to an AZ. You can have multiple subnets inside an AZ. You cannot define a VPC that goes beyond a region, but there are ways that allows you to communicate different VPCs.

#### VPC IP addressing

The VPC IP addressing can be either in IPv4 and IPv6. The IPv4 address can be chosen by the customer but the IPv6 address is chosen from AWS range as a /56 and the customer can define /64 subnets inside these /56 global address.

Each one of the EC2 instances that you launch has an IPv4 address and optionally can have an IPv6 address. What cannot happen is that an EC2 instance has only an IPv6 address, it must have an IPv4 address. As a consequence, the IPv6 addresses within an /56 subnet has to have an associated IPv4 address.

When you create a VPC, you must specify a range of IPv4 addresses for the VPC in the form of a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. This is the primary CIDR block for your VPC. To add a CIDR block to your VPC, the following rules apply:

- The VPC block size is between a /16 netmask (65536 IP addresses) and /28 netmask (16 IP addresses).
- The CIDR block must not overlap with any existing CIDR block that's associated with the VPC.
- You cannot increase or decrease the size of an existing CIDR block once it is created.
- After you've created your VPC, you can associate secondary CIDR blocks with the VPC. You can have a maximum of 6 CIDR blocks per VPC. You have a limit on the number of routes you can add to a route table. You cannot associate a CIDR block if this results in you exceeding your limits.
- The CIDR block must not be the same or larger than the CIDR range of a route in any of the VPC route tables. For example, if you have a route with a destination of 10.0.0.0/24 to a virtual private gateway, you cannot associate a CIDR block of the same range or larger. However, you can associate a CIDR block of 10.0.0.0/25 or smaller.

- Within these blocks, you can create subnets, that can be public or private. The IP address that you use determine that if it possible to make it publicly routable. You should create subnets with a publicly routable CIDR block that falls outside of the private IPv4 address ranges specified in RFC 1918.
- When you create a subnet with an IP address range within the VPC CIDR blocks, there are some reserved IPs addresses that the customer cannot use. They are the +0,+1,+2,+3 of the subnet together with the broadcast address of the subnet. For example, for the IPv4 range `10.0.0.0/24` they would be:
  - `10.0.0.0` as the network address.
  - `10.0.0.1` for the VPC router.
  - `10.0.0.2` for the DNS server. For VPCs with multiple CIDR blocks, the IP address of the DNS server is located in the primary CIDR. AWS also reserve the base of each subnet range plus two for all CIDR blocks in the VPC.
  - `10.0.0.3` for future use.
  - `10.0.0.255`: Network broadcast address. AWS do not support broadcast in a VPC.

To calculate the total number of IP addresses of a given CIDR Block, you simply need to follow the 2 easy steps below. Let's say you have a CIDR block `/27`:

1. Subtract 32 with the mask number :  $(32 - 27) = 5$
2. Raise the number 2 to the power of the answer in Step 1 :  $2^5 = 32$

The answer to Step 2 is the total number of IP addresses available in the given CIDR netmask. Don't forget that in AWS, the first 4 IP addresses and the last IP address in each subnet CIDR block are not available for you to use, and cannot be assigned to an instance.

## IPv6 addressing

If you want to use IPv6 addresses in EC2 instances, it is necessary to work as dual stack, as you cannot disable IPv4 support for your VPC and subnets. The amount of IPv6 addresses you can use is limited by the number of IPv4 address you have.

There are 2 types of IPv6 addresses:

- **Global Unicast Address (GUA)**. AWS assigns a fixed size `/56` GUA per VPC which is random assigned. Each subnet that you create must have a `/64` range. If IPv6 is enabled, EC2 instances has a GUA automatically/manually assigned at or after launch.
- **Link Local Address (LLA)**. In IPv6 world, there are no MAC addresses and as a consequence it is no necessary to do NAT. In IPv4, NAT is needed in order to cover the shortage of available public IPv4 addresses, but in IPv6 there is no a foreseeable shortage of IPv6 addresses. LLA is also assigned to EC2 instance. It has a link-local prefix: `fe80::/64` and is a modified EUI-64 format obtained through a 48-bit MAC address. The LLA is used to communicate with the VPC router and to talk with other instances only within a subnet. You cannot cross a subnet with LLAs.

## Bring Your Own IP Addresses (BYOIP)

You can bring part or all of your public IPv4 address range from your on-premises network to your AWS account. You continue to own the address range, but AWS advertises it on the internet. After you bring the address range to AWS, it appears in your account as an address pool. You can create an Elastic IP address from your address pool and use it with your AWS resources, such as EC2 instances, NAT gateways, and Network Load Balancers.

---

**Important:** BYOIP is not available in all Regions.

---

The following requirements have to be met:

- The address range must be registered with your Regional internet registry (RIR), such as the American Registry for Internet Numbers (ARIN), R?seaux IP Europ?ens Network Coordination Centre (RIPE), or Asia-Pacific Network Information Centre (APNIC). It must be registered to a business or institutional entity and can not be registered to an individual person.
- The most specific address range that you can specify is /24.
- You can bring each address range to one Region at a time.
- You can bring five address ranges per Region to your AWS account.
- The addresses in the IP address range must have a clean history. We might investigate the reputation of the IP address range and reserve the right to reject an IP address range if it contains an IP address that has poor reputation or is associated with malicious behavior.
- The following are supported:
  - ARIN - “Direct Allocation” and “Direct Assignment” network types
  - RIPE - “ALLOCATED PA”, “LEGACY”, and “ASSIGNED PI” allocation statuses
  - APNIC ? “ALLOCATED PORTABLE” and “ASSIGNED PORTABLE” allocation statuses

To ensure that only you can bring your address range to your AWS account, you must authorize Amazon to advertise the address range. You must also provide proof that you own the address range through a signed authorization message.

A Route Origin Authorization (ROA) is a cryptographic statement about your route announcements that you can create through your RIR. It contains the address range, the Autonomous System numbers (ASN) that are allowed to advertise the address range, and an expiration date. An ROA authorizes Amazon to advertise an address range under a specific AS number. However, it does not authorize your AWS account to bring the address range to AWS. To authorize your AWS account to bring an address range to AWS, you must publish a self-signed X509 certificate in the Registry Data Access Protocol (RDAP) remarks for the address range. The certificate contains a public key, which AWS uses to verify the authorization-context signature that you provide. You should keep your private key secure and use it to sign the authorization-context message.

## VPC DHCP

It is not supported to use an specific DHCP that you provide, you have to use the DHCP given by AWS. You need to configure the DHCP options set as show below and attach it to your VPC.

If you want to change the DCHP options, the only way to do it is by creating a new DHCP options set and attach it to the VPC. You cannot change a DHCP options set when it is live.

The *Domain name* is the name you will like to have inside your EC2 instance, completing unqualified DNS hostnames. In *Domain name servers* field you can include up to 4 Domain name servers to be configured in the VPC instances. In AWS, there are some DNS servers available. You can use up to 4 *NTP servers*.

When you want to do a DNS request from an instance, you can query the +2 IP address of the subnet to which it belongs to. Another way to do a DNS request is by querying 169.254.169.253 which proxies to the +2 DNS resolver.

There are no DNS hostnames automatically assigned for IPv6 addresses. When you launch a EC2 instance you get automatically a private DNS hostname with this form:

## Create DHCP options set ✕

Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network. The options field of a DHCP message contains configuration parameters.

**Name tag**  ⓘ

Specify at least one of the following configuration parameters

**Domain name**  ⓘ

**Domain name servers**  ⓘ

**NTP servers**  ⓘ

**NetBIOS name servers**  ⓘ

**NetBIOS node type**  ⓘ

Cancel Yes, Create

```
ip-private-ipv4-address.region.compute.internal
```

And optionally a public DNS hostname with this form:

```
ec2-public-ipv4-address.region.amazonaws.com
```

## Security groups

Security groups are stateful firewalls. When an EC2 instance talk to another EC2 instance needs to create an entry in the connections table in order to cross the stateful firewall. It will look at the outbound rules of the EC2 instance initiating the connection and look at the inbound rules of the EC2 instances receiving the connection. If both the outbound rules of the source EC2 and the inbound rules of the destination EC2 allows the connection, then it is established. When the response from this connection comes back to the firewall, it is allowed.

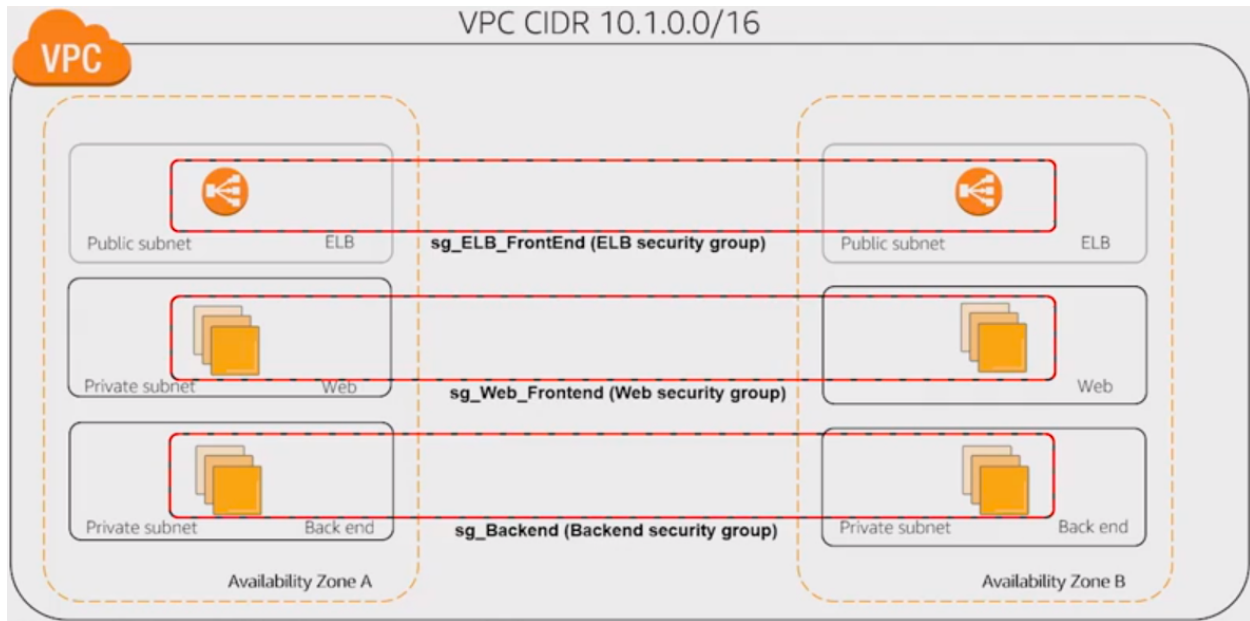
Suppose you have a VPC as illustrated below. Initially, we could think that if we want ELB Frontend to communicate with the web frontend servers, we need to know ELB IP address, which can change dynamically. Analogously, we have the same situation between the Web Frontend and the Back end servers.

The correct way to deal with these scenarios is by tiering security groups. The `sg_ELB_Frontend` allow traffic from any IP address via HTTPS. The `sg_Web_Frontend` allow traffic from only from ELB\_Frontend via port 8443 by using the `sg_ELB_Frontend` ID as the source in the security group.

The *Port* field can be a number or a range (for example: 1000-2000). You cannot define Ports separated by commas (for example: 1000,1001,1002). Instead, you have to define different rules for each of this ports and define a range of addresses if they are contiguous.

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, You can have a maximum of 120 (60 inbound and 60 outbound) rules per security group and up to 5 security groups attached to each network interface of the EC2 instance. Security groups act at the instance level,





VPC Dashboard

Filter by VPC: None

Virtual Private Cloud

Your VPCs

Subnets

Route Tables

Internet Gateways

DHCP Options Sets

Create Security Group Delete Security Group

Filter VPC security groups

<input type="checkbox"/>	Name tag	Group ID	Group Name	VPC	Description
<input type="checkbox"/>	sg_ELB_FrontEnd	sg-9142e0f7	sg_ELB_FrontEnd	vpc-9df15bf9 (10.50.0.0/16...	ELB Security Group
<input type="checkbox"/>	sg_Backend	sg-3e40e258	sg_Backend	vpc-9df15bf9 (10.50.0.0/16...	Backend Security Group
<input type="checkbox"/>	sg_Web_Frontend	sg-7640e210	sg_Web_Frontend	vpc-9df15bf9 (10.50.0.0/16...	Web Security Group

Filter VPC security groups

<input type="checkbox"/>	Name tag	Group ID	Group Name	VPC	Description
<input checked="" type="checkbox"/>	sg_ELB_FrontEnd	sg-9142e0f7	sg_ELB_FrontEnd	vpc-9df15bf9	ELB Security Group
<input type="checkbox"/>	sg_Backend	sg-3e40e258	sg_Backend	vpc-9df15bf9 (10.50.0.0/16...	Backend Security Group
<input type="checkbox"/>	sg_Web_Frontend	sg-7640e210	sg_Web_Frontend	vpc-9df15bf9	Web Security Group

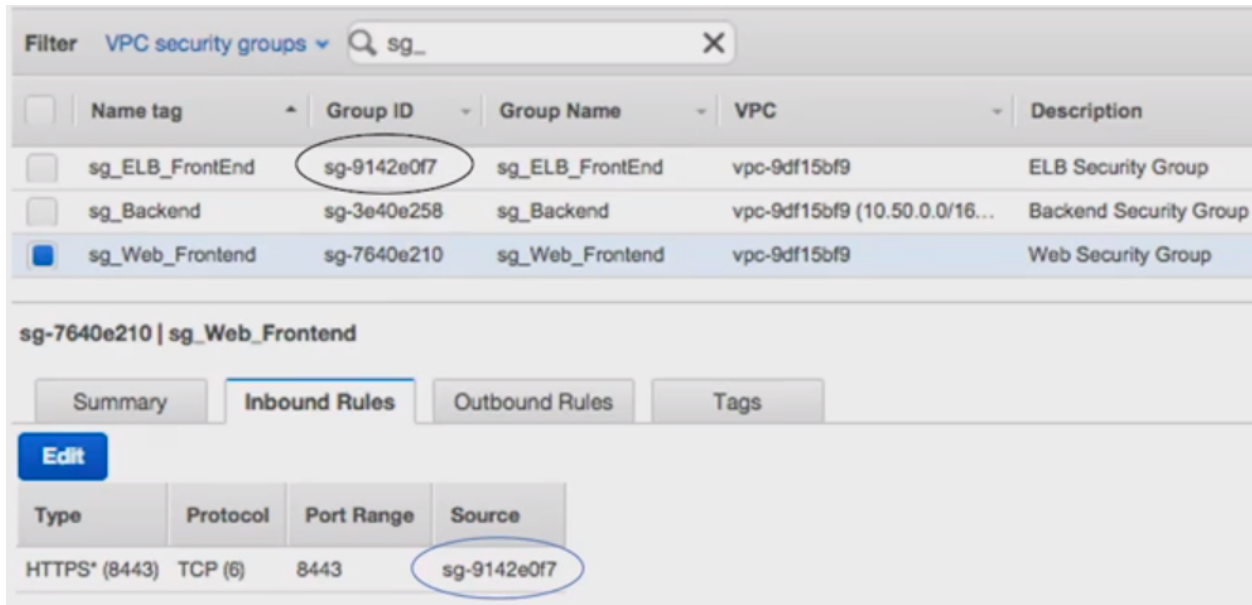
sg-9142e0f7 | sg\_ELB\_FrontEnd

Summary Inbound Rules Outbound Rules Tags

Edit

Type	Protocol	Port Range	Source
HTTPS (443)	TCP (6)	443	0.0.0.0/0





not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

## Network ACLs

A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. Network ACLs are stateless firewall rules. If you have a client that wants to communicate with an EC2 instance within a subnet in which the NACLs applied, some inbound rules allowing this traffic need to be present. When the EC2 replies, the outbound rules of the NACLs applied as well because it is a stateless firewall. As the response can use ephemeral ports, the outbound rules should allow all ports.

With NACLs, you can define ALLOW and DENY rules. The default ACLs allow all traffic in and all traffic out. Most customers use NACLs rules with DENY. In security groups, there are no ALLOW, if it is not allowed, is denied by default. They have a rule number that defines the priority, that is, the order in which they are applied. Network ACL Rules are evaluated by rule number, from lowest to highest, and executed immediately when a matching allow/deny rule is found.

The following are the basic things that you need to know about network ACLs:

- Your VPC automatically comes with a modifiable default network ACL. By default, it allows all inbound and outbound IPv4 traffic and, if applicable, IPv6 traffic.
- You can create a custom network ACL and associate it with a subnet. By default, each custom network ACL denies all inbound and outbound traffic until you add rules.
- Each subnet in your VPC must be associated with a network ACL. If you don't explicitly associate a subnet with a network ACL, the subnet is automatically associated with the default network ACL.
- You can associate a network ACL with multiple subnets; however, a subnet can be associated with only one network ACL at a time. When you associate a network ACL with a subnet, the previous association is removed.
- A network ACL contains a numbered list of rules that we evaluate in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with the network ACL. The highest number that you can use for a rule is 32766. We recommend that you start by creating rules in increments (for example, increments of 10 or 100) so that you can insert new rules where you need to later on.

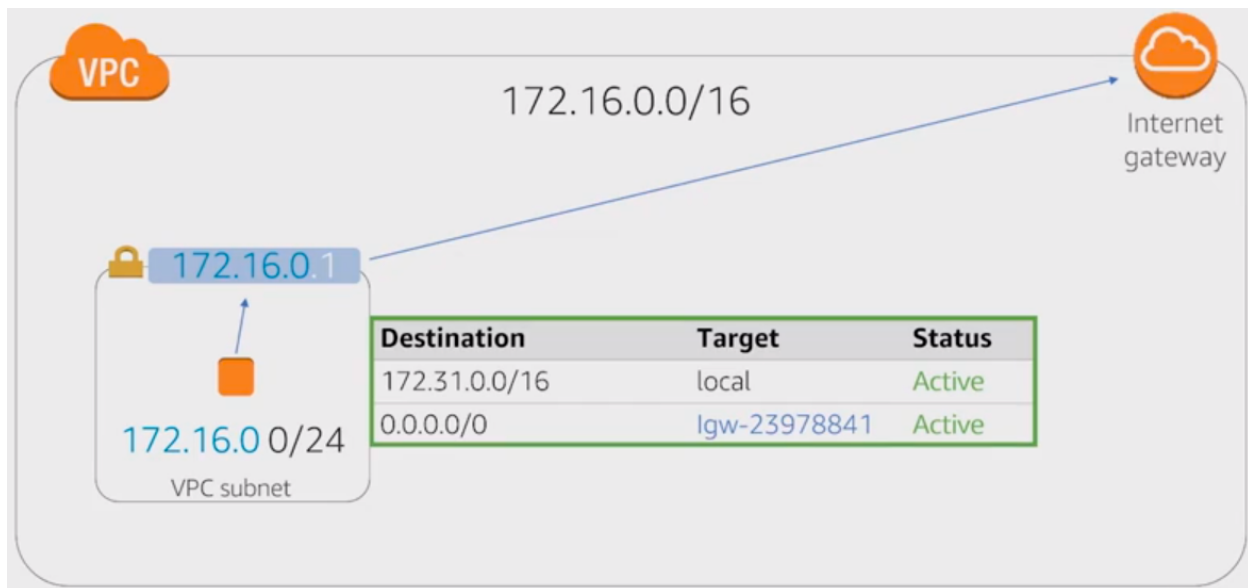
- A network ACL has separate inbound and outbound rules, and each rule can either allow or deny traffic.
- Network ACLs are stateless; responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

AWS re:Invent 2018: Your Virtual Data Center: VPC Fundamentals and Connectivity Options (NET201)

## Amazon VPC: Route tables and gateways

### VPC subnets route table

The route tables allows you to route to some gateway. The route table get applied to router of the subnet itself (+1 of the subnet itself). The default gateway of the EC2 instance (0.0.0.0/0) always will be pointing to the router of the subnet. In this router, you will have a route table with an entry with destination the VPC CIDR blocks and the target local. Local means that you can route to every resource in the VPC. The local entry in the route table has priority over the rest of the routes. The other entry in the route table has a destination 0.0.0.0/0 and target the internet gateway. The **internet gateway** allow an EC2 instance to route traffic to Internet or any resource in the Internet route traffic to the EC2 instance.



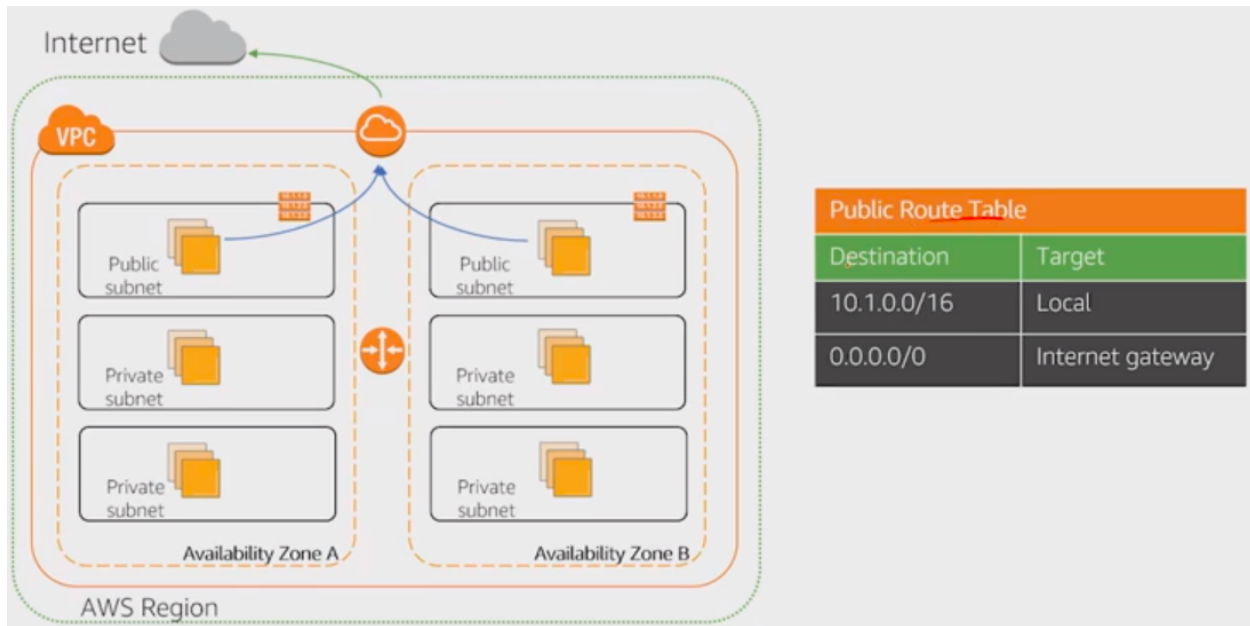
### VPC: Gateways

A **NAT gateway** is a managed address translation gateway.

IPv6 addresses on EC2 instances are public addresses, so that if we connect them to internet gateway there would be reachable from Internet. The **egress-only internet gateway** is a gateway for IPv6 only, that allows EC2 instances with IPv6 addresses to reach Internet, but they are not reachable from Internet.

### VPC Routing: Public subnet

What makes a subnet public is its route table. It has an entry with destination 0.0.0.0/0 and target the Internet Gateway. You would need a public IP address attached to the network interface of EC2 instance in this subnet. The packets coming from the EC2 instance to the Internet Gateway have a private IP address and they are translated to the public IP address attached to it thanks to the Internet Gateway.



You do not need to manage the Internet Gateway. It scales out or in depending on the amount the traffic it has to handle. The maximum throughput it can support is 5Gbps.

To enable access to or from the Internet for instances in a VPC subnet, you must do the following:

- Attach an internet gateway to your VPC.
- Ensure that your subnet's route table points to the Internet gateway.
- Ensure that instances in your subnet have a globally unique IP address (public IPv4 address, Elastic IP address, or IPv6 address).
- Ensure that your network access control lists and security groups allow the relevant traffic to flow to and from your instance.

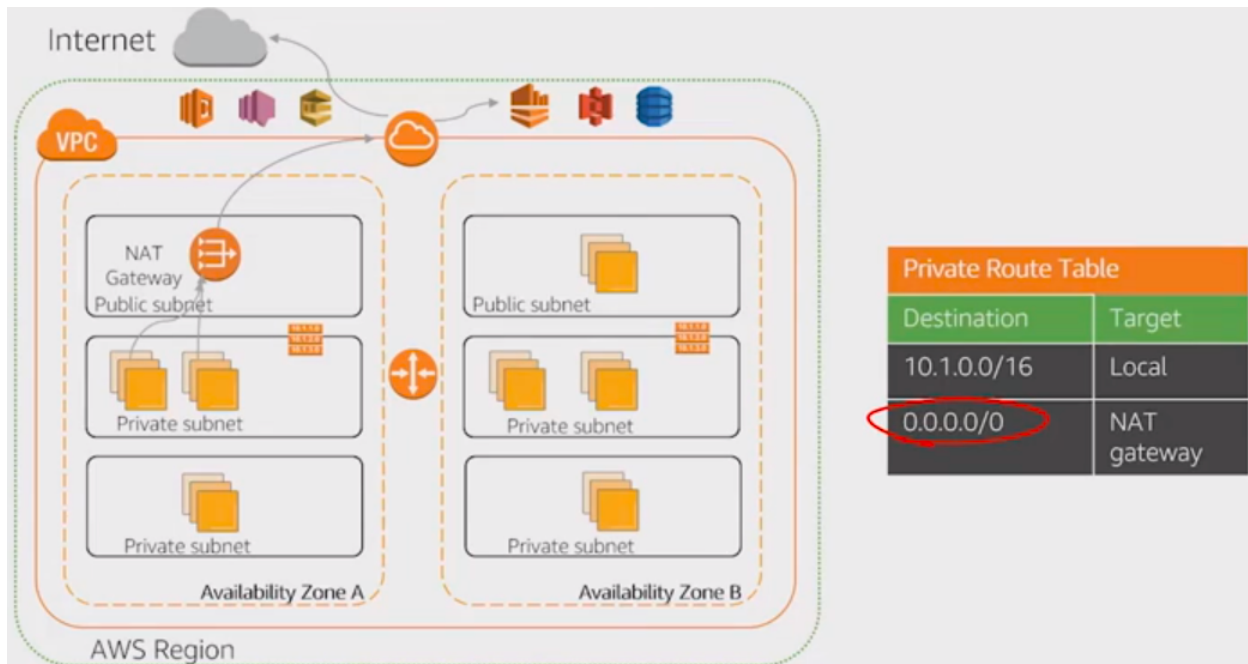
### VPC Routing: Private subnet

If an EC2 in a private subnet wants to communicate to the Internet, this subnet needs to be attached to a NAT gateway which lives in a public subnet. There is an entry in the route table in which the destination is 0.0.0.0/0 and the target is the NAT gateway.

The NAT gateway allows the EC2 with a private address to reach the Internet but the EC2 instance is not reachable from the Internet. The NAT gateway requires an Elastic IP address that will be used to translate the private address of the EC2 instance. Even if you are using a NAT gateway, you are still passing through an Internet Gateway to reach the Internet. You have one Internet Gateway per VPC but you can have multiple NAT gateways inside your VPC. The reason to have more than one is that they are highly available within an AZ, not across AZs.

You can achieve HA across AZs by launching a NAT gateway in each AZ. If one AZ is down, all subnets associated to that AZ are down. If you have high availability, you would have multiple subnets spanning multiple zones: each zone's private subnets would point to a NAT Gateway in that zone's public subnet, with separate routing tables for each zone.

A NAT gateway serves separate subnets and can scale up to 45 Gbps. There is a limit of 55000 connections towards the same destination. If you need more than 45 Gbps or more than 55000 connections towards the same destination, then you will need more than one NAT gateway. In that case, you will need another subnet with a different route table that will point towards the second NAT gateway.



NAT gateway hourly usage and data processing rates apply. Amazon EC2 charges for data transfer also apply. NAT gateways are not supported for IPv6 traffic? use an egress-only internet gateway instead.

To create a NAT gateway, you must specify the public subnet in which the NAT gateway should reside. You must also specify an Elastic IP address to associate with the NAT gateway when you create it. The Elastic IP address cannot be changed once you associate it with the NAT Gateway.

After you've created a NAT gateway, you must update the route table associated with one or more of your private subnets to point Internet-bound traffic to the NAT gateway. This enables instances in your private subnets to communicate with the internet. Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone. You have a limit on the number of NAT gateways you can create in an Availability Zone.

AWS offers two kinds of NAT devices: a NAT gateway or a NAT instance. It is recommended to use NAT gateways, as they provide better availability and bandwidth over NAT instances. The NAT Gateway service is also a managed service that does not require your administration efforts. A NAT instance is launched from a NAT AMI.

Here is a diagram showing the differences between NAT gateway and NAT instance:

### EC2 instance as in-line next-hop

If you want to implement your own NAT gateway or IDS or IPS, ... by using an EC2 instance, then you will need to use it as in-line next-hop. There are several methods to implement it.

#### Method 1

You will need to define an entry in the route table with destination 0.0.0.0/0 and the target is an ENI (Elastic Network Interface) of an EC2 instance within a public subnet. This EC2 instance has reachability to the Internet via an Internet Gateway.

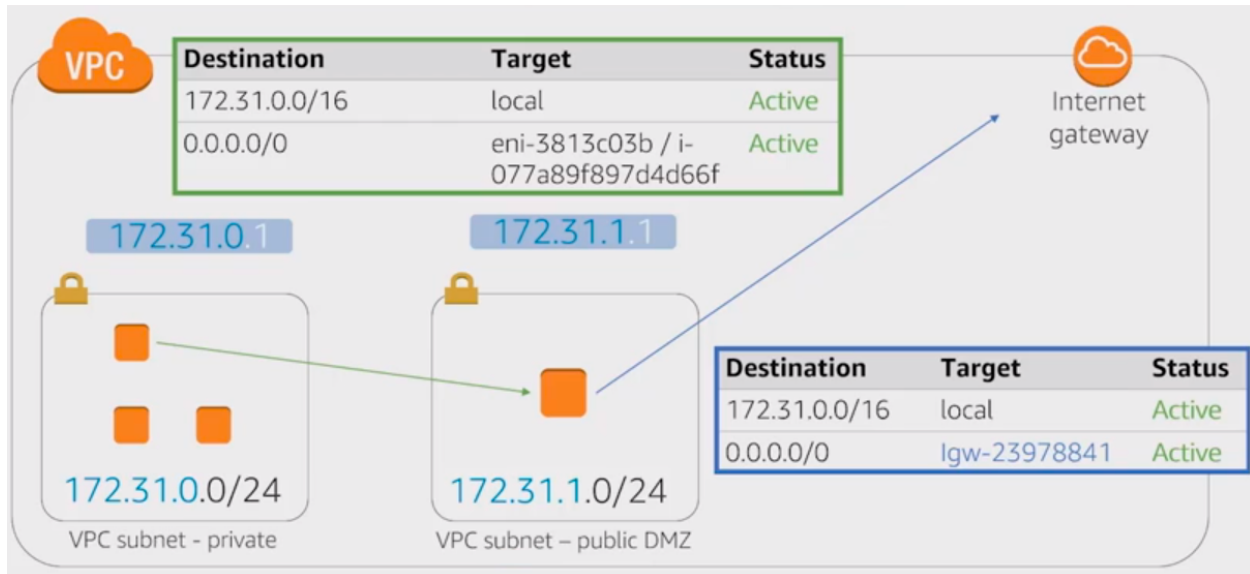
The responsible for translating the private address of the EC2 to the public address is the Internet Gateway.





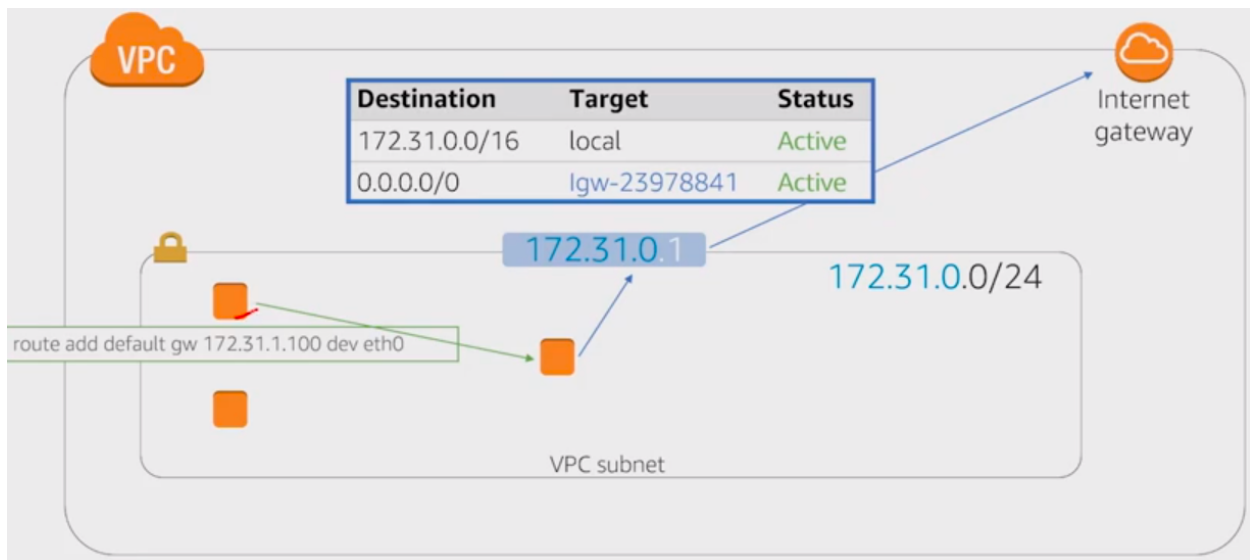
Tutorials Dojo

Attribute	NAT gateway	NAT instance
Availability	Highly available. NAT gateways in each Availability Zone are implemented with redundancy. Create a NAT gateway in each Availability Zone to ensure zone-independent architecture.	Use a script to manage failover between instances
Bandwidth	Can scale up to 45 Gbps.	Depends on the bandwidth of the instance type
Maintenance	Manage by AWS	Manage by you.
Performance	Software is optimized for handling NAT traffic	A generic Amazon Linux AMI that's configured to perform NAT
Cost	Charged depending on the number of NAT gateways you use, duration of usage, and amount of data that you send through the NAT gateways.	Charged depending on the number of NAT instances that you use, duration of usage, and instance type and size.
Type and size	Uniform offering; you don't need to decide on the type or size.	Choose a suitable instance type and size, according to your predicted workload
Public IP addresses	Choose the Elastic IP address to associate with a NAT gateway at creation.	Use an elastic IP address or a public IP address with a NAT instance. You can change the public IP address at any time by associating a new elastic IP address with the instance.
Private IP addresses	Automatically selected from the subnet's IP address range when you create the gateway.	Assign a specific private IP address from the subnet's IP address range when you launch the instance.
Security groups	Cannot be associated with a NAT gateway	Associate with your NAT instance and the resources behind your NAT instance to control inbound and outbound traffic.
Network ACLs	Use a network ACL to control the traffic to and from the subnet in which your NAT gateway resides.	Use a network ACL to control the traffic to and from the subnet in which your NAT instance resides.
Flow logs	Use flow logs to capture the traffic.	Use flow logs to capture the traffic.
Port Forwarding	Not supported.	Manually customize the configuration to support port forwarding.
Bastion Servers	Not supported.	Use as a bastion server.
Traffic Metrics	Monitor your NAT gateway using cloudwatch.	View Cloudwatch metrics for the instance.
Timeout Behavior	When a connection times out, a NAT gateway returns an RST packet to any resources behind the NAT gateway that attempt to continue the connection (it does not send a FIN packet).	When a connection times out, a NAT instance sends a FIN packet to resources behind the NAT instance to close the connection.
IP Fragmentation	Supports forwarding of IP fragmented packets for the UDP protocol. Does not support fragmentation for the TCP and ICMP protocols. Fragmented packets for these protocols will get dropped.	Supports reassembly of IP fragmented packets for the UDP, TCP, and ICMP protocols.



## Method 2

Another method is to configure a route inside the EC2 and redirects it to another EC2 instance inside the same subnet. The latter EC2 instance sends the traffic to the default gateway (+1 of subnet network) and its route table has the entry for destination 0.0.0.0/0 pointing to the Internet Gateway. The responsible for translating the private address of the EC2 to the public address is the Internet Gateway.



## VPC endpoints for AWS services

It allows to communicate with some services directly within your VPC. There are two types of VPC: interface and gateway.

Table 7: VPC endpoints for AWS services

Endpoint Type	Description	Supported Services
Interface (powered by PrivateLink)	An elastic network interface with a private IP address that serves as an entry point for the traffic destined to a supported AWS service.	Amazon Kinesis, Elastic Load Balancing API, Amazon EC2 API, AWS Systems Manager, AWS Service Catalog, Amazon SNS, AWS SNS.
Gateway	A gateway that is a target for a specified route in your route table, used for traffic destined to a supported AWS service.	Amazon S3, Amazon DynamoDB.

The characteristics of an **Interface VPC endpoint** are the following:

- There is one interface per AZ. It is an ENI from an EC2 instance which lives in an AZ.
- It does not support endpoint policies.
- You can apply a security group.
- You access over AWS Direct Connect, but not through AWS VPN.
- It makes requests to the service using endpoint-specific DNS hostnames. Optionally, you can use Amazon Route 53 DNS private hosted zone to make requests to the service using its default public DNS name.

**Note:** PrivateLink

PrivateLink is a purpose-built technology designed to access AWS services, while keeping all the network traffic within the AWS network. There is no need for proxies, NAT gateways or Internet Gateways.

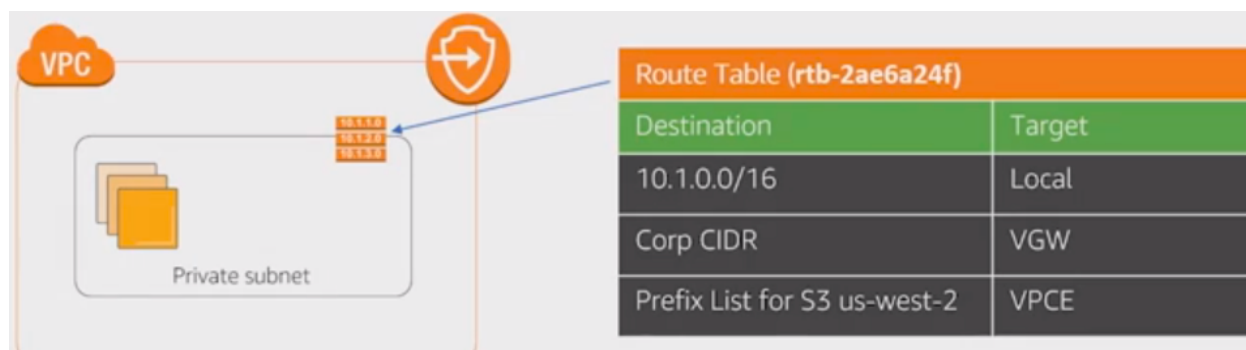
The characteristics of an **Gateway VPC endpoint** are the following:

- It supports multiple AZs.
- It supports the use of endpoint policies.
- It cannot be accessed neither over AWS Direct Connect nor over VPN.
- It makes requests to the service using its default public DNS name.

An example of creating an Amazon S3 VPC endpoint is the following:

```
$ aws ec2 create-vpc-endpoint --vpc-id vpc-40f18d25 --service-name com.amazonaws.us-west-2.s3 --route-table-ids rtb-2ae6a24f
```

You need an entry in the route table with destination the prefix list for the S3 endpoint and the target the VPC endpoint.



That prefix list is built by AWS. It is a logical route destination target that dynamically translates to service IPs. Amazon S3 IP ranges change over time and Amazon S3 prefix lists abstract change.

```
$ aws ec2 describe-prefix-lists

{
  "PrefixLists": [
    {
      "PrefixListName": "com.amazonaws.us-east-1.s3",
      "Cidrs": [
        "54.231.0.0/17"
      ],
      "PrefixListId": "pl-63a5400a"
    }
  ]
}
```

The VPC endpoint policy allows you to control VPC access to Amazon S3. An example could be the following:

```
{
  "Statement": [
    {
      "Sid": "vpce-restrict-to-my_secure_bucket",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3::my_secure_bucket",
        "arn:aws:s3::my_secure_bucket/*"
      ]
    }
  ]
}
```

Having a VPC endpoint policy in place does not mean that you have access to a bucket. There are bucket policy that can allow or deny access to it. For instance, you can restrict that the bucket must be accesible only from the S3 VPC endpoint:

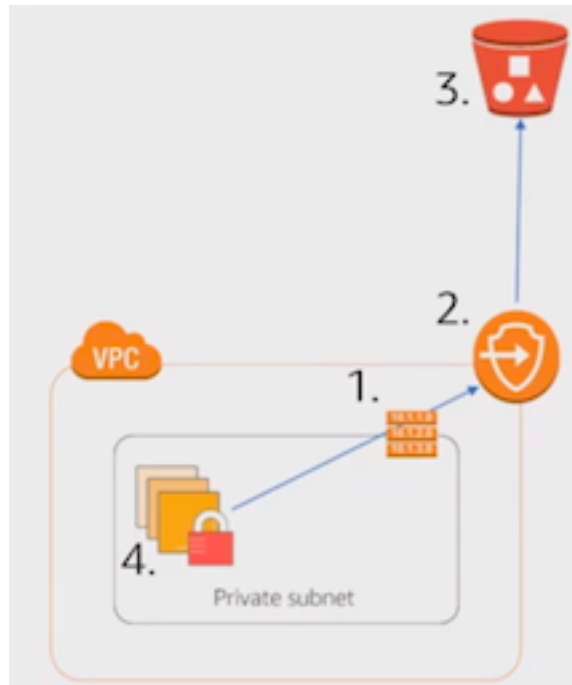
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "bucket-restrict-to-specific-vpce",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3::my_secure_bucket",
        "arn:aws:s3::my_secure_bucket/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:sourceVpce": [
            "vpce-bc42a4e5"
          ]
        }
      }
    }
  ]
}
```

As a summary, you can control VPC access to Amazon S3 via several security layers:

1. Route table association.
2. VPC endpoint policy.



3. Bucket policy.
4. EC2 security groups with prefix list.



**Note:** Prefix list. You can use prefix list with security groups but not with NACLs.

## Amazon EC2 networking

### Elastic network interface

An elastic network interface (ENI) is a logical networking component in a VPC that represents a virtual network card. You can attach a network interface to an EC2 instance in the following ways:

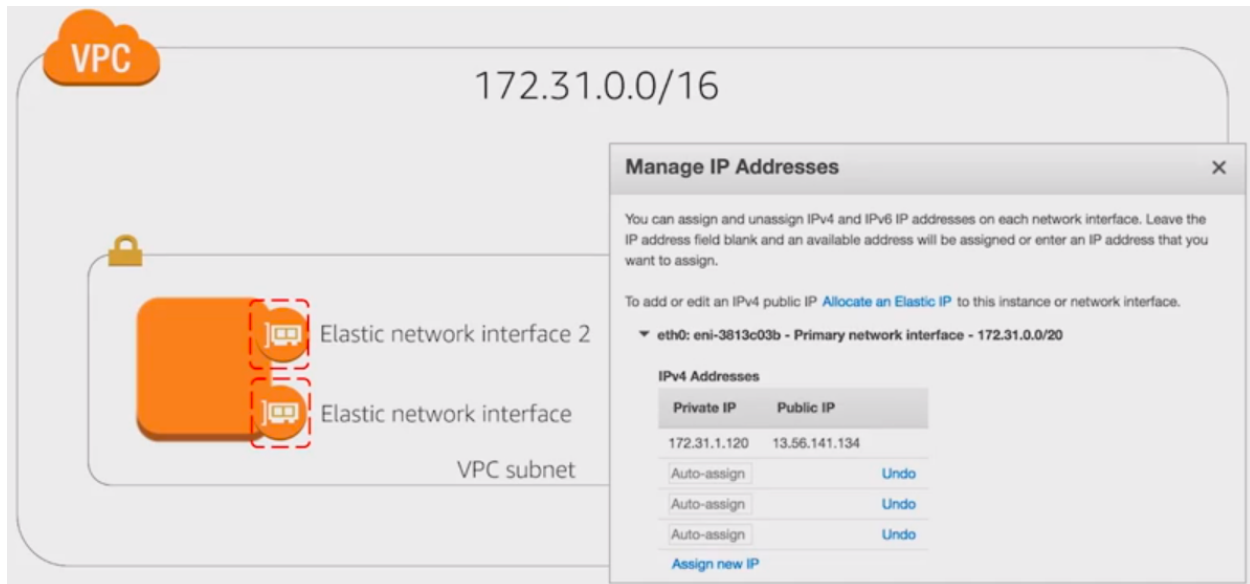
- When it's running (hot attach).
- When it's stopped (warm attach).
- When the instance is being launched (cold attach).

You always have an ENI attached to an instance, it is the default ENI and cannot be modified. You can attach more than one ENI to an EC2 instance and they can be moved to another EC2 instance. In general, the ENIs of an EC2 instance are attached to different subnets. Each ENI have a private IP address and a public IP address, optionally you can have up to 4 private IP addresses and 4 public IP addresses.

The ENIs will be attached to different VPC subnets in the same AZ. The ENIs are used to talk among EC2 instances but also between an EC2 instance and an EBS volume. Some EC2 instance types support EBS optimized dedicated throughput, which allows you to have dedicated throughput to access an ENS volume.

If we need to optimize network performance between 2 EC2 instances, the supported throughput is based on the instace type.

The 25 Gigabit and 10 Gigabit performance is measured one way. you have to double that for birdirectional (full duplex). Sometimes the Amazon EC2 network performance is classified as high, moderate or low. It is a function of



Instance type	EBS-optimized by default	Maximum bandwidth (Mbps)	Maximum throughput (MB/s, 128 KB I/O)	Maximum IOPS (16 KB I/O)
c1.xlarge		1,000	125	8,000
c3.xlarge		500	62.5	4,000
c3.2xlarge		2,000	250	16,000
c3.4xlarge	Yes	500	62.5	4,000
c4.large	Yes	750	93.75	6,000
c4.2xlarge	Yes	1,000	125	8,000
c4.4xlarge	Yes	2,000	250	16,000
c4.8xlarge	Yes	4,000	500	32,000
c5.large*	Yes	2,250	281	16,000
c5.xlarge*	Yes	2,250	281	16,000
c5.2xlarge*	Yes	2,250	281	16,000
c5.4xlarge	Yes	2,250	281	16,000
c5.9xlarge	Yes	4,500	563	32,000
c5.18xlarge	Yes	9,000	1,125	64,000
d2.xlarge	Yes	750	93.75	6,000

Instance type	vCPU	Mem (GiB)	Storage (GB)	Networking Perf.	CPU	Clock Speed (GHz)	Intel AVX	Intel AVX2	Intel AVX-512	Intel Turbo	EBS OPT	Enhanced Networking
c5.large	2	4	EBS Only	Up to 10 Gbps	Intel Xeon Platinum	3.0	No	No	Yes	Yes	Yes	Yes
c5.xlarge	4	8	EBS Only	Up to 10 Gbps	Intel Xeon Platinum	3.0	No	No	Yes	Yes	Yes	Yes
c5.2xlarge	8	16	EBS Only	Up to 10 Gbps	Intel Xeon Platinum	3.0	No	No	Yes	Yes	Yes	Yes
c5.4xlarge	16	32	EBS Only	Up to 10 Gbps	Intel Xeon Platinum	3.0	No	No	Yes	Yes	Yes	Yes
c5.9xlarge	36	72	EBS Only	10 Gigabit	Intel Xeon Platinum	3.0	No	No	Yes	Yes	Yes	Yes
c5.18xlarge	72	144	EBS Only	25 Gigabit	Intel Xeon Platinum	3.0	No	No	Yes	Yes	Yes	Yes

the instance size, but not all instances are created equal, so it is recommended to test it with iperf or another network performance measurement tool. Burstable network performance are defined in C5, I3 and R4 EC2 instance types. It is supported 5 GB maximum per instance on egress traffic destined outside the VPC. You can go above the 5 GB by using placement groups or VPC endpoints.

## Enhanced networking

There are 3 main strategies to get towards the 25 Gbps: Device pass-through, Elastic Network Adapter (ENA) and Intel Data Plane Development Kit (DPDK).

### Device pass-through

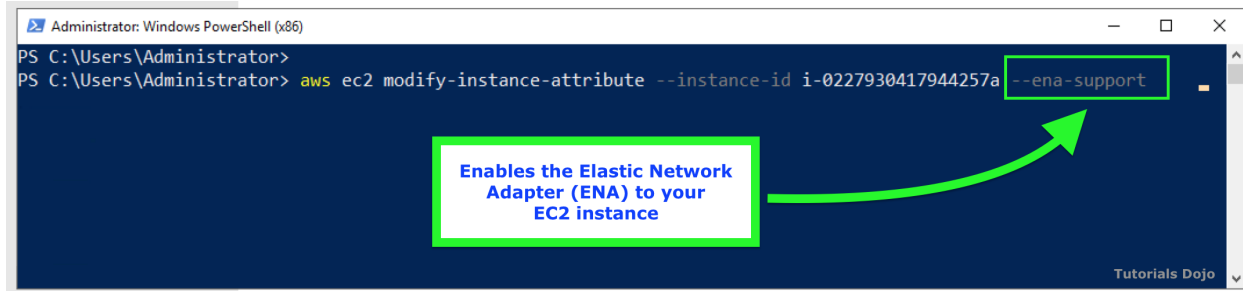
In virtualization, single root input/output virtualization or SR-IOV is a specification that allows the isolation of the PCI Express resources for manageability. It allows PCIe devices to appear as multiple separate physical PCIe devices using Virtual Functions (VF). It provides higher bandwidth, higher packet per second (PPS) performance, and consistently lower inter-instance latencies. Enhanced networking requires a specialized driver, which means:

- Your instance OS must know about it.
- Amazon EC2 must be notified that your instance can use it.

### Elastic Network Adapter (ENA)

It is the next generation of enhanced networking that supports HW checksums, multi-queue, receive side steering. It can achieve 25 Gbps of throughput in a placement group. It is an open-source Amazon network driver. There are items for advanced tuning:

- Multiple Tx/Rx queue pairs (the maximum number is advertised by the device through the administration queue).
- CPU cache line optimizes data placement.
- Checksum offload and TCP transmit segmentation offload (TSO).
- Support for receive-side scaling (RSS) for multi-core scaling.
- Depending on ENA device, support for low-latency queue (LLQ), which saves more microseconds.



An Elastic Fabric Adapter (EFA) is simply an Elastic Network Adapter (ENA) with added capabilities. It provides all of the functionality of an ENA, with additional OS-bypass functionality. OS-bypass is an access model that allows HPC and machine learning applications to communicate directly with the network interface hardware to provide low-latency, reliable transport functionality.

The OS-bypass capabilities of EFAs are not supported on Windows instances. If you attach an EFA to a Windows instance, the instance functions as an Elastic Network Adapter, without the added EFA capabilities.

### Intel Data Plane Development Kit

It is a set of libraries and drivers for fast packet processing. It is supported for both the Intel 82599 Virtual Interface and the ENA driver. It is a software accelerator that runs in user space, bypassing the Linux kernel and providing access to NICs, CPUs, and memory for a packet processing application. The DPDK can:

- Receive and send packets within the minimum number of CPU cycles (usually fewer than 80 cycles).
- Develop fast capture algorithms (tcpdump-like)
- Run third-party fast path stacks.

There are some use cases: firewalls, real-time communication processing, HPC, and network appliances.

### Placement groups

When you want a group of EC2 instances to talk very fast and with high throughput, you may need to include them inside placement groups.

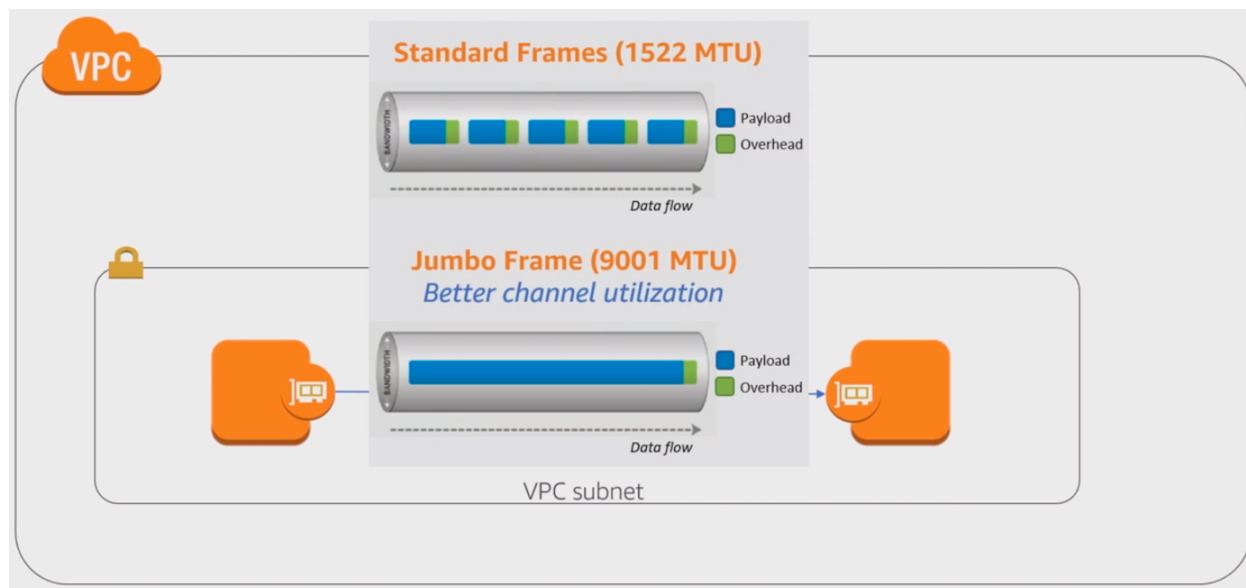
When you launch EC2 instances in the same AZ, it doesn't mean that they are going to be in the same data center. An AZ can have more than one data center. To force the EC2 instances to be very close (perhaps in the same rack) and allows you to have full Amazon EC2 bandwidth.

Use placement groups when you need high, consistent instance-to-instance bandwidth. The maximum network throughput is limited by the slower of the two instances. A placement group can span peered VPCs in the same region (may not get full bi-directional bandwidth). It can launch multiple instance types into a placement group. All traffic is limited to 5 Gb/s when exiting VPC.

It is recommended not to have different instance types in a placement group because it is more difficult to launch them very close among them.

### Jumbo frames

An standard frame in AWS has a MTU of 1522. It means that your payload has 1522 bytes and the overhead is quite big. In Jumbo frames, what allows you to do is allocate more capacity for the payload (9001 bytes) and a smaller overhead.



Every single device in between the EC2 instances must support 9001 MTU, which is not going to work when you go out the Internet. It is supported inside VPCs, but if you go through a Virtual Private Gateway or a VPN gateway, then it is not supported.

**Note:** Enable the ICMP message Type 3, Code 4 (Destination Unreachable: fragmentation needed and don't fragment set). This setting instructs the original host to adjust the MTU until the packet can be transmitted.

## Multicast

Multicast is a network capability that allows one-to-many distribution of data. With multicasting, one or more sources can transmit network packets to subscribers that typically reside within a multicast group. However, take note that Amazon VPC does not support multicast or broadcast networking.

If you need multicast, then you can use an overlay multicast. An overlay multicast is a method of building IP level multicast across a network fabric supporting unicast IP routing, such as Amazon Virtual Private Cloud (Amazon VPC).

## 1.6.2 Design and implement hybrid IT network architectures at scale

In AWS, there are 3 connectivity options:

- **Public Internet**, via public IPs or elastic IPs, taking into account that traffic out is charged by AWS.
- **VPN**, using IPSec authentication and encryption. It has 2 primary options: AWS managed VPN and software VPN (Amazon EC2).
- **AWS Direct Connect**, which establishes a private connection with AWS, separate from the Internet. It allows you to have a consistent network experience and the ability to connect through many locations globally. You can have sub-1Gps, 1Gps, 10Gps options.

Depending on the requirements (security, cost, management) that your customer has, the possible options could be different.

## Virtual Private Networks

There are mainly 2 types of VPN: site-to-site VPN and client-to-site VPN. In site-to-site VPN, you are connecting the corporate data center over the Internet towards my VPC. In the client-to-site VPN, you have users that use VPN clients to EC2 instances.

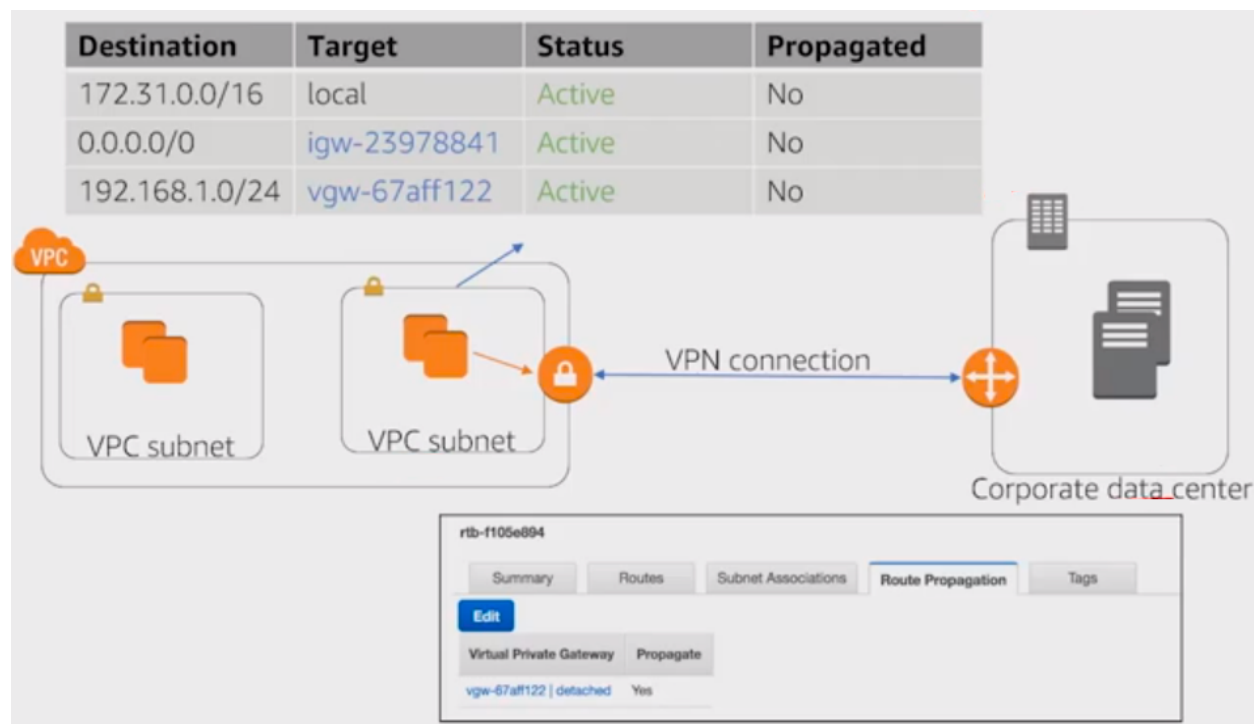
### Site-to-site VPN

There are two options of site-to-site VPN:

- Terminating the VPN connection in a **VPN Gateway (VGW)**, which is an AWS managed service that only supports IPSec protocol to setup VPNs.
- Terminating the VPN connection in 2 endpoints (for HA) on top EC2 instances.

A VGW is a fully managed gateway endpoint for your VPC with built-in multiple AZ HA. You only can have 1 VGW per VPC. It is responsible for hybrid IT connectivity, using VPN and AWS Direct Connect. It is self-sustainable entity that can be created without the requirement of a preexisting VPC (that is, it can run in detached mode). You have the ability to:

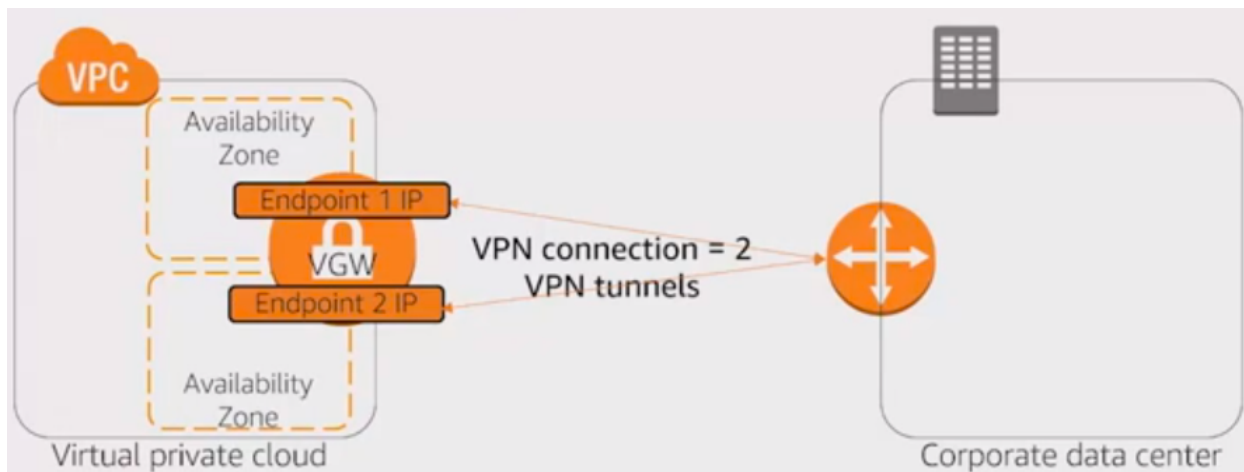
- Attach to any VPC in the same account and AWS region.
- Detach one VGW and attach another one to a VPC.
- Choose ASN at creation (BYO-ASN).



In the previous example, the IP addressing of the corporate data center is 192.168.1.0/24. In the route table, we can notice that the 192.168.1.0/24 has not been propagated so it has to be manually configured and not learned via BGP routing protocol.

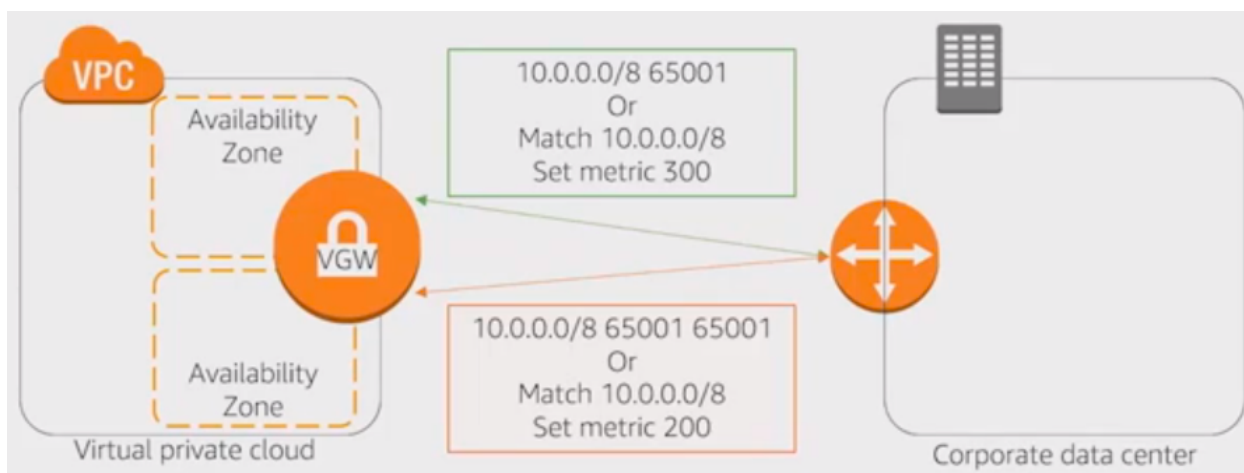
You get 2 public IP addresses with a VGW that are across 2 different service providers. The customer gateway located in the corporate data center has only 1 IP address and you have to establish more than 1 VPN connections, unless it is redundant in different service providers. You need to establish 2 IPSec tunnels towards AWS per VGW connection

because you 2 endpoints with different IP addresses. You can specify the preshared key of each one of the 2 IPSec tunnels and you can also specify the inside tunnel IP in between the two sides of the VPN.



As you have 2 tunnels, there is a risk of asymmetric routing. To avoid it and prevents to send traffic from one path and not getting it returned to the other path, you can use BGP with one of these 2 options:

- Prepend an AS path.
- Set the multi-exit discriminator (MED) attribute.



You can connect from different data centers into the same VGW. One of the features of VGW is that you can use as a CloudHub. AWS VPN CloudHub is an AWS functionality, you only need to establish 2 VPN connections and by doing that you will be able to use this CloudHub to send traffic between the 2 corporate data centers and there is no need to have direct connection between the two.

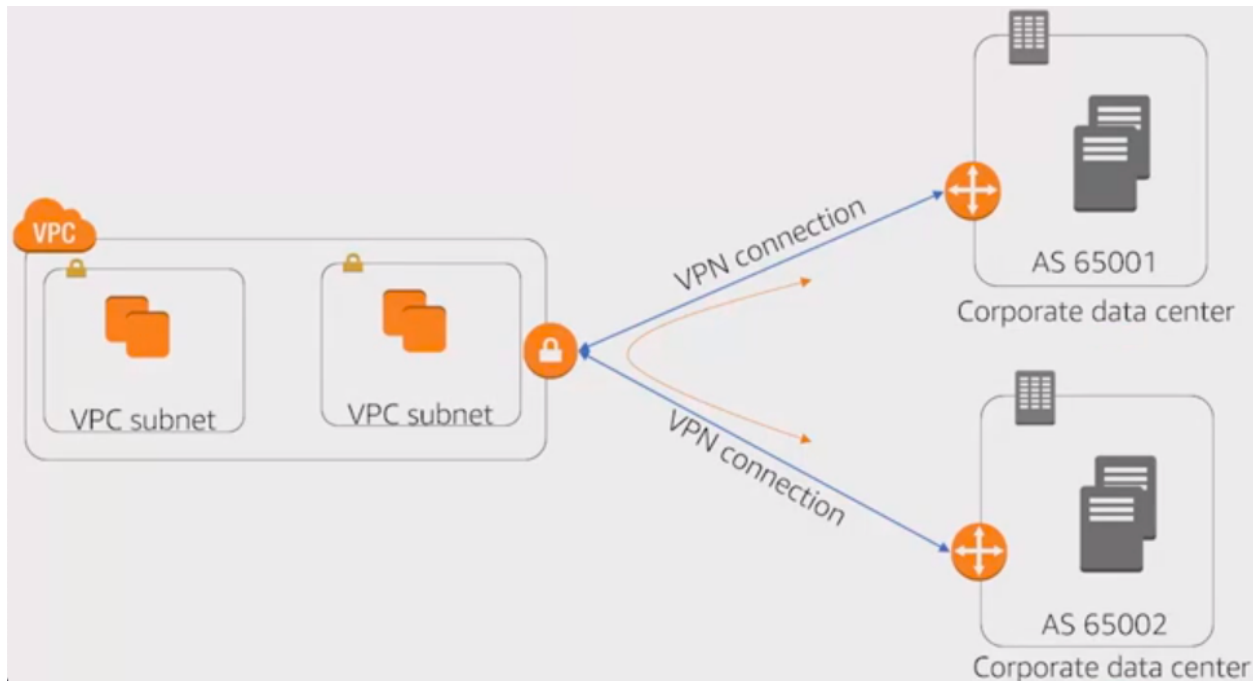
VGW doesn't initiate IPSec negotiation. AWS uses an on-demand DPD mechanism so that:

- If AWS receives no traffic from a VPN peer for 10 seconds, AWS sends a DPD "R-U-THERE" message.
- If the VPN peer does not respond to 3 successive DPDs, the VPN peer is considered dead and AWS closes the tunnel.

Static VPN (policy-based VPN) is limited to one unique security association (SA) pair per tunnel (one inbound and one outbound). Instead, you should:

- Limit the number of encryption domains (networks) that are allowed access to the VPC and consolidate.





- Configure the policy to allow “any” network (0.0.0.0/0) from behind your VPN termination endpoint to the VPC CIDR.

## Monitoring

You can monitor VPN tunnels using CloudWatch, which collects and processes raw data from the VPN service into readable, near real-time metrics. These statistics are recorded for a period of 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. VPN metric data is automatically sent to CloudWatch as it becomes available. The following metrics are available for your VPN tunnels:

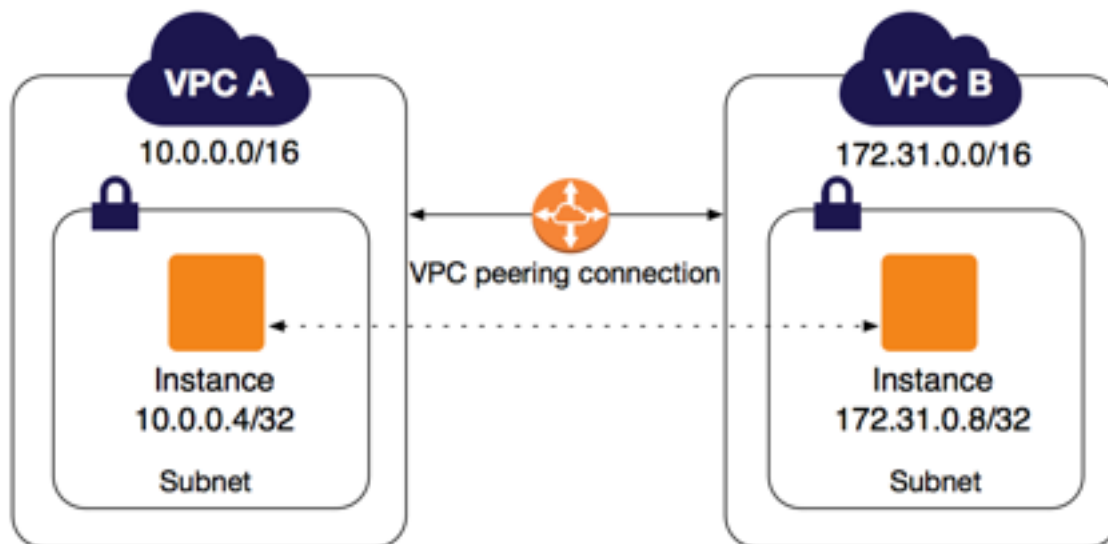
- **TunnelState.** The state of the tunnel. For static VPNs, 0 indicates DOWN and 1 indicates UP. For BGP VPNs, 1 indicates ESTABLISHED and 0 is used for all other states.
- **TunnelDataIn.** The bytes received through the VPN tunnel. Each metric data point represents the number of bytes received after the previous data point. Use the Sum statistic to show the total number of bytes received during the period. This metric counts the data after decryption.
- **TunnelDataOut.** The bytes sent through the VPN tunnel. Each metric data point represents the number of bytes sent after the previous data point. Use the Sum statistic to show the total number of bytes sent during the period. This metric counts the data before encryption.

## 1.6.3 Connecting networks

### VPC peering connection

A VPC peering connection is a network connection between two VPCs that enables you to route traffic between them using private IP addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region.





AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck.

[AWS Knowledge Center Videos: “What is VPC Peering?”](#)

The following VPC peering connection configurations are not supported.

- Overlapping CIDR Blocks.
- Transitive Peering.
- Edge to Edge Routing Through a Gateway or Private Connection.

### Overlapping CIDR Blocks

You cannot create a VPC peering connection between VPCs with matching or overlapping IPv4 CIDR blocks.



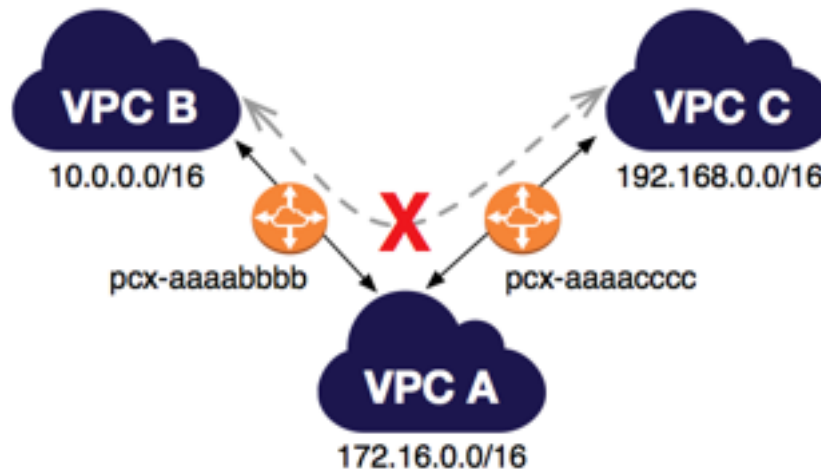
If the VPCs have multiple IPv4 CIDR blocks, you cannot create a VPC peering connection if any of the CIDR blocks overlap (regardless of whether you intend to use the VPC peering connection for communication between the non-overlapping CIDR blocks only).

This limitation also applies to VPCs that have non-overlapping IPv6 CIDR blocks. Even if you intend to use the VPC peering connection for IPv6 communication only, you cannot create a VPC peering connection if the VPCs have matching or overlapping IPv4 CIDR blocks. Communication over IPv6 is not supported for an inter-region VPC peering connection.



### Transitive Peering

You have a VPC peering connection between VPC A and VPC B (pcx-aaaabbbb), and between VPC A and VPC C (pcx-aaaacccc). There is no VPC peering connection between VPC B and VPC C. You cannot route packets directly from VPC B to VPC C through VPC A.



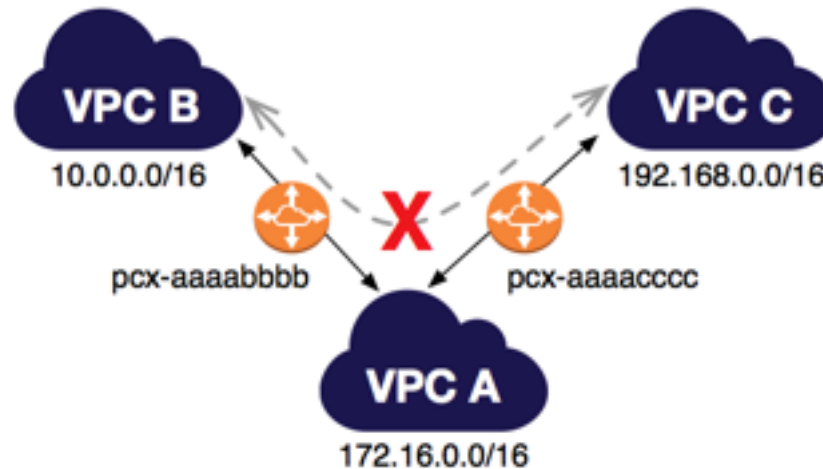
### Edge to Edge Routing Through a Gateway or Private Connection

If either VPC in a peering relationship has one of the following connections, you cannot extend the peering relationship to that connection:

- A VPN connection or an AWS Direct Connect connection to a corporate network.
- An Internet connection through an Internet gateway.
- An Internet connection in a private subnet through a NAT device.
- A VPC endpoint to an AWS service; for example, an endpoint to Amazon S3.
- (IPv6) A ClassicLink connection. You can enable IPv4 communication between a linked EC2-Classic instance and instances in a VPC on the other side of a VPC peering connection. However, IPv6 is not supported in EC2-Classic, so you cannot extend this connection for IPv6 communication.

For example, if VPC A and VPC B are peered, and VPC A has any of these connections, then instances in VPC B cannot use the connection to access resources on the other side of the connection. Similarly, resources on the other side of a connection cannot use the connection to access VPC B.

[AWS re:Invent 2018: AWS Direct Connect: Deep Dive \(NET403\)](#)



### Bastion hosts

The best way to implement a bastion host is to create a small EC2 instance which should only have a security group from a particular IP address for maximum security. This will block any SSH Brute Force attacks on your bastion host. It is also recommended to use a small instance rather than a large one because this host will only act as a jump server to connect to other instances in your VPC and nothing else.

### 1.6.4 Elastic Load Balancing (ELB)

In Elastic Load Balancing, there are various security features that you can use such as Server Order Preference, Predefined Security Policy, Perfect Forward Secrecy and many others.

Perfect Forward Secrecy is a feature that provides additional safeguards against the eavesdropping of encrypted data through the use of a unique random session key. This prevents the decoding of captured data, even if the secret long-term key is compromised. Perfect Forward Secrecy is used to offer SSL/TLS cipher suites for CloudFront and Elastic Load Balancing.

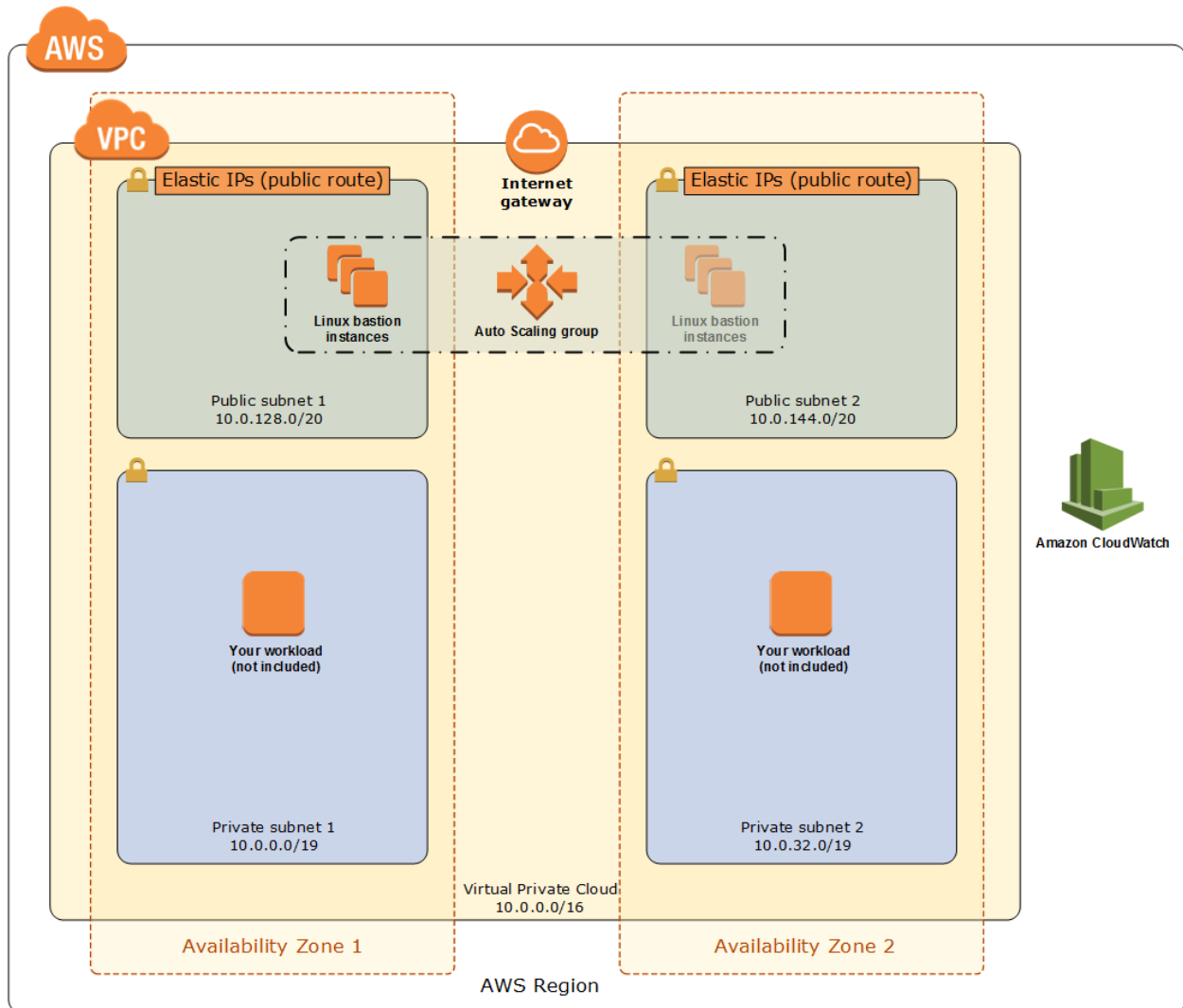
Elastic Load Balancing supports three types of load balancers. You can select the appropriate load balancer based on your application needs.

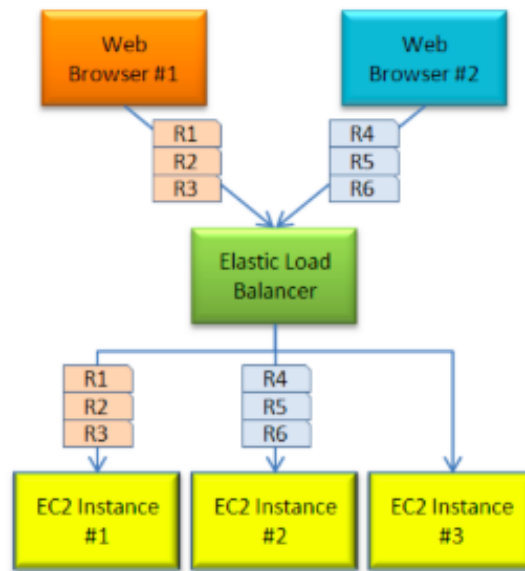
If you need flexible application management and TLS termination then we recommend that you use Application Load Balancer. If extreme performance and static IP is needed for your application then we recommend that you use Network Load Balancer. If your application is built within the EC2 Classic network then you should use Classic Load Balancer.

To ensure that a Classic Load Balancer stops sending requests to instances that are de-registering or unhealthy while keeping the existing connections open, use connection draining. This enables the load balancer to complete in-flight requests made to instances that are de-registering or unhealthy.

When you enable connection draining, you can specify a maximum time for the load balancer to keep connections alive before reporting the instance as de-registered. The maximum timeout value can be set between 1 and 3,600 seconds (the default is 300 seconds). When the maximum time limit is reached, the load balancer forcibly closes connections to the de-registering instance.

By default, a Classic Load Balancer routes each request independently to the registered instance with the smallest load. However, you can use the sticky session feature (also known as session affinity), which enables the load balancer to bind a user's session to a specific instance. This ensures that all requests from the user during the session are sent to the same instance.





The key to managing sticky sessions is to determine how long your load balancer should consistently route the user's request to the same instance. If your application has its own session cookie, then you can configure Elastic Load Balancing so that the session cookie follows the duration specified. If your application does not have its own session cookie, then you can configure Elastic Load Balancing to create a session cookie by specifying your own stickiness duration.

Sticky session feature of the Classic Load Balancer can provide session management, however, take note that this feature has its limitations such as, in the event of a failure, you are likely to lose the sessions that were resident on the failed node. In the event that the number of your web servers change when your Auto Scaling kicks in, it's possible that the traffic may be unequally spread across the web servers as active sessions may exist on particular servers. If not mitigated properly, this can hinder the scalability of your applications.

An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it evaluates the listener rules in priority order to determine which rule to apply, and then selects a target from the target group for the rule action. You can configure listener rules to route requests to different target groups based on the content of the application traffic. Routing is performed independently for each target group, even when a target is registered with multiple target groups.


Application Load Balancers support path-based routing, host-based routing and support for containerized applications.

Application Load Balancers support Weighted Target Groups routing. With this feature, you will be able to do weighted routing of the traffic forwarded by a rule to multiple target groups. This enables various use cases like blue-green, canary and hybrid deployments without the need for multiple load balancers. It even enables zero-downtime migration between on-premises and cloud or between different compute types like EC2 and Lambda.

You can use path conditions to define rules that forward requests to different target groups based on the URL in the request (also known as path-based routing). Each path condition has one path pattern. If the URL in a request matches the path pattern in a listener rule exactly, the request is routed using that rule.

A path pattern is case-sensitive, can be up to 128 characters in length, and can contain any of the following characters. You can include up to three wildcard characters.

**THEN**

**1. Redirect to...** 

Original value: #{port}


**Host** Original value: #{host}

**Path** Original value: #{path}

**Query** Original value: #{query}

AlbWt-LB A2 | HTTP:80 (1 rules)

Rule limits for condition values, wildcards, and total rules.

RULE ID	IF (all match)	THEN
last arn...3de0d	✓ Requests otherwise not routed	<p><b>1. Forward to...</b> </p> <p>Target group : Weight (0-999)</p> <p>blue 50 ✕ Traffic distribution 50%</p> <p>green 50 ✕ Traffic distribution 50%</p> <p>Select a target group 0 ✕</p> <p>Group-level stickiness <input checked="" type="checkbox"/></p> <p>+ Add action</p>

```

A?Z, a?z, 0?9
_ - . $ / ~ " ' @ : +
& (using &amp;)
* (matches 0 or more characters)
? (matches exactly 1 character)
Example path patterns

```

```

/img/*
/js/*

```

AWS re:Invent 2018: [REPEAT 1] Elastic Load Balancing: Deep Dive and Best Practices (NET404-R1)

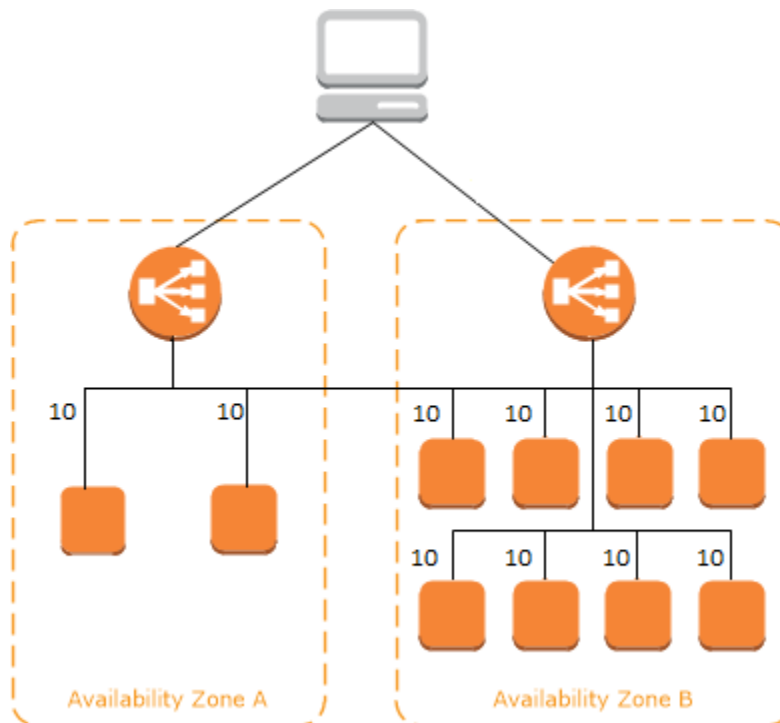
### Cross-zone load balancing

Cross-zone load balancing reduces the need to maintain equivalent numbers of instances in each enabled Availability Zone, and improves your application's ability to handle the loss of one or more instances.

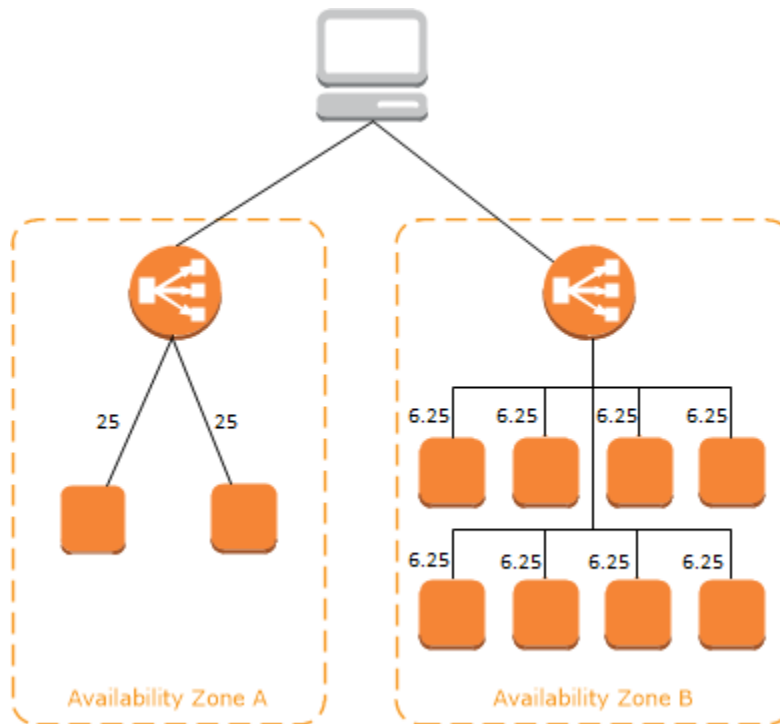
When you create a Classic Load Balancer, the default for cross-zone load balancing depends on how you create the load balancer. With the API or CLI, cross-zone load balancing is disabled by default. With the AWS Management Console, the option to enable cross-zone load balancing is selected by default. After you create a Classic Load Balancer, you can enable or disable cross-zone load balancing at any time.

The following diagrams demonstrate the effect of cross-zone load balancing. There are two enabled Availability Zones, with 2 targets in Availability Zone A and 8 targets in Availability Zone B. Clients send requests, and Amazon Route 53 responds to each request with the IP address of one of the load balancer nodes. This distributes traffic such that each load balancer node receives 50% of the traffic from the clients. Each load balancer node distributes its share of the traffic across the registered targets in its scope.

If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



If cross-zone load balancing is disabled, each of the 2 targets in Availability Zone A receives 25% of the traffic and each of the 8 targets in Availability Zone B receives 6.25% of the traffic. This is because each load balancer node can route its 50% of the client traffic only to targets in its Availability Zone.



### 1.6.5 Amazon Route 53

When you sign up for Route 53 the first thing you do is create a Hosted Zone. This is where your DNS data will be kept. When you do that, you receive 4 name servers where you can delegate your domain. Then you specify your Fully Qualified Domain Name, which is the domain you have purchased with a DNS registrar. This could be external, or you can use Route 53 to purchase a domain. Proxy and privacy services are options with Amazon's Route 53 registrar, allowing you to hide partial or all your information within the DNS query.

The Hosted Zone will contain record sets, which are the DNS translations you want to perform for that specific domain.

Amazon Route 53's DNS services does not support DNSSEC at this time. However, their domain name registration service supports configuration of signed DNSSEC keys for domains when DNS service is configured at another provider. More information on configuring DNSSEC for your domain name registration can be found [Configuring DNSSEC for a Domain](#).

Amazon Route 53 currently supports the following DNS record types:

- A (address record)
- AAAA (IPv6 address record)
- CNAME (canonical name record)
- CAA (certification authority authorization)
- MX (mail exchange record)
- NAPTR (name authority pointer record)
- NS (name server record)

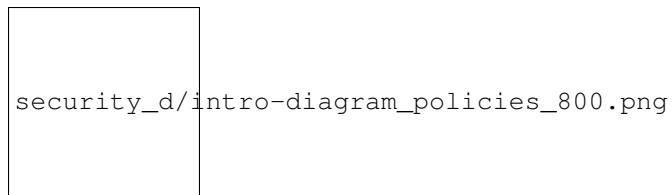


- PTR (pointer record)
- SOA (start of authority record)
- SPF (sender policy framework)
- SRV (service locator)
- TXT (text record)

## 1.7 AWS Identity and Access Management (IAM)

### 1.7.1 IAM

AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.



A **user** is a permanent named operator. It could be a human or it could be a machine. Their credentials are permanent and they stay with that named user until there is a forced rotation, whether it's a name and password, whether it's an access key, secret key combination, etc. This is my *authentication method* for named users in the system.

A **group** is a collection of users. Groups can have many users, users can belong to many groups.

A **role** is an operation (human or machine) and its credentials are temporary. It is the authentication method for the user.

An IAM role is similar to a user in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials (password or access keys) associated with it. Instead, if a user assumes a role, temporary security credentials are created dynamically and provided to the user.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but not want to embed AWS keys within the app (where they can be difficult to rotate and where users can potentially extract them). Sometimes you want to give AWS access to users who already have identities defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.

Permissions happens in a separate object known as the **policy document**. A policy specifies resources and the operations that can be performed on these resources. It is a JSON document and attaches either a permanent named user or to a group of users, or to a role. It lists the specific API or wildcard group of APIs that are white-listed.

Identities in AWS exist in these forms:

- **IAM users:** Users created within the account. IAM users are created with no permissions by default.
- **Roles:** Temporary identities used by EC2 instances, Lambdas, and external users.
- **Federation:** Users with Active Directory identities or other corporate credentials have role assigned in IAM.

- **Web Identity Federation:** Users with web identities from Amazon.com or other Open ID provier have role assigned using Security Token System (STS).

Amazon Resource Names (ARNs) uniquely identify AWS resources. We require an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies, Amazon Relational Database Service (Amazon RDS) tags, and API calls.

[Secure Access to AWS Services Using AWS Identity and Access Management \(IAM\) Roles](#)

[How to Enable Multi-Factor Authentication \(MFA\) for Your AWS User Account](#)

[AWS re:Invent 2018: \[REPEAT 1\] Become an IAM Policy Master in 60 Minutes or Less \(SEC316-R1\)](#)

[Deep Dive on AWS Single Sign-On - AWS Online Tech Talks](#)

## **SAML 2.0-based Federation**

AWS supports identity federation with SAML 2.0 (Security Assertion Markup Language 2.0), an open standard that many identity providers (IdPs) use, such as Active Directory. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create an IAM user for everyone in your organization. By using SAML, you can simplify the process of configuring federation with AWS, because you can use the IdP's service instead of writing custom identity proxy code.

If your identity store is not compatible with SAML 2.0, then you can build a custom identity broker application to perform a similar function. The broker application authenticates users, requests temporary credentials for users from AWS, and then provides them to the user to access AWS resources.

The application verifies that employees are signed into the existing corporate network's identity and authentication system, which might use LDAP, Active Directory, or another system. The identity broker application then obtains temporary security credentials for the employees.

To get temporary security credentials, the identity broker application calls either `AssumeRole` or `GetFederationToken` to obtain temporary security credentials, depending on how you want to manage the policies for users and when the temporary credentials should expire. The call returns temporary security credentials consisting of an AWS access key ID, a secret access key, and a session token. The identity broker application makes these temporary security credentials available to the internal company application. The app can then use the temporary credentials to make calls to AWS directly. The app caches the credentials until they expire, and then requests a new set of temporary credentials.

## **1.7.2 AWS Organizations**

AWS Organizations offers policy-based management for multiple AWS accounts. With Organizations, you can create groups of accounts, automate account creation, apply and manage policies for those groups. Organizations enables you to centrally manage policies across multiple accounts, without requiring custom scripts and manual processes. It allows you to create Service Control Policies (SCPs) that centrally control AWS service use across multiple AWS accounts.

You can use an IAM role to delegate access to resources that are in different AWS accounts that you own. You share resources in one account with users in a different account. By setting up cross-account access in this way, you don't need to create individual IAM users in each account. In addition, users don't have to sign out of one account and sign into another in order to access resources that are in different AWS accounts.

You can use the consolidated billing feature in AWS Organizations to consolidate payment for multiple AWS accounts or multiple AISPL accounts. With consolidated billing, you can see a combined view of AWS charges incurred by all of your accounts. You can also get a cost report for each member account that is associated with your master account. Consolidated billing is offered at no additional charge. AWS and AISPL accounts can't be consolidated together.

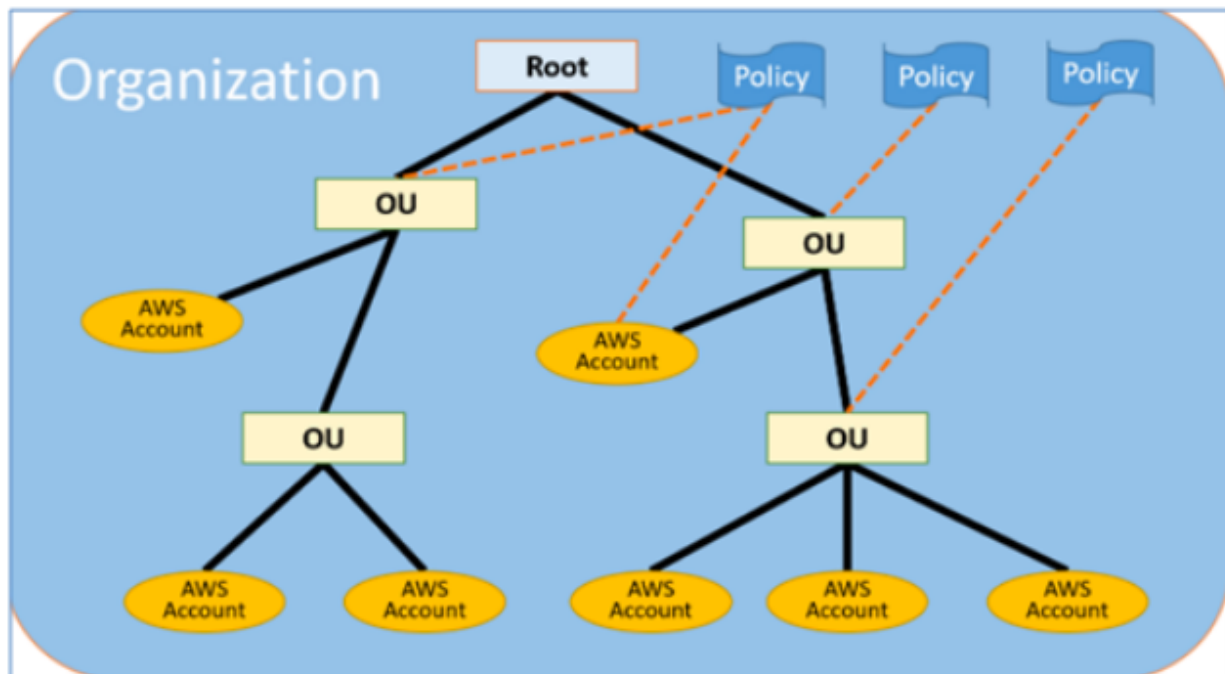
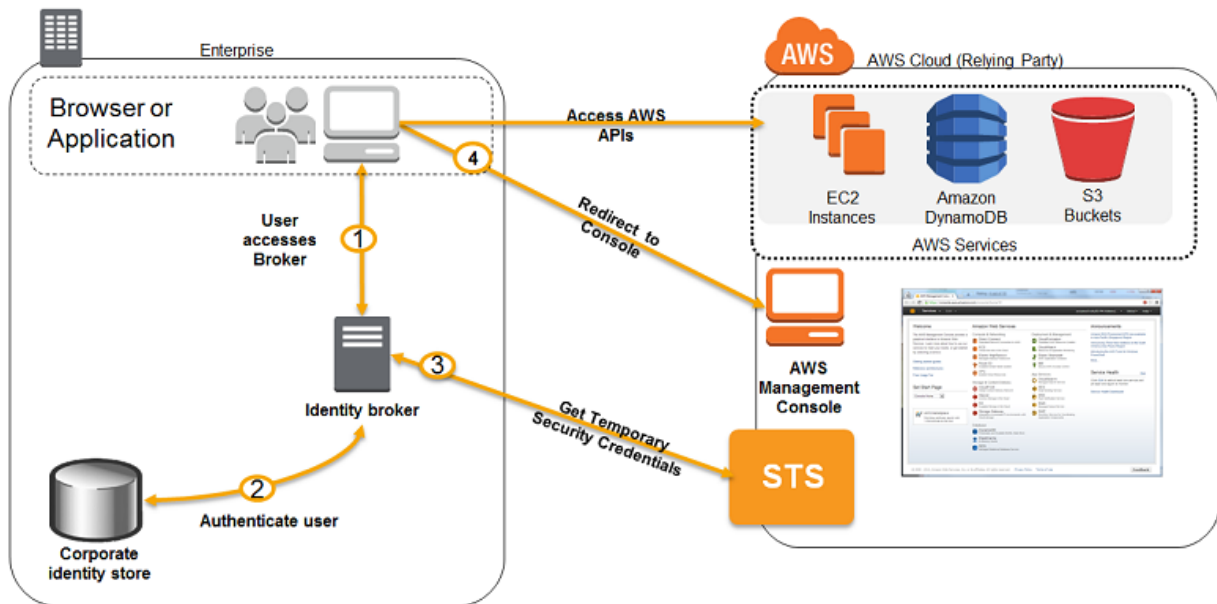


Fig. 60: AWS Organizations

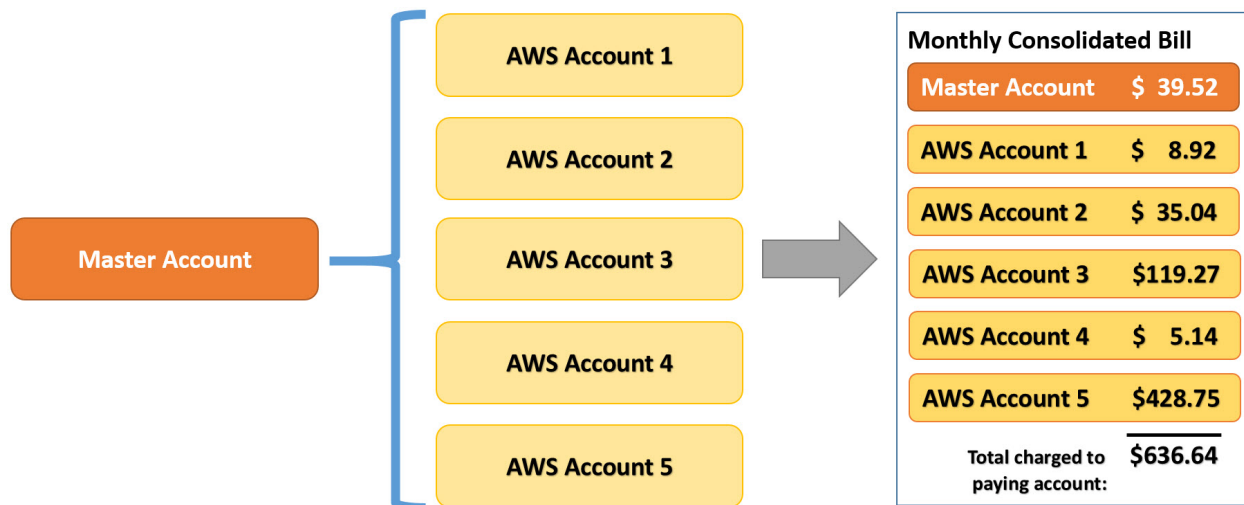


Fig. 61: AWS Organizations consolidated billing

### 1.7.3 AWS Security Token Service

AWS Security Token Service (AWS STS) is the service that you can use to create and provide trusted users with temporary security credentials that can control access to your AWS resources. Temporary security credentials work almost identically to the long-term access key credentials that your IAM users can use.

In this diagram, IAM user Alice in the Dev account (the role-assuming account) needs to access the Prod account (the role-owning account). Here's how it works:

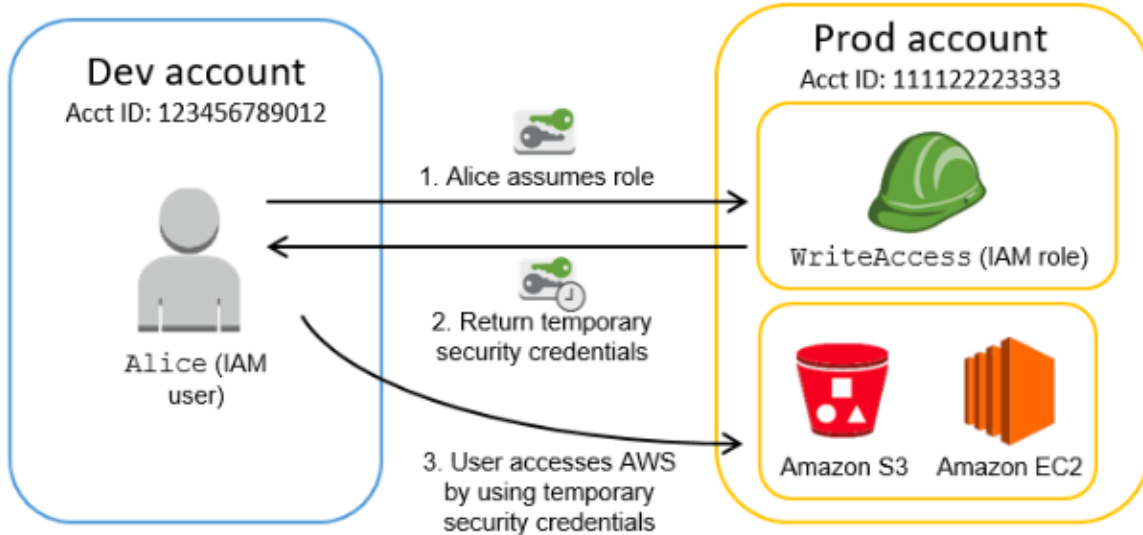
1. Alice in the Dev account assumes an IAM role (WriteAccess) in the Prod account by calling AssumeRole.
2. STS returns a set of temporary security credentials.
3. Alice uses the temporary security credentials to access services and resources in the Prod account. Alice could, for example, make calls to Amazon S3 and Amazon EC2, which are granted by the WriteAccess role.

### 1.7.4 Amazon Cognito

You can use Amazon Cognito to deliver temporary, limited-privilege credentials to your application so that your users can access AWS resources. Amazon Cognito identity pools support both authenticated and unauthenticated identities. You can retrieve a unique Amazon Cognito identifier (identity ID) for your end user immediately if you're allowing unauthenticated users or after you've set the login tokens in the credentials provider if you're authenticating users.

### 1.7.5 AWS Directory Service

AWS Directory Service provides multiple ways to use Amazon Cloud Directory and Microsoft Active Directory (AD) with other AWS services. Directories store information about users, groups, and devices, and administrators use them to manage access to information and resources. AWS Directory Service provides multiple directory choices for customers who want to use existing Microsoft AD or Lightweight Directory Access Protocol (LDAP)-aware applications in the cloud. It also offers those same choices to developers who need a directory to manage users, groups, devices, and access.



### 1.7.6 AWS Web Application Firewall (WAF)

It helps protect web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources unexpectedly.

AWS WAF gives customers control over which traffic to allow or block to their web applications by defining customizable web security rules. Customers can use AWS WAF to create rules that block common attack patterns, such as SQL injection or cross-site scripting, as well as rules tailored to their specific application.

New rules can be deployed within minutes in response to changing traffic patterns. Also, AWS WAF includes a full-featured API that customers can use to automate the creation, deployment, and maintenance of web security rules. AWS WAF integrates with services such as CloudFront and ELB, allowing customers to enforce security at the AWS edge, before traffic even reaches customer application infrastructure.

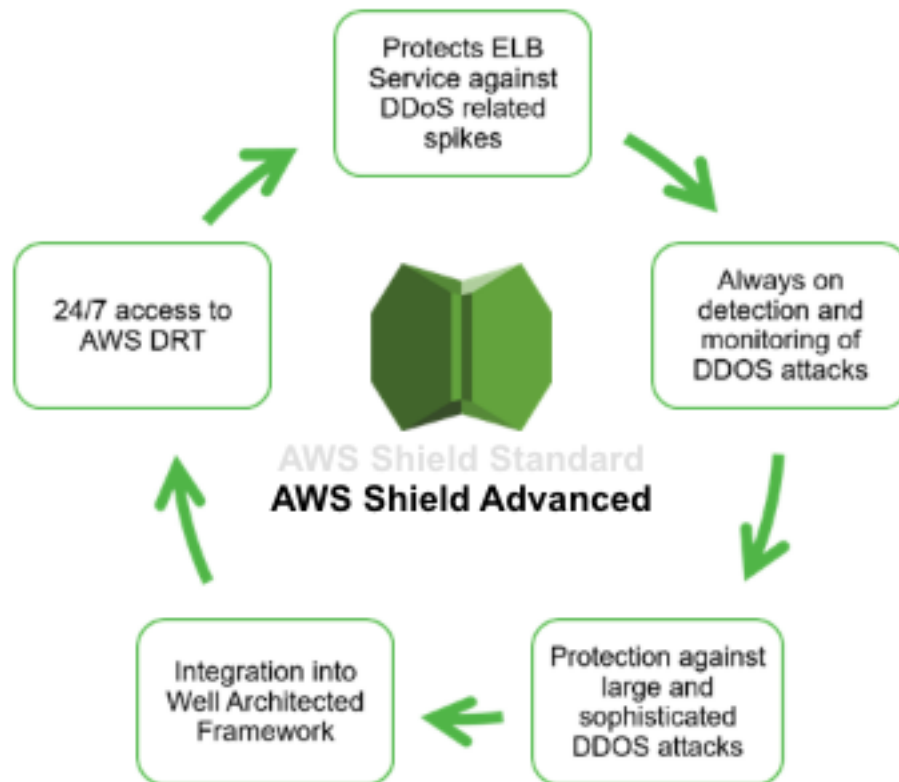
### 1.7.7 AWS Shield

AWS Shield is a managed DDoS protection service that safeguards applications running on AWS. There are 2 tiers of AWS Shield: Standard and Advanced.



AWS Shield Standard protects applications running on AWS by providing always-on detection and automatic inline mitigations of DDoS attacks, to minimize application downtime and performance degradation. All AWS customers benefit from the automatic protections of AWS Shield Standard, at no additional charge. It defends against most common, frequently occurring network and transport layer DDoS attacks that target web sites or applications. It integrates with Amazon CloudFront and Amazon Route 53, to provide comprehensive availability protection against many types of infrastructure attacks.

A combination of traffic signatures, anomaly algorithms and other analysis techniques are employed to detect malicious traffic in real-time. Automated mitigation techniques are built into this service, applied inline to your applications so there is no latency impact. Several techniques, such as deterministic packet filtering and priority-based traffic shaping are used to automatically mitigate attacks without impacting your applications. You can mitigate application layer attacks by writing rules using AWS WAF, only paying for what you use. It is a self service and there is no need to engage AWS support to receive protection against DDoS attacks.



Customers can also subscribe to AWS Shield Advanced. It provides additional detection and mitigation against large and sophisticated DDoS attacks. It provides near real-time visibility and integrates with AWS WAF. It gives customers 24x7 access to the AWS DDoS Response Team (DRT) and protection against DDoS-related spikes in their EC2, ELB, CloudFront and Route 53 charges. DRT can be engaged before, during, or after an attack. The DRT will triage incidents, identify root causes, and apply mitigations on your behalf. They can also assist you with post attack analysis. AWS Shield Advanced is available globally on Amazon CloudFront and Amazon Route 53 edge locations.

Using advanced routing techniques, this service tier provides additional capacity to protect against larger DDoS attacks. For application layer attacks, you can use AWS WAF to set up proactive rules like Rate-based Blacklisting to automatically stop bad traffic or respond immediately, all at no additional charge. You'll have complete visibility into DDoS attacks with near real-time notification through Amazon CloudWatch and detailed diagnostics on the management console. AWS Shield Advanced monitors application layer traffic to your Elastic IP addresses, ELB, CloudFront or Route 53 resources. It detects application layer attacks like HTTP floods or DNS query floods by baselining traffic on your resource and identifying anomalies. If any of these services scale up in response to a DDoS attack, AWS will provide service credits for charges due to usage spikes.

AWS Shield Standard protects your Amazon Route 53 Hosted Zones from infrastructure layer DDoS attacks, including reflection attacks or SYN floods that frequently target DNS. A variety of techniques, such as header validation and priority-based traffic shaping, automatically mitigate these attacks.

AWS Shield Advanced provides greater protection, visibility into attacks on your Route 53 infrastructure, and help from the AWS response team for extreme scenarios.

When using Amazon CloudFront, AWS Shield Standard provides comprehensive protection against infrastructure layer attacks like SYN floods, UDP floods, or other reflection attacks. Always-on detection and mitigation systems automatically scrub bad traffic at layer 3 and 4 to protect your application.

With AWS Shield Advanced the AWS response team actively applies any mitigations necessary for sophisticated infrastructure layer 3 or 4 attacks using traffic engineering. Application layer attacks, like HTTP floods, are also protected.

The always-on detection system baselines customers' steady-state application traffic and monitors for anomalies. Using AWS WAF, you can customize any application layer mitigation.

For other custom applications, not based on TCP (for example UDP and SIP), you cannot use services like Amazon CloudFront or ELB. In these cases, you often need to run your application directly on internet-facing EC2 instances.

AWS Shield Standard protects your EC2 instance from common infrastructure layer 3 or 4 attacks, such as UDP reflection attacks that include DNS, NTP, and SSDP reflection attacks. Techniques, such as priority-based traffic shaping, are automatically engaged when a well-defined DDoS attack signature is detected.

With AWS Shield Advanced on Elastic IP address, you have enhanced detection that automatically recognizes the type of AWS resource and size of EC2 instance and applies pre-defined mitigators. You can create custom mitigation profiles, and during the attack, all your Amazon VPC NACLs are automatically enforced at the border of the AWS network, giving you access to additional bandwidth and scrubbing capacity to mitigate large volumetric DDoS attacks. It also protects against SYN floods or other vectors, such as UDP floods.

Apart from using the managed DDoS protection service AWS Shield, you can protect your system from DoS attack by doing the following:

- Use an Amazon CloudFront service for distributing both static and dynamic content.
- Use an Application Load Balancer with Auto Scaling groups for your EC2 instances then restrict direct Internet traffic to the rest of resources by deploying them to a private subnet.
- Setup alerts in Amazon CloudWatch to look for high `Network In` and CPU utilization metrics.

[Best Practices for DDoS Mitigation on AWS](#)

## 1.7.8 AWS Certificate Manager

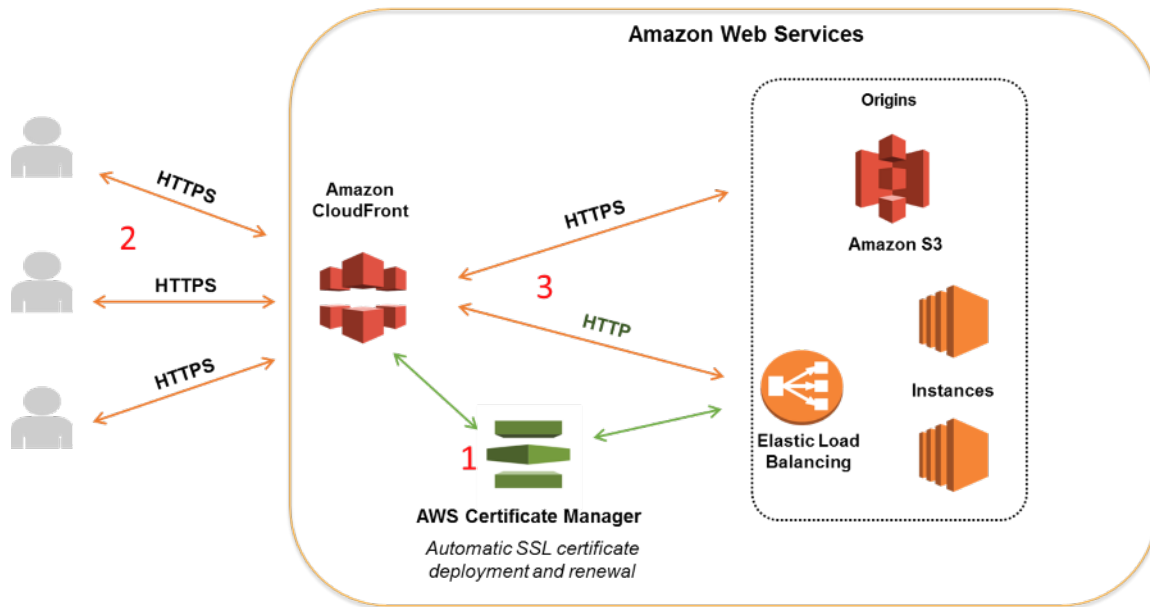
AWS Certificate Manager lets you import third-party certificates from the AWS Certificate Manager console, as well as programmatically. If AWS Certificate Manager is not available in your region, use AWS CLI to upload your third-party certificate to the IAM certificate store.

## 1.7.9 Amazon Inspector

Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. The service automatically assesses applications for vulnerabilities or deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed report with prioritized steps for remediation.

It is agent-based, API-driven, and delivered as a service. This makes it easy for you to build right into your existing DevOps process, decentralizing and automating vulnerability assessments, and empowering your DevOps teams to make security assessment an integral part of the deployment process.





### 1.7.10 CloudHSM

AWS Key Management Service (KMS) is a multi-tenant, managed service that allows you to use and manage encryption keys. AWS CloudHSM is a cloud-based hardware security module (HSM) that enables you to easily generate and use your own encryption keys on the AWS Cloud. Both services offer a high level of security for your cryptographic keys. AWS CloudHSM provides a dedicated, FIPS 140-2 Level 3 HSM under your exclusive control, directly in your Amazon Virtual Private Cloud (VPC).

You should consider using AWS CloudHSM over AWS KMS if you require:

- Keys stored in dedicated, third-party validated hardware security modules under your exclusive control.
- FIPS 140-2 compliance.
- Integration with applications using PKCS#11, Java JCE, or Microsoft CNG interfaces.
- High-performance in-VPC cryptographic acceleration (bulk crypto).

Attempting to log in as the administrator more than twice with the wrong password zeroizes your HSM appliance. When an HSM is zeroized, all keys, certificates, and other data on the HSM is destroyed. You can use your cluster's security group to prevent an unauthenticated user from zeroizing your HSM.

Amazon does not have access to your keys nor to the credentials of your Hardware Security Module (HSM) and therefore has no way to recover your keys if you lose your credentials. Amazon strongly recommends that you use two or more HSMs in separate Availability Zones in any production CloudHSM Cluster to avoid loss of cryptographic keys.

## 1.8 Elasticity, High Availability, and Monitoring

### 1.8.1 Understanding the basics

**Fault tolerance** refers to the ability of a system to remain operational even if some components of that system fail. It can be seen as the built-in redundancy of an application's components. The user does not suffer any impact from a fault, the SLA is met.



**High availability** is a concept regarding the entire system. It goes to ensure that:

- Systems are generally functioning and accessible, but it may perform in a degraded state.
- Downtime is minimized as much as possible.
- Minimal human intervention is required.
- Minimal upfront financial investment

A comparison of High Availability on premises versus AWS would be the following:

- Usually, *on premises* could be very expensive and only applied to mission-critical applications.
- On AWS you have the options to expand availability and recoverability among whatever services you choose.

## High Availability

The HA service tools are the following:

- **Elastic Load Balancers.** It distributes incoming traffic (loads) among your instances. It can send metrics to Amazon CloudWatch. ELB can be a trigger to notify high latency or if your services are becoming over-utilized. It can also be customized, for example you can configure it to recognize unhealthy EC2 instances. It can be public or internal-facing and it can use multiple different protocols.
- **Elastic IP addresses.** They are useful in providing greater fault tolerance for your application. They are static IP addresses designed for dynamic cloud computing. It allows you to mask a failure of an instance of software by allowing your users to utilize the same IP addresses with replacement resources. Your users continue to access applications even if an instance fails.
- **Amazon Route 53.** It is an authoritative DNS service that it used to translate domain names into IP addresses. It supports:
  - Simple routing
  - Latency-based routing
  - Health checks
  - DNS failovers
  - Geo-location routing
- **Auto Scaling** launches or terminates instances based on specific conditions. It is designed to assist you building a flexible system that can adjust or modify capacity depending on changes on customer demand. You can create new resources on demand or have scheduled provisioning. This is used to keep your applications and systems running no matter the load is. We can identify 3 types of autoscaling services:
  - **Amazon EC2 Auto Scaling** helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.
  - **Application Auto Scaling** allows you to configure automatic scaling for the following resources:
    - \* Amazon ECS services
    - \* Spot Fleet requests
    - \* Amazon EMR clusters
    - \* AppStream 2.0 fleets
    - \* DynamoDB tables and global secondary indexes
    - \* Aurora replicas
    - \* Amazon SageMaker endpoint variants

- \* Custom resources provided by your own applications or services. For more information, see the GitHub repository.
- \* Amazon Comprehend document classification endpoints
- \* Lambda function provisioned concurrency
- **AWS Auto Scaling** which allows you to scale a number of services at the same time and in a very simple fashion. For example, if you have an application that is using EC2 instances and DynamoDB tables, you can setup the automatic provisioning and scaling of all these resources from one single interface.
- **Amazon CloudWatch** is a distributed statistics gathering system. It collects and tracks your metrics of your AWS infrastructure. You can create and use your own custom metrics. If there are metrics that surpass a threshold CloudWatch in conjunction with Auto Scaling can scale automatically to ensure HA of your architecture.

## **Fault tolerance**

The fault tolerant service tools are the following:

- **Amazon Simple Queue Service (SQS)** which can be used as the backbone of your fault tolerant application. It is a highly reliable distributed messaging system. It helps you to ensure that the queue is always available.
- **Amazon Simple Storage Service (S3)** which provides highly durable and fault tolerant data storage. It stores the data redundantly across multiple different devices, across multiple facilities in a region.
- **Amazon Relational Database Service (RDS)** which is web service tool for you to set up, operate and scale relational databases. It provides HA and fault tolerance by offering several features to enhance the reliability on your critical databases. Some of these features include automatic backups, snapshots, and multi-AZ deployments.

## **1.8.2 AWS management tools**

AWS management tools can be classified in 4 main categories with tools that all integrated and interoperable:

- **Provisioning and entitlement:** AWS CloudFormation, AWS Service Catalog.
- **Configuration management:** AWS OpsWorks.
- **Monitoring:** Amazon CloudWatch, AWS Cost Explorer.
- **Operations and compliance management:** AWS CloudTrail, AWS Config, AWS Systems Manager.

## **AWS Cost Explorer**

Cost Explorer helps you visualize and manage your AWS costs and usages over time. It offers a set of reports you can view data with for up to the last 13 months, forecast how much you're likely to spend for the next 3 months, and get recommendations for what Reserved Instances to purchase. You use Cost Explorer to identify areas that need further inquiry and see trends to understand your costs.

## **Amazon CloudWatch**

Amazon CloudWatch is a monitoring service that allows you to monitor your AWS resources and the applications you run on AWS in real time.

Using Amazon CloudWatch alarm actions, you can create alarms that automatically stop, terminate, reboot, or recover your EC2 instances. You can use the stop or terminate actions to help you save money when you no longer need an

instance to be running. You can use the reboot and recover actions to automatically reboot those instances or recover them onto new hardware if a system impairment occurs.

## Metrics

Metrics are data about the performance of your systems. By default, several services provide free metrics for resources (such as Amazon EC2 instances, Amazon EBS volumes, and Amazon RDS DB instances). You can also enable detailed monitoring for some resources, such as your Amazon EC2 instances, or publish your own application metrics. Amazon CloudWatch can load all the metrics in your account (both AWS resource metrics and application metrics that you provide) for search, graphing, and alarms. Metric data is kept for 15 months, enabling you to view both up-to-the-minute data and historical data.

Some Amazon CloudWatch features include collecting and tracking metrics like CPU utilization, data transfer, as well as disk I/O and utilization. We can also monitor services for cloud resources and applications via collecting and monitoring log files. Additionally, you have the ability to set alarms on any of your metrics so that you can send notifications or take other automated actions.

CloudWatch has available Amazon EC2 Metrics for you to use for monitoring:

- *CPU Utilization* identifies the processing power required to run an application upon a selected instance.
- *Disk Read operations* metric measures the completed read operations from all instance store volumes available to the instance in a specified period of time.
- *Disk Write operations* metric measures the completed write operations to all instance store volumes available to the instance in a specified period of time.
- *Disk Read bytes* metric is used to determine the volume of the data the application reads from the hard disk of the instance. This can be used to determine the speed of the application.
- *Disk write bytes* metric is used to determine the volume of the data the application writes onto the hard disk of the instance. This can be used to determine the speed of the application.
- *Network In* measures the number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to a single instance.
- *Network Out* measures the number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic from a single instance.
- *Network Packets In* measures the number of packets received on all network interfaces by the instance. This metric identifies the volume of incoming traffic in terms of the number of packets on a single instance. This metric is available for basic monitoring only.
- *Network Packets Out* measures the number of packets sent out on all network interfaces by the instance. This metric identifies the volume of outgoing traffic in terms of the number of packets on a single instance. This metric is available for basic monitoring only.
- *Metadata No Token* measures the number of times the instance metadata service was accessed using a method that does not use a token.

However, there are certain metrics that are not readily available in CloudWatch such as memory utilization, disk space utilization, and many others which can be collected by setting up a custom metric. You need to prepare a custom metric using CloudWatch Monitoring Scripts which is written in Perl. You can also install CloudWatch Agent to collect more system-level metrics from Amazon EC2 instances. Here's the list of custom metrics that you can set up:

- Memory utilization
- Disk swap utilization
- Disk space utilization

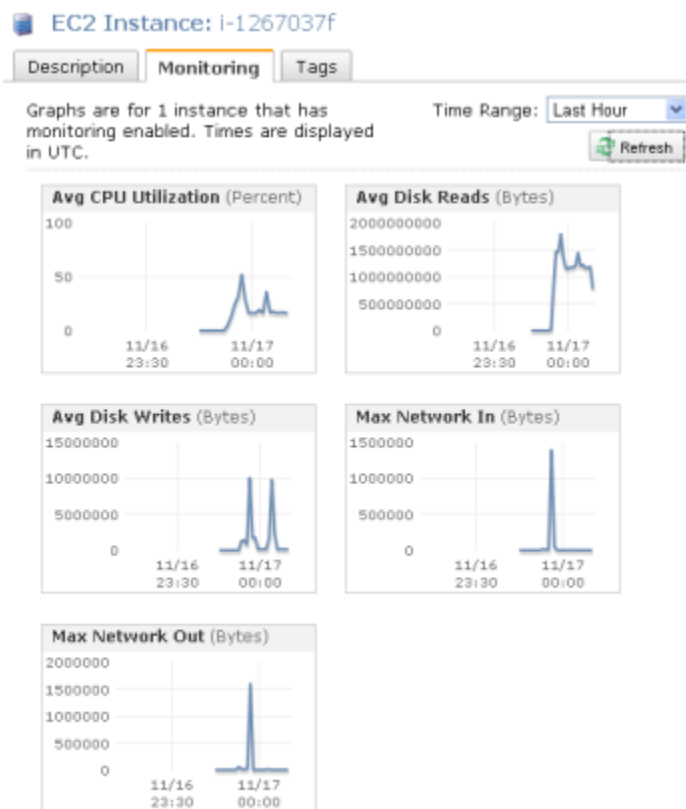


Fig. 62: Sample Auto Scaling group

- Page file utilization
- Log collection

Take note that there is a multi-platform CloudWatch agent which can be installed on both Linux and Windows-based instances. You can use a single agent to collect both system metrics and log files from Amazon EC2 instances and on-premises servers. This agent supports both Windows Server and Linux and enables you to select the metrics to be collected, including sub-resource metrics such as per-CPU core. It is recommended that you use the new agent instead of the older monitoring scripts to collect metrics and logs.

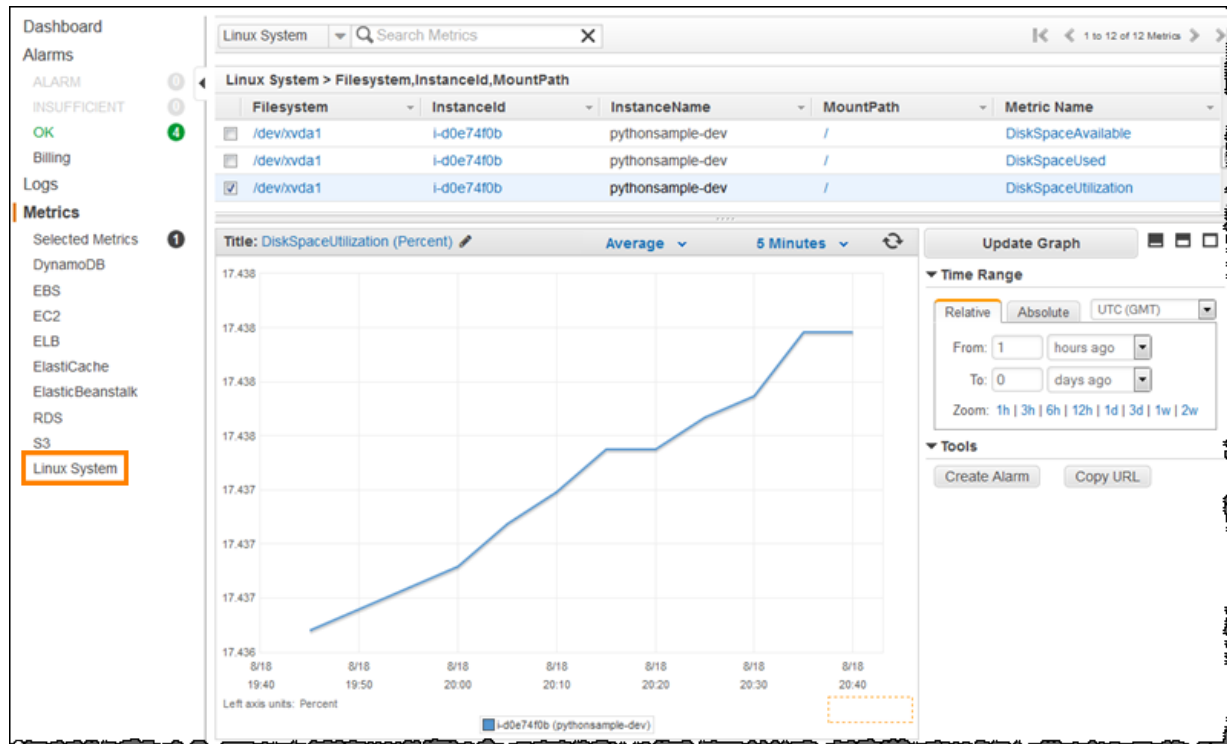


Fig. 63: CloudWatch agent to collect both system metrics and log files

Collect Metrics and Logs from Amazon EC2 instances with the CloudWatch Agent

## CloudWatch Logs

CloudWatch Logs agent provides an automated way to send log data to CloudWatch Logs from Amazon EC2 instances. The CloudWatch Logs agent is comprised of the following components:

- A plug-in to the AWS CLI that pushes log data to CloudWatch Logs.
- A script (daemon) that initiates the process to push data to CloudWatch Logs.
- A cron job that ensures that the daemon is always running.

AWS re:Invent 2018: CloudWatch Logs Insights Customer Use Case

## AWS CloudTrail

AWS CloudTrail is a service which enables compliance, governance, operational auditing and risk auditing in your accounts. It enables to track public activity across teams, accounts, and organizations in one place, in a consistent

format. It allows you to explore public activity using a single set of tools, and respond to activity in minutes.

## Use cases

- **Simplify compliance workflows.** Keep track of API usage in a single location, simplifying audit and compliance processes.
- **Enhance security analysis.** Perform security analysis and detecting user behavior patterns across services, users, and accounts.
- **Monitor data exfiltration risks.** Stay alert to data exfiltration risks by collecting activity data on Amazon S3 objects through object-level API events.
- **Perform operational troubleshooting.** Simplify root cause analysis using CloudTrail events, to reduce time to resolution.

## AWS CloudTrail events

An event in CloudTrail is the record of a single invocation of an AWS REST API and contains not only the name of the event but a lot of information related with this API invocation. This activity can be an action taken by a user, role, or service that is monitorable by CloudTrail. CloudTrail events provide a history of both API and non-API account activity made through the AWS Management Console, AWS SDKs, command line tools, and other AWS services. It is integrated with over 130 AWS services. It automatically gathers usage activity. It records event details, such as operation, principal, request and response attributes, the time it was made, etc. It delivers events to central locations.

There are two types of events that can be logged in CloudTrail: management events and data events. Both of them can be categorized as a read event or a write event, where read events are generally more frequent than write events. By default, trails log management events, but not data events.

**Management events** are resource control actions, such as update and delete actions on an Amazon EC2 instance. They are generally infrequent compared to data events. It is available from nearly all services.

**Data events** are fine-grained actions, such as reading from an object in Amazon S3. They can be very high frequency events.

The events are deliver to Amazon S3, and optionally, to Amazon CloudWatch logs. The central collection can be across accounts and regions if desired. The delivery takes typically less 15 minutes at 99th percentile, and some services have delivery times of less than 5 minutes at 99th percentile.

## Configuring trails

A trail is a resource which turns on event capture and delivery. It includes a set of event filters to define which events you are interested in, and defines a set of delivery destinations to select where you want the events stored. It can be set up through AWS Managment console, AWS API, or AWS CLI and you can define more than one trail.

A trail can be applied to all regions or a single region. As a best practice, create a trail that applies to all regions in the AWS partition in which you are working. This is the default setting when you create a trail in the CloudTrail console.

For most services, events are recorded in the region where the action occurred. For global services such as AWS Identity and Access Management (IAM), AWS STS, Amazon CloudFront, and Route 53, events are delivered to any trail that includes global services, and are logged as occurring in US East (N. Virginia) Region.

## Setting up your event logs

By default, CloudTrail event log files are encrypted using Amazon S3 server-side encryption (SSE). You can also choose to encrypt your log files with an AWS Key Management Service (AWS KMS) key. You can store your log files in your bucket for as long as you want. You can also define Amazon S3 lifecycle rules to archive or delete log files automatically. If you want notifications about log file delivery and validation, you can set up Amazon SNS notifications.

AWS re:Invent 2018: Augmenting Security & Improving Operational Health w/ AWS CloudTrail (SEC323)

### 1.8.3 Amazon EC2 Auto Scaling

Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. Using Auto Scaling removes the guesswork of how many EC2 instances you need at a point in time to meet your workload requirements.

When you run your applications on EC2 instances, it is critical to monitoring the performance of your workload using Amazon CloudWatch (for example, CPU utilization). EC2 resources requirements can vary over time, with periods with more demand and others with less demand. Auto Scaling allows you adjust capacity as needed (i.e. Capacity Management) based on conditions that you define and it is especially powerful in environments with fluctuating performance requirements. It allows you to maintain performance and minimize costs. Auto Scaling really answers 2 critical questions:

1. How can I ensure that my workload has enough EC2 resources to meet fluctuating performance requirements?
2. How can I automate EC2 resource provisioning to occur on-demand?

It matches several reliability design principles: Scale horizontally, Stop guessing capacity and Manage change in automation. If Auto Scaling adds more instances, this is termed as *scaling out*. When Auto Scaling terminates instances, this is *scaling in*.

There are 3 components required for auto-scaling:

1. Create a **launch configuration** or **launch template** determines what will be launched by Auto Scaling, i.e. the EC2 instance characteristics you need to specify: AMI, instance type, security groups, SSH keys, AWS IAM instance profile and user data to apply to the instance.

A launch template is similar to a launch configuration, in that it specifies instance configuration information. Included are the ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and the other parameters that you use to launch EC2 instances. However, defining a launch template instead of a launch configuration allows you to have multiple versions of a template. With versioning, you can create a subset of the full set of parameters and then reuse it to create other templates or template versions. For example, you can create a default template that defines common configuration parameters such as tags or network configurations, and allow the other parameters to be specified as part of another version of the same template.

We recommend that you use launch templates instead of launch configurations to ensure that you can use the latest features of Amazon EC2, such as T2 Unlimited instances.

2. Create a **Auto Scaling group**. It is a logical group of instances for your service and defines where the deployment takes place and some boundaries for the deployment. You define which VPC to deploy the instances, in which load balancer to interact with, and specify the boundaries for a group: the *minimum*, the *maximum*, and the *desired* size of the Auto Scaling Group. If you set a minimum of 1, if the number of servers goes below 1, another instance will be launched. If you set a maximum of 4, you will never have more than 4 instances in your group. The desire capacity is the number of instances that should be running at any given time (for example 2). The Auto Scaling is going to launch instances or terminate instances in order to meet the desired capacity. You can select the health check type.

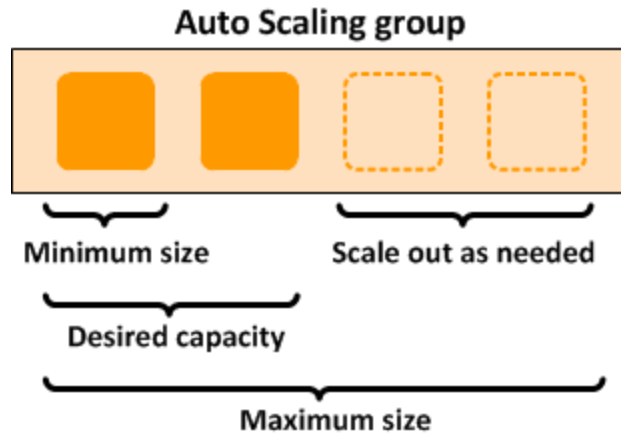


Fig. 64: Sample Auto Scaling group

There is a Health Check Grace Period setting (in seconds) in the Auto Scaling group configuration. This indicates how long Auto Scaling should wait until it starts using the ELB health check (which, in turn, has settings for how often to check and how many checks are required to mark an instance as Healthy/Unhealthy).

3. Define a least one **Auto Scaling policy**, which specifies how and when to scale in or scale out, that is, to launch or terminate EC2 instances.

If you are using Auto Scaling and you want new instances, they must meet the following criteria:

- The instance is in the running state.
- The AMI used to launch the instance must still exist.
- The instance is not a member of another Auto Scaling group.
- The instance is launched into one of the Availability Zones defined in your Auto Scaling group.
- If the Auto Scaling group has an attached load balancer, the instance and the load balancer must both be in EC2-Classic or the same VPC. If the Auto Scaling group has an attached target group, the instance and the load balancer must both be in the same VPC.

## Auto Scaling policies

There 4 possible types of auto scaling policies: manual scaling, scheduled scaling, dynamic scaling, predictive scaling.

## Manual Scaling

At any time, you can change the size of an existing Auto Scaling group manually.

The screenshot shows the "Manual Scaling" section of the AWS Management Console. It features three input fields for setting the size of the Auto Scaling group: "Desired Capacity" (set to 1), "Min" (set to 1), and "Max" (set to 5). Each field has an information icon (i) to its left.

Fig. 65: Manual Scaling



## Scheduled scaling

Scaling based on a schedule allows you to set your own scaling schedule for predictable load changes. For example, every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling actions based on the predictable traffic patterns of your web application. Scaling actions are performed automatically as a function of time and date.

To configure your Auto Scaling group to scale based on a schedule, you create a scheduled action. The scheduled action tells Amazon EC2 Auto Scaling to perform a scaling action at specified times. To create a scheduled scaling action, you specify the start time when the scaling action should take effect, and the new minimum, maximum, and desired sizes for the scaling action. At the specified time, Amazon EC2 Auto Scaling updates the group with the values for minimum, maximum, and desired size specified by the scaling action. You can create scheduled actions for scaling one time only or for scaling on a recurring schedule.

---

**Note:** For scaling based on predictable load changes, you can also use the predictive scaling feature of AWS Auto Scaling.

---

## Dynamic scaling

When you configure dynamic scaling, you must define how to scale in response to changing demand and require you to create CloudWatch alarms for the scaling policies. You create conditions that define high and low thresholds for the alarms to trigger adding or removing instances. Condition-based policies make your Auto Scaling dynamic and able to meet fluctuating requirements. It is best practice to create at least one Auto Scaling policy to specify when to scale out and at least one policy to specify to scale in. You can attach one or more Auto Scaling policies to an Auto Scaling group. It supports the following types of scaling policies: target tracking scaling, simple scaling, and step scaling.

If you are scaling based on a utilization metric that increases or decreases proportionally to the number of instances in an Auto Scaling group, we recommend that you use target tracking scaling policies. Otherwise, we recommend that you use step scaling policies.

For an advanced scaling configuration, your Auto Scaling group can have more than one scaling policy. For example, you can define one or more target tracking scaling policies, one or more step scaling policies, or both. This provides greater flexibility to cover multiple scenarios.

When there are multiple policies in force at the same time, there's a chance that each policy could instruct the Auto Scaling group to scale out (or in) at the same time. When these situations occur, Amazon EC2 Auto Scaling chooses the policy that provides the largest capacity for both scale out and scale in.

The approach of giving precedence to the policy that provides the largest capacity applies even when the policies use different criteria for scaling in. For example, if one policy terminates three instances, another policy decreases the number of instances by 25 percent, and the group has eight instances at the time of scale in, Amazon EC2 Auto Scaling gives precedence to the policy that provides the largest number of instances for the group. This results in the Auto Scaling group terminating two instances (25 percent of 8 = 2). The intention is to prevent Amazon EC2 Auto Scaling from removing too many instances.

## Target tracking scaling

Increase or decrease the current capacity of the group based on a target value for a specific metric. This is similar to the way that your thermostat maintains the temperature of your home: you select a temperature and the thermostat does the rest.

For example, you have a web application that currently runs on two instances and you want the CPU utilization of the Auto Scaling group to stay at around 50 percent when the load on the application changes. This gives you extra

capacity to handle traffic spikes without maintaining an excessive amount of idle resources. You can configure your Auto Scaling group to scale automatically to meet this need.

The screenshot shows the configuration for a target tracking scaling policy. The fields are as follows:

- Name:** Scale Group Size
- Metric type:** Average CPU Utilization
- Target value:** (empty input field)
- Instances need:** 300 seconds to warm up after scaling
- Disable scale-in:** ☐

Fig. 66: Dynamic Scaling with target tracking

One common configuration to have dynamic Auto Scaling is to create CloudWatch alarms based on performance information from your EC2 instances or a load balancer. When a performance threshold is breached, a CloudWatch alarm triggers an Auto Scaling event which either scales out or scales in EC2 instances in the environment.

The screenshot shows the configuration for a CloudWatch alarm. The fields are as follows:

- Whenever:** CPUUtilization
- is:**
- for:**  consecutive period(s)

The screenshot shows the configuration for an AutoScaling Action. The fields are as follows:

- Whenever this alarm:** State is ALARM
- From resource type:** AutoScaling
- From the:** IREASG
- Take this action:** Increase Group Size - Add 2 instances

Fig. 67: Sample CloudWatch alarm

CloudWatch can monitor metrics such as CPU, network traffic and queue size. CloudWatch has a feature called CloudWatch Logs that allows you pick up logs from EC2 instances, AWS Lambdas or CloudTrail. You can store the logs in the CloudWatch logs. You can also convert logs into metrics by extracting metrics using patterns. CloudWatch provides default metric across many AWS services and resources. You can also define custom metrics for your applications.

### Simple scaling

Increase or decrease the current capacity of the group based on a single scaling adjustment. With simple scaling, after a scaling activity is started, the policy must wait for the scaling activity or health check replacement to complete and

the cooldown period to expire before responding to additional alarms. Cooldown periods help to prevent the initiation of additional scaling activities before the effects of previous activities are visible.

The **cooldown period** helps to ensure that your Auto Scaling group doesn't launch or terminate additional instances before the previous scaling activity takes effect. You can configure the length of time based on your instance warmup period or other application needs. The default is 300 seconds. This gives newly launched instances time to start handling application traffic. After the cooldown period expires, any suspended scaling actions resume. If the CloudWatch alarm fires again, the Auto Scaling group launches another instance, and the cooldown period takes effect again. However, if the additional instance was enough to bring the performance back down, the group remains at its current size.

## Step scaling

With step scaling, you choose scaling metrics and threshold values for the CloudWatch alarms that trigger the scaling process as well as define how your scalable target should be scaled when a threshold is in breach for a specified number of evaluation periods. Step scaling policies increase or decrease the current capacity of a scalable target based on a set of scaling adjustments, known as step adjustments. The adjustments vary based on the size of the alarm breach. After a scaling activity is started, the policy continues to respond to additional alarms, even while a scaling activity is in progress. Therefore, all alarms that are breached are evaluated by Application Auto Scaling as it receives the alarm messages.

When you configure dynamic scaling, you must define how to scale in response to changing demand. For example, you have a web application that currently runs on two instances and you want the CPU utilization of the Auto Scaling group to stay at around 50 percent when the load on the application changes. This gives you extra capacity to handle traffic spikes without maintaining an excessive amount of idle resources. You can configure your Auto Scaling group to scale automatically to meet this need. The policy type determines how the scaling action is performed.

The screenshot displays two step scaling policies in the AWS Management Console. The top policy, 'Increase Group Size', is triggered by the 'WebApp\_ScaleOut' alarm. It defines three step adjustments: adding 2 instances when CPU utilization is between 50% and 60%, adding 4 instances between 60% and 70%, and adding 6 instances when CPU utilization is 70% or higher. The bottom policy, 'Decrease Group Size', is triggered by the 'WebApp\_ScaleIn' alarm. It defines two step adjustments: removing 2 instances when CPU utilization is between 30% and 20%, and removing 4 instances when CPU utilization is 20% or higher. Both policies include a 300-second cooldown period.

**Policy 1: Increase Group Size**

- Name:** Increase Group Size
- Execute policy when:** WebApp\_ScaleOut
- breaches the alarm threshold:** CPUUtilization >= 50 for 5 consecutive periods of 60 seconds for the metric dimensions AutoScalingGroupName = WebApp-WebServerGroup-13H20HQTSS519C
- Take the action:**
  - Add 2 instances when 50 <= CPUUtilization < 60
  - Add 4 instances when 60 <= CPUUtilization < 70
  - Add 6 instances when 70 <= CPUUtilization < +infinity
- Instances need:** 300 seconds to warm up after each step

**Policy 2: Decrease Group Size**

- Name:** Decrease Group Size
- Execute policy when:** WebApp\_ScaleIn
- breaches the alarm threshold:** CPUUtilization <= 30 for 10 consecutive periods of 60 seconds for the metric dimensions AutoScalingGroupName = WebApp-WebServerGroup-13H20HQTSS519C
- Take the action:**
  - Remove 2 instances when 30 >= CPUUtilization > 20
  - Remove 4 instances when 20 >= CPUUtilization > -infinity

Fig. 68: Dynamic Scaling with step scaling

## Predictive scaling

Using data collected from your actual EC2 usage and further informed by billions of data points drawn from Amazon.com observations, we use well-trained Machine Learning models to predict your expected traffic (and EC2 usage) including daily and weekly patterns. The model needs at least one day's of historical data to start making predictions; it is re-evaluated every 24 hours to create a forecast for the next 48 hours.

It performs a regression analysis between load metric and scaling metric and schedules scaling actions for the next 2 days, hourly, and repeats this process every day.

## Termination policies

Using termination policies, you can control which instances you prefer to terminate first when a scale-in event occurs. It also describes how to enable instance scale-in protection to prevent specific instances from being terminated during automatic scale in.

---

**Note:** Auto Scaling groups with different types of purchase options are a unique situation. Amazon EC2 Auto Scaling first identifies which of the two types (Spot or On-Demand) should be terminated. If you balance your instances across Availability Zones, it chooses the Availability Zone with the most instances of that type to maintain balance. Then, it applies the default or customized termination policy.

---

## Default Termination Policy

The default termination policy is designed to help ensure that your instances span Availability Zones evenly for high availability. The default policy is kept generic and flexible to cover a range of scenarios. The default termination policy behavior is as follows:

1. Determine which Availability Zones have the most instances, and at least one instance that is not protected from scale in.
2. Determine which instances to terminate so as to align the remaining instances to the allocation strategy for the On-Demand or Spot Instance that is terminating. This only applies to an Auto Scaling group that specifies allocation strategies.

For example, after your instances launch, you change the priority order of your preferred instance types. When a scale-in event occurs, Amazon EC2 Auto Scaling tries to gradually shift the On-Demand Instances away from instance types that are lower priority.

3. Determine whether any of the instances use the oldest launch template or configuration:
  - 3.1. [For Auto Scaling groups that use a launch template]

Determine whether any of the instances use the oldest launch template unless there are instances that use a launch configuration. Amazon EC2 Auto Scaling terminates instances that use a launch configuration before instances that use a launch template.
  - 3.2. [For Auto Scaling groups that use a launch configuration]

Determine whether any of the instances use the oldest launch configuration.
4. After applying all of the above criteria, if there are multiple unprotected instances to terminate, determine which instances are closest to the next billing hour. If there are multiple unprotected instances closest to the next billing hour, terminate one of these instances at random.

Note that terminating the instance closest to the next billing hour helps you maximize the use of your instances that have an hourly charge. Alternatively, if your Auto Scaling group uses Amazon Linux or Ubuntu, your EC2 usage is billed in one-second increments.

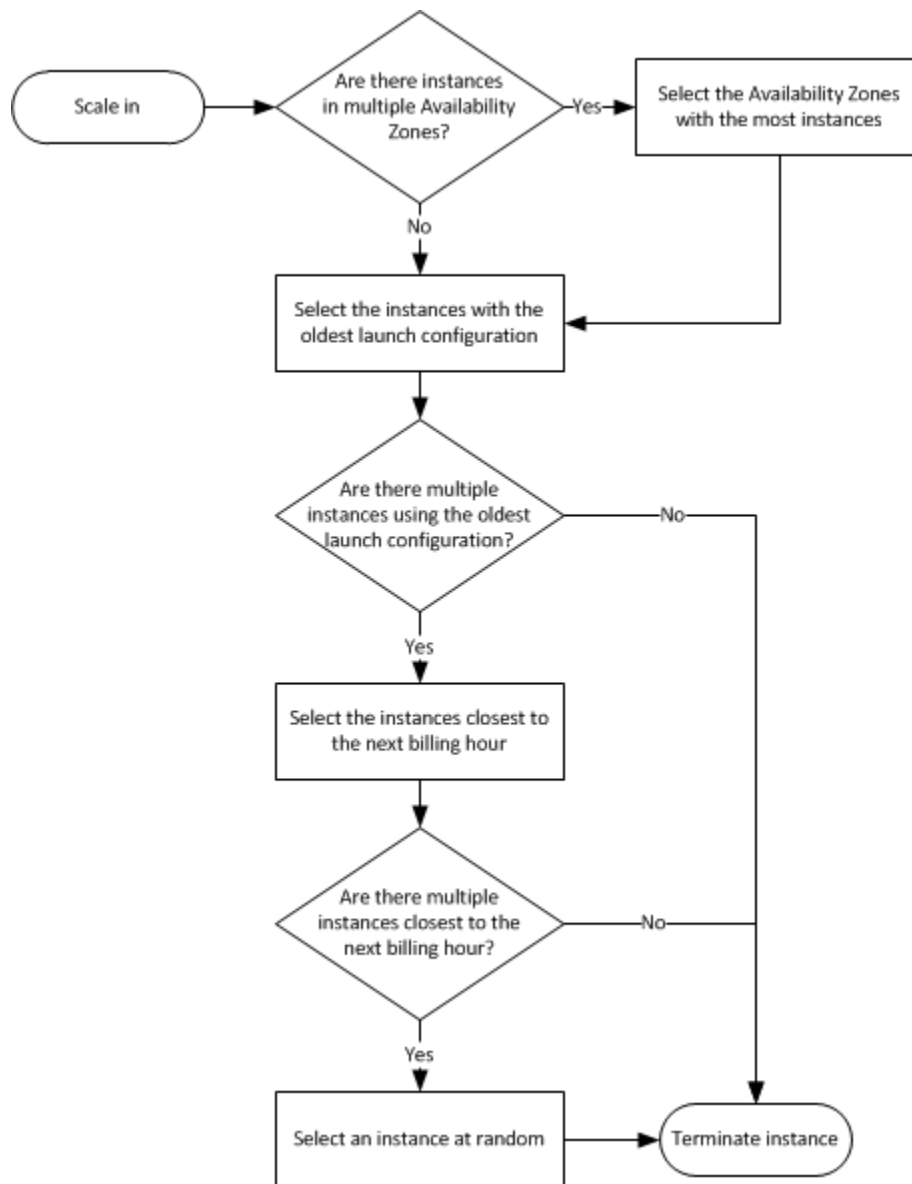


Fig. 69: Default termination policy flowchart

### Customizing the Termination Policy

You have the option of replacing the default policy with a customized one to support common use cases like keeping instances that have the current version of your application.

When you customize the termination policy, if one Availability Zone has more instances than the other Availability Zones that are used by the group, your termination policy is applied to the instances from the imbalanced Availability Zone. If the Availability Zones used by the group are balanced, the termination policy is applied across all of the Availability Zones for the group.

Amazon EC2 Auto Scaling supports the following custom termination policies:

- **OldestInstance.** Terminate the oldest instance in the group. This option is useful when you're upgrading the instances in the Auto Scaling group to a new EC2 instance type. You can gradually replace instances of the old type with instances of the new type.
- **NewestInstance.** Terminate the newest instance in the group. This policy is useful when you're testing a new launch configuration but don't want to keep it in production.
- **OldestLaunchConfiguration.** Terminate instances that have the oldest launch configuration. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.
- **ClosestToNextInstanceHour.** Terminate instances that are closest to the next billing hour. This policy helps you maximize the use of your instances that have an hourly charge.
- **Default.** Terminate instances according to the default termination policy. This policy is useful when you have more than one scaling policy for the group.
- **OldestLaunchTemplate.** Terminate instances that have the oldest launch template. With this policy, instances that use the noncurrent launch template are terminated first, followed by instances that use the oldest version of the current launch template. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.
- **AllocationStrategy.** Terminate instances in the Auto Scaling group to align the remaining instances to the allocation strategy for the type of instance that is terminating (either a Spot Instance or an On-Demand Instance). This policy is useful when your preferred instance types have changed. If the Spot allocation strategy is lowest-price, you can gradually rebalance the distribution of Spot Instances across your N lowest priced Spot pools. If the Spot allocation strategy is capacity-optimized, you can gradually rebalance the distribution of Spot Instances across Spot pools where there is more available Spot capacity. You can also gradually replace On-Demand Instances of a lower priority type with On-Demand Instances of a higher priority type.

## Instance Scale-In Protection

To control whether an Auto Scaling group can terminate a particular instance when scaling in, use instance scale-in protection. You can enable the instance scale-in protection setting on an Auto Scaling group or on an individual Auto Scaling instance. When the Auto Scaling group launches an instance, it inherits the instance scale-in protection setting of the Auto Scaling group. You can change the instance scale-in protection setting for an Auto Scaling group or an Auto Scaling instance at any time.

Instance scale-in protection starts when the instance state is `InService`. If you detach an instance that is protected from termination, its instance scale-in protection setting is lost. When you attach the instance to the group again, it inherits the current instance scale-in protection setting of the group.

If all instances in an Auto Scaling group are protected from termination during scale in, and a scale-in event occurs, its desired capacity is decremented. However, the Auto Scaling group can't terminate the required number of instances until their instance protection settings are disabled.

Instance scale-in protection does not protect Auto Scaling instances from the following:

- Manual termination through the Amazon EC2 console, the `terminate-instances` command, or the `TerminateInstances` action. To protect Auto Scaling instances from manual termination, enable Amazon EC2 termination protection.
- Health check replacement if the instance fails health checks. To prevent Amazon EC2 Auto Scaling from terminating unhealthy instances, suspend the `ReplaceUnhealthy` process.
- Spot Instance interruptions. A Spot Instance is terminated when capacity is no longer available or the Spot price exceeds your maximum price.

## Use cases

### Automate provisioning of instances

One of the use cases of Auto Scaling is automating the provision of EC2 instances. When the instances come up, they need to have some software and applications installed. You have to approaches to achieve it:

- You can use AMIs with all required configuration and software for this purpose, this is called the **golden image**. This golden image can be specified in the launch template.
- You can define a **base AMI** and install code and configuration as needed through user data in the launch template, AWS CodeDeploy, AWS Systems Manager, or even with configuration tools such as Puppet, Chef, and Ansible.

Listing 4: Sample user data

```
#!/bin/bash

# Install updates
sudo yum update -y;

# Install AWS CodeDeploy agent
cd /home/ec2-user;
wget https://aws-codedeploy-us-east-1.s3.region-identifier.amazonaws.com/latest/
↪install \ -o install &&
chmod +x ./install &&
sudo ./install auto && sudo service codedeploy-agent start;
```

### Perform additional actions with lifecycle hooks

Another use case is when the EC2 instance comes up, you want to execute additional actions when the instance is in pending state or terminating state such as:

- Assign EC2 IP address or ENI on launch.
- Register new instances with DNS, external monitoring systems, firewalls.
- Load existing state from Amazon S3 or other system.
- Pull down log files before instance is terminated.
- Investigate issues with an instance before terminating it.
- Persist instance state to external system.

The EC2 instances in an Auto Scaling group have a path, or lifecycle, that differs from that of other EC2 instances. The lifecycle starts when the Auto Scaling group launches an instance and puts it into service. The lifecycle ends when you terminate the instance, or the Auto Scaling group takes the instance out of service and terminates it. The following illustration shows the transitions between instance states in the Amazon EC2 Auto Scaling lifecycle.

---

**Note:** You are billed for instances as soon as they are launched, including the time that they are not yet in service.

---

You can receive notification when state transitions happen. You can rely on notifications to react to changes that happened. It is available via Amazon SNS and Amazon CloudWatch Events.



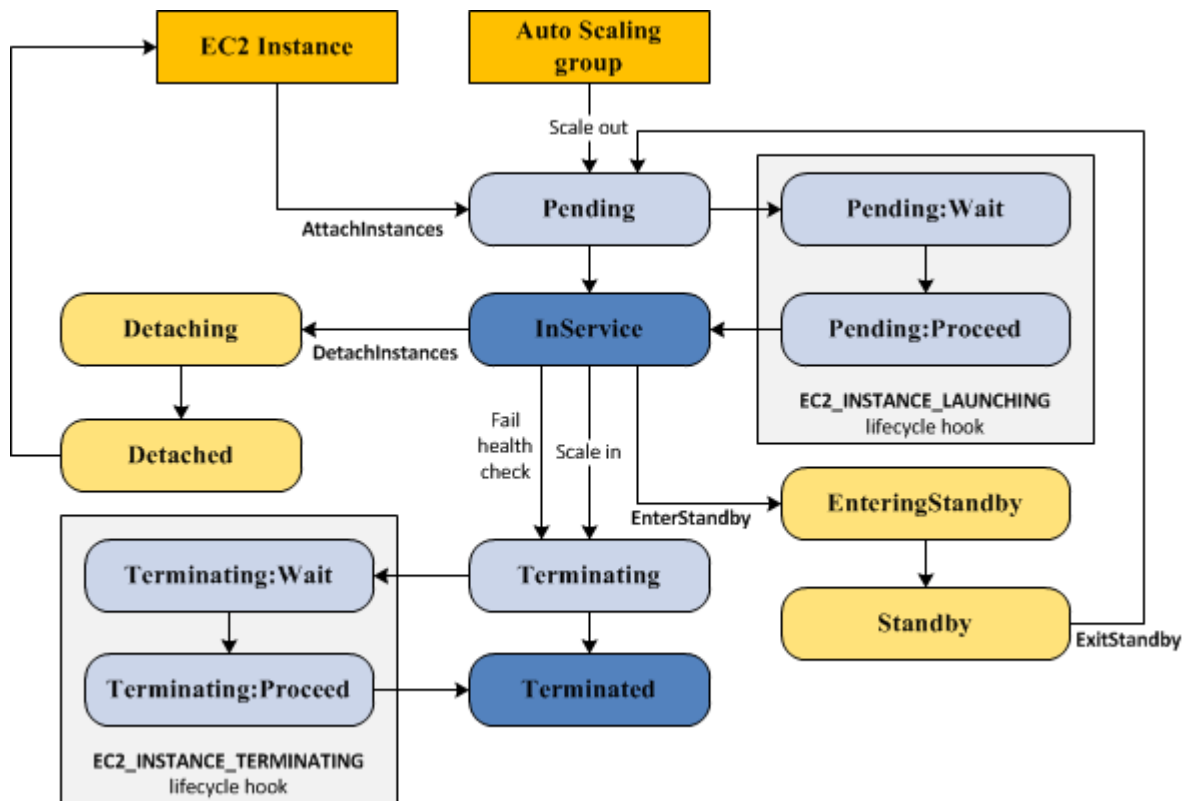


Fig. 70: Auto Scaling Lifecycle

### Register instances behind load balancer

You have full integration with ELB allowing you to automatically register instances behind Application Load Balancer, Network Load Balancer, and Classic Load Balancer.

### Reduce paging frequency

### Replace unhealthy instances

When the load balancer determines that an instance is unhealthy, it stops routing requests to that instance. The load balancer resumes routing requests to the instance when it has been restored to a healthy state. There are 3 ways of checking the status of your EC2 instances:

1. Via the Auto Scaling group. The default health checks for an Auto Scaling group are **EC2 status checks only**. If an instance state is different from `running` or system health check equals `impaired`, the Auto Scaling group considers the instance unhealthy and replaces it. If you attached one or more load balancers or target groups to your Auto Scaling group, the group does not, by default, consider an instance unhealthy and replace it if it fails the load balancer health checks.
2. Via the **ELB health checks**. However, you can optionally configure the Auto Scaling group to use Elastic Load Balancing health checks. This ensures that the group can determine an instance's health based on additional tests provided by the load balancer. The load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances. These tests are called health checks. Load balancer health checks fail if ELB health equals `OutOfService`.



If you configure the Auto Scaling group to use Elastic Load Balancing health checks, it considers the instance unhealthy if it fails either the EC2 status checks or the load balancer health checks. If you attach multiple load balancers to an Auto Scaling group, all of them must report that the instance is healthy in order for it to consider the instance healthy. If one load balancer reports an instance as unhealthy, the Auto Scaling group replaces the instance, even if other load balancers report it as healthy.

3. **Via custom health checks.** You can manually mark instances as unhealthy. You can integrate with external monitoring systems.

Why did Auto Scaling Group terminate my healthy instance(s)?

## Balance capacity across AZs

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is disabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone.

With Application Load Balancers, cross-zone load balancing is always enabled. With Network Load Balancers, cross-zone load balancing is disabled by default. When you create a Classic Load Balancer, the default for cross-zone load balancing depends on how you create the load balancer. With the API or CLI, cross-zone load balancing is disabled by default. With the AWS Management Console, the option to enable cross-zone load balancing is selected by default.

For example, suppose we have 6 EC2 instances across 2 AZs behind an Elastic Load Balancer. In this case, we will have 3 EC2 instances in each AZ. If one of the AZs goes down, then Auto Scaling is going to terminate the 3 EC2 instances that were in this AZ and launch 3 EC2 instances in the AZ that is alive. Whenever the failed AZ is restored, then Auto Scaling rebalance the number of EC2 instances and setting 3 EC2 instances in each AZ.

## Use spot instances to reduce costs

You can reduce costs, optimize performance and eliminate operational overhead by automatically scaling instances across instance families and purchasing models (spot, on-demand, and reserved instances) in a single Auto Scaling group.

You can specify what percentage of your group capacity should be fulfilled by on-demand instances, and spot instances to optimize cost. Use a prioritized list for on-demand instance types to scale capacity during an urgent, unpredictable event to optimize performance.

AWS re:Invent 2015: All You Need To Know About Auto Scaling (CMP201)

AWS re:Invent 2018: Capacity Management Made Easy with Amazon EC2 Auto Scaling (CMP377)

Introduction to Amazon EC2 Auto Scaling

## 1.8.4 Scaling your databases

## 1.9 Automation

### 1.9.1 AWS CloudFormation

AWS CloudFormation is a Infrastructure as code that enables provisioning and management the entire lifecycle of your AWS infrastructure. The process to execute Infrastructure as code with CloudFormation is the following:

1. Code your CloudFormation template in YAML or JSON directly or use sample templates.

2. Upload local files or from an S3 bucket.
3. Create a stack using console, API, or CLI.
4. Stacks and resources are provisioned.

The following scenarios demonstrate how AWS CloudFormation can help: Simplify infrastructure management, quickly replicate your infrastructure, easily control and track changes to your infrastructure.

The benefits you get from CloudFormation are:

1. As group of resources get deployed or update your resources, if one of your resources fails, Cloud Formation keeps your state and is able to rollback to the last known good state.
2. CloudFormation knows about stabilization of resources. When a resource has been created, it is guaranteed that you can do a describe or a list to the resources.

There is no additional charge for AWS CloudFormation. You only pay for the AWS resources that are created (e.g. Amazon EC2 instances, Elastic Load Balancing load balancers, etc.)

### [AWS CloudFormation Deep Dive and Recent Enhancements](#)

## **Template Anatomy**

A template is a JSON- or YAML-formatted text file that describes your AWS infrastructure. The following examples show an AWS CloudFormation template structure and its sections.

## **JSON**

The following example shows a JSON-formatted template fragment.

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  }
}
```

(continues on next page)

(continued from previous page)

```
},  
  
"Outputs" : {  
  set of outputs  
}  
}
```

## YAML

The following example shows a YAML-formatted template fragment.

```
---  
AWSTemplateFormatVersion: "version date"  
  
Description:  
  String  
  
Metadata:  
  template metadata  
  
Parameters:  
  set of parameters  
  
Mappings:  
  set of mappings  
  
Conditions:  
  set of conditions  
  
Transform:  
  set of transforms  
  
Resources:  
  set of resources  
  
Outputs:  
  set of outputs
```

## Template Sections

Templates include several major sections. Some sections in a template can be in any order. However, as you build your template, it can be helpful to use the logical order shown in the following list because values in one section might refer to values from a previous section. Take note that all of the sections here are optional, except for Resources, which is the only one required.

- Format Version
- Description
- Metadata
- Parameters
- Mappings
- Conditions

- Transform
- Resources (required)
- Outputs

## Enterprise management

**Drift detection** allows you to detect if configuration changes were made to your stack resources outside of CloudFormation via the AWS Management Console, CLI, and SDKs. you can use the diff viewer in the console to pinpoint the changes. It supports the most commonly used resources. You can define automatic drift alerts via AWS Config rule. You can remediate by updating the live configuration values to match the template values.

**Stacksets** extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation.

**Dynamic references** provide a compact, powerful way for you to specify external values that are stored and managed in other services, such as the Systems Manager Parameter Store, in your stack templates. When you use a dynamic reference, CloudFormation retrieves the value of the specified reference when necessary during stack and change set operations. CloudFormation currently supports the following dynamic reference patterns:

- *ssm*, for plaintext values stored in AWS Systems Manager Parameter Store
- *ssm-secure*, for secure strings stored in AWS Systems Manager Parameter Store
- *secretsmanager*, for entire secrets or specific secret values that are stored in AWS Secrets Manager

## Developer productivity

**CloudFormation Linter** is an open source tool that allows you to validate CloudFormation YAML/JSON templates against the CloudFormation spec and additional checks. Includes checking valid values for resource properties and best practices. CloudFormation Linter has plugins for Atom, VisualStudio Code, Sublime, and VIM. It can be run headless in pipelines. It can process multiple files. It can handle conditions/Fn::if. It has SAM local integration.

The **AWS CloudFormation Template Schema** is intended to improve the authoring experience for our customers. It is a simple code process which converts our existing Resource Specifications files into a JSON Schema formatted document. This schema can be integrated into many publicly available IDEs such as Visual Studio Code & PyCharm to provide inline syntax checking and code completion.

**Taskcat** is an open source tool that tests AWS CloudFormation templates. It deploys your AWS CloudFormation template in multiple AWS Regions and generates a report with a pass/fail grade for each region. You can specify the regions and number of Availability Zones you want to include in the test, and pass in parameter values from your AWS CloudFormation template. taskcat is implemented as a Python class that you import, instantiate, and run.

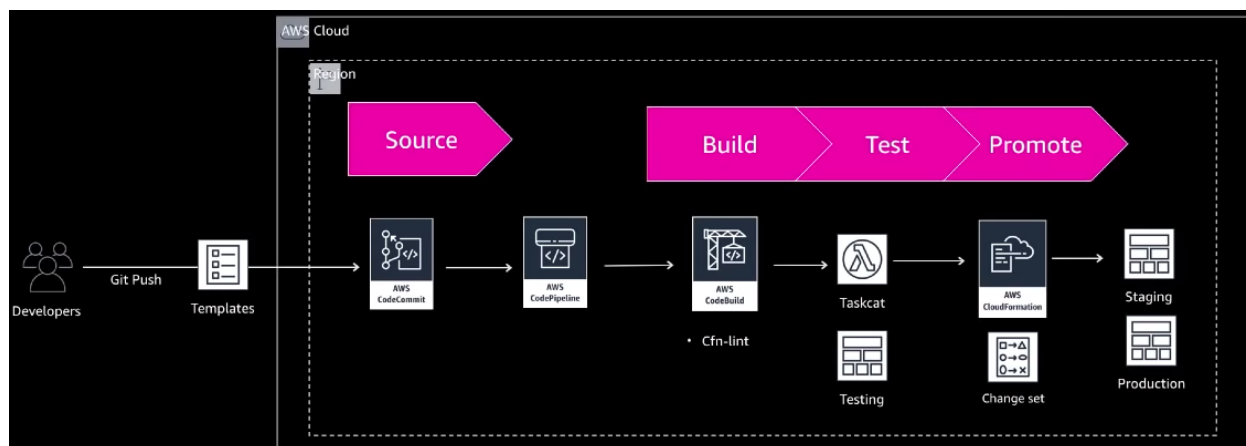
An example of pipeline for deploying AWS Cloudformation templates is illustrated below.

**Macros** allows you to write short-hand, abbreviated instructions that expand automatically before deployment. For instance you can use it to:

- Add utility functions, iteration loops, string, . . .
- Ensure resources are defined to comply with standards.
- Easy to share code to reuse across stacks.

The key benefit of macros is once deployed, downstream users can be isolated from program details. Macros are AWS Lambda functions and can use all supported languages. You can run macros without change sets. You have [Macro examples](#).

**AWS CDK (Cloud Development Kit)** supports TypeScript/JavaScript, Java, .NET, Python.



## 1.9.2 AWS OpsWorks

AWS OpsWorks is a configuration management service that provides managed instances of Chef and Puppet. Chef and Puppet are automation platforms that allow you to use code to automate the configurations of your servers. OpsWorks lets you use Chef and Puppet to automate how servers are configured, deployed, and managed across your Amazon EC2 instances or on-premises compute environments. OpsWorks has three offerings - AWS Opsworks for Chef Automate, AWS OpsWorks for Puppet Enterprise, and AWS OpsWorks Stacks.

Controlling User Session Access to Instances in AWS System Manager Session Manager

How do I troubleshoot issues deleting instances when using AWS OpsWorks Stacks?

## 1.9.3 A little more hands off

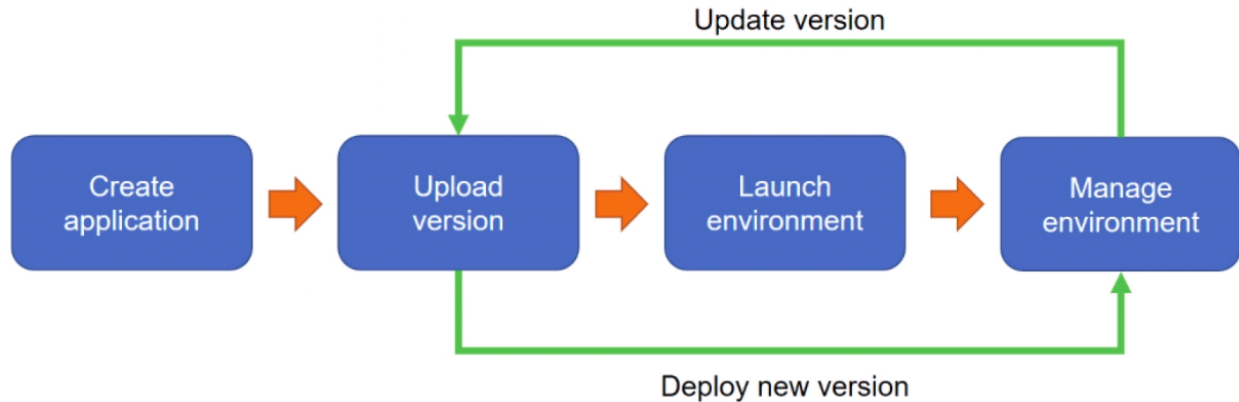
## 1.9.4 AWS Elastic Beanstalk

AWS Elastic Beanstalk is a PaaS with the following features:

- It allows a quick deployment of your applications. Any code that you have previously written on some specific language can be simply placed over the platform.
- It reduces the management complexity. You don't need to manage the whole system.
- It is possible to keep full control over it, allowing you to choose the instance type, the DB and adjust Auto Scaling according to your needs. Besides, it allows you to update your application, access server log files, and enables HTTPS on the load balancer according to the needs of your application.
- It supports a large range of platforms: Packer Builder; Single container, multicontainer, or preconfigured Docker; Go; Java SE; Java with Tomcat; .NET on Windows Server with IIS; Node.js; PHP; Python; Ruby.

The steps to deploy and update your servers are based only on the creation of your application. After that, you upload the versions to BeanStalk, then launch all the needed environments in the cloud according to the needs of your application. After that, you can manage your environment, and if you need to write a new version, you just update the version.

AWS Elastic Beanstalk stores your application files and optionally, server log files in Amazon S3. If you are using the AWS Management Console, the AWS Toolkit for Visual Studio, or AWS Toolkit for Eclipse, an Amazon S3 bucket will be created in your account and the files you upload will be automatically copied from your local client to Amazon S3. Optionally, you may configure Elastic Beanstalk to copy your server log files every hour to Amazon S3. You do this by editing the environment configuration settings.



With CloudWatch Logs, you can monitor and archive your Elastic Beanstalk application, system, and custom log files from Amazon EC2 instances of your environments. You can also configure alarms that make it easier for you to react to specific log stream events that your metric filters extract. The CloudWatch Logs agent installed on each Amazon EC2 instance in your environment publishes metric data points to the CloudWatch service for each log group you configure. Each log group applies its own filter patterns to determine what log stream events to send to CloudWatch as data points. Log streams that belong to the same log group share the same retention, monitoring, and access control settings. You can configure Elastic Beanstalk to automatically stream logs to the CloudWatch service.

[AWS re:Invent 2017: Manage Your Applications with AWS Elastic Beanstalk \(DEV305\)](#)

## 1.10 Caching

### 1.10.1 Caching overview

#### 1.10.2 CloudFront

CloudFront provides improved latency, packet loss and overall quality. It avoids conflicts and network interconnect capacity. It offers greater operational control.

You can control how long your objects stay in a CloudFront cache before CloudFront forwards another request to your origin. Reducing the duration allows you to serve dynamic content. Increasing the duration means your users get better performance because your objects are more likely to be served directly from the edge cache. A longer duration also reduces the load on your origin.

Typically, CloudFront serves an object from an edge location until the cache duration that you specified passes — that is, until the object expires. After it expires, the next time the edge location gets a user request for the object, CloudFront forwards the request to the origin server to verify that the cache contains the latest version of the object.

The `Cache-Control` and `Expires` headers control how long objects stay in the cache. The `Cache-Control max-age` directive lets you specify how long (in seconds) you want an object to remain in the cache before CloudFront gets the object again from the origin server. The minimum expiration time CloudFront supports is 0 seconds for web distributions and 3600 seconds for RTMP distributions.

#### File versions

To control the versions of files that are served from your distribution, you can either invalidate files or give them versioned file names. If you want to update your files frequently, AWS recommends that you primarily use file versioning for the following reasons:

- Versioning enables you to control which file a request returns even when the user has a version cached either locally or behind a corporate caching proxy. If you invalidate the file, the user might continue to see the old version until it expires from those caches.
- CloudFront access logs include the names of your files, so versioning makes it easier to analyze the results of file changes.
- Versioning provides a way to serve different versions of files to different users.
- Versioning simplifies rolling forward and back between file revisions.
- Versioning is less expensive. You still have to pay for CloudFront to transfer new versions of your files to edge locations, but you don't have to pay for invalidating files.

## Origin Access Identity (OAI)

When you create or update a distribution in CloudFront, you can add an origin access identity (OAI) and automatically update the bucket policy to give the origin access identity permission to access your bucket. Alternatively, you can choose to manually change the bucket policy or change ACLs, which control permissions on individual objects in your bucket. You can update the Amazon S3 bucket policy using either the AWS Management Console or the Amazon S3 API:

- Grant the CloudFront origin access identity the applicable permissions on the bucket.
- Deny access to anyone that you don't want to have access using Amazon S3 URLs.

## SNI Custom SSL

SNI Custom SSL relies on the SNI extension of the Transport Layer Security protocol, which allows multiple domains to serve SSL traffic over the same IP address by including the hostname which the viewers are trying to connect to.

Amazon CloudFront delivers your content from each edge location and offers the same security as the Dedicated IP Custom SSL feature. SNI Custom SSL works with most modern browsers, including Chrome version 6 and later (running on Windows XP and later or OS X 10.5.7 and later), Safari version 3 and later (running on Windows Vista and later or Mac OS X 10.5.6. and later), Firefox 2.0 and later, and Internet Explorer 7 and later (running on Windows Vista and later).

Some users may not be able to access your content because some older browsers do not support SNI and will not be able to establish a connection with CloudFront to load the HTTPS version of your content. If you need to support non-SNI compliant browsers for HTTPS content, it is recommended to use the Dedicated IP Custom SSL feature.

## Lambda@Edge

**Lambda@Edge** is a feature of Amazon CloudFront that lets you run code closer to users of your application, which improves performance and reduces latency. With **Lambda@Edge**, you don't have to provision or manage infrastructure in multiple locations around the world. You pay only for the compute time you consume - there is no charge when your code is not running.

With **Lambda@Edge**, you can enrich your web applications by making them globally distributed and improving their performance — all with zero server administration. **Lambda@Edge** runs your code in response to events generated by the Amazon CloudFront content delivery network (CDN). Just upload your code to AWS Lambda, which takes care of everything required to run and scale your code with high availability at an AWS location closest to your end user.

**Lambda@Edge** lets you run Lambda functions to customize the content that CloudFront delivers, executing the functions in AWS locations closer to the viewer. The functions run in response to CloudFront events, without provisioning or managing servers. You can use Lambda functions to change CloudFront requests and responses at the following points:

AWS

Tags

SSL Certificate

☐

Default CloudFront Certificate (\*.cloudfront.net)

Choose this option if you want your users to use HTTPS or HTTP to access your content with the CloudFront domain name (such as https://d11111abcdef8.cloudfront.net/logo.jpg).

Important: If you choose this option, CloudFront requires that browsers or devices support TLSv1 or later to access your content.

☒

Custom SSL Certificate (example.com):

Choose this option if you want your users to access your content by using an alternate domain name, such as https://www.example.com/logo.jpg.

You can use certificates that you created in AWS Certificate Manager (ACM) in the US-East (N. Virginia) Region, or you can use certificates stored in the IAM certificate store.

\*.leeatk.com

↺

Request an ACM certificate

[Learn more](#) about using custom SSL/TLS certificates with CloudFront.

[Learn more](#) about using ACM.

Custom SSL Client Support

☒

Only Clients that Support Server Name Indication (SNI)

CloudFront serves your content over HTTPS only to clients that support SNI. Older browsers and other clients that do not support SNI can not access your content over HTTPS.

[Learn More](#)

☐

All Clients (\$600/month prorated charge applies. [Learn about pricing.](#))

CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS. Any client can access your content.

[Learn More](#)

Fig. 71: SNI Custom SSL

- After CloudFront receives a request from a viewer (viewer request).
- Before CloudFront forwards the request to the origin (origin request).
- After CloudFront receives the response from the origin (origin response).
- Before CloudFront forwards the response to the viewer (viewer response).

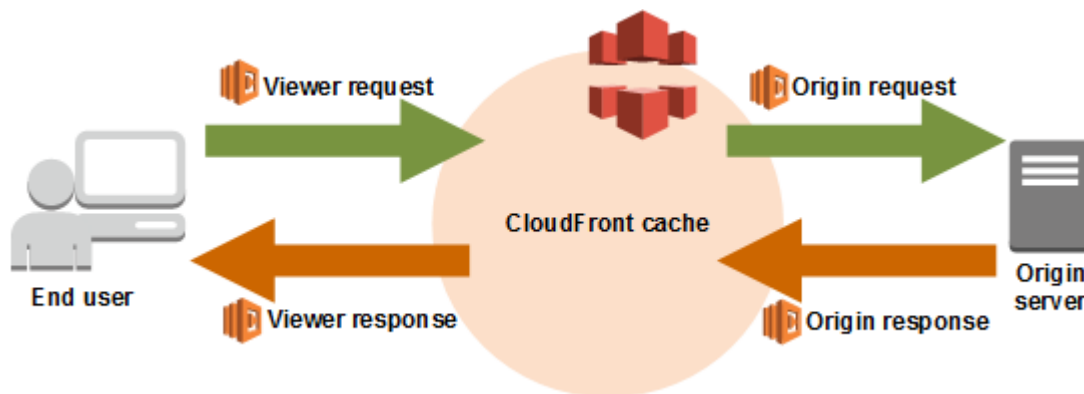


Fig. 72: Cloudfront events that trigger Lambda functions

## CloudFront signed URLs and signed cookies

CloudFront signed URLs and signed cookies provide the same basic functionality: they allow you to control who can access your content. If you want to serve private content through CloudFront and you're trying to decide whether to use signed URLs or signed cookies, consider the following:

Use **signed URLs** for the following cases:



- You want to use an RTMP distribution. Signed cookies aren't supported for RTMP distributions.
- You want to restrict access to individual files, for example, an installation download for your application.
- Your users are using a client (for example, a custom HTTP client) that doesn't support cookies.

Use **signed cookies** for the following cases:

- You want to provide access to multiple restricted files, for example, all of the files for a video in HLS format or all of the files in the subscribers' area of a website.
- You don't want to change your current URLs.

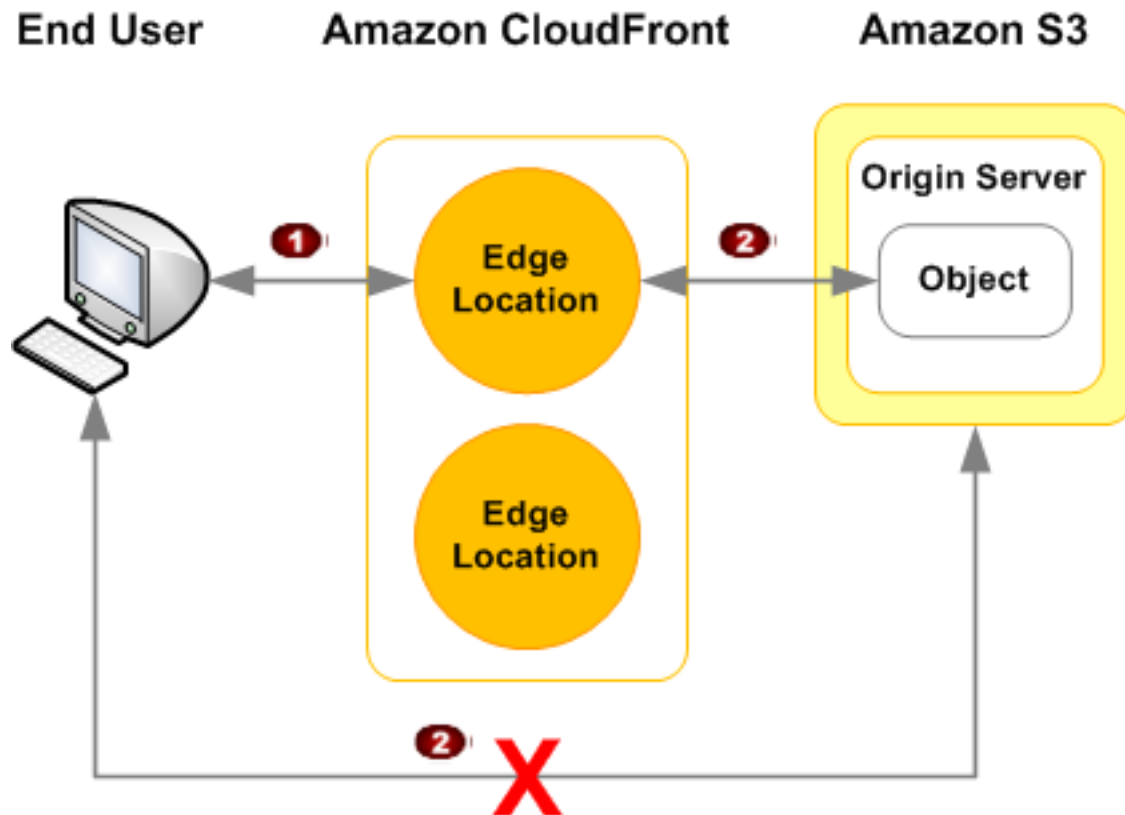


Fig. 73: Private Content in CloudFront

## Pricing

To estimate the cost of using CloudFront, you need to consider the following:

- **Traffic distribution.** Data transfer and request pricing vary across geographic regions, and pricing is based on the edge location through which your content is served.
- **Requests.** The number and type of requests made and the geographic region in which the requests are made.
- **Data transfer out.** The amount of data transferred out of your Amazon CloudFront edge locations.

The benefit you get by caching dynamic content is that request and the response ride over the AWS backbone instead the public Internet.

You can set origins as a S3 for static content. For dynamic content, you can setup as origin EC2 instances, ELB instances, and HTTP servers. It supports SSL so that private content is protected.

### 1.10.3 Amazon ElastiCache

Amazon ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory data store or cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory data stores, instead of relying entirely on slower disk-based databases.

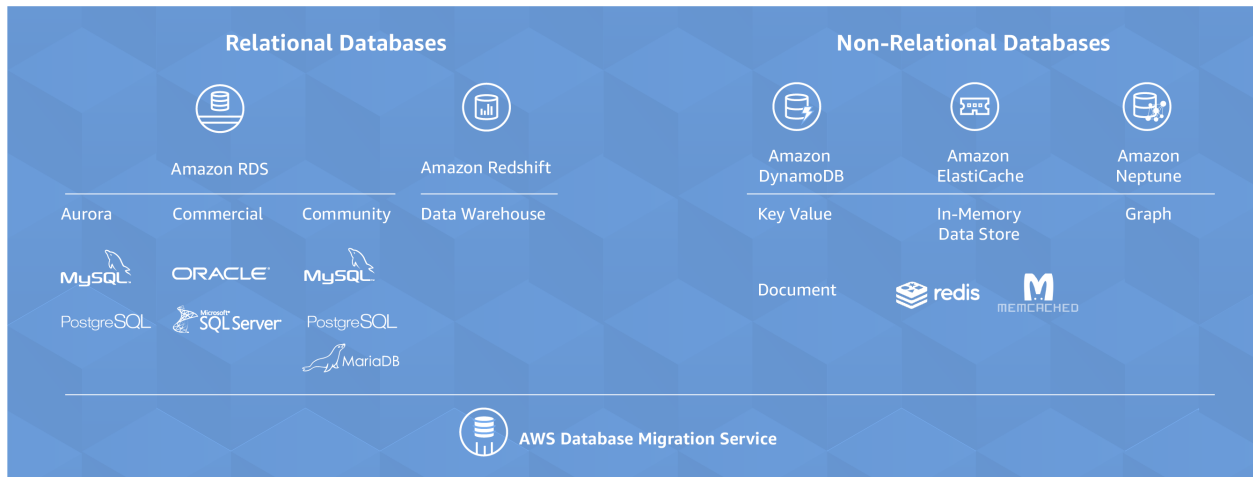
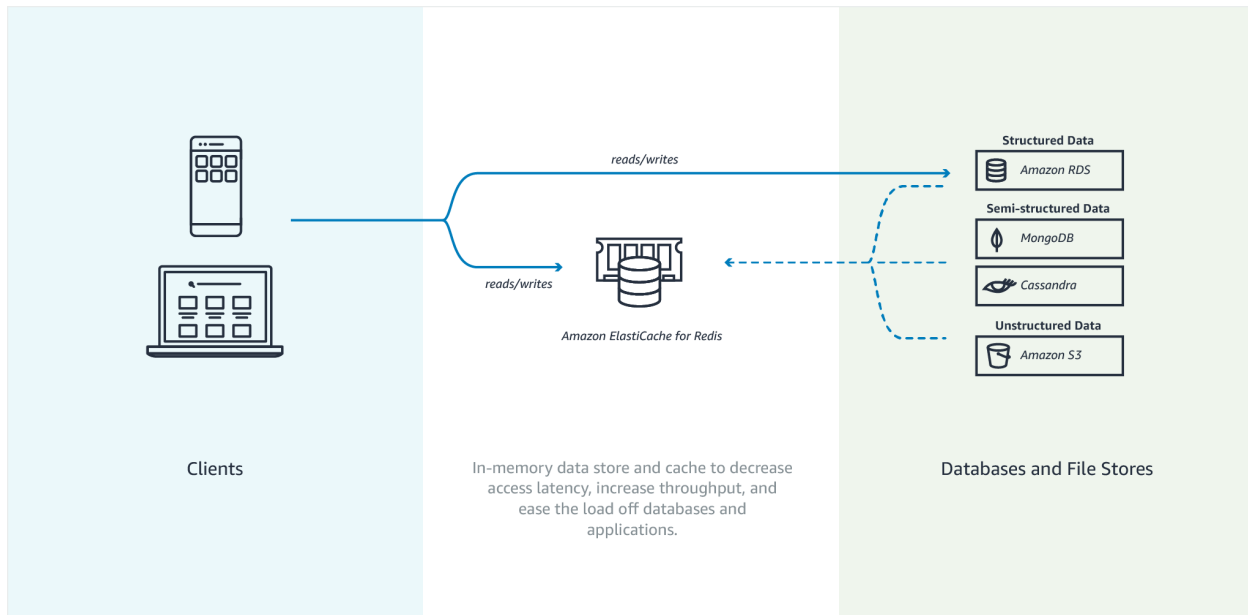


Fig. 74: Amazon ElastiCache

The primary purpose of an in-memory key-value store is to provide ultra-fast (submillisecond latency) and inexpensive access to copies of data. Most data stores have areas of data that are frequently accessed but seldom updated. Additionally, querying a database is always slower and more expensive than locating a key in a key-value pair cache. Some database queries are especially expensive to perform, for example, queries that involve joins across multiple tables or queries with intensive calculations.

By caching such query results, you pay the price of the query once and then are able to quickly retrieve the data multiple times without having to re-execute the query.



## Redis

Using Redis AUTH command can improve data security by requiring the user to enter a password before they are granted permission to execute Redis commands on a password-protected Redis server. Hence, Option 3 is the correct answer.

To require that users enter a password on a password-protected Redis server, include the parameter `--auth-token` with the correct password when you create your replication group or cluster and on all subsequent commands to the replication group or cluster.

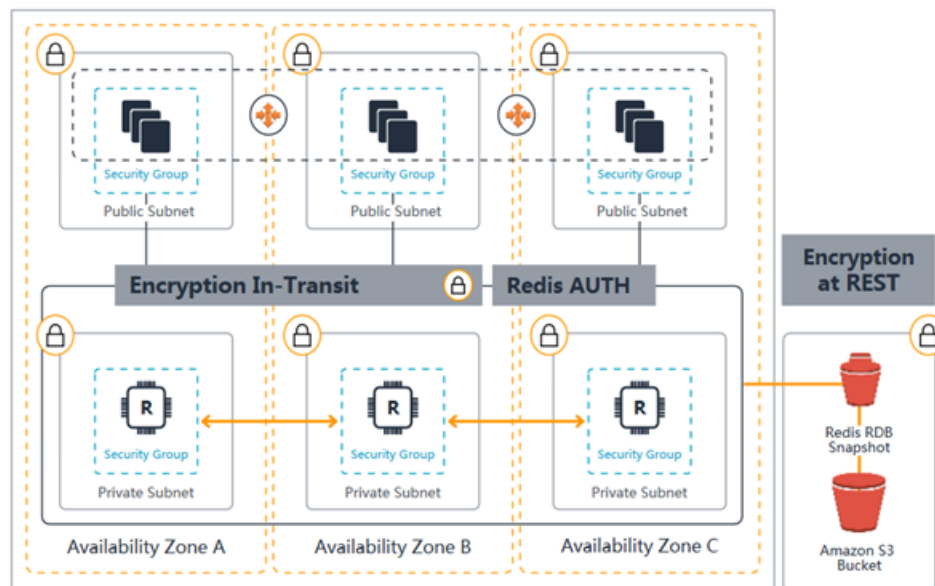


Fig. 75: Amazon ElastiCache authentication and encryption

### 1.10.4 AWS Global Accelerator

AWS Global Accelerator is a service that improves the availability and performance of your applications with local or global users. It provides static IP addresses that act as a fixed entry point to your application endpoints in a single or multiple AWS Regions, such as your Application Load Balancers, Network Load Balancers or Amazon EC2 instances.

AWS Global Accelerator uses the AWS global network to optimize the path from your users to your applications, improving the performance of your TCP and UDP traffic. AWS Global Accelerator continually monitors the health of your application endpoints and will detect an unhealthy endpoint and redirect traffic to healthy endpoints in less than 1 minute.



Many applications, such as gaming, media, mobile applications, and financial applications, need very low latency for a great user experience. To improve the user experience, AWS Global Accelerator directs user traffic to the nearest application endpoint to the client, thus reducing internet latency and jitter. It routes the traffic to the closest edge location via Anycast, then by routing it to the closest regional endpoint over the AWS global network. AWS Global Accelerator quickly reacts to changes in network performance to improve your users' application performance.

AWS Global Accelerator and Amazon CloudFront are separate services that use the AWS global network and its edge locations around the world. CloudFront improves performance for both cacheable content (such as images and videos) and dynamic content (such as API acceleration and dynamic site delivery). Global Accelerator improves performance for a wide range of applications over TCP or UDP by proxying packets at the edge to applications running in one or more AWS Regions. Global Accelerator is a good fit for non-HTTP use cases, such as gaming (UDP), IoT (MQTT), or Voice over IP, as well as for HTTP use cases that specifically require static IP addresses or deterministic, fast regional failover. Both services integrate with AWS Shield for DDoS protection.

## 1.11 Building decoupled architectures

### 1.11.1 Asynchronous messaging

If loose-coupling is important, especially in a system that requires high resilience and has unpredictable scale, another option is asynchronous messaging. Asynchronous messaging is a fundamental approach for integrating independent systems, or building up a set of loosely coupled systems that can operate, scale, and evolve independently and flexibly. As Tim Bray said, "If your application is cloud-native, or large-scale, or distributed, and doesn't include a messaging component, that's probably a bug."

When you have a monolithic application, all the communication among the different components occur internally. When you decouple the application into a microservices architecture, the communication among them can be done by messaging solutions.

In the messaging process we have several actors: the producers, the message, and the consumers. When sending messages is up to the producers and the consumers to define what the format and the contents are. You can also send attributes along with your messages, that are key/values pairs that you want to attach to your messages. These attributes can have a business meaning (for instance: `CustomerId=1234, MessageType=NewBooking`) or technical meaning (for instance: `SourceHost=ec2-203-0-113-25.compute-1.amazonaws.com, ProgramName=WebServer-PID:9989`). The payload can be encrypted but the attributes cannot. The attributes can be used to filter messages.

The AWS messaging services can be classified in terms of the use:

- **Modern application development**, which are ideal for born-in-the-cloud apps. They have simple APIs and offer unlimited throughput. We have Amazon SQS for dealing queues, Amazon SNS for subscribing to topics, Amazon Kinesis for dealing streams.
- **App migration**, which are API-compatible, feature-rich and use standard protocols. We have Amazon MQ as a message broker.

AWS re:Invent 2018: Choosing the Right Messaging Service for Your Distributed App (API305)

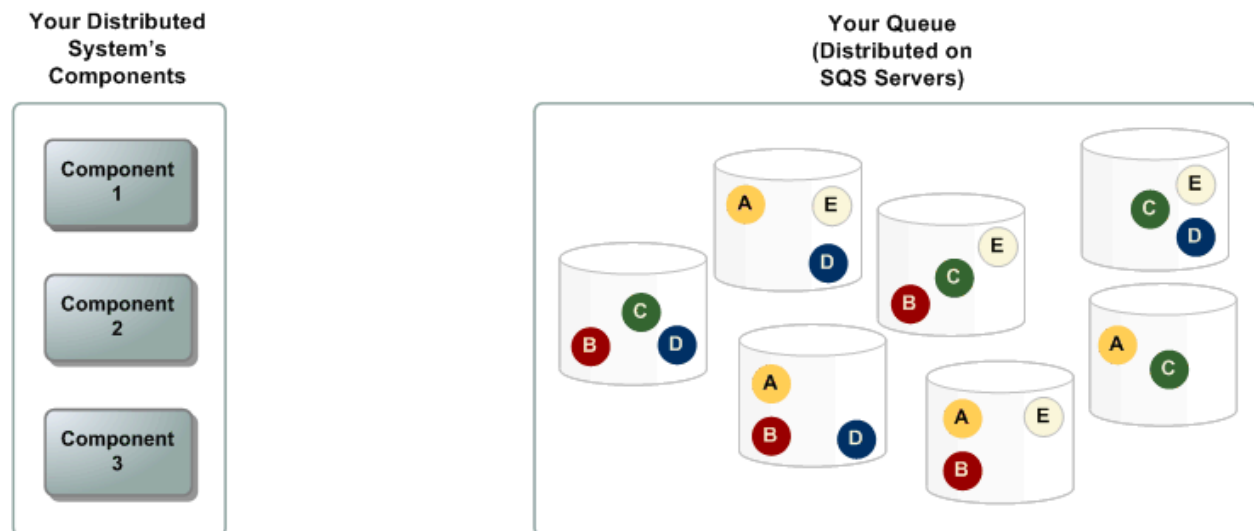
## 1.11.2 Amazon SQS

### Basic Amazon SQS Architecture

#### Distributed Queues

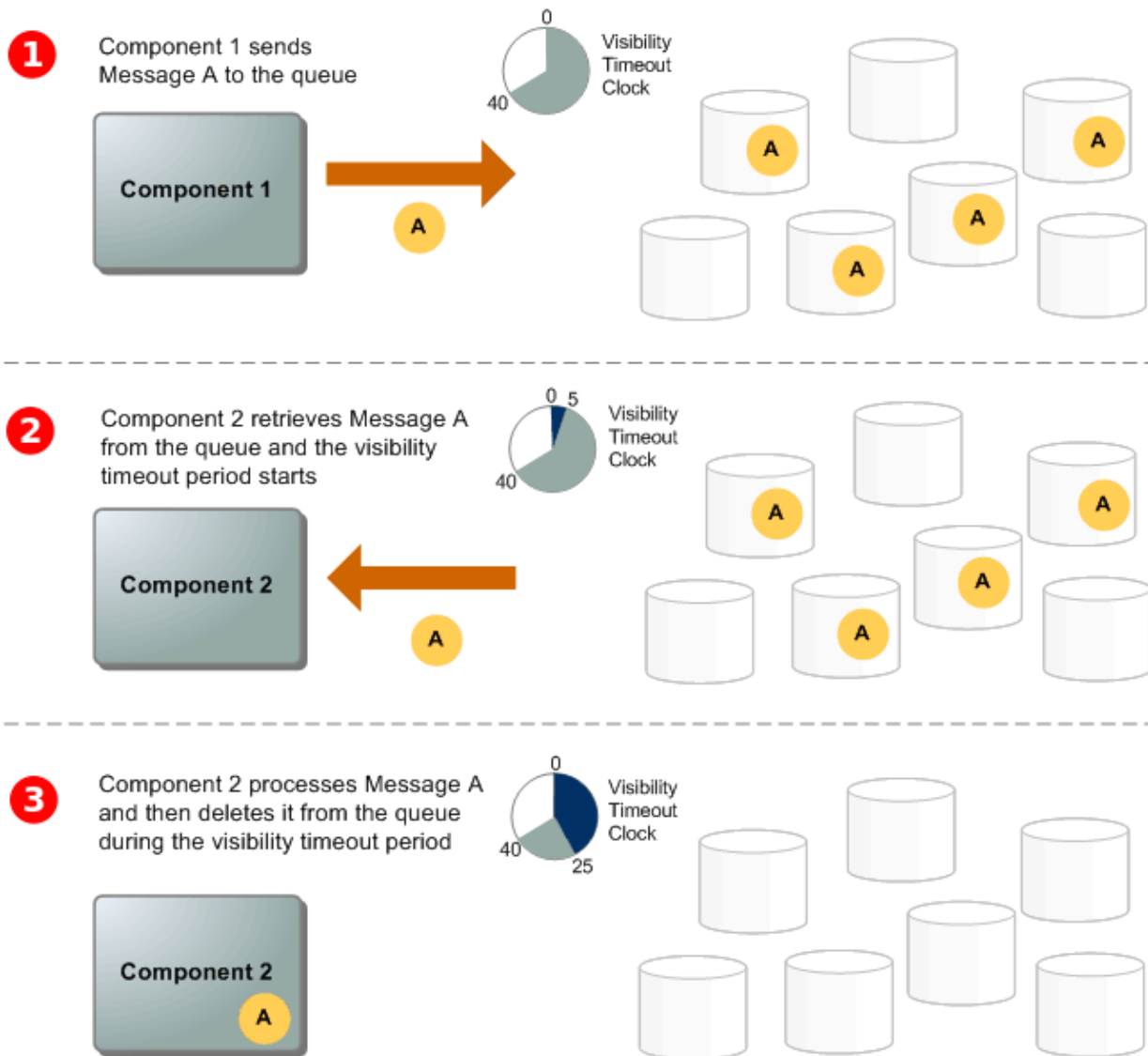
There are three main parts in a distributed messaging system: the components of your distributed system, your queue (distributed on Amazon SQS servers), and the messages in the queue.

In the following scenario, your system has several producers (components that send messages to the queue) and consumers (components that receive messages from the queue). The queue (which holds messages A through E) redundantly stores the messages across multiple Amazon SQS servers.



#### Message Lifecycle

The following scenario describes the lifecycle of an Amazon SQS message in a queue, from creation to deletion.



1. A producer (component 1) sends message A to a queue, and the message is distributed across the Amazon SQS servers redundantly.
2. When a consumer (component 2) is ready to process messages, it consumes messages from the queue, and message A is returned. While message A is being processed, it remains in the queue and isn't returned to subsequent receive requests for the duration of the visibility timeout.
3. The consumer (component 2) deletes message A from the queue to prevent the message from being received and processed again when the visibility timeout expires.

---

**Note:** Amazon SQS automatically deletes messages that have been in a queue for more than maximum message retention period. The default message retention period is 4 days. However, you can set the message retention period to a value from 60 seconds to 1,209,600 seconds (14 days) using the `SetQueueAttributes` action. Once the message retention limit is reached, your messages are automatically deleted.

---

Amazon SQS uses short polling by default, querying only a subset of the servers (based on a weighted random distribution) to determine whether any messages are available for inclusion in the response. Short polling works for scenarios that require higher throughput. However, you can also configure the queue to use Long polling instead, to reduce cost.

The `ReceiveMessageWaitTimeSeconds` is the queue attribute that determines whether you are using Short or Long polling. By default, its value is zero which means it is using Short polling. If it is set to a value greater than zero, then it is Long polling. Quick facts about SQS Long Polling:

- Long polling helps reduce your cost of using Amazon SQS by reducing the number of empty responses when there are no messages available to return in reply to a `ReceiveMessage` request sent to an Amazon SQS queue and eliminating false empty responses when messages are available in the queue but aren't included in the response.
- Long polling reduces the number of empty responses by allowing Amazon SQS to wait until a message is available in the queue before sending a response. Unless the connection times out, the response to the `ReceiveMessage` request contains at least one of the available messages, up to the maximum number of messages specified in the `ReceiveMessage` action.
- Long polling eliminates false empty responses by querying all (rather than a limited number) of the servers. Long polling returns messages as soon any message becomes available.

Amazon SQS supports dead-letter queues, which other queues (source queues) can target for messages that can't be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system because they let you isolate problematic messages to determine why their processing doesn't succeed.

## Amazon SQS Standard Queues

Amazon SQS offers standard as the default queue type. Standard queues support a nearly unlimited number of transactions per second (TPS) per API action (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`). Standard queues support at-least-once message delivery. However, occasionally (because of the highly distributed architecture that allows nearly unlimited throughput), more than one copy of a message might be delivered out of order. Standard queues provide best-effort ordering which ensures that messages are generally delivered in the same order as they're sent.

You can get duplicate messages for instance in this scenario:

1. A producer sends a message to the queue.
2. The queue stores the message durably.
3. There is a networking problem when the producer was calling send message.

4. The producer gets a timeout. It doesn't know if SQS got the message or it didn't.
5. The producer retry the sent.
6. SQS standard will store a duplicate of the message.

When you want to consume messages, the only thing you want to do is to call receive message and provide the queue URL. You do not have to tell SQS which message you want to receive, it is the responsibility of SQS to select the best message to give to you.

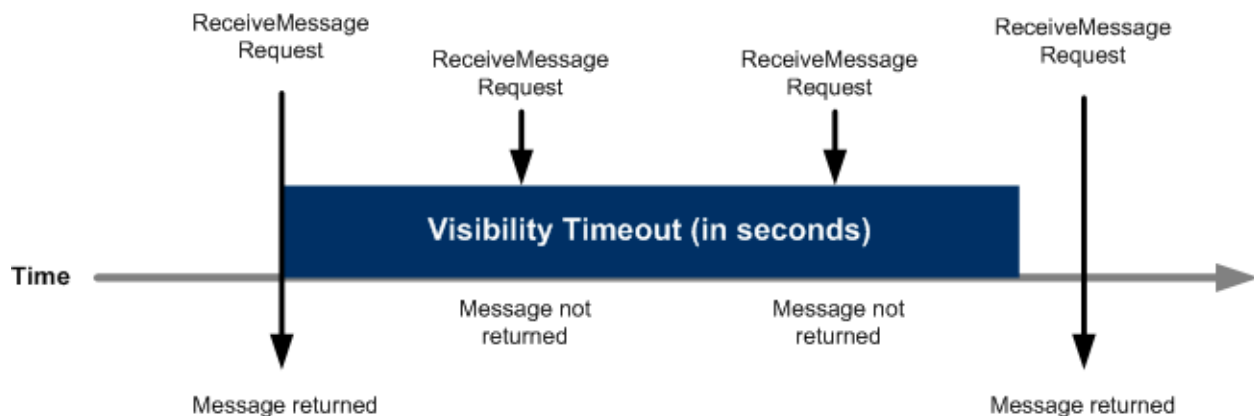
When the consumer gets the receive message, the SQS get the message and gives it to the consumer. The consumer can start working on it, but notice that the message is still in the queue. It is not immediately removed, it is invisible, and you can control the invisibility timeout. This invisibility timeout makes sure that if another consumer wants to fetch another message, SQS won't give you this particular message because some consumer is already working on it.

When the consumer successfully consumes the message, call the delete message on the message that it got, which actually achieves the removal of the message. Only when the consumer acknowledges that it successfully consumed the message, the message is removed from the queue. This guarantees that the message is consumed at least once.

When the consumer has a problem consuming the message, the easiest solution for the consumer is just forget about the message and do nothing. What happen next is that the invisibility timeout on the message it was working on expires, and the message is available for consumption again.

### Amazon SQS Visibility Timeout

When a consumer receives and processes a message from a queue, the message remains in the queue. Amazon SQS doesn't automatically delete the message. Because Amazon SQS is a distributed system, there's no guarantee that the consumer actually receives the message (for example, due to a connectivity issue, or due to an issue in the consumer application). Thus, the consumer must delete the message from the queue after receiving and processing it.



Immediately after a message is received, it remains in the queue. To prevent other consumers from processing the message again, Amazon SQS sets a visibility timeout, a period of time during which Amazon SQS prevents other consumers from receiving and processing the message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours.

### Message Ordering

A standard queue makes a best effort to preserve the order of messages, but more than one copy of a message might be delivered out of order. If your system requires that order be preserved, we recommend using a FIFO (First-In-First-Out) queue or adding sequencing information in each message so you can reorder the messages when they're received.



## At-Least-Once Delivery

Amazon SQS stores copies of your messages on multiple servers for redundancy and high availability. On rare occasions, one of the servers that stores a copy of a message might be unavailable when you receive or delete a message.

If this occurs, the copy of the message isn't deleted on that unavailable server, and you might get that message copy again when you receive messages. Design your applications to be idempotent (they should not be affected adversely when processing the same message more than once).

## Amazon SQS FIFO (First-In-First-Out) Queues

FIFO queues have all the capabilities of the standard queue. FIFO (First-In-First-Out) queues are designed to enhance messaging between applications when the order of operations and events is critical, or where duplicates can't be tolerated. FIFO queues also provide exactly-once processing but have a limited number of transactions per second (TPS):

- By default, with batching, FIFO queues support up to 3,000 messages per second (TPS), per API action (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`). To request a quota increase, submit a support request.
- Without batching, FIFO queues support up to 300 messages per second, per API action (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`).

---

**Note:** Amazon SNS isn't currently compatible with FIFO queues.

The name of a FIFO queue must end with the `.fifo` suffix. The suffix counts towards the 80-character queue name quota. To determine whether a queue is FIFO, you can check whether the queue name ends with the suffix.

---

Typically what you need is to process messages in sequence for specific subgroup of messages, such as Customer ID, but you can work with multiple customers in parallel. To send the message to the FIFO queue, the producer must tell what the message group for which the message belongs to. It is just a tag that you put in the message and it is not necessary to pre-create this group. There is no limitation of the number of messages that you can send.

Imagine an scenario where:

1. A producer sends a message to the queue.
2. The queue stores the message durably.
3. There is a networking problem when the producer was calling send message.
4. The producer gets a timeout. It doesn't know if SQS got the message or it didn't.
5. The producer retry the sent.
6. SQS keeps track of the identifiers of the messages sent to it in the last 5 minutes, even if they are already consumed. As a consequence, it is able to detect that it is retry of sending the same message and no duplicate is introduced in the queue. An OK is returned to the producer, because the message is already present.

When the consumer calls receive and FIFO decides the group you are going to get messages. The SQS get the message and gives it to the consumer. The consumer can start working on it, but notice that the message is still in the queue. It is not immediately removed, it is invisible, and you can control the invisibility timeout. The difference with standard queues is that no other consumer can receive messages from the same group as selected for this message. This is how it is preserved the order of the messages within the group. The entire group is lock until the consumer finishes processing the message.

When the consumer successfully consumes the message, call the delete message on the message that it got, which actually achieves the removal of the message. Only when the consumer acknowledges that it successfully consumed

the message, the message is removed from its group within the queue. This guarantees that the message is consumed only once. The group is unlocked and another messages from this group can be consumed for the same or another consumer. You cannot guarantee which consumer is going to get the next message, there is no consumer affinity.

When the consumer has a problem consuming the message, the easiest solution for the consumer is just forget about the message and do nothing. What happen next is that the invisibility timeout on the message it was working on expires, and the group is available for consumption again.

## Limits

A single Amazon SQS message queue can contain an unlimited number of messages. However, there is a 120,000 limit for the number of inflight messages for a standard queue and 20,000 for a FIFO queue. Messages are inflight after they have been received from the queue by a consuming component, but have not yet been deleted from the queue.

### 1.11.3 Amazon SNS

The most important characteristics of SNS are the following:

- It is a flexible, fully managed pub/sub messaging and mobile communications service.
- It coordinates the delivery of messages to subscribing endpoints and clients, therefore enabling you to send different information to different subscribers.
- It is easy to setup, operate and send reliable communications.
- It allows you to decouple and scale microservices, distributed systems and serverless communications.

Amazon SNS allows you to have pub/sub messaging for different systems in Amazon, like AWS Lambda, HTTP/S and Amazon SQS. Amazon SNS Mobile Notifications allows you to do similar publishing but to different mobile systems, like ADM, APNS, Baidu, GCM, MPNS, and WNS.

## Amazon SNS topics

The objective of SNS is to send something and deliver it to multiple destinations. It is a pub/sub model in which you publish something and you have multiple subscribers via SNS topics. In this case, we have producers but no consumers.

With a topic, you can publish messages to it or configure subscriptions or destinations you want to deliver messages. The destinations that you can configure are Amazon SQS (except FIFO queue which are not supported yet), AWS Lambda, HTTP/s endpoint, mobile app, SMS, e-mail.

When a producer publish a message, it gets an acknowledge, before even it is delivered to each destination. It means that you will see the same latency of publishing locations whether you have one destination or multiple destinations.

What happens internally in SNS is that a fanout is performed. For each subscribed destination, a copy of the message will be sent. You can configure filters in destinations that prevent to arrive a several destinations. You can interpret this stage as multiple internal queues that you do not see that keeps track of each individual destination. Finally, the copies of the messages are sent to the destinations.

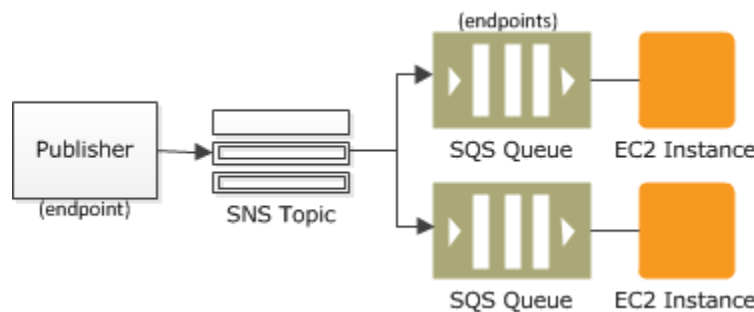
If one of the channels fail for any reason (for instance, it was an HTTP endpoint and the web server was not running), we receive notifications for the successful deliveries and we keep track of the delivery that failed. We will retry the failed message sending, the number of retries depend on the destination. For an SQS queue or an AWS Lambda it will retry a large amount of times and the message will probably be delivered. For HTTP endpoints, you have to define a delivery policy. Each delivery policy is comprised of four phases:

1. *Immediate Retry Phase (No Delay)*. This phase occurs immediately after the initial delivery attempt. There is no delay between retries in this phase.

2. *Pre-Backoff Phase*. This phase follows the Immediate Retry Phase. Amazon SNS uses this phase to attempt a set of retries before applying a backoff function. This phase specifies the number of retries and the amount of delay between them.
3. *Backoff Phase*. This phase controls the delay between retries by using the retry-backoff function. This phase sets a minimum delay, a maximum delay, and a retry-backoff function that defines how quickly the delay increases from the minimum to the maximum delay. The backoff function can be arithmetic, exponential, geometric, or linear.
4. *Post-Backoff Phase*. This phase follows the backoff phase. It specifies a number of retries and the amount of delay between them. This is the final phase.

## Fanout

A “fanout” pattern is when an Amazon SNS message is sent to a topic and then replicated and pushed to multiple Amazon SQS queues, HTTP endpoints, or email addresses. This allows for parallel asynchronous processing. For example, you could develop an application that sends an Amazon SNS message to a topic whenever an order is placed for a product. Then, the Amazon SQS queues that are subscribed to that topic would receive identical notifications for the new order. The Amazon EC2 server instance attached to one of the queues could handle the processing or fulfillment of the order, while the other server instance could be attached to a data warehouse for analysis of all orders received.



When a consumer receives and processes a message from a queue, the message remains in the queue. Amazon SQS doesn't automatically delete the message. Because Amazon SQS is a distributed system, there's no guarantee that the consumer actually receives the message (for example, due to a connectivity issue, or due to an issue in the consumer application). Thus, the consumer must delete the message from the queue after receiving and processing it.

Immediately after the message is received, it remains in the queue. To prevent other consumers from processing the message again, Amazon SQS sets a visibility timeout, a period of time during which Amazon SQS prevents other consumers from receiving and processing the message. The default visibility timeout for a message is 30 seconds. The maximum is 12 hours.

### 1.11.4 Amazon MQ

Amazon MQ is a managed message broker service for Apache ActiveMQ that makes it easy to set up and operate message brokers in the cloud. Connecting your current applications to Amazon MQ is easy because it uses industry-standard APIs and protocols for messaging, including JMS, NMS, AMQP, STOMP, MQTT, and WebSocket. Using standards means that in most cases, there's no need to rewrite any messaging code when you migrate to AWS.

Amazon MQ, Amazon SQS, and Amazon SNS are messaging services that are suitable for anyone from startups to enterprises. If you're using messaging with existing applications and want to move your messaging service to the cloud quickly and easily, it is recommended that you consider Amazon MQ. It supports industry-standard APIs and protocols so you can switch from any standards-based message broker to Amazon MQ without rewriting the messaging code in your applications.

If you are building brand new applications in the cloud, then it is highly recommended that you consider Amazon SQS and Amazon SNS. Amazon SQS and SNS are lightweight, fully managed message queue and topic services that scale almost infinitely and provide simple, easy-to-use APIs. You can use Amazon SQS and SNS to decouple and scale microservices, distributed systems, and serverless applications, and improve reliability.

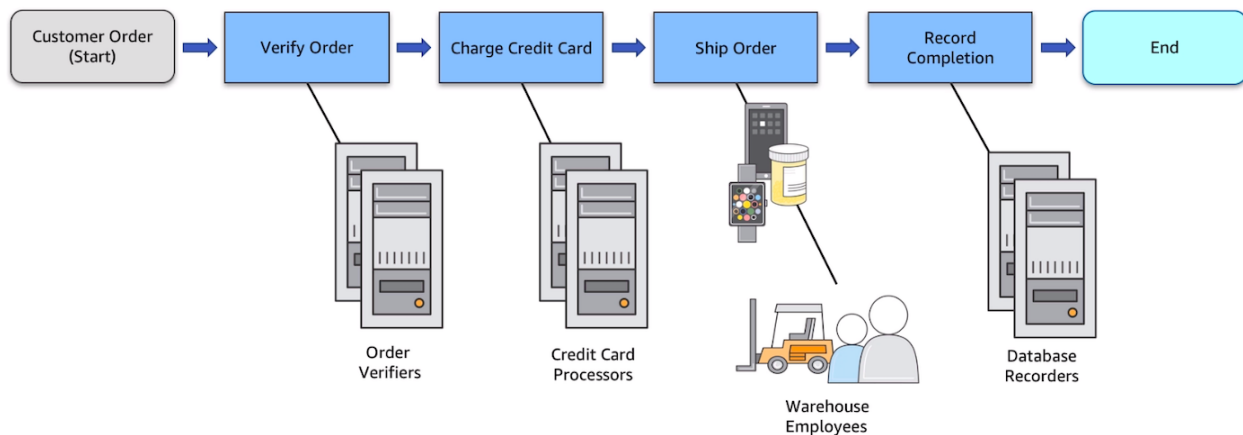
### 1.11.5 Amazon Simple Workflow Service (SWF)

#### Introduction

It is a managed workflow service that helps developers build, run, and scale applications that coordinate work across distributed components. You can use Amazon SWF as a fully managed state tracker and task coordinator for your background jobs that require sequential and parallel steps. An application can consist of several different tasks to be performed and a certain sequence driven by a set of conditions. SWF makes it easy to architect and implement and coordinate these tasks in AWS cloud.

When building solutions to coordinate tasks in a distributed environment, the developer has to account for several variables. Tasks that drive a processing step can be long running and may fail, timeout or require a restart. They often complete with varying throughputs and latencies. Tracking and visualization tasks in all these cases is not only challenging but also undifferentiated work. As applications and tasks scale up, you see different distributed system problems, for example: you must ensure that a task is assigned once and that the outcome is tracked reliably through unexpected failures and outages.

By using Amazon SWF you can easily manage your application tasks and how to coordinate them.



The tasks in this workflow are sequential. An order must be verified before making a charge in the credit card. A credit card must be charged successfully before an order must be shipped. An order must be shipped before being recorded. Even so, because SWF supports distributed processes, these tasks can be carried out in different locations. If the tasks are programmatic in nature, they can also be written in different programming languages or using different tools.

In addition to sequential processing of tasks, SWF also supports workflow with parallel processing of tasks. Parallel tasks are performed at the same time and may be carried out independently by different applications or human workers. Your workflow makes decisions on how to proceed once one or more parallel tasks have been completed.

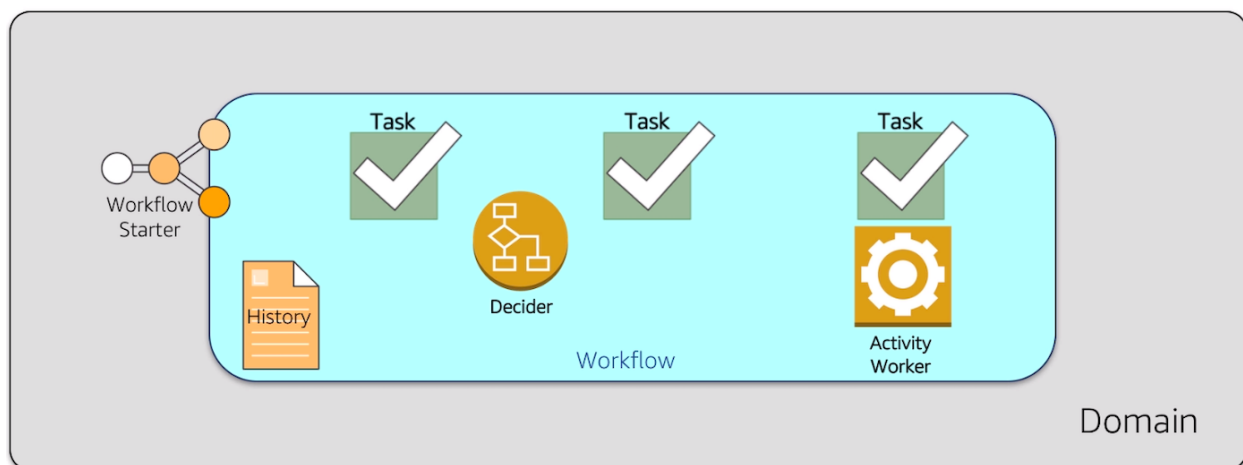
The benefits of SWF are:

- **Logical separation.** The service promotes the separation between the control flow of your background jobs stepwise logic and the actual units of work that contains your unique business logic. This allows you to separately manage, maintain and scale state machinery of your application from the core business logic. As your business requirements change, you can easily change application logic without having to worry about your state machinery, tasks dispatch and flow control.

- **Reliable.** It runs on Amazon HA data centers so the state tracking and tasks processing engine is available whenever application need them. SWF redundantly stores tasks, reliably dispatches them to application components, track the progress and keeps the related state.
- **Simple.** It is simple to use and replaces the complexity of custom-coded solution and process automation software with a fully managed cloud web service. This eliminates the need for developers to manage infrastructure plumbing for the process automation so they can focus on the functionality of the application.
- **Scale.** SWF scales with your application usage. No manual administration of your workflow service is required as you add more workflows to your application or increase the complexity of your workflows.
- **Flexible.** It allows you write your application components and coordination logic in any programming language and run them in the cloud or on-premises.

## Overview

Domain is a collection of related workflows. A workflow starter is any application that can initiate workflow executions. Workflows are collections of actions and actions are tasks or workflow steps. Decider implements the workflow coordination logic. Activity workers implements actions. Workflow history is the detailed, complete and consistent record of every event that occur since the workflow execution started. Additionally, in the course of its operations SWF interacts with a number of different types of programmatic actors. Actors can be workflow starters, deciders or activity workers. These actors communicate with SWF through its API. You can develop these actors in any programming language.



Each workflow runs on an AWS resource called the **Domain**. Domains provide a way of scoping SWF resources within your AWS account. All the components of your workflow such as workflow types and activities types must be specified to be in a domain. It is possible to have more than a workflow in a domain. However, workflows in different domains can interact one with another. When setting up a new workflow, before you set up any of the other workflow components you have to register a domain if you have not done so. When you register a domain you have to specify a workflow history retention period. This period is the length of time SWF would continue to retain information about the workflow execution after the workflow execution is complete.

**Workflows** coordinate and manage the execution of activities that can be run asynchronously across multiple computing devices and that can feature both sequential and parallel processing. The workflow starter starts the workflow instance, also referred as the workflow execution, and can interact with an instance during execution for purposes such as pass additional data to the workflow worker or obtaining the current workflow state. The workflow starter uses a workflow client to start the workflow execution and interacts with the workflow as needed during execution and handles clean up. The workflow starter can be a locally run application, a web application, the AWS CLI or even the AWS management console.

SWF interacts with activity workers and deciders by providing them with work assignments known as **tasks**. There are 3 types of tasks in Amazon SWF:

- *Activity task* tells an activity worker to perform its function, such as to check inventory or charge a credit card. The activity task contains all the information that the activity worker needs to perform its function.
- *Lambda task* is similar to an activity task but executes the a Lambda function an set up a traditional SWF activity.
- *Decision task* tells the decider that the state of the workflow execution has changed, so that the decider can determine the next activity that needs to be performed. It contains to current workflow history. SWF schedule the decision task when the workflow starts and whenever the state of a workflow changes, such as when an activity task complete.

Each decision task contains a paginated view of the entire workflow execution history. The decider analyzes the workflow execution history and responds back to SWF with a set of decisions that specify what should occur next in the workflow execution. Essentially, every decision task gives the decider an opportunity to assess the workflow and provides direction back to SWF. To ensure that no conflicting decisions are processed, SWF assigns each decision task to exactly one decider and allows one decision task at a time to be active in a workflow execution.

A **decider** is an implementation of a workflow coordination logic. Deciders control the flow of activity tasks in a workflow execution. Whenever a change occur in workflows executions, such as the completion of an activity task, SWF creates a decision task that contains the workflow history up to that point in time and assigns the task to a decider. When the decider receives the decision task from the SWF, it analyzes the workflow execution history to determine the next appropriate steps in the workflow execution. The decider communicates these steps back to SWF using decisions. A decision is a SWF data type that can represent next actions.

An **activity worker** is a process or thread that performs the activity tasks that are part of your workflow. The activity tasks represent one of the tasks that you identified in your application. Each activity worker pulls SWF for new tasks that is appropriate for that activity worker to perform and certain tasks can only be performed by certain activity workers. After receiving a task, the activity worker processes the tasks to completion and reports SWF that the task was completed and provides the result. The activity worker polls then for a new task. The activity worker is associated with a workflow execution continuing this way: processing tasks until the workflow execution itself is complete. Multiple activity workers can process tasks of the same activity type.

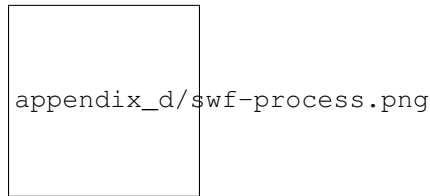
**Workflow history** is a detailed, complete, and consistent record of every event that occurred since the workflow execution started. An event represents a discrete change in the workflow execution state, such as a new activity being scheduled or running activity being completed. The workflow history contains every event that causes the execution state of the workflow execution to change, such as scheduled completed activities, tasks timeouts and signals. Operations that don't change the state of the workflow execution don't typically appear in the workflow history. For example, the workflow history doesn't show pull attempts or the use of visibility operations. The Workflow history has a set of key benefits:

- It enables applications to be stateless because all information about a workflow execution is stored in the Workflow history.
- For each workflow execution the Workflow history provides of what activities were scheduled, the current status, and the results. The workflow execution uses this information to determine the next steps.
- The history provides the detailed audit trail that you can use to monitor running workflow executions and verify completed workflow executions.

The process that SWF follows is the following:

1. A workflow starter kick outs your workflow execution. For example, this can be a web server front end.
2. SWF receives the start workflow execution request and then scheduled a Decision task.
3. The decider receives the task from SWF, reviews the history and applies the coordination logic to determine the activity that needs to be performed.
4. SWF receives the decision, schedule the activity task and waits the activity task to complete or time out.

5. SWF assigns the activity to a worker that performs the tasks and returns the results to SWF.
6. SWF receives the results of the activity, add them to the workflow history, and schedule the decision task.
7. This process repeats itself for each activity in your workflow.



Deciders and activity workers communicates with SWF using long polling. With this approach, the decider or activity worker periodically initiates communication with SWF notifying SWF of visibility to accept the task and then specify the tasks list to get tasks from. If the task is available in a specified task list, SWF returns it immediately in the response. If no task is available, SWF holds the TCP connection open up to 60 seconds, so that if a task becomes available during that time, it can be returned in the same connection.

## Use Cases

In general, customers have used SWF to build applications for video encoding, social commerce, infrastructure provisioning, mapreduce pipelines, business process management, and several other use cases.

## Other considerations

By default, each workflow execution can run for a maximum of 1 year in Amazon SWF. This means that it is possible that in your workflow, there are some tasks which require manual action that renders it idle.

Amazon SWF does not take any special action if a workflow execution is idle for an extended period of time. Idle executions are subject to the timeouts that you configure. For example, if you have set the maximum duration for an execution to be 1 day, then an idle execution will be timed out if it exceeds the 1 day limit. Idle executions are also subject to the Amazon SWF limit on how long an execution can run (1 year).

## 1.12 Microservices and serverless architectures

### 1.12.1 Amazon ECS

Amazon ECS enables you to inject sensitive data into your containers by storing your sensitive data in either AWS Secrets Manager secrets or AWS Systems Manager Parameter Store parameters and then referencing them in your container definition. This feature is supported by tasks using both the EC2 and Fargate launch types. Secrets can be exposed to a container in the following ways:

- To inject sensitive data into your containers as environment variables, use the `secrets` container definition parameter.
- To reference sensitive information in the log configuration of a container, use the `secretOptions` container definition parameter.

Within your container definition, specify `secrets` with the name of the environment variable to set in the container and the full ARN of either the Secrets Manager secret or Systems Manager Parameter Store parameter containing the sensitive data to present to the container. The parameter that you reference can be from a different Region than the container using it, but must be from within the same account.

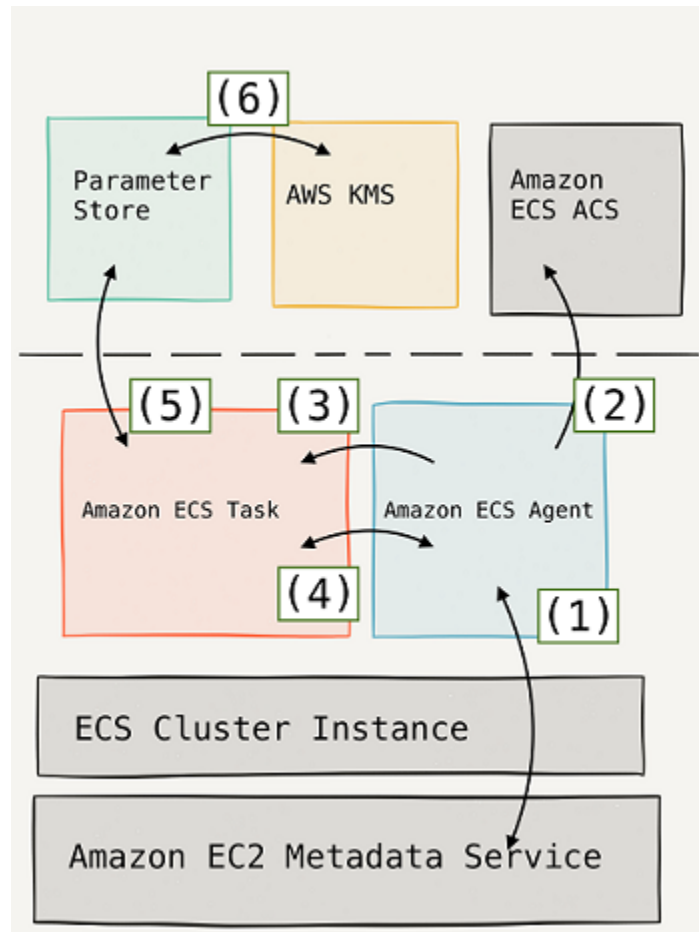


Fig. 76: AWS Lambda



### 1.12.2 AWS Lambda

There are 4 main principles that allows us to identify if a service is serverless or not:

- There are no servers to provision or manage.
- It scales with usage.
- You pay for value. You don't need to pay for the time the resource is idle.
- It has availability and fault tolerance built in. In AWS, serverless take advantage of the availability and fault tolerance given by AWS global infrastructure.

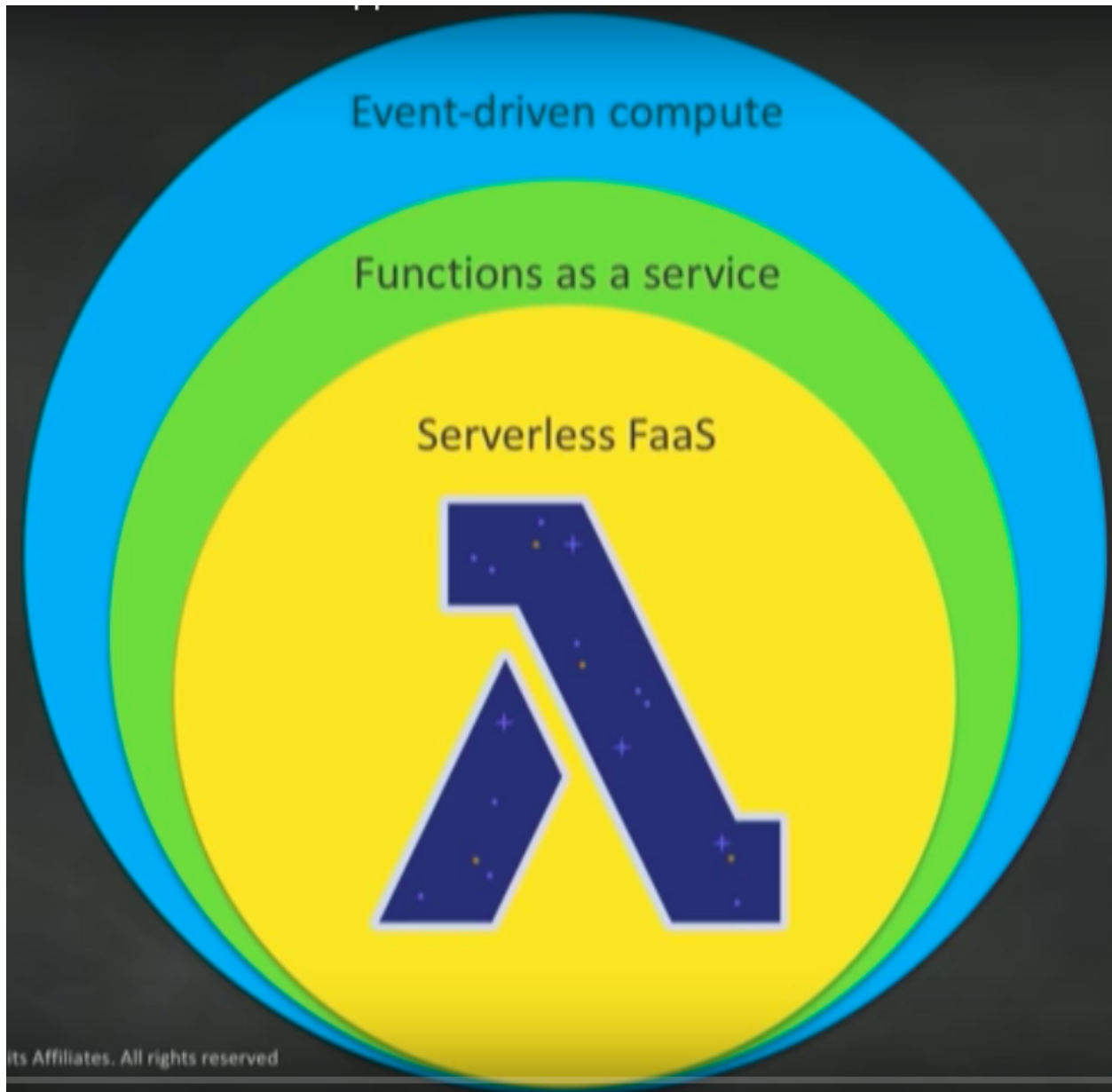


Fig. 77: AWS Lambda

AWS Lambda handles load balacing, auto scaling, failures, security isolation, OS management, ... for you.

Serverless applications have 3 components: the event source, that triggers the AWS Lambda function, which calls some **services**.

- **Event sources** can be changes in changes in data state, requests to endpoints, changes in resource state.
- **Functions** that can be developed in Node.js, Python, Java, C#, Go, Ruby, Runtime API.

A Lambda function consists of code and any associated dependencies. In addition, a Lambda function also has configuration information associated with it. Initially, you specify the configuration information when you create a Lambda function. Lambda provides an API for you to update some of the configuration data. The anatomy of a Lambda function is as follows:

- **Handler() function** is the function to be executed upon invocation.
- **Event object** is data sent during Lambda function invocation.
- **Context object** are the methods available to interact with runtime information (request ID, log group, more).

AWS Lambda runtime API and layers are features that allow developers to share, discover, and deploy both libraries and languages as part of their serverless applications. Runtime API enables developers to use Lambda with any programming language. Layers let functions easily share code. Upload layer once, reference within any function. Layers promote separation of responsibilities, lets developers focus on writing business logic. Combined, runtime API and layers allow developers to share any programming language or language version with others.

There are 3 types of Lambda execution models:

- **Synchronous (push)**. For instance, an Amazon API Gateway request to AWS Lambda functions and expect some response.
- **Asynchronous (event)**. For instance, a Amazon SNS invokes AWS Lambda function and it makes some task on it without sending nothing to the client.
- **Poll-based**. For instance, Amazon DynamoDB Streams or Amazon Kinesis streams are continuously poll and Lambda makes an invocation for you.

The Lambda permission model allows you to establish fine-grained security controls for both execution and invocation:

- **Execution policies**. They define what AWS resources/API calls can this function access via IAM and is used in streaming invocations. For instance: Lambda function A can read from DynamoDB table users.
- **Function policies** are used for sync and async invocations. For example: Actions on bucket X can invoke Lambda function Z.

## Introduction to AWS Lambda & Serverless Applications

### Building APIs with Amazon API Gateway

## Environment Variables

When you create or update Lambda functions that use environment variables, AWS Lambda encrypts them using the AWS Key Management Service. When your Lambda function is invoked, those values are decrypted and made available to the Lambda code.

Lambda encrypts environment variables with a key that it creates in your account (an AWS managed customer master key). Use of this key is free. You can also choose to provide your own key for Lambda to use instead of the default key.

However, if you wish to use encryption helpers and use KMS to encrypt environment variables after your Lambda function is created, you must create your own AWS KMS key and choose it instead of the default key. The default key will give errors when chosen. Creating your own key gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.

**Environment variables**

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

Key	Value	Encrypt	Code	Remove
password	AQICAHgdCwJ7eNzG0cBk9Q6nDD21wmtlCvWz2AsE75No	Encrypt	Code	Remove

▼ Encryption configuration

Enable helpers for encryption in transit [Info](#)

☒

AWS KMS key to encrypt in transit

arn:aws:kms:us-east-1:8420 ?key/2defc6c2-ab8a-499f-87de-

⚠ AWS KMS call failed for reason: User: arn:aws:iam::84205 :user/koko is not authorized to perform: kms:Encrypt on resource: arn:aws:kms:us-east-1:84205 :2defc6c2-ab8a-499f-87de-

AWS KMS key to encrypt at rest [Info](#)

Choose an AWS KMS key to encrypt the environment variables at rest, or simply let Lambda manage the encryption.

☒ (default) aws/lambda

☐ Use a customer master key

Fig. 78: AWS Lambda Environment Variables

When you provide the key, only users in your account with access to the key can view or manage environment variables on the function. Your organization may also have internal or external requirements to manage keys used for encryption and control when they are rotated.

You can also encrypt environment variable values client-side before sending them to Lambda, and decrypt them in your function code. This obscures secret values in the Lambda console and API output, even for users who have permission to use the key. In your code, you retrieve the encrypted value from the environment and decrypt it by using the AWS KMS API.

## Networking

You can configure a function to connect to a virtual private cloud (VPC) in your account. Use Amazon Virtual Private Cloud (Amazon VPC) to create a private network for resources such as databases, cache instances, or internal services. Connect your function to the VPC to access private resources during execution.

AWS Lambda runs your function code securely within a VPC by default. However, to enable your Lambda function to access resources inside your private VPC, you must provide additional VPC-specific configuration information that includes VPC subnet IDs and security group IDs. AWS Lambda uses this information to set up elastic network interfaces (ENIs) that enable your function to connect securely to other resources within your private VPC.

Lambda functions cannot connect directly to a VPC with dedicated instance tenancy. To connect to resources in a dedicated VPC, peer it to a second VPC with default tenancy.

Your Lambda function automatically scales based on the number of events it processes. If your Lambda function accesses a VPC, you must make sure that your VPC has sufficient ENI capacity to support the scale requirements of your Lambda function. It is also recommended that you specify at least one subnet in each Availability Zone in your Lambda function configuration.

By specifying subnets in each of the Availability Zones, your Lambda function can run in another Availability Zone if one goes down or runs out of IP addresses. If your VPC does not have sufficient ENIs or subnet IPs, your Lambda function will not scale as requests increase, and you will see an increase in invocation errors with EC2 error types like `EC2ThrottledException`. For asynchronous invocation, if you see an increase in errors without corresponding CloudWatch Logs, invoke the Lambda function synchronously in the console to get the error responses.

## Monitoring

AWS Lambda automatically monitors functions on your behalf, reporting metrics through Amazon CloudWatch. These metrics include total invocation requests, latency, and error rates. The throttles, Dead Letter Queues errors and Iterator age for stream-based invocations are also monitored.

You can monitor metrics for Lambda and view logs by using the Lambda console, the CloudWatch console, the AWS CLI, or the CloudWatch API.

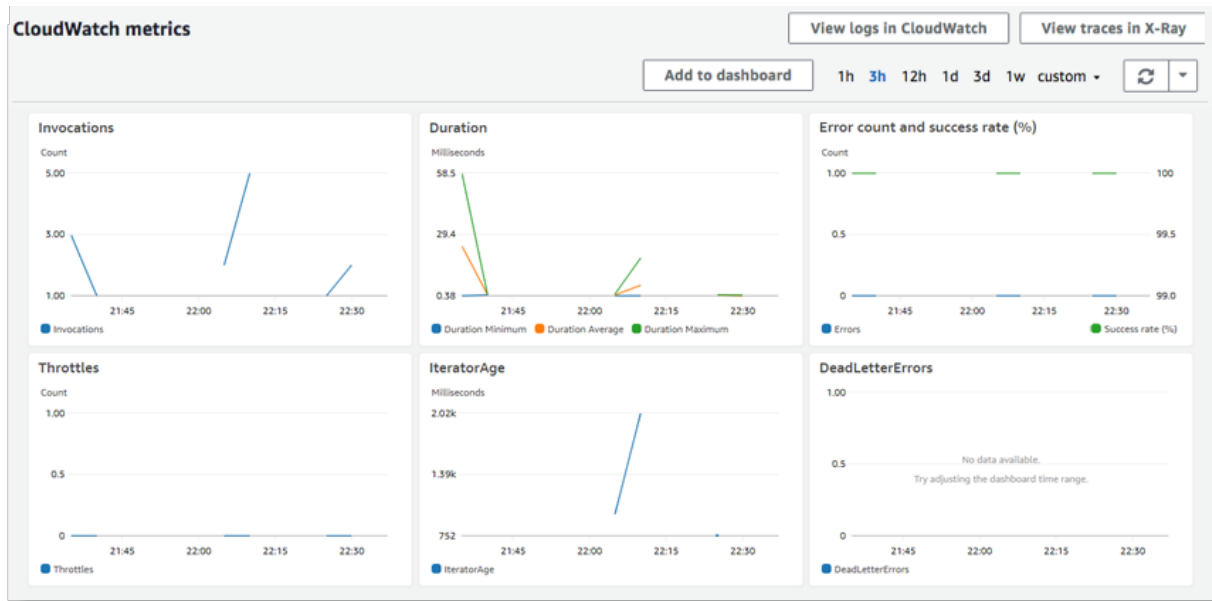


Fig. 79: AWS Lambda metrics functions list

## Pricing

You buy compute time in 100ms increments. There is no hourly, daily, or monthly minimums and no per-device fees. You never pay for idle time. The free tier covers 1 million requests and 400,000 GBs of compute every month, every customer.

Lambda exposes only a memory control, with the percentage of CPU core and network capacity allocated to a function proportionally. If your code is CPU, network or memory-bound, then it could be cheaper to choose more memory.

You pay for the AWS resources that are used to run your Lambda function. To prevent your Lambda function from running indefinitely, you specify a timeout. When the specified timeout is reached, AWS Lambda terminates execution of your Lambda function. It is recommended that you set this value based on your expected execution time. The default timeout is 3 seconds and the maximum execution duration per request in AWS Lambda is 900 seconds, which is equivalent to 15 minutes.

## Limits

By default, the AWS Lambda limits the total concurrent executions across all functions within a given region to 1000. By setting a concurrency limit on a function, Lambda guarantees that allocation will be applied specifically to that function, regardless of the amount of traffic processing the remaining functions. If that limit is exceeded, the function will be throttled but not terminated.

### 1.12.3 AWS Step Functions

AWS Step Functions provides serverless orchestration for modern applications. Orchestration centrally manages a workflow by breaking it into multiple steps, adding flow logic, and tracking the inputs and outputs between the steps. As your applications execute, Step Functions maintains application state, tracking exactly which workflow step your application is in, and stores an event log of data that is passed between application components. That means that if networks fail or components hang, your application can pick up right where it left off.

Application development is faster and more intuitive with Step Functions, because you can define and manage the workflow of your application independently from its business logic. Making changes to one does not affect the other. You can easily update and modify workflows in one place, without having to struggle with managing, monitoring and maintaining multiple point-to-point integrations. Step Functions frees your functions and containers from excess code, so your applications are faster to write, more resilient, and easier to maintain.

### 1.12.4 Amazon API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. With a few clicks in the AWS Management Console, you can create an API that acts as a “front door” for applications to access data, business logic, or functionality from your back-end services, such as workloads running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS Lambda, or any web application. Since it can use AWS Lambda, you can run your APIs without servers.

Amazon API Gateway creates a unified API frontend for multiple microservices. It provides DDoS protection and throttling for your backend. It allows you to authenticate and authorize requests to a backend. It can throttle, meter and monetize API usage by third-party developers.

Amazon API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management. Amazon API Gateway has no minimum fees or startup costs. You pay only for the API calls you receive and the amount of data transferred out.

Amazon API Gateway provides throttling at multiple levels including global and by service call. Throttling limits can be set for standard rates and bursts. For example, API owners can set a rate limit of 1,000 requests per second for a specific method in their REST APIs, and also configure Amazon API Gateway to handle a burst of 2,000 requests per second for a few seconds. Amazon API Gateway tracks the number of requests per second. Any request over the limit will receive a 429 HTTP response. The client SDKs generated by Amazon API Gateway retry calls automatically when met with this response.

You can add caching to API calls by provisioning an Amazon API Gateway cache and specifying its size in gigabytes. The cache is provisioned for a specific stage of your APIs. This improves performance and reduces the traffic sent to your back end. Cache settings allow you to control the way the cache key is built and the time-to-live (TTL) of the data stored for each method. Amazon API Gateway also exposes management APIs that help you invalidate the cache for each stage.

All of the APIs created with Amazon API Gateway expose HTTPS endpoints only. Amazon API Gateway does not support unencrypted (HTTP) endpoints. By default, Amazon API Gateway assigns an internal domain to the API that automatically uses the Amazon API Gateway certificate. When configuring your APIs to run under a custom domain name, you can provide your own certificate for the domain.

Building APIs with Amazon API Gateway

### 1.12.5 AWS X-Ray

You can use AWS X-Ray to trace and analyze user requests as they travel through your Amazon API Gateway APIs to the underlying services. API Gateway supports AWS X-Ray tracing for all API Gateway endpoint types: regional,

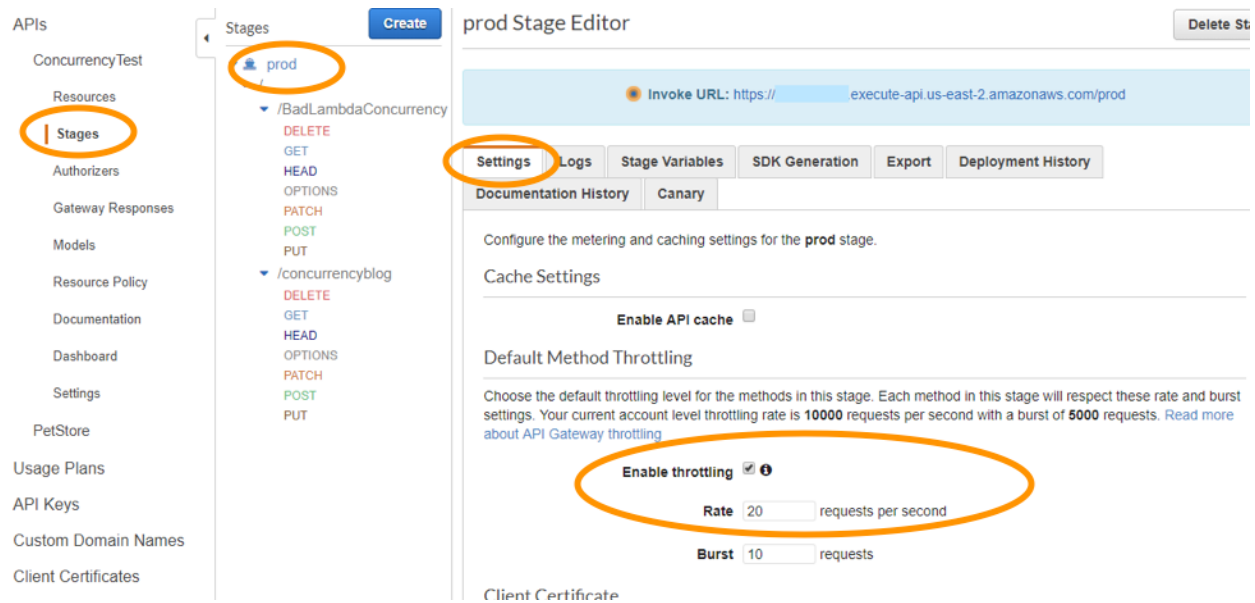


Fig. 80: Amazon API Gateway settings

edge-optimized, and private. You can use AWS X-Ray with Amazon API Gateway in all regions where X-Ray is available.

X-Ray gives you an end-to-end view of an entire request, so you can analyze latencies in your APIs and their backend services. You can use an X-Ray service map to view the latency of an entire request and that of the downstream services that are integrated with X-Ray. And you can configure sampling rules to tell X-Ray which requests to record, at what sampling rates, according to criteria that you specify. If you call an API Gateway API from a service that's already being traced, API Gateway passes the trace through, even if X-Ray tracing is not enabled on the API.

You can enable X-Ray for an API stage by using the API Gateway management console, or by using the API Gateway API or CLI.

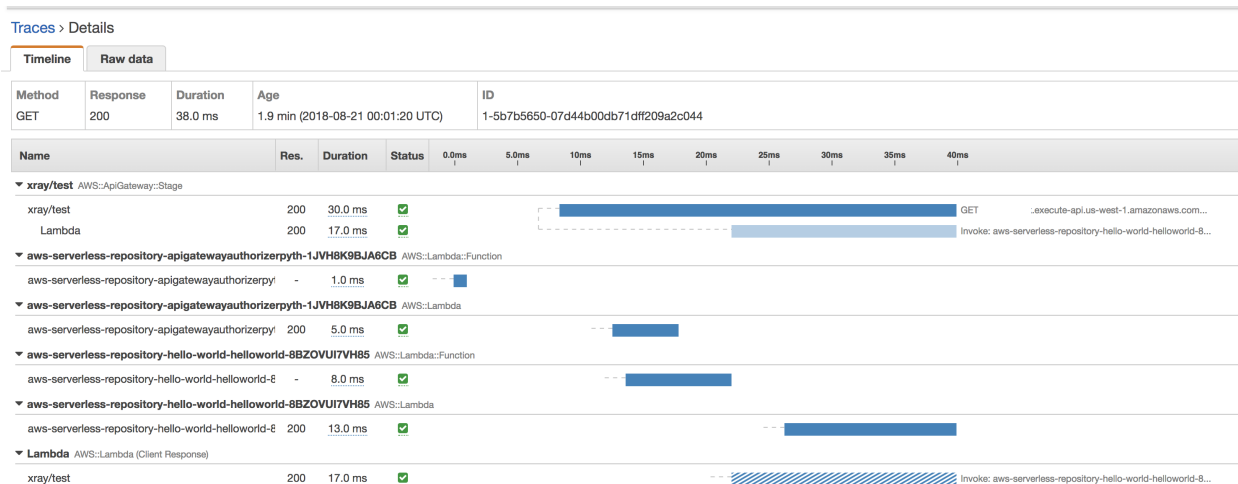


Fig. 81: Amazon API Gateway trace view with AWS X-Ray

### 1.12.6 Amazon ECS

In Amazon EC2 container service, Docker is the only container platform supported by EC2 container service presently.

## 1.13 RTO/RPO and backup recovery setup

### 1.13.1 Disaster planning

Backup and Recovery Approaches Using AWS

### 1.13.2 Recovery strategies

#### Pilot Light

The term pilot light is often used to describe a DR scenario in which a minimal version of an environment is always running in the cloud. The idea of the pilot light is an analogy that comes from the gas heater. In a gas heater, a small flame that's always on can quickly ignite the entire furnace to heat up a house. This scenario is similar to a backup-and-restore scenario.

For example, with AWS you can maintain a pilot light by configuring and running the most critical core elements of your system in AWS. When the time comes for recovery, you can rapidly provision a full-scale production environment around the critical core.

AWS re:Invent 2018: Deep Dive: Hybrid Cloud Storage Arch. w/Storage Gateway, ft. CME Grp. (STG305-R)

## 1.14 Optimizations and review

### 1.14.1 AWS Trusted Advisor

AWS Trusted Advisor is an online tool that provides you real time guidance to help you provision your resources following AWS best practices. AWS Trusted Advisor analyzes your AWS environment and provides best practice recommendations in 5 categories:

- Cost Optimization
- Performance
- Security
- Fault Tolerance
- Service Limits

The status of the check is shown by using color coding on the dashboard page:

- Green: no problem detected.
- Yellow: investigation recommended.
- Red: action recommended.

For each check, you can review a detailed description of the recommended best practice, a set of alert criteria, guidelines for action, and a list of useful resources on the topic.

Architecting for the Cloud



### 1.14.2 AWS Budgets

AWS Budgets gives you the ability to set custom budgets that alert you when your costs or usage exceed (or are forecasted to exceed) your budgeted amount.

Budgets can be tracked at the monthly, quarterly, or yearly level, and you can customize the start and end dates. You can further refine your budget to track costs associated with multiple dimensions, such as AWS service, linked account, tag, and others. Budget alerts can be sent via email and/or Amazon Simple Notification Service (SNS) topic.

You can also use AWS Budgets to set a custom reservation utilization target and receive alerts when your utilization drops below the threshold you define. RI utilization alerts support Amazon EC2, Amazon RDS, Amazon Redshift, and Amazon ElastiCache reservations.

Budgets can be created and tracked from the AWS Budgets dashboard or via the Budgets API.

### 1.14.3 AWS Config

AWS Config is a service that enables customers to assess, audit, and evaluate the configurations of AWS resources. AWS Config continuously monitors and records AWS resource configurations and allows customers to automate the evaluation of recorded configurations against desired configurations and notify you if something is not compliant.

With Config, customers can review changes in configurations and relationships between AWS resources, dive into detailed resource configuration histories, and determine the overall compliance against the configurations specified in their internal or best-practices guidelines. This enables customers to simplify compliance auditing, security analysis, change management, and operational troubleshooting.

In addition, AWS Config can compare configuration changes against best-practices configuration rules, which are available from AWS. Any deviations can generate notification or automated remediation events that trigger actions using services like AWS Lambda. Config can also be used to track resource inventory of the environment.

When you set up AWS Config, you complete the following:

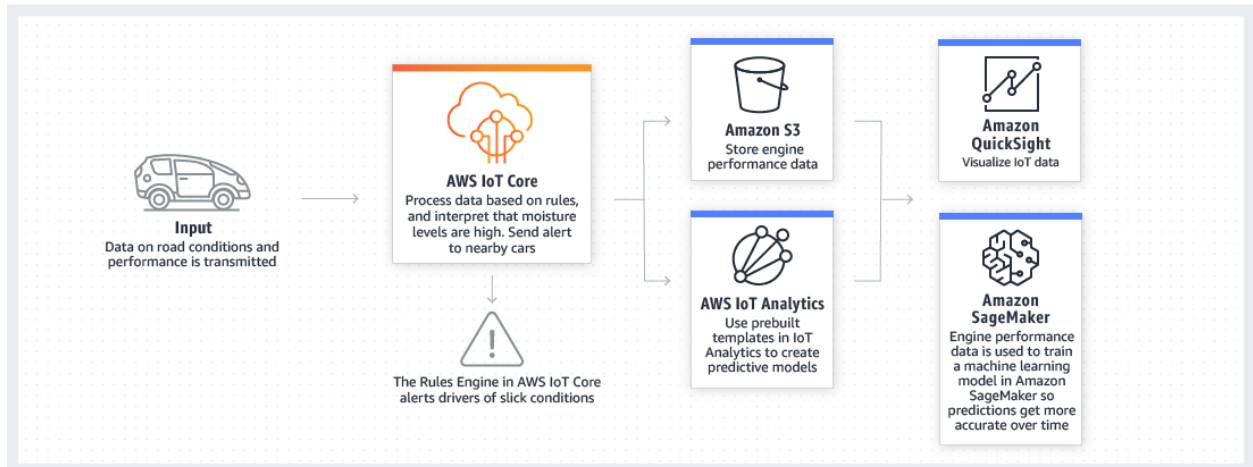
- Specify the resource types that you want AWS Config to record.
- Set up an S3 bucket to receive a configuration snapshot on request and configuration history.
- Set up an SNS topic to send configuration stream notifications.
- Grant AWS Config the permissions it needs to access S3 bucket and the SNS topic.
- Specify the rules that you want AWS Config to use to evaluate compliance information for the recorded resource types.

## 1.15 Appendix

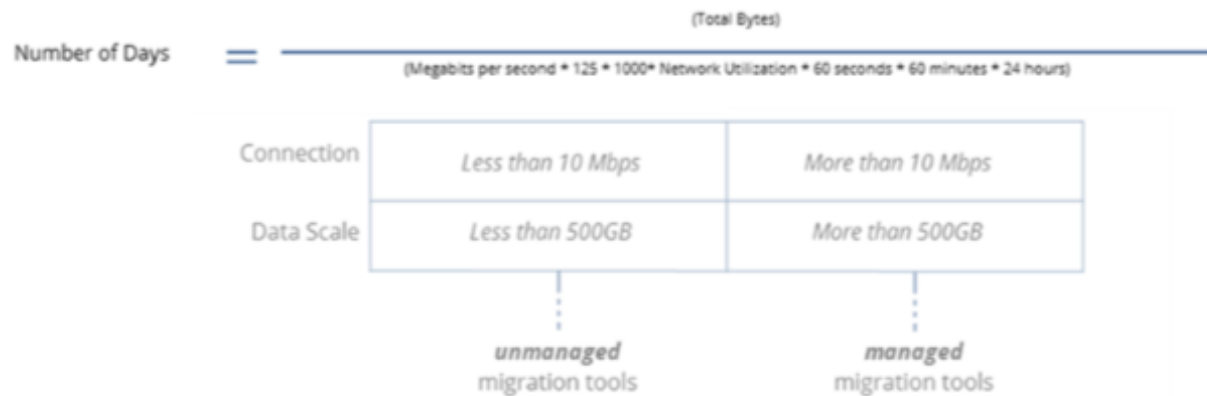
### 1.15.1 AWS IoT Core

AWS IoT Core is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT Core provides secure communication and data processing across different kinds of connected devices and locations so you can easily build IoT applications.





### Minimize risk while moving data quickly



Migration Tools	
Unmanaged	Managed
<ul style="list-style-type: none"> <li>Amazon S3 Command Line Interface</li> <li>Amazon Glacier Command Line Interface</li> </ul>	<ul style="list-style-type: none"> <li>AWS Direct Connect</li> <li>AWS Snowball</li> <li>Amazon S3 Transfer Acceleration</li> <li>AWS Storage Gateway</li> <li>Amazon Kinesis Firehose</li> <li>Technology Partnerships</li> </ul>

## 1.15.2 Cloud Migration

## 1.15.3 Cloud Economics

The cloud economics offering support AWS APN Partners in multiple areas, including **business value** and **cloud financial management**. For business value, the cloud value framework consists of 4 pillars:

1. Cost savings (typical focus)
2. Staff productivity
3. Operational resilience
4. Business agility

The last 3 being the most compelling cloud benefits.

The cloud financial management offering is to make sure customers reap the cost benefits associated with running their workloads on AWS.

### Cost savings

See section *AWS pricing*

### Staff productivity

As enterprises move to AWS, a few common patterns emerge at the IT staff level. Tactical, undifferentiated work previously required for traditional data centers, like provisioning resources, moves from manual to automated. This saves staff time and reduces time to market. This allows customers' resources to move to more strategic work.

As AWS maturity increases, customers learn how to further improve their businesses with AWS. They adopt new services and technologies, which can result in additional cost reductions and accelerated time to market.

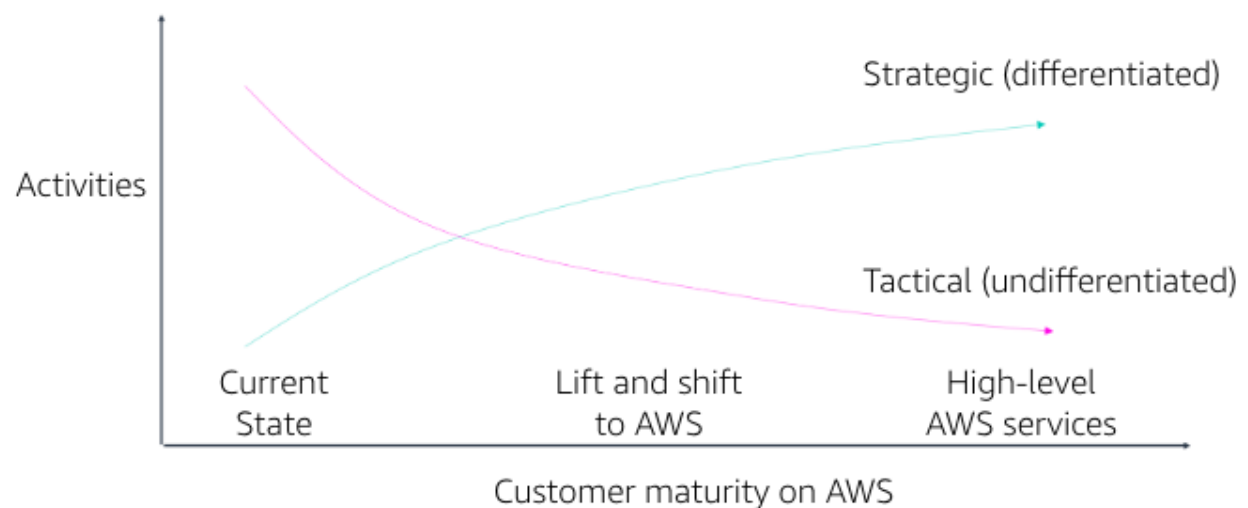


Fig. 82: AWS maturity versus activities

IT team members who used to work on projects like storage array deployments and server refreshes can transition to become DevOps specialists. By being integrated into the dev team, they can support the development of new products and services.

Task	Typical Reduction	Description	Calculating server benefits
Server budgeting and planning	Up to 90%	No capital server budget or plan in the AWS Cloud	
Server purchasing process	Up to 75%	Instance purchasing requires minimal effort in comparison to server purchasing	
Long-term capacity planning	Up to 75%	Capacity planning is simply a matter of initiating new instances based on thresholds, and much of this can be automated	
Project budgeting and planning	Up to 75%	Project budgeting and planning effort should be significantly reduced	
Prepare detailed implementation plans	Up to 75%	Implementation plans will be reduced since instance initiation is very straightforward	
Arrange repair for hardware in occasion of hardware failure	Up to 100%	Not necessary with AWS	
Install/upgrade/remove software	Up to 50%	Simplify and automate OS patching / updating	

Fig. 83: Server benefits

Task	Typical Reduction	Description	Calculating network benefits
Forecast network traffic for capacity planning	Up to 50%	Network monitoring simplified, requiring less investment in data collection and planning	
Ensure that the network infrastructure is up and running	Up to 80%	Not necessary with AWS	
Arrange for return of defective parts and warranty service manufacturer or supplies	Up to 100%	Not necessary with AWS	
Install/refresh network hardware	Up to 100%	Not necessary with AWS	
Network maintenance/upgrades, including service packs, patches, and hot fixes	Up to 100%	Not necessary with AWS	

Fig. 84: Network benefits

Task	Typical Reduction	Description	Calculating storage benefits
Installation of hardware	Up to 100%	No physical arrays to implement	
Capacity planning	Up to 50%	Still a required task but far less time required due to AWS options	
Apply storage system updates, patches, and configuration changes	Up to 100%	No longer required—AWS manages system updates	
Arrange repair for hardware in occasion of hardware failure	Up to 100%	No longer required—AWS manages hardware failure	
Update system on new versions	Up to 100%	No longer required—AWS manages system updates	
Regular preventative maintenance (daily, weekly, etc.), boot and shut down systems when needed	Up to 90%	Minimal effort required—AWS manages preventative maintenance	
Install/upgrade/remove software	Up to 90%	Minimal effort required—AWS manages storage software	

Fig. 85: Storage benefits

Task	Typical Reduction	Description	Calculating application benefits
Create new development environment	Up to 80%	Provisioning time for new development on AWS requires clicks	
Monitor security	Up to 50%	Still a required task, but reduced with AWS capabilities	
Test code	Up to 40%	Still a required task, but reduced with AWS capabilities	
Scale out systems	Up to 25%	Still a required task, but reduced with AWS capabilities	
Build security capability and fixes	Up to 25%	Still a required task, but reduced with AWS capabilities	
Test execution	Up to 80%	Still a required task, but reduced with AWS capabilities	

Fig. 86: Application benefits

Task	Typical Reduction	Description	Calculating facilities benefits
Manage the installation of all data center devices and services	Up to 90%	Still a required task, but only for certain infrastructure assets (WAN, firewall, and so forth)	
Facilities project budgeting and planning	Up to 95%	Minimal physical assets to implement (WAN, firewall, and so forth)	
Prepare detailed implementation plans	Up to 95%	Minimal physical assets to implement (WAN, firewall, and so forth)	
Manage power and cooling systems	Up to 100%	No physical assets to manage (CRAC, in-row cooling, utilities, and so forth)	
Manage the installation of all data center devices and services	Up to 95%	Minimal physical assets to implement (WAN, firewall, and so forth)	
Maintain power, space, cooling, devices	Up to 95%	Minimal physical assets to implement (WAN, firewall, and so forth)	
Disaster recovery planning and testing	Up to 80%	Still a required task for failover purposes but greatly simplified without hardware layer	

Fig. 87: Facilities benefits

Task	Typical Reduction	Description	Calculating security benefits
Long term capacity planning	Up to 75%	Still a required task, but only for WAN, internet traffic, and software security	
Set up new security devices	Up to 80%	Still a required task, but no longer required for most physical assets	
Security tool administration of vulnerability scanners	Up to 50%	Reduced effort since AWS provides significant vulnerability protection	
Antivirus tool administration	Up to 100%	Part of AWS security capabilities	
Security audits / make policy recommendations	Up to 40%	Simplified due to AWS shared responsibility model	
Threat, vulnerability / risk assessments	Up to 75%	Reduced effort since AWS provides significant security capability	
Network security policies, application security, access control / corporate data safeguards	Up to 30%	Easier to implement policies, controls, and compliance policy	

Fig. 88: Security benefits

## Operational resilience

Operational resilient IT organizations depend on the health of 4 cornerstones: operations, security, software, and infrastructure.

### Operations failures

Some primary causes of operations failures are:

- Human errors, such as lack of clearly defined procedures or user privilege.
- Configuration errors in hardware or operating system settings and startup scripts.
- Procedural errors, like restoring the wrong backup or forgetting to restart a device.
- Commonplace accidents in the data center, like tripping over power cords, dropping equipment, or disconnecting devices.

AWS leverages automation; manages services from end to end; provides system-wide visibility for usage, performance, and operational metrics; enables security and governance configuration; and monitors API access.

### Security: causes for breaches

The causes for security breaches include:

- Malware, such as worms, viruses, and trojan horses.
- Network attacks, like open ports, SYN floods, and fragmented packets.
- Unpatched applications or operating systems.
- Security issues, such as password disclosures, social engineering, credentials not stored securely, non-strict password policies, and poor privilege and access management.
- Poor or limited authentication.

AWS has a [Shared Responsibility Model](#), which means that AWS shares security responsibilities with customers. In this model, AWS is responsible for the security of everything from the hypervisor level to the operating system.

AWS managed services move the line of responsibility higher.

AWS helps to reduce security risks in numerous ways:

- Leverages AWS automation and tools available to help customers mitigate the most severe security risks, including denial of service attacks.
- Provides the AWS Identity Access Management (IAM), service to centrally manage users and credentials, which helps customers reduce or eliminate the existence of “rogue servers”.
- Leverages our roster of 30 plus compliance certifications and accreditations to help our customers build secure, compliance-ready environment.

### Software: causes for failure

Common causes for software resilience failures include:

- Resource exhaustion, like runaway processes, memory leaks, and file growth.
- Computational or logic errors, such as faulty references, de-allocated memory, corrupt pointers, sync errors, and race conditions.

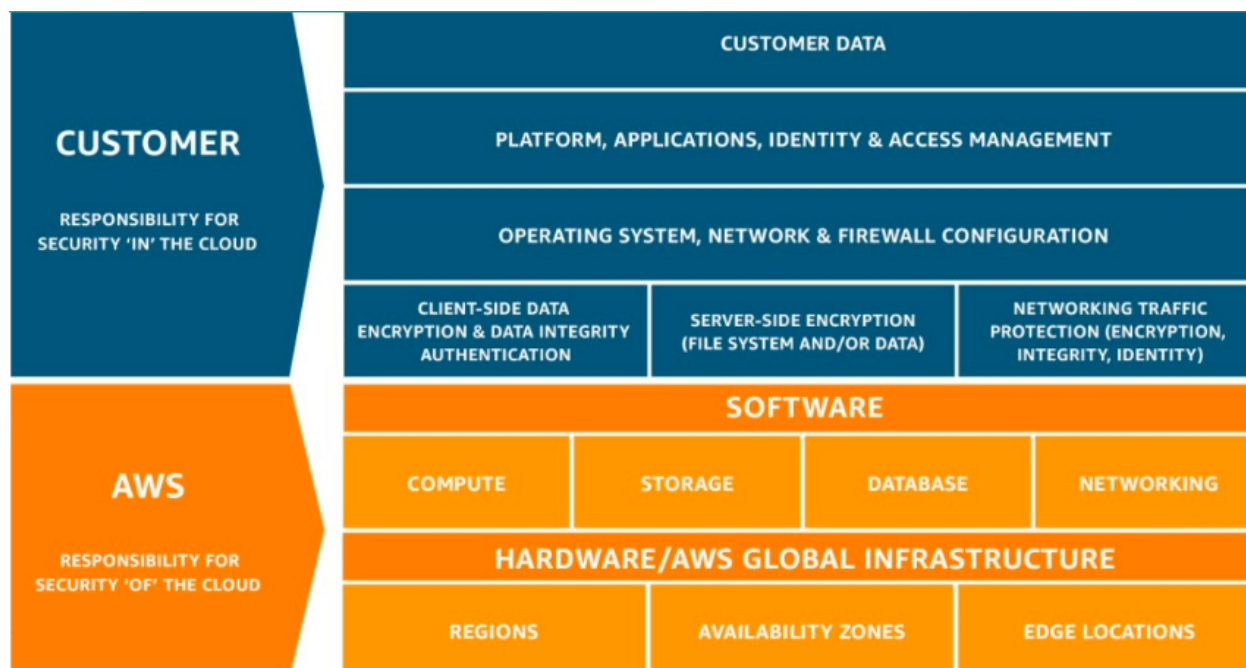


Fig. 89: AWS shared security model

- Inadequate monitoring, such as the inability to identify issues.
- Failed upgrades, such as intercompatibility and integrations.

AWS provides services in a way that allows customers to increase or decrease the resources they need and have AWS manage the changes. To provide software resilience, AWS:

- Offer blue and green deployments that allow for quick rollbacks.
- Automates continuous integration and continuous delivery workflow.
- Runs smaller code deployments to reduce unit, integration, and system bugs.
- Provides current and secure resources with OS patching.
- Creates and manages a collection of related AWS resources.

### Infrastructure: causes for failure

Causes for infrastructure failure include:

- Hardware failure of servers, storage, or networks.
- Natural disaster, like hurricanes, floods, and earthquakes.
- Power outages, including failed power supplies and batteries.
- Volumetric attacks, such as DDoS, DNS amplification; or UDP/ICMP floods.

AWS helps reduce infrastructure failures in numerous ways:

- AWS continues to expand their world-class infrastructure and leads the industry in improving data centers on a massive scale.
- Customers can run applications and failover across multiple Availability Zones and Regions.

- AWS systems are designed to be highly available and durable. S3 is designed to provide eleven 9s of durability and four 9s of availability. Amazon Ec2 is designed for four 9s of availability, and Amazon EBS volumes are designed for five 9s of availability.
- As a standard, each AWS Availability Zone in each Region is redundantly connected to multiple tier-one transit providers.
- At AWS every compute instance is served by two independent power sources, each with utility, UPS, and back-up generator power.

## Business agility

Here, you can see a list of KPIs for measuring business agility:

Key Performance Indicators (KPIs)	
New applications launched per year	Mean time to resolution (MTTR) (hours)
Time to market for new applications	Response time to defects (hours)
Time to provision new environments (days)	Customer retention (%)
Deployment frequency (revolutions/year)	Adoption of new features (%)
Time to deploy to production (weeks)	"Value" per release (\$ revenue potential)
Time to deploy to test (days)	Employee retention (%)
Features per release	Employee absenteeism (%)
Total number of incidents/defects	Employee Net Performance Score/satisfaction
Percentage of total defects found in test	Customer Net Performance Score/satisfaction

## Time to market for new applications

Some of the most important activities that a healthy business must do to continue to grow and innovate are to scope, prioritize, and take on new initiatives. You can think about the initiative process like a project funnel.

## Code throughput and systems stability

DevOps practices help customers deliver software faster, more reliably, and with fewer errors. Two key DEvOps-related IT performance dimensions are code throughput and systems stability.

Lead time for changes and deployment frequency correspond to code throughput. Throughput is measured by how frequently a team is able to deploy code and how quickly it can move from committing code to deploying it.

Change failure rate and Mean Time to Recover (MTTR) correspond to systems stability. Stability is measured by how quickly a system can recover from downtime and how many changes succeed versus how many fail.

## Cloud financial management

Cloud financial management includes four key areas.



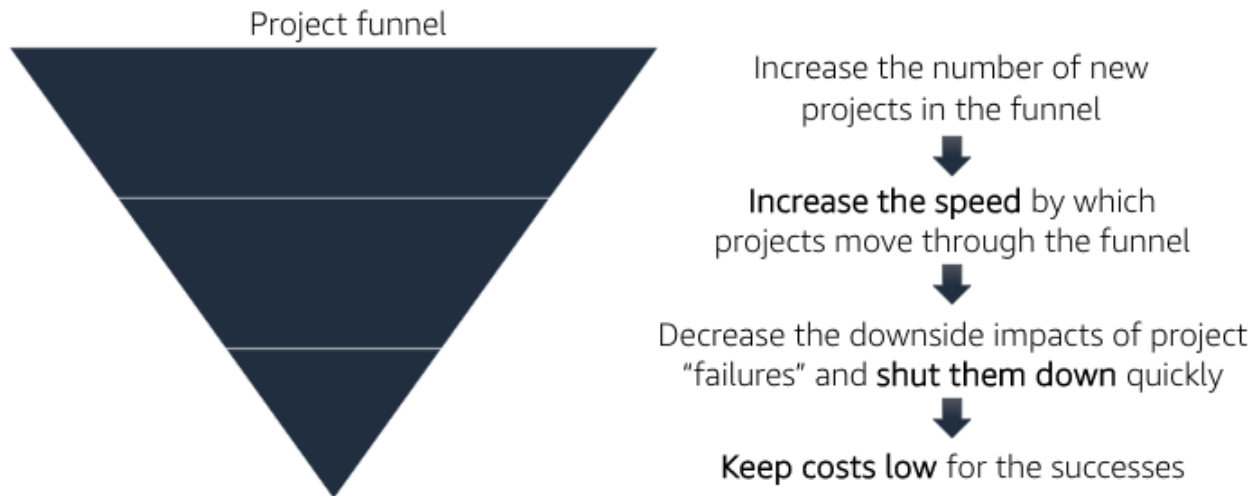
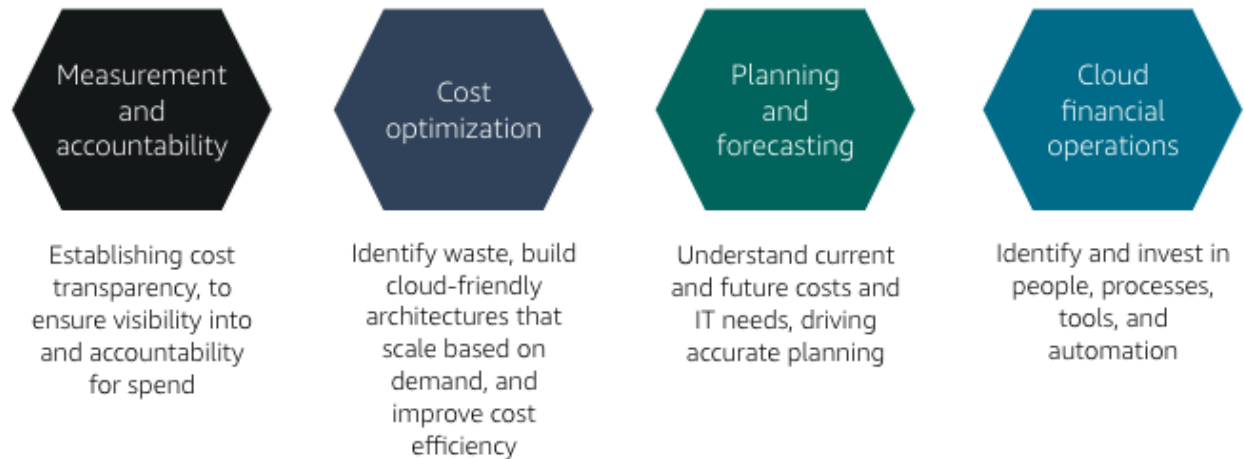


Fig. 90: Innovate by increasing “fail fast” while reducing risks and costs



To enable cost transparency you must have the right tagging scheme and apply it to all areas of spending. User-defined tags allow customers to label their resources so they can manage them. At a minimum, from cost perspective, customers should use the following 5 tags:

- What cost center does it belong to? This may belong to more than one.
- What application or workload does it support?
- Who owns it?
- What is the expiration date? When should it be turned off? This helps with Reserved Instance purchasing.
- Automation tags can state directions such as “shut me down on the weekend” for and non-production environment, or “This instance runs non-critical workloads and can be freed up for disaster recovery in case of a malfunction on a different Availability Zone.”

An ideal tool for measuring and monitoring should provide:

- Cost and usage data.
- Optimization recommendations.
- Other information that helps teams make data-driven, cost-based decisions.

AWS Cost Explorer is a free tool that helps customers dive deeper into cost and usage data to identify trends, pinpoint cost drivers, and detect anomalies.

AWS has identified 4 key pillars of cost optimization best practices:

### **Right-sizing instances**

This means selecting the least expensive instance available that meets the functional and performance requirements. Right-sizing is the process of reviewing deployed resources and seeking opportunities to downsize when possible. For example, if only CPU and RAM are underused, a customer can switch to a smaller size instance.

AWS Cost Explorer generates EC2 instance rightsizing recommendations by scanning your past usage over the previous 14 days. From there, AWS removes Spot Instance usage and any instances it believes that you terminated. AWS then analyzes the remaining instance usage to identify idle and underutilized instances:

- Idle instances are instances that have lower than 1% maximum CPU utilization.
- Underutilized instances are instances with maximum CPU utilization between 1% and 40%.

When AWS Cost Explorer identifies an idle instance, it will generate a termination recommendation. When it identifies an underutilized instance, AWS simulates covering that usage with a smaller instance within the same family.

### **Increasing application elasticity**

#### **Turn off non-production instances**

In regards to increasing application elasticity for cost optimization, reviewing production versus non-production instances is key. Instances that are non-production, such as dev, test, and QA, may not need to run during nonworking hours, such as nights and weekends. In these cases, these servers can be shut down, and will stop incurring charges if they are not Reserved Instances. Typically when a nonproduction instance has a usage percentage less than or equal to 36%, it is less expensive to use On-Demand pricing versus Reserved Instances.

A customer can create an AWS Lambda function that can automate the starting and stopping of instances based on parameters like idling and time of day:

[How do I stop and start Amazon EC2 instances at regular intervals using Lambda?](#)

### **Automatic scaling**

Automatic scaling helps to ensure the correct number of instances are available to handle the workload of an application. Both the minimum and maximum number of instances can be specified, and automatic scaling ensures that you never go below or above the thresholds.

This provides the customer the opportunity to provision and pay for a baseline and then automatically scale for peak when demand spikes, which lowers costs with no performance impact.

Automatic scaling can be scheduled based on predefined times or performance. Recently, with the introduction of predictive scaling for EC2, AWS will use data collected from your actual EC2 usage and billions of data points drawn from AWS' own observations, in combination with well-trained machine learning models to predict expected traffic (and EC2 usage) including daily and weekly patterns. The prediction process produces a scaling plan that can drive one or more groups of auto scaled EC2 instances. This allows you to automate the process of proactively scaling, ahead of daily and weekly peaks, improving overall user experience for your site or business, and avoid over-provisioning, which will reduce your EC2 costs.

## Choosing the right pricing model

See section *Pricing options*

## Optimizing storage

See section *Object Storage Classes*

# 1.16 Domains

## 1.16.1 Domain 1: Design Resilient Architectures

### 1.1 Choose reliable/resilient storage

The reason you want to make sure your storage is resilient is that this way, even in the event of a disaster, you don't lose data or state.

## Storage options

### EC2 Instance Store

Amazon EC2 Instance Store

This lives on your physical HW where your EC2 compute instance run.

- They are ephemeral volumes. When the instance terminates or stops, they are lost.
- Only certain EC2 instances types have instance stores.
- The size is fixed since this store is on the physical host.
- The disk type whether it is SSD or HDD is also fixed based on the instance type and so it is the capacity. Both of these are a function of the instance type.
- Application-level durability. While your application is running you can rely on the instance store but since it is ephemeral you cannot on much beyond that.

Generally, you want to use the instance store for caching or for storing other temporary data that you are replicating somewhere else. This way you can get the fastest access that the instance store provide and at the same time you are not affected by the ephemeral of the instance store.

### Elastic Block Store

This an attachable storage that connects to one EC2 instance at a time.

- Different types
- It supports encryption and snapshots.
- Some EBS volume types support provisioned IOPS, meaning that you can configure the volume for a larger or smaller number of read and writes per second.
- The EBS volume persists beyond the lifecycle than an EC2 instance. You can stop or even terminate a EC2 instance and still preserve the EBS volume.

- You can attach multiple volumes to a single instance. However, you can only attach one instance to a EBS volume at a time. When you have multiple EBS volumes attached to your instance, you can use RAID 0 and striping to achieve higher throughput and IOPS.

Think of EBS volumes as durable, attachable storage for your EC2 instances. These are the 4 types of EBS volumes.

Volume Type	Solid-State Drives (SSD)		Hard disk Drives (HDD)	
	General Purpose SSD (gp2)	Provisioned IOPS SSD (io1)	Throughput Optimized HDD (st1)	Cold HDD (sc1)
Use Cases	<ul style="list-style-type: none"> <li>• Recommended for most workloads</li> <li>• System boot volumes</li> <li>• Virtual desktops</li> <li>• Low-latency interactive apps</li> <li>• Dev and test environments</li> </ul>	<ul style="list-style-type: none"> <li>• Critical business applications that require sustained IOPS performance, or more than 10,000 IOPS or 160 MiB/s of throughput per volume</li> <li>• Large database workloads</li> </ul>	<ul style="list-style-type: none"> <li>• Streaming workloads requiring consistent, fast throughput at a low price</li> <li>• Big data</li> <li>• Data warehouses</li> <li>• Log processing</li> <li>• Cannot be a boot volume</li> </ul>	<ul style="list-style-type: none"> <li>• Throughput-oriented storage for large volumes of data that is infrequently accessed</li> <li>• Scenarios where the lowest storage cost is important</li> <li>• Cannot be a boot volume</li> </ul>
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max. IOPS/Volume	10,000	32,000	500	250
Max. Throughput/Volume	160 MiB/s	500 MiB/s	500 MiB/s	250 MiB/s
Dominant Performance Attribute	IOPS	IOPS	MiB/s	MiB/s

There are 2 primary types of EBS volumes: SSD and HDD and each one has 2 subtypes. SSD volumes have much better IOPS, which is read/write operations per second. This is because SSD are Solid State disks with no moving parts. HDD or Hard Disk Drives have good throughput and gives you lower IOPS.

SSD is good for random access while HDD are great for sequential access. HDD are also cheaper than SSD. HDD are good for sequential access because with this type of access you are going to have large blocks of data to read in and fewer read/write operations. For example, if you are processing log files or Big Data workloads made up of sequences of records, HDD will give great performance at a lower price.

With SSD and HDD, there are 2 subtypes of EBS volumes. General Purpose SSD (gp2) is the cheaper SSD volume, Provisioned IOPS SSD (io1) is more expensive. Provisioned IOPS gives you the ability to scale up the number of reads/writes and gives you a better performance at a higher price. Similarly, for Throughput Optimized HDD (st1) is more expensive and has better specs than Cold HDD (sc1).

## Amazon EFS

### Introduction to Amazon Elastic File System (EFS)

This is similar to EBS because you mount it as a disk in your EC2 instances.

- It gives you shared file storage in the AWS Cloud. Multiple EC2 instances can access the same EFS volume.
- It gives you petabyte-scale file system
- The capacity is elastic, it scales up and down as you grow or shrink your footprint.
- It supports NFS v4.0 and 4.1 (NFSv4) protocol.
- It is compatible with Linux-based AMIs for Amazon EC2. It is not currently supported for Windows.

You create an EFS volume and then create mount points for it in a particular VPC. The EFS volumen can only attach to a single VPC at a time. Within this VPC, you get mount targets that your EC2 instances can then connect to.

## Amazon S3

For object storage, the primary option is S3.

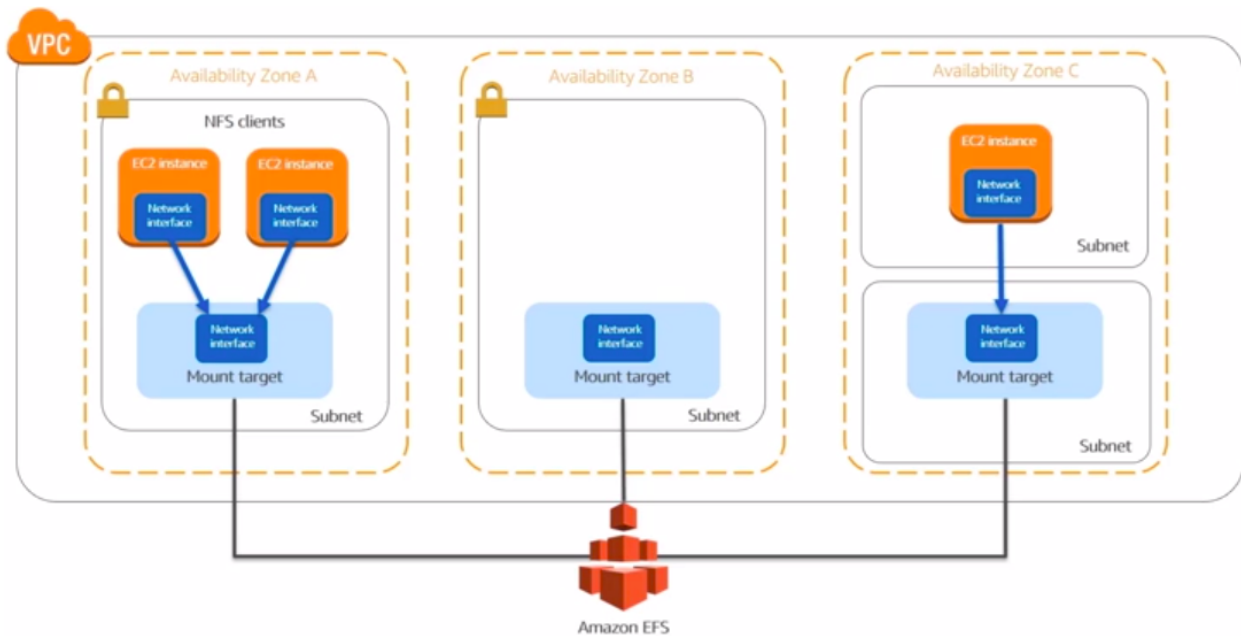


Fig. 91: Amazon EFS architecture

- The consistency model. To give you limited storage S3 is implemented as a distributed system. With distributed systems the consistency model can be strongly consistent or eventually consistent. S3 gives you strong consistency for new objects and it gives you eventual consistency for updates. Eventual consistency means that if you update an object and you do a read/write afterwards, you may get the previous version of the object. This is because it is a scalable distributed system.
- It gives you several storage classes: S3 Standard, S3 Standard-IA. The storage classes offer a tradeoff:
  - Standard is cheaper for access, meaning uploads and downloads.
  - Standard-IA is cheaper for storage, but more expensive for access.
- It offers several options for Server Side Encryption:
  - SSE-S3: S3 manages master key used to encrypt files on the server side.
  - SSE-KMS: KMS manages master key used to encrypt files on the server side.
  - SSE-C: Customer provides and uses master key used to encrypt files on the server side.
- It enables encryption of data in transit by supporting uploads and downloads using HTTPS.
- It allows you enabling version on a bucket. Previous versions of files remain available even if the file is updated or deleted. This is a great way to ensure that your files are protected from accidental deletes or overwrites.
- You can control S3 access through IAM user policies, through bucket policies or through ACLs.
- It supports multi-part uploads, where you upload a large file in parallel parts.
- It has an Internet-accessible API.
- It has virtually unlimited capacity.
- It is region scoped.
- It is designed for 11 nines of durability.

## Amazon Glacier

Suppose you have data that you want to hold on for a long period of time. You want to keep them on cold storage so that it always there, but you are not planning to access it very often. This is where Glacier comes in.

- It is a solution for data backup and archive storage.
- It provides archives which correspond to files and vaults which are a collection of archives.
- It has 3 retrieval types: expedited, standard, and bulk. They offer different tradeoffs between latency and cost:
  - Bulk is the cheapest but take longer and can take up to 12 hours at this time.
  - Expedited is more expensive but faster and can take up to 5 minutes at this time.
- It encrypts data by default.
- A great way to use Glacier is to set up S3 lifecycle policies on your buckets. This lifecycle policies can move data into Glacier automatically after a period of time. This is great for archiving old data.
- It is region scoped.
- It is designed for 11 nines of durability.

## 1.2 Determine how to design decoupling mechanisms using AWS services

### High Availability

Decoupling ensures that if one tier or component fails, the others are not impacted because they are decoupled.



Fig. 92: Example of tightly-coupled system

This is a tightly-coupled system: The web server takes requests and then sends an email over the email service. When the email service goes down, the web server is forced to become unoperational. The email service going down can be a perfectly normal event: maybe you bring down the email server to upgrade it to a new version of the email service, or to upgrade the HW or maybe it can be a failure of the email service. Regardless, the impact of failures is large in tightly coupled systems.



Fig. 93: Example of decoupled system

This system has an SQS for holding messages for the email service. The Web server has not to wait for the email service to do its jobs, it queues the work and moves on. This kind of asynchronous interaction is much more resource

efficient of the web server side. Furthermore, if the email service goes down, the web server is not impacted, it can continue to function. Not only that: none of the messages are lost, the messages are safely persisted in the SQS queue and it will wait for the email service to come back again. Once the service comes back online, the messages can be processed.

### Scalability

Another benefit from decoupling is scalability. In this system, a web server is calling a logging service. If the web server sends a high volume of requests to the logging service, it can overwhelm it and caused it to become overloaded.

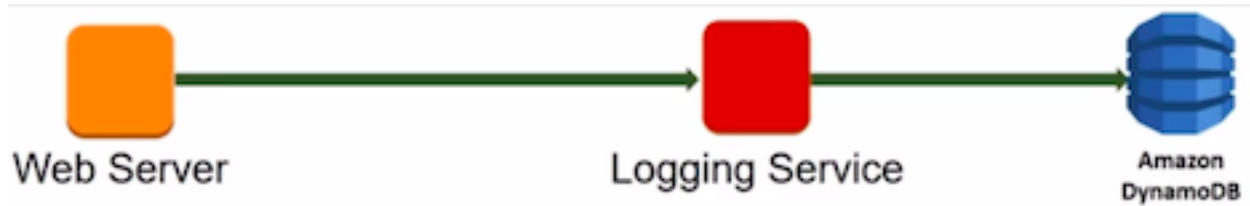


Fig. 94: Example of a overloaded system

We can solve this problem by decoupling the web server from the logging server using SQS queue. The logging service can scale out when the volume of requests goes up. When the volume of requests goes down, the logging service can scale down. The scale down is important to keep our costs down when we are not using the extra servers.

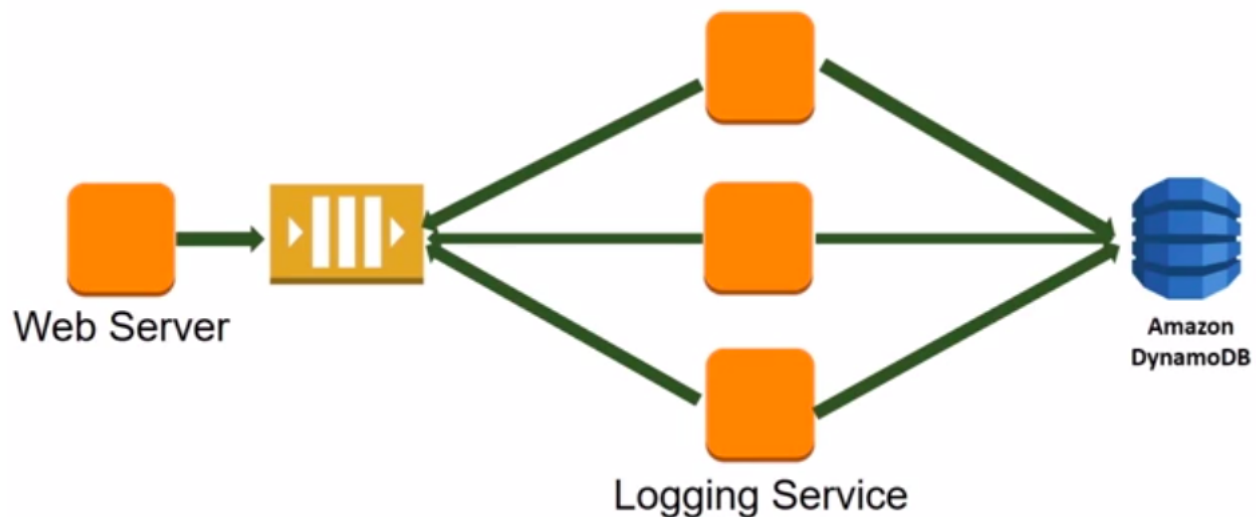


Fig. 95: Example of a decoupled system preventing from overloading

We can use a load balancer for decoupling. It distributes the incoming requests across the servers running the logging services. The load balancer is useful when the backend service return response is important. If there is no response required by the caller (in this case the web server), a queue can be used instead.

### Identity of components

Suppose we have an external client making requests to web server inside a VPC. The request and the response goes over the public Internet. If the web server goes offline, the client will have an error. If we have some automation implemented so that we replace the web server with another server which is based on the same image. The problem is



that the new server has a new IP address assigned by the VPC. This means that the client is not able to get up to the IP is propagated to the client through DNS.



Fig. 96: System with the new web server with a different IP

One way to solve this problem is to use an elastic IP address. If the server goes offline, the new instance will use the same Elastic IP address. An Elastic IP address is an IP address that can move from one instance to another. We have decoupled the identity of the server.

### 1.3 Determine how to design a multi-tier architecture solution

Multi-tier architectures are naturally decoupled. Each tier can scale independently. The entire application may not need to scale when scaling event occurs.

### 1.4 Determine how to design high availability and/or fault tolerant architectures

When you are operating at scale, in case of an event of failure or an unusual or exceptional event, you want to be treated as a normal operational event. You want your application to stay highly available in case of a fault and to continue to keep providing value and service to your users.

#### Fault tolerance

The more loose your system is coupled, the more easily it scales and the more fault-tolerant it can be.



Fig. 97: Tightly-coupled systems versus Loosely-coupled systems



## Test axioms

- Expect “Single AZ” will never be a right answer.
- Using AWS managed services should always be preferred.
- Fault tolerant and HA are not the same thing.
- Expect that everything will fail at some point and design accordingly

## 1.16.2 Domain 2: Define Performant Architectures

### 2.1 Choose performant storage and databases

#### Elastic Block Store

The EBS volume types offer different tradeoff in performance. SSD are more expensive and offer more perform better in IOPS, meaning random reads and writes. HDD are cheaper and perform better for sequential reads and writes.

	Solid-State Drives (SSD)		Hard Disk Drives (HDD)	
	General Purpose	Provisioned IOPS	Throughput-Optimized	Cold
Max volume size	16 TiB	16 TiB	16 TiB	16 TiB
Max IOPS/volume	10,000	32,000	500	250
Max throughput/volume	160 MiB/s	500 MiB/s	500 MiB/s	250 MiB/s

Fig. 98: EBS volume types performance

Another key to highly performant architectures is to offload all static content to S3 instead of keeping them on the servers. Instead of folders from your webserver for your html, javascript, CSS, images, video files, and any other static content, you want to move all of them to S3. This dramatically improves your webserver performance. It takes load out of the webserver and frees CPU and memory for serving dynamic content.

#### Amazon S3 buckets

To upload your data to S3 you need:

1. Create a bucket in one of the AWS Regions. The bucket name becomes the subdomain of the S3 object URL.
2. Upload any number of objects to the bucket.

S3 objects automatically get a URL assigned to them that is based on the bucket name. You can have a virtual host-based URL or a path-based URL. The virtual host-based URL contains the bucket name as part of the domain name:

Listing 5: Virtual host-based URL

```
http://[bucket name].s3.amazonaws.com
http://[bucket name].s3-aws-region.amazonaws.com
```

The path-based URL has the bucket name as the bucket as the first part of the path:

**Listing 6: Path-based URL**

```
http://s3-aws-region.amazonaws.com/[bucket name]/[key]
```

You can use an URL with the region name or without it, and it will redirect to the region that corresponds. Buckets are always tight to a region, even though you can reference them without the region name. Buckets name are globally unique. If a call a bucket “mybucket”, no other bucket in any region can have the same name. The full path after the bucket name is called the key.

**Lifecycle policies**

A common pattern with data files is that initially they are hot, there is a lot of interest in processing them, but over time they are not accessed that often, finally there are rarely accessed. However, you need to hold them for some time. This is where lifecycle policies come in. They can move cold data to cheaper storage and the hot data to the more accessible storage.

**Performant storage on databases**

We have several options:

- Relational Database with **Amazon RDS**. You should use RDS when:
  - You are using joins, complex transactions or complex queries.
  - You have a medium-to-high query/write rate.
  - Do not scale more than a single worker node/shard.
  - High durability.

You should not use RDS (but use DynamoDB) when:

- Massive read/write rates (e.g. 150K writes/second).
- Sharding.
- You use simple GET/PUT requests and queries.
- You need a engine that is not provided by AWS or you want a customized RDBMS.

The RDS master DB can be scaled up by using a bigger instance type for it. The other way to scale RDS is by using read replicas. Read replicas are supported with Aurora, PostgreSQL, MySQL and MariaDB. Using read replicas you can offloads the read requests to the read replicas and take some of the load out of the master DB.

- Managed NoSQL database with **Amazon DynamoDB**, adequate for access patterns that match the key-value paradigm. You do not specify how much space you need when you create a DynamoDB table. The DynamoDB grows as your data footprint changes. You do specify the throughput (how many reads/writes) you need and DynamoDB scales to allocate resources based on throughput capacity requirements (read/write). The throughput is specified in read capacity units (rcus) and write capacity units (wcus).
  - Read capacity unit (1 read per second for an item up to 4 KB in size). Each rcu gives you:
    - \* One strongly consistent read per second, meaning that you read something and read it back and you are guaranteed to have the latest value.
    - \* If you willing to relax the strong consistency requirement. You can get two eventually consistent reads per second.
  - Write capacity unit (1 write for second for an item up to 1 KB in size).



Fig. 99: RDS read replicas

- **Data Warehouse with Amazon Redshift.** Redshift gives you a SQL interface, it is useful if you analytic queries instead of transactional queries. You are not inserting or getting a single row rather you want to compute aggregate numbers across an entire table.

## 2.2 Apply caching to improve performance

### Amazon CloudFront

Caching can improve the performance of your application without requiring to redesigning or rewriting the core logic or algorithms. You can cache the data from your application at different levels. You can cache at the web level using a CDN such as CloudFront.

### Amazon ElastiCache

You can apply caching at the application and database levels. You can use ElastiCache to cache what you otherwise you repeatedly fetch from the DB backend. By using a cache with your DB backend, you can take some of your load out of your databases. Moreover, you can improve the RTT of your queries because you are heading the cache instead of the DB for many of them.

ElastiCache gives you 2 different types of caches:

- **Memcached.** It is simpler and easier to setup. It has the following features:
  - Multithreading.
  - Low maintenance.
  - Easy horizontal scalability and auto discovery.
- **Redis.** It is more sophisticated and gives you support for data types. It is more than a simple key-value store.

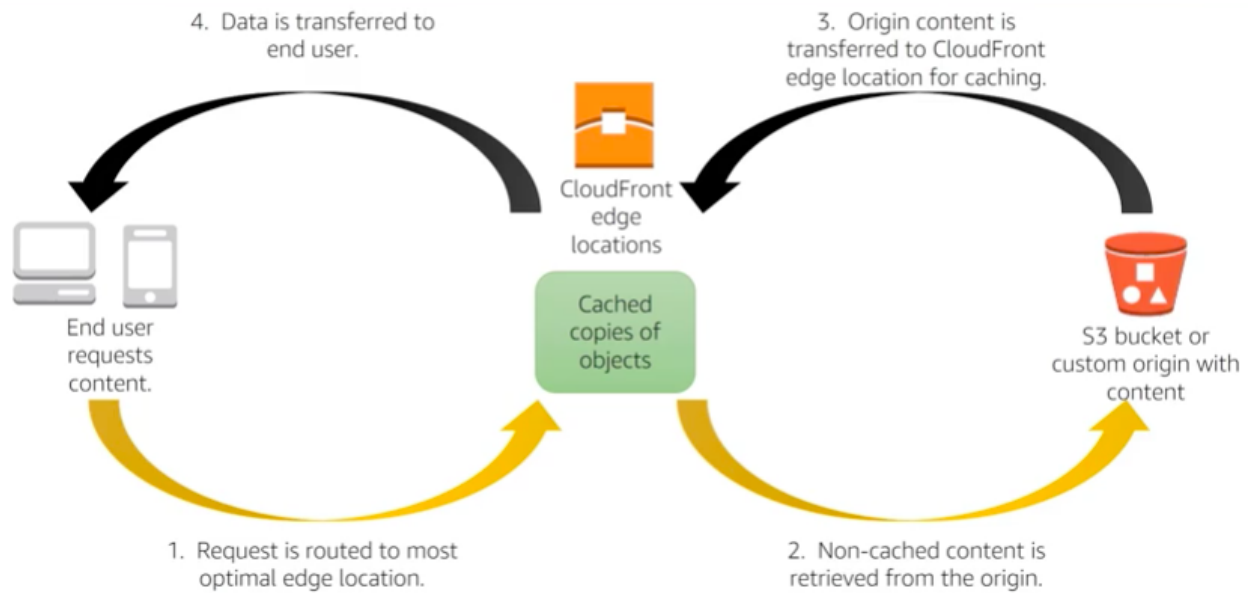


Fig. 100: Caching in CloudFront

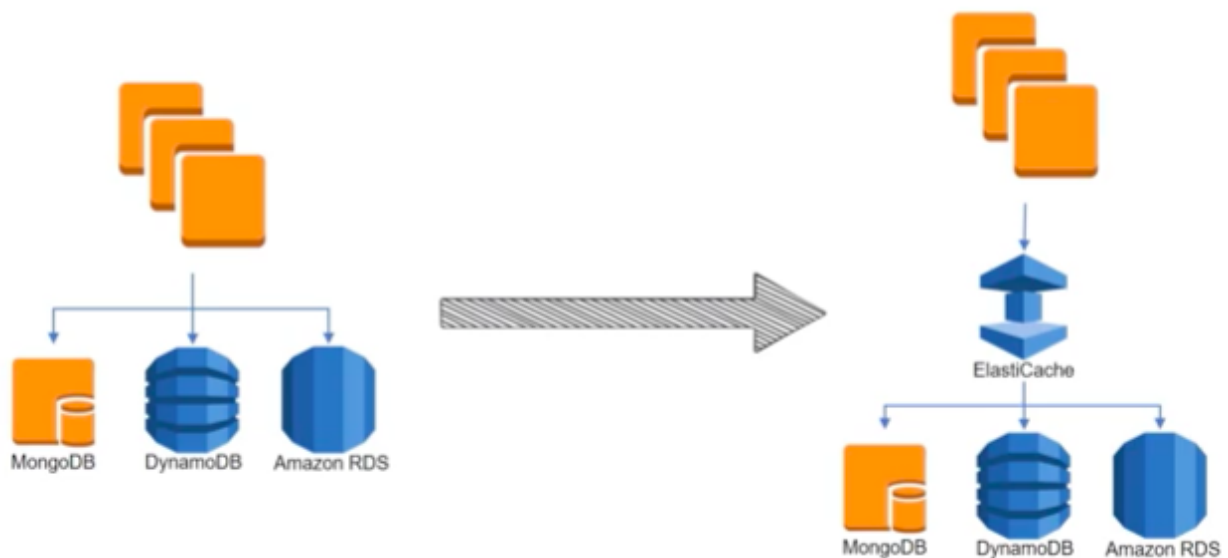


Fig. 101: RDS read replicas

- Support for data structures.
- Persistence.
- Atomic operations.
- Pub/sub messaging.
- Read replicas/failover.
- Cluster mode/sharded clusters.

## 2.3 Design solutions for elasticity and scalability

To make your applications perform well regardless of how much traffic they receive, we want to be able to scale them. We want to grow them in response to traffic spikes, this improves performance. At the same time, we want to shrink when the traffic subsides, this helps to keep our costs low. There are 2 types of scaling for your applications:

- **Vertical scaling** is when change in the specifications of the instances (CPU, memory). If you replace a smaller instance with a larger one, is also called scaling up. If you replace a larger one with a smaller one, then it scales down.
- **Horizontal scaling** is when you increase or decrease the number of instances (add or remove instances as needed). It is called scaling in and scaling out. The easiest way to do horizontal scaling is through Auto Scaling.

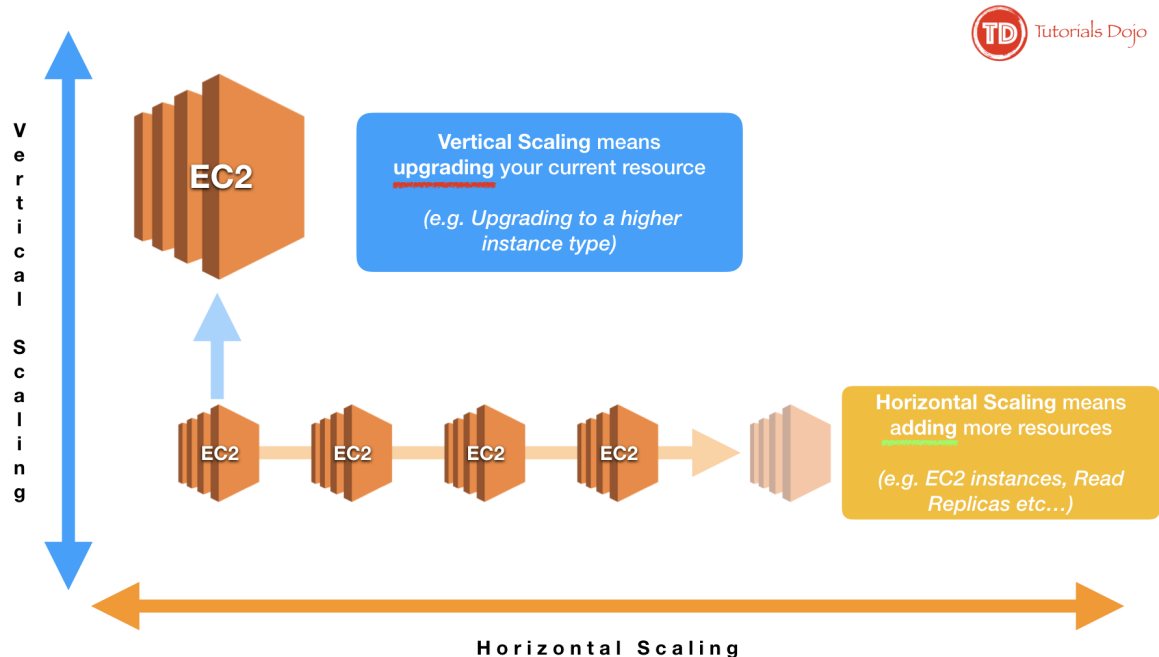


Fig. 102: Vertical scaling vs Horizontal scaling

### Test axioms

- If data is unstructured, Amazon S3 is generally the storage solution.
- Use caching strategically to improve performance.
- Know when and why to use Auto Scaling.

- Choose the instance and database type that makes the most sense for your workload and performance need.

## 1.16.3 Domain 3: Specify Secure Applications and Architectures

### 3.1 Determine how to secure application tiers

Amazon Web Services: Overview of Security Processes

AWS Security Best Practices

IAM Best Practices

### 3.2 Determine how to secure data

We can break down the problem of securing data into 2 problems:

- *Data in transit* when data moves in and out of AWS or within AWS.
- *Data at rest* when data is stored in Amazon S3 or EBS or some other systems in AWS.

#### Data in transit

For securing the transfer of data in and out of your AWS infrastructure, you can use:

- SSL over web.
- VPN for IPSec for data moving between corporate data centers and AWS.
- IPSec over AWS Direct Connect.
- Import/Export services via Snowball and Snowball mobile that keep the data encrypted.

Data sent to the AWS API, the AWS API calls use HTTPS/SSL by default.

#### Data at rest

Data stored at Amazon S3 is private by default, it requires AWS credentials for access. It can be accessed over HTTP or HTTPS. There is an audit mode of access to all objects. It supports ACL and policies over buckets, prefixes (directory/folder) and objects. There are several options for encryption to choose on:

- **Server-side encryption.** Request Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects. Options:
  - Amazon S3-Managed keys (SSE-S3).
  - KMS-Managed keys (SSE-KMS).
  - Customer-provided keys (SSE-C)
- **Client-side encryption** that requires encrypting the data before sending it to AWS service. Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools. Options:
  - *KMS managed master encryption keys (CSE-KMS).* When using an AWS KMS-managed customer master key to enable client-side data encryption, you provide an AWS KMS customer master key ID (CMK ID) to AWS.

- *Customer managed master encryption keys (CSE-C)*. Your client-side master keys and your unencrypted data are never sent to AWS. It's important that you safely manage your encryption keys because if you lose them, you can't decrypt your data. Sometimes the CSE-C is required for strict compliance and regulatory requirements of the company.

You can store keys using:

- *Key Management Service* which is a customer software-based key management integrated with many AWS services (for instance Amazon EBS, S3, RDS, Readshift, Elastic Transcode, WorkMail, EMR) and you can use it directly from application.
- *AWS CloudHSM* which is a HW-based key management that you can use it directly from application and it is FIPS 140-2 compliant.

### Test axioms

- Lock down the root user.
- Security groups only allow. Network ACLs allow explicit deny.
- Prefer IAM roles to access keys.

### 3.3 Define the networking infrastructure for a single VPC application

It is recommended to use subnets to define Internet accessibility.

- *Public subnets*. To support inbound/outbound access to the public Internet, include a routing table entry to an internet gateway.
- *Private subnets*. They do not have a routing table entry to an internet gateway. It is not directly accessible from the public Internet. To support restricted, outbound-only public Internet access, typically use a “jump box” (NAT/proxy/bastion host).

There are 2 main mechanisms to restricting access to a VPC: Security groups and Network ACLs.

	Security Groups	Access Control Lists
Access Type	Specify a port, protocol, source IP	Specify a port, protocol, source IP
Rules	Explicit Allow only	Explicit Allow or Deny
State	Stateful	Stateless
Application	Applied to ENIs	Applied to subnets
Association	Associated with single VPC	Associated with single VPC
Supports	VPC and EC2 Classic	VPC only

Fig. 103: Security groups versus Network ACLs

The different tiers of an application can be separated through security groups. You can use security groups to control traffic into, out of, and between resources.

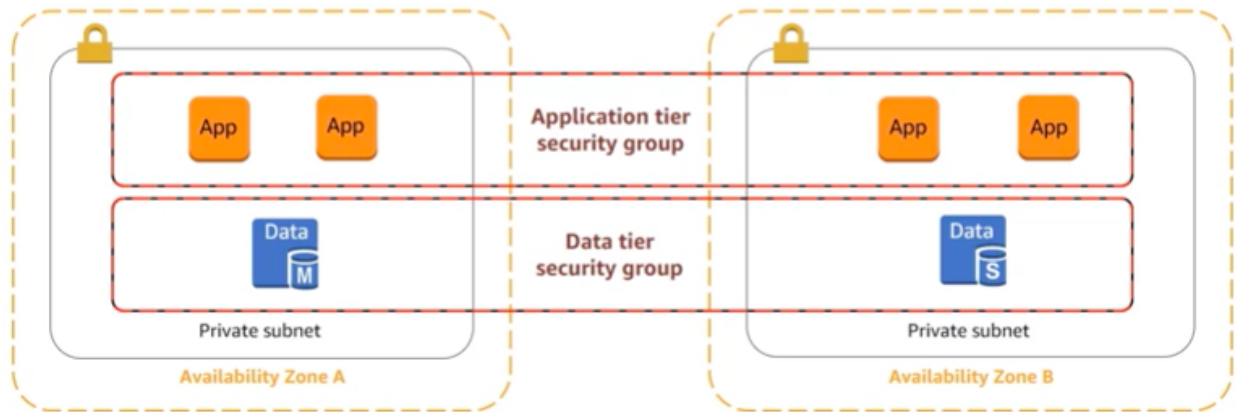


Fig. 104: Security groups

The services to get traffic in or out of a VPC are:

- Internet Gateway: It enables connecting to the Internet.
- Virtual Private Gateway: It enables connecting to customer's data center through a VPN.
- Direct Connect: It sets up a Dedicated pipe between customer's data center and AWS.
- VPC peering: It enables connecting VPCs to each others.
- NAT gateways: It allows Internet traffic from private subnets.

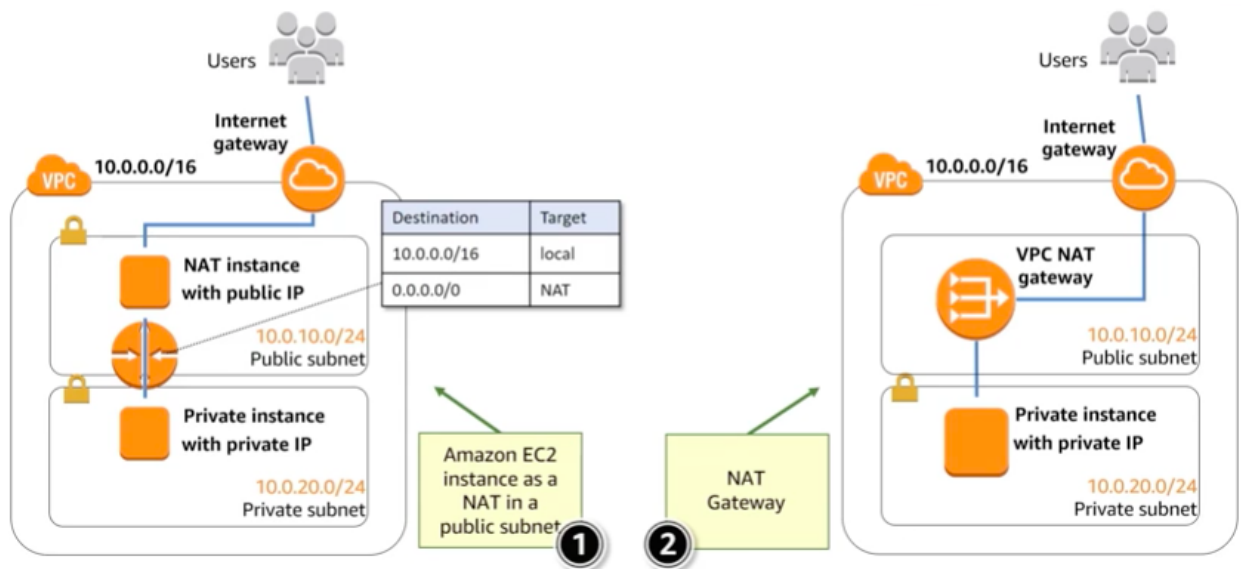


Fig. 105: Outbound traffic from private instances

The NAT gateway is more scalable than a NAT instance and it is managed service. A NAT instance is cheaper because it is a single instance.

AWS re:Invent 2018: Your Virtual Data Center: VPC Fundamentals and Connectivity Options (NET201)

Linux Bastion Hosts on AWS

Amazon Virtual Private Cloud User Guide



## 1.16.4 Domain 4: Design Cost-Optimized Architectures

### 4.1 Determine how to design cost-optimized storage

Considerations for estimating the cost of using Amazon S3 are the following:

1. Storage class.
2. Storage amount.
3. Number of requests.
4. Amount of data transfer.

Considerations for estimating the cost of using Amazon EBS are the following:

1. How many and what type of volumes.
2. Input/output operations per second (IOPS)
3. How frequently I am making snapshots and how long I am storing these snapshots.
4. Data transfer.

### 4.2 Determine how to design cost-optimized compute

Considerations for estimating the cost of using Amazon EC2 are the following:

1. Clock hours of server time.
2. Machine configuration.
3. Machine purchase type.
4. Number of instances.
5. Load balancing.
6. Detailed monitoring.
7. Auto Scaling.
8. Elastic IP addresses.
9. Operating systems and software packages.

Amazon EC2 pricing factor are the following:

- EC2 instance families.
- Tenancy: default or dedicated
- Pricing options.

---

**Note:** Instance storage.

Instance storage is free but is ephemeral.

---

## **Serverless architectures**

Another way to save cost is by using serverless architectures. In these architectures, you don't pay for the idle time. The compute logic could be through AWS Lambda, static content via Amazon S3, Amazon DynamoDB for storing state, and Amazon API Gateway to attach a REST endpoint to Lambda that can be called via Web from a browser or other HTTP client.

## **Amazon CloudFront**

CloudFront can help reduce your costs by avoiding fetching S3 data by caching it on CloudFront. Considerations for estimating the cost of using Amazon CloudFront are the following:

1. Traffic distribution.
2. Requests.
3. Data transfer out.

Caching with CloudFront can have positive impacts on both performance and cost-optimization.

## **Test axioms**

- If you know it's going to be on, reserve it.
- Any unused CPU time is a waste of money.
- Use the most cost-effective data storage service and class.
- Determine the most cost-effective EC2 pricing model and instance type for each workload.

## **1.16.5 Domain 5: Define Operationally-Excellent Architectures**

### **5.1 Choose design features in solutions that enable operational excellence**

#### **Test axioms**

- IAM roles are easier and safer than keys and passwords.
- Monitor metrics across the system.
- Automate responses to metrics where appropriate.
- Provide alerts for anomalous conditions.

## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`