# **AWS IAM**

IAM is the AWS Identity and Access Management Service. IAM is used to securely control individual and group access to AWS resources. IAM makes it easy to provide multiple users secure access to AWS resources. IAM can be used to manage:

* Users.
* Groups.
* Access policies.
* Roles.
* User credentials.
* User password policies.
* Multi-factor authentication (MFA).
* API keys for programmatic access (CLI).

Provides centralized control of your AWS account. Enables shared access to your AWS account.

By default, new users are created with NO access to any AWS services – they can only login to the AWS console.

Permission must be explicitly granted to allow a user to access an AWS service.

IAM users are individuals who have been granted access to an AWS account.

Each IAM user has three main components:

A username.

A password.

Permissions to access various resources.

You can apply granular permissions with IAM. You can assign users individual security credentials such as access keys, passwords, and multi-factor authentication devices.

IAM is not used for application-level authentication.

Identity Federation (including AD, Facebook etc). can be configured allowing secure access to resources in an AWS account without creating an IAM user account.

Multi-factor authentication (MFA) can be enabled/enforced for the AWS account and for individual users under the account.

MFA uses an authentication device that continually generates random, six-digit, single-use authentication codes.

You can authenticate using an MFA device in the following three ways:

Through the AWS Management Console – the user is prompted for a user name, password, and authentication code.

Using the AWS API – restrictions are added to IAM policies and developers can request temporary security credentials and pass MFA parameters in their AWS STS API requests.

Using the AWS CLI by obtaining temporary security credentials from STS (aws sts get-session-token).

Want to see how to setup MFA? In the brief AWS Hands-on Labs video tutorial below, you’ll learn how to activate a virtual Multi-factor Authentication (MFA) for your AWS Root Account. In under 5 minutes, we cover: Deleting the Root Account Access Key and Activating Multi-Factor Authentication.

It is a best practice to use MFA for all users and to use U2F or hardware MFA devices for all privileged users.

IAM is universal (global) and does not apply to regions.

IAM is eventually consistent.

IAM replicates data across multiple data centres around the world.

The “root account” is the account created when your setup the AWS account. It has complete Admin access and is the only account that has this access by default.

It is a best practice to not use the root account for anything other than billing.

Power user access allows all permissions except the management of groups and users in IAM.

Temporary security credentials consist of the AWS access key ID, secret access key, and security token.

IAM can assign temporary security credentials to provide users with temporary access to services/resources.

To sign-in you must provide your account ID or account alias in addition to a user name and password.

The sign-in URL includes the account ID or account alias, e.g.:

[https://My\_AWS\_Account\_ID.signin.aws.amazon.com/console/.](https://my_aws_account_id.signin.aws.amazon.com/console/)

Alternatively you can sign-in at the following URL and enter your account ID or alias manually:

[https://console.aws.amazon.com/.](https://console.aws.amazon.com/)

IAM integrates with many different AWS services.

IAM supports PCI DSS compliance.

AWS recommend that you use the AWS SDKs to make programmatic API calls to IAM.

However, you can also use the IAM Query API to make direct calls to the IAM web service.

IAM Elements

Principals:

An entity that can take an action on an AWS resource.

Your administrative IAM user is your first principal.

You can allow users and services to assume a role.

IAM supports federated users.

IAM supports programmatic access to allow an application to access your AWS account.

IAM users, roles, federated users, and applications are all AWS principals.

Requests:

Principals send requests via the Console, CLI, SDKs, or APIs.

Requests are:

Actions (or operations) that the principal wants to perform.

Resources upon which the actions are performed.

Principal information including the environment from which the request was made.

Request context – AWS gathers the request information:

Principal (requester).

Aggregate permissions associated with the principal.

Environment data, such as IP address, user agent, SSL status etc.

Resource data, or data that is related to the resource being requested.

Authentication:

A principal sending a request must be authenticated to send a request to AWS.

To authenticate from the console, you must sign in with your user name and password.

To authenticate from the API or CLI, you must provide your access key and secret key.

Authorization:

IAM uses values from the request context to check for matching policies and determines whether to allow or deny the request.

IAM policies are stored in IAM as JSON documents and specify the permissions that are allowed or denied.

IAM policies can be:

User (identity) based policies.

Resource-based policies.

IAM checks each policy that matches the context of your request.

If a single policy has a deny action IAM denies the request and stops evaluating (explicit deny).

Evaluation logic:

By default, all requests are denied (implicit deny).

An explicit allow overrides the implicit deny.

An explicit deny overrides any explicit allows.

Only the root user has access to all resources in the account by default.

Actions:

Actions are defined by a service.

Actions are the things you can do to a resource such as viewing, creating, editing, deleting.

Any actions on resources that are not explicitly allowed are denied.

To allow a principal to perform an action you must include the necessary actions in a policy that applies to the principal or the affected resource.

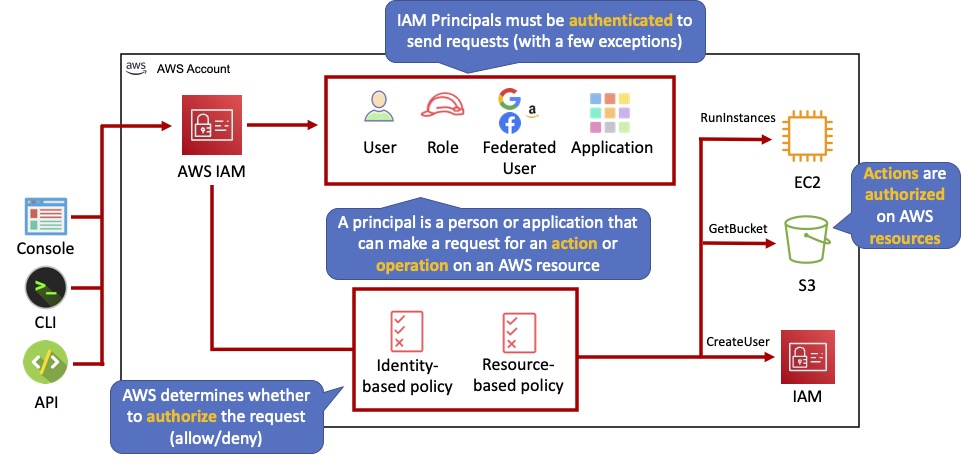
Resources:

A resource is an entity that exists within a service.

E.g. EC2 instances, S3 buckets, IAM users, and DynamoDB tables.

Each AWS service defines a set of actions that can be performed on the resource.

After AWS approves the actions in your request, those actions can be performed on the related resources within your account.



**Authentication Methods**

**Console password:**

A password that the user can enter to sign into interactive sessions such as the AWS Management Console.

You can allow users to change their own passwords.

You can allow selected IAM users to change their passwords by disabling the option for all users and using an IAM policy to grant permissions for the selected users.

**Access Keys:**

A combination of an access key ID and a secret access key.

You can assign two active access keys to a user at a time.

These can be used to make programmatic calls to AWS when using the API in program code or at a command prompt when using the AWS CLI or the AWS PowerShell tools.

You can create, modify, view, or rotate access keys.

When created IAM returns the access key ID and secret access key.

The secret access is returned only at creation time and if lost a new key must be created.

Ensure access keys and secret access keys are stored securely.

Users can be given access to change their own keys through IAM policy (not from the console).

You can disable a user’s access key which prevents it from being used for API calls.

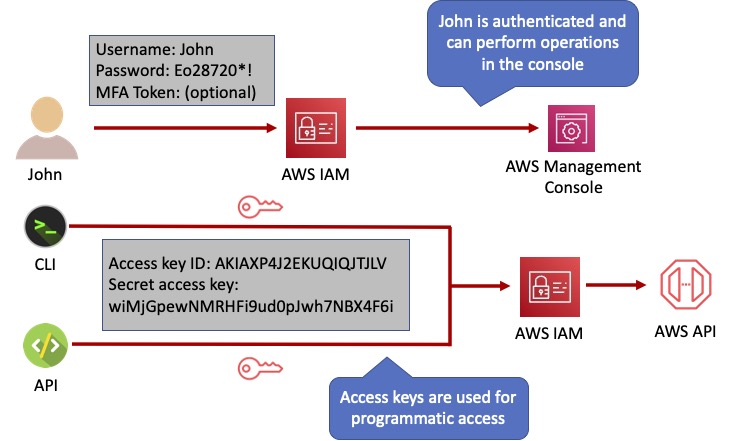
Server certificates:s

SSL/TLS certificates that you can use to authenticate with some AWS services.

AWS recommends that you use the AWS Certificate Manager (ACM) to provision, manage and deploy your server certificates.

Use IAM only when you must support HTTPS connections in a region that is not supported by ACM.

The following diagram shows the different methods of authentication available with IAM:



## Users

An IAM user is an entity that represents a person or service.

Can be assigned:

An access key ID and secret access key for programmatic access to the AWS API, CLI, SDK, and other development tools.

A password for access to the management console.

By default, users cannot access anything in your account.

The account root user credentials are the email address used to create the account and a password.

The root account has full administrative permissions, and these cannot be restricted.

Best practice for root accounts:

Don’t use the root user credentials.

Don’t share the root user credentials.

Create an IAM user and assign administrative permissions as required.

Enable MFA.

IAM users can be created to represent applications, and these are known as “service accounts”.

You can have up to 5000 users per AWS account.

Each user account has a friendly name and an ARN which uniquely identifies the user across AWS.

A unique ID is also created which is returned only when you create the user using the API, Tools for Windows PowerShell, or the AWS CLI.

You should create individual IAM accounts for users (best practice not to share accounts).

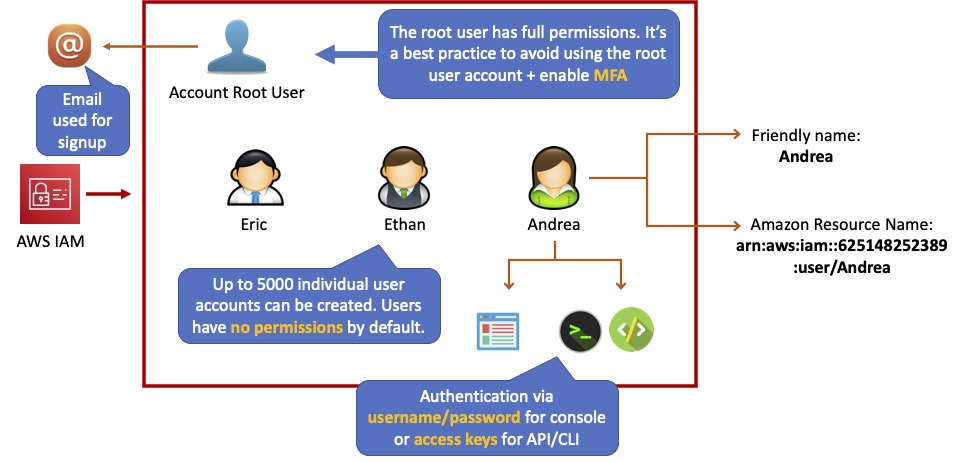
The Access Key ID and Secret Access Key are not the same as a password and cannot be used to login to the AWS console.

The Access Key ID and Secret Access Key can only be generated once and must be regenerated if lost.

A password policy can be defined for enforcing password length, complexity etc. (applies to all users).

You can allow or disallow the ability to change passwords using an IAM policy.

Access keys and passwords should be changed regularly.



## Groups

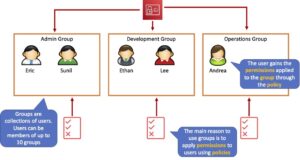
Groups are collections of users and have policies attached to them.

A group is not an identity and cannot be identified as a principal in an IAM policy.

Use groups to assign permissions to users.

Use the principal of least privilege when assigning permissions.

You cannot nest groups (groups within groups).



## Roles

Roles are created and then “assumed” by trusted entities and define a set of permissions for making AWS service requests.

With IAM Roles you can delegate permissions to resources for users and services without using permanent credentials (e.g. user name and password).

IAM users or AWS services can assume a role to obtain temporary security credentials that can be used to make AWS API calls.

You can delegate using roles.

There are no credentials associated with a role (password or access keys).

IAM users can temporarily assume a role to take on permissions for a specific task.

A role can be assigned to a federated user who signs in using an external identity provider.

Temporary credentials are primarily used with IAM roles and automatically expire.

Roles can be assumed temporarily through the console or programmatically with the AWS CLI, Tools for Windows PowerShell, or API.

IAM roles with EC2 instances:

IAM roles can be used for granting applications running on EC2 instances permissions to AWS API requests using instance profiles.

Only one role can be assigned to an EC2 instance at a time.

A role can be assigned at the EC2 instance creation time or at any time afterwards.

When using the AWS CLI or API instance profiles must be created manually (it’s automatic and transparent through the console).

Applications retrieve temporary security credentials from the instance metadata.

Role Delegation:

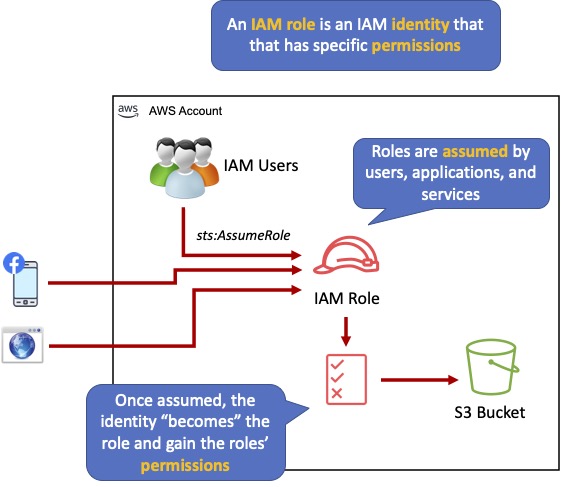
Create an IAM role with two policies:

Permissions policy – grants the user of the role the required permissions on a resource.

Trust policy – specifies the trusted accounts that are allowed to assume the role.

Wildcards (\*) cannot be specified as a principal.

A permissions policy must also be attached to the user in the trusted account.



## Policies

Policies are documents that define permissions and can be applied to users, groups, and roles.

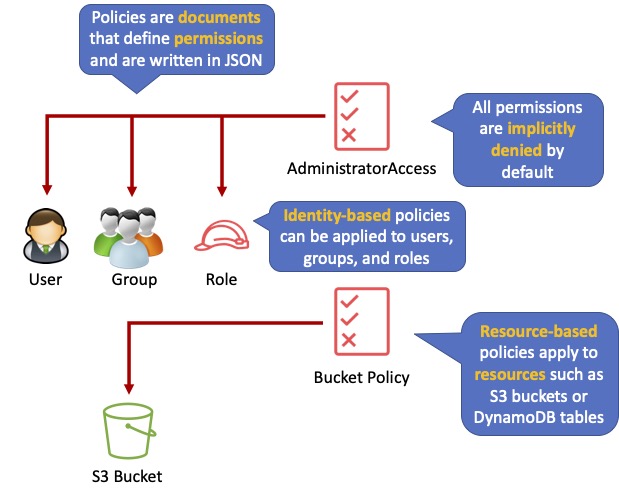
Policy documents are written in JSON (key value pair that consists of an attribute and a value).

All permissions are implicitly denied by default.

The most restrictive policy is applied.

The IAM policy simulator is a tool to help you understand, test, and validate the effects of access control policies.

The Condition element can be used to apply further conditional logic.



Inline Policies vs Managed Policies

There are 3 types of policies:

Managed policies.

Customer managed policies.

Inline policies.

Managed Policy:

Created and administered by AWS.

Used for common use cases based on job function.

Save you having to create policies yourself.

Can be attached to multiple users, groups, or roles within and across AWS accounts.

Cannot change the permissions assigned.

Customer Managed Policy:

Standalone policy that you create and administer in your own AWS account.

Can be attached to multiple users, groups, and roles – but only within your own account.

Can be created by copying an existing managed policy and then customizing it.

Recommended for use cases where the existing AWS Managed Policies don’t meet the needs of your environment.

Inline Policy:

Inline policies are embedded within the user, group, or role to which it is applied.

Strict 1:1 relationship between the entity and the policy.

When you delete the user, group, or role in which the inline policy is embedded, the policy will also be deleted.

In most cases, AWS recommends using Managed Policies instead of inline policies.

Inline policies are useful when you want to be sure that the permissions in a policy are not inadvertently assigned to any other user, group, or role.

AWS Managed and Customer Managed Policies

An AWS managed policy is a standalone policy that is created and administered by AWS.

Standalone policy means that the policy has its own Amazon Resource Name (ARN) that includes the policy name.

AWS managed policies are designed to provide permissions for many common use cases.

You cannot change the permissions defined in AWS managed policies.

Some AWS managed policies are designed for specific job functions.

The job-specific AWS managed policies include:

Administrator.

Billing.

Database Administrator.

Data Scientist.

Developer Power User.

Network Administrator.

Security Auditor.

Support User.

System Administrator.

View-Only User.

You can create standalone policies that you administer in your own AWS account, which we refer to as customer managed policies.

You can then attach the policies to multiple principal entities in your AWS account.

When you attach a policy to a principal entity, you give the entity the permissions that are defined in the policy.

IAM Policy Evaluation Logic

By default, all requests are implicitly denied. (Alternatively, by default, the AWS account root user has full access).

An explicit allow in an identity-based or resource-based policy overrides this default.

If a permissions boundary, Organizations SCP, or session policy is present, it might override the allow with an implicit deny.

An explicit deny in any policy overrides any allows.

A few concepts should be known to understand the logic:

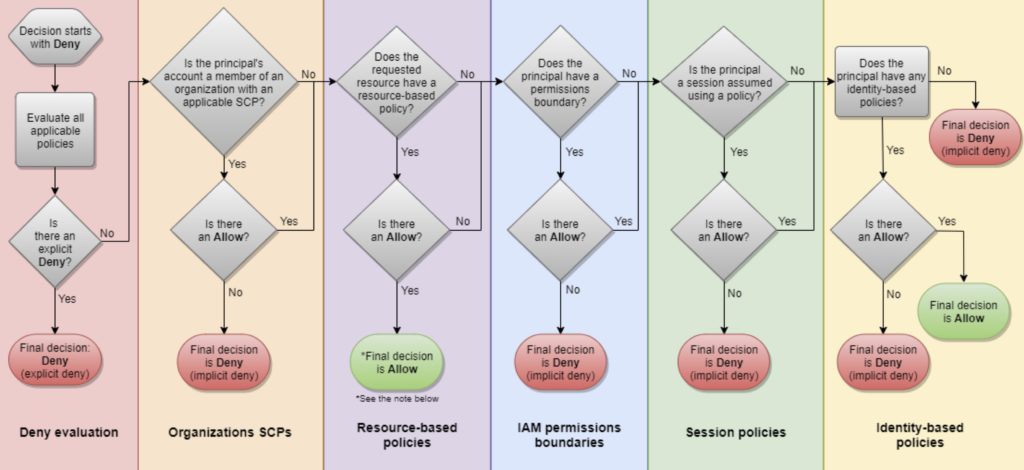
Identity-based policies – Identity-based policies are attached to an IAM identity (user, group of users, or role) and grant permissions to IAM entities (users and roles).

Resource-based policies – Resource-based policies grant permissions to the principal (account, user, role, or federated user) specified as the principal.

IAM permissions boundaries – Permissions boundaries are an advanced feature that sets the maximum permissions that an identity-based policy can grant to an IAM entity (user or role).

AWS Organizations service control policies (SCPs) – Organizations SCPs specify the maximum permissions for an organization or organizational unit (OU). Session policies – Session policies are advanced policies that you pass as parameters when you programmatically create a temporary session for a role or federated user.

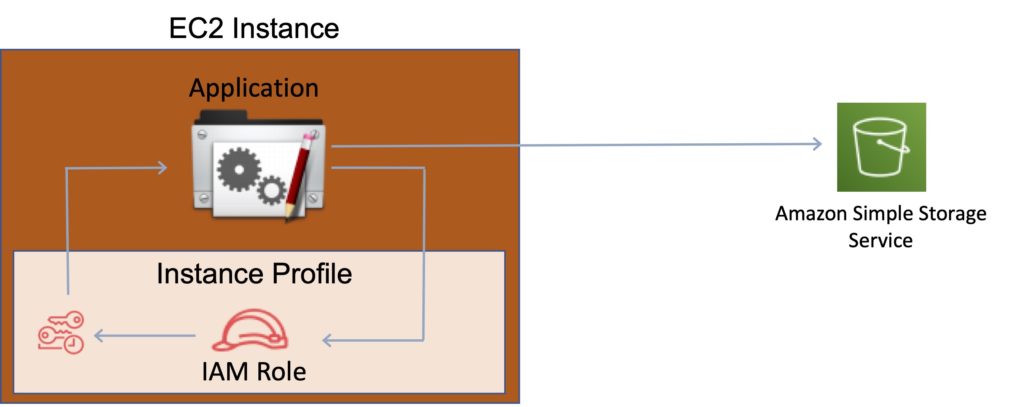
The following flowchart details the IAM policy evaluation logic:



IAM Instance Profiles

An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.

An instance profile can contain only one IAM role, although a role can be included in multiple instance profiles.



You can use the following AWS CLI commands to work with instance profiles in an AWS account:

Create an instance profile: aws iam create-instance-profile

Add a role to an instance profile: aws iam add-role-to-instance-profile

List instance profiles: aws iam list-instance-profiles, aws iam list-instance-profiles-for-role

Get information about an instance profile: aws iam get-instance-profile

Remove a role from an instance profile: aws iam remove-role-from-instance-profile

Delete an instance profile: aws iam delete-instance-profile

AWS Security Token Service

The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for IAM users or for users that you authenticate (federated users).

By default, AWS STS is available as a global service, and all AWS STS requests go to a single endpoint at[https://sts.amazonaws.com](https://sts.amazonaws.com/)

You can optionally send your AWS STS requests to endpoints in any region (can reduce latency).

Credentials will always work globally.

STS supports AWS CloudTrail, which records AWS calls for your AWS account and delivers log files to an S3 bucket.

Temporary security credentials work almost identically to long-term access key credentials that IAM users can use, with the following differences:

Temporary security credentials are short-term.

They can be configured to last anywhere from a few minutes to several hours.

After the credentials expire, AWS no longer recognizes them or allows any kind of access to API requests made with them.

Temporary security credentials are not stored with the user but are generated dynamically and provided to the user when requested.

When (or even before) the temporary security credentials expire, the user can request new credentials, if the user requesting them still has permission to do so.

Advantages of STS are:

You do not have to distribute or embed long-term AWS security credentials with an application.

You can provide access to your AWS resources to users without having to define an AWS identity for them (temporary security credentials are the basis for IAM Roles and ID Federation).

The temporary security credentials have a limited lifetime, so you do not have to rotate them or explicitly revoke them when they’re no longer needed.

After temporary security credentials expire, they cannot be reused (you can specify how long the credentials are valid for, up to a maximum limit).

The AWS STS API action returns temporary security credentials that consist of:

An access key which consists of an access key ID and a secret ID.

A session tokens.

Expiration or duration of validity.

Users (or an application that the user runs) can use these credentials to access your resources.

With STS you can request a session token using one of the following APIs:

AssumeRole – can only be used by IAM users (can be used for MFA).

AssumeRoleWithSAML – can be used by any user who passes a SAML authentication response that indicates authentication from a known (trusted) identity provider.

AssumeRoleWithWebIdentity – can be used by an user who passes a web identity token that indicates authentication from a known (trusted) identity provider.

GetSessionToken – can be used by an IAM user or AWS account root user (can be used for MFA).

GetFederationToken – can be used by an IAM user or AWS account root user.

AWS recommends using Cognito for identity federation with Internet identity providers.

Users can come from three sources.

Federation (typically AD):

Uses SAML 2.0.

Grants temporary access based on the users AD credentials.

Does not need to be a user in IAM.

Single sign-on allows users to login to the AWS console without assigning IAM credentials.

Federation with Mobile Apps:

Use Facebook/Amazon/Google or other OpenID providers to login.

Cross Account Access:

Lets users from one AWS account access resources in another.

To make a request in a different account the resource in that account must have an attached resource-based policy with the permissions you need.

Or you must assume a role (identity-based policy) within that account with the permissions you need.

There are a couple of ways STS can be used.

Scenario 1:

Develop an Identity Broker to communicate with LDAP and AWS STS.

Identity Broker always authenticates with LDAP first, then with AWS STS.

Application then gets temporary access to AWS resources.

Scenario 2:

Develop an Identity Broker to communicate with LDAP and AWS STS.

Identity Broker authenticates with LDAP first, then gets an IAM role associated with the user.

Application then authenticates with STS and assumes that IAM role.

Application uses that IAM role to interact with the service.

Cross Account Access

Useful for situations where an AWS customer has separate AWS account – for example for development and production resources.

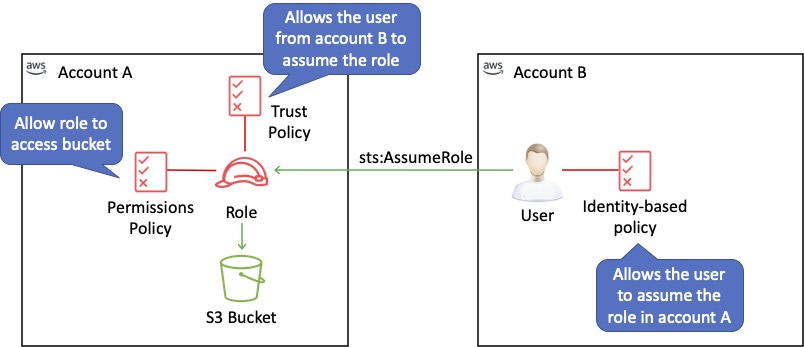
Cross Account Access makes is easier to work productively within a multi-account (or multi-role) AWS environment by making is easy to switch roles within the AWS Management Console.

Can sign-in to the console using your IAM user name and then switch the console to manage another account without having to enter another user name and password.

Lets users from one AWS account access resources in another.

To make a request in a different account the resource in that account must have an attached resource-based policy with the permissions you need.

Or you must assume a role (identity-based policy) within that account with the permissions you need.



IAM Best Practices

To secure AWS resources it is recommended that you follow these best practices:

Lock away your AWS account root user access keys.

Use roles to delegate permissions.

Grant least privilege.

Get started using permissions with AWS managed policies.

Validate your policies.

Use customer managed policies instead of inline policies.

Use access levels to review IAM permissions.

Configure a strong password policy for your users.

Enable MFA.

Use roles for applications that run on Amazon EC2 instances.

Do not share access keys.

Rotate credentials regularly.

Remove unnecessary credentials.

Use policy conditions for extra security.

Monitor activity in your AWS account.

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud.

With Amazon EC2 you launch virtual server instances on the AWS cloud.

Each virtual server is known as an “instance”.

You use preconfigured templates for your instances known as Amazon Machine Images (AMIs).

Each AMI includes the information needed to launch your EC2 instance (including the operating system and any included software packages).

Amazon EC2 currently supports a variety of operating systems including:

Amazon Linux.

Ubuntu.

Windows Server.

MacOS.

Red Hat Enterprise Linux.

SUSE Linux Enterprise Server.

Fedora.

Debian.

CentOS.

Gentoo Linux.

Oracle Linux.

FreeBSD.

EC2 compute units (ECUs) provide the relative measure of the integer processing power of an Amazon EC2 instance.

With EC2 you have full control at the operating system layer (root/admin access).

Key pairs are used to securely connect to EC2 instances:

A key pair consists of a public key that AWS stores, and a private key file that you store.

For Windows AMIs, the private key file is required to obtain the password used to log into your instance.

For Linux AMIs, the private key file allows you to securely SSH (secure shell) into your instance.

Metadata and User Data:

User data is data that is supplied by the user at instance launch in the form of a script.

Instance metadata is data about your instance that you can use to configure or manage the running instance.

User data is limited to 16KB.

User data and metadata are not encrypted.

Instance metadata is available at http://169.254.169.254/latest/meta-data/ (the trailing “/” is required).

Instance user data is available at: http://169.254.169.254/latest/user-data.

The IP address 169.254.169.254 is a link-local address and is valid only from the instance.

On Linux you can use the curl command to view metadata and user data, e.g.“curl http://169.254.169.254/latest/meta-data/”.

The Instance Metadata Query tool allows you to query the instance metadata without having to type out the full URI or category names.

EC2 Instance types

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases.

Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications.

Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

|  |  |  |
| --- | --- | --- |
| Category | Families | Purpose/Design |
| General Purpose | Mac, T4g, T3, T3a, T2, M6g, M6i, M5, M5a, M5n, M5zn, M5, A1 | General Purpose Instances provide a balance on compute, memory, and networking resources, and can be used for a variety of diverse workloads. |
| Compute Optimized | C6g, C6gn, C6i, C5, C5a, C5n, C4 | Compute optimized are ideal for compute bound applications that benefit from high performance processors. |
| Memory Optimized | R6g, R5, R5a, R5b, R5n, R4, X2gd, X1e, X1, High Memory, z1d | Memory optimized instances are designed to deliver fast performance for workloads that process large data sets in memory. |
| Accelerated Computing | P4, P3, P2, DL1, Inf1, G5, G4dn, G4ad, G3, F1, VT1 | Accelerated Computing instances use hardware accelerators, or co-processors to perform functions such as floating-point number calculations, graphics processing, or data pattern matching. |
| Storage Optimized | I3, I3en, D2, D4, D3en, H1 | This instance family provides Non-Volatile Memory Express (NVMe) SSD-Backed instance storage optimized for low latency, very high random I/O performance, high sequential read throughput and high IOPS at a low cost. |

Launching EC2 Instances

Choose an Amazon Machine Image (AMI).

Choose whether to auto-assign a public IP – default is to use the subnet setting.

Can add an instance to a placement group (more about this below).

Instances can be assigned to IAM roles which configures them with credentials to access AWS resources.

Termination protection can be enabled and prevents you from terminating an instance.

Basic monitoring is enabled by default (5-minute periods), detailed monitoring can be enabled (1-minute periods, chargeable).

Can define shared or dedicated tenancy.

T2 unlimited allows applications to burst past CPU performance baselines as required (chargeable).

Can add a script to run on startup (user data).

Can join to a directory (Windows instances only).

There is an option to enable an Elastic GPU (Windows instances only).

Storage options include adding additional volumes and choosing the volume type.

Use Amazon Elastic File System (EFS) for mounting a shared filesystem to multiple EC2 instances.

Non-root volumes can be encrypted.

Root volumes can be encrypted at launch.

There is an option to create tags (or can be done later).

You can select an existing security group or create a new one.

You must create or use an existing key pair – this is required if you want to access your instances via SSH. However, you can also attach the ‘AmazonEC2RoleforSSM’ IAM role to your EC2 instance to allow connection to your instance via Systems Manager (Session Manager).

Amazon Machine Images

An Amazon Machine Image (AMI) provides the information required to launch an instance.

An AMI includes the following:

A template for the root volume for the instance (for example, an operating system, an application server, and applications).

Launch permissions that control which AWS accounts can use the AMI to launch instances.

A block device mapping that specifies the volumes to attach to the instance when it’s launched.

AMIs are regional. You can only launch an AMI from the region in which it is stored. However, you can copy AMIs to other regions using the console, command line, or the API.

Volumes attached to the instance are either EBS or Instance store:

Amazon Elastic Block Store (EBS) provides persistent storage. EBS snapshots, which reside on Amazon S3, are used to create the volume.

Instance store volumes are ephemeral (non-persistent). That means data is lost if the instance is shut down. A template stored on Amazon S3 is used to create the volume.

Billing and provisioning

There are several options for how you consume and pay for Amazon EC2 instances.

On demand

Pay for hours used with no commitment.

Low cost and flexibility with no upfront cost.

Ideal for auto scaling groups and unpredictable workloads.

Good for dev/test.

Spot

Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud.

Spot Instances are available at up to a 90% discount compared to On-Demand prices.

You can use Spot Instances for various stateless, fault-tolerant, or flexible applications such as big data, containerized workloads, CI/CD, web servers, high-performance computing (HPC), and other test & development workloads.

You can request Spot Instances by using the Spot management console, CLI, API or the same interface that is used for launching On-Demand instances by indicating the option to use Spot.

You can also select a Launch Template or a pre-configured or custom Amazon Machine Image (AMI), configure security and network access to your Spot instance, choose from multiple instance types and locations, use static IP endpoints, and attach persistent block storage to your Spot instances.

New pricing model: The Spot price is [determined](https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/) by long term trends in supply and demand for EC2 spare capacity.

You don’t have to bid for Spot Instances in the new pricing model, and you just pay the Spot price that’s in effect for the current hour for the instances that you launch.

Spot Instances receive a two-minute interruption notice when these instances are about to be reclaimed by EC2, because EC2 needs the capacity back.

Instances are not interrupted because of higher competing bids.

To reduce the impact of interruptions and optimize Spot Instances, diversify, and run your application across multiple capacity pools.

Each instance family, each instance size, in each Availability Zone, in every Region is a separate Spot pool.

You can use the RequestSpotFleet API operation to launch thousands of Spot Instances and diversify resources automatically.

To further reduce the impact of interruptions, you can also set up Spot Instances and Spot Fleets to [respond](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html#interruption-behavior) to an interruption notice by stopping or hibernating rather than terminating instances when capacity is no longer available.

Reserved

Purchase (or agree to purchase) usage of EC2 instances in advance for significant discounts over On-Demand pricing.

Provides a capacity reservation when used in a specific AZ.

AWS Billing automatically applies discounted rates when you launch an instance that matches your purchased RI.

Capacity is reserved for a term of 1 or 3 years.

EC2 has three RI types: Standard, Convertible, and Scheduled.

Standard = commitment of 1 or 3 years, charged whether it’s on or off.

Scheduled = reserved for specific periods of time, accrue charges hourly, billed in monthly increments over the term (1 year).

Scheduled RIs match your capacity reservation to a predictable recurring schedule.

For the differences between standard and convertible RIs, see the table below.

RIs are used for steady state workloads and predictable usage.

Ideal for applications that need reserved capacity.

Upfront payments can reduce the hourly rate.

Can switch AZ within the same region.

Can change the instance size within the same instance type.

Instance type modifications are supported for Linux only.

Cannot change the instance size of Windows RIs.

Billed whether running or not.

Can sell reservations on the AWS marketplace.

Can be used in Auto Scaling Groups.

Can be used in Placement Groups.

Can be shared across multiple accounts within Consolidated Billing.

If you don’t need your RI’s, you can try to sell them on the Reserved Instance Marketplace.

|  |  |  |
| --- | --- | --- |
|  | Standard | Convertible |
| Terms | 1 year, 3 year | 1 year, 3 year |
| Average discount off On-Demand price | 40% – 60% | 31% – 54% |
| Change AZ, instance size, networking type | Yes via ModifyReservedInstance API or console | Yes via ExchangeReservedInstance API or console |
| Change instance family, OS, tenancy, payment options | No | Yes |
| Benefit from price reductions | No | Yes |

RI Attributes:

Instance type – designates CPU, memory, networking capability.

Platform – Linux, SUSE Linux, RHEL, Microsoft Windows, Microsoft SQL Server.

Tenancy – Default (shared) tenancy, or Dedicated tenancy.

Availability Zone (optional) – if AZ is selected, RI is reserved, and discount applies to that AZ (Zonal RI). If no AZ is specified, no reservation is created but the discount is applied to any instance in the family in any AZ in the region (Regional RI).

Comparing Amazon EC2 Pricing Models

The following table provides a brief comparison of On-demand, Reserved and Spot pricing models:

|  |  |  |
| --- | --- | --- |
| On-Demand | Reserved | Spot |
| No upfront fee | Options: No upfront, partial upfront or all upfront | No upfront fee |
| Charged by hour or second | Charged by hour or second | Charged by hour or second |
| No commitment | 1-year or 3-year commitment | No commitment |
| Ideal for short term needs or unpredictable workloads | Ideal for steady-state workloads and predictable usage | Ideal for cost-sensitive, compute intensive use cases that can withstand interruption |

You are limited to running up to a total of 20 On-Demand instances across the instance family, purchasing 20 Reserved Instances, and requesting Spot Instances per your dynamic spot limit per region (by default).

Dedicated hosts

Physical servers dedicated just for your use.

You then have control over which instances are deployed on that host.

Available as On-Demand or with Dedicated Host Reservation.

Useful if you have server-bound software licenses that use metrics like per-core, per-socket, or per-VM.

Each dedicated host can only run one EC2 instance size and type.

Good for regulatory compliance or licensing requirements.

Predictable performance.

Complete isolation.

Most expensive option.

Billing is per host.

Dedicated instances

Virtualized instances on hardware just for you.

Also uses physically dedicated EC2 servers.

Does not provide the additional visibility and controls of dedicated hosts (e.g. how instances are placed on a server).

Billing is per instance.

May share hardware with other non-dedicated instances in the same account.

Available as On-Demand, Reserved Instances, and Spot Instances.

Cost additional $2 per hour per region.

The following table describes some of the differences between dedicates instances and dedicated hosts:

|  |  |  |
| --- | --- | --- |
| Characteristic | Dedicated Instances | Dedicated Hosts |
| Enables the use of dedicated physical servers | X | X |
| Per instance billing (subject to a $2 per region fee) | X |  |
| Per host billing |  | X |
| Visibility of sockets, cores, host ID |  | X |
| Affinity between a host and instance |  | X |
| Targeted instance placement |  | X |
| Automatic instance placement | X | X |
| Add capacity using an allocation request |  | X |

Partial instance-hours consumed are billed based on instance usage.

Instances are billed when they’re in a running state – need to stop or terminate to avoid paying.

Charging by the hour or second (by the second with Linux instances only).

Data between instances in different regions is charged (in and out).

Regional Data Transfer rates apply if at least one of the following is true, but are only charged once for a given instance even if both are true:

The other instance is in a different Availability Zone, regardless of which type of address is used.

Public or Elastic IP addresses are used, regardless of which Availability Zone the other instance is in.

Networking

Networking Limits (per region or as specified):

|  |  |
| --- | --- |
| Name | Default Limit |
| EC2-Classic Elastic IPs | 5 |
| EC2-VPC Elastic IPs | 5 |
| VPCs | 5 |
| Subnets per VPC | 200 |
| Security groups per VPC | 500 |
| Rules per VPC security group | 50 |
| VPC security groups per elastic network interface | 5 |
| Network interfaces | 350 |
| Network ACLs per VPC | 200 |
| Rules per network ACL | 20 |
| Route tables per VPC | 200 |
| Entries per route table | 50 |
| Active VPC peering connections | 50 |
| Outstanding VPC peering connection requests | 25 |
| Expiry time for an unaccepted VPC peering connection | 168 |

IP Addresses

There are three types of IP address that can be assigned to an Amazon EC2 instance:

Public – public address that is assigned automatically to instances in public subnets and reassigned if instance is stopped/started.

Private – private address assigned automatically to all instances.

Elastic IP – public address that is static.

Public IPv4 addresses are lost when the instance is stopped but private addresses (IPv4 and IPv6) are retained.

Public IPv4 addresses are retained if you restart the instance.

Elastic IPs are retained when the instance is stopped.

Elastic IP addresses are static public IP addresses that can be remapped (moved) between instances.

All accounts are limited to 5 elastic IPs per region by default; however this is a soft limit which can be raised by a service limit increase to AWS Support.

AWS charges for elastic IP’s when they’re not being used.

An Elastic IP address is for use in a specific region only.

You can assign custom tags to your Elastic IP addresses to categorize them.

By default, EC2 instances come with a private IP assigned to the primary network interface (eth0).

Public IP addresses are assigned for instances in public subnets (VPC).

DNS records for elastic IP’s can be configured by filling out a form.

Secondary IP addresses can be useful for hosting multiple websites on a server or redirecting traffic to a standby EC2 instance for HA.

You can choose whether secondary IP addresses can be reassigned.

You can associate a single private IPv4 address with a single Elastic IP address and vice versa.

When reassigned the IPv4 to Elastic IP association is maintained.

When a secondary private address is unassigned from an interface, the associated Elastic IP address is disassociated.

You can assign or remove IP addresses from EC2 instances while they are running or stopped.

When you stop and start and EC2 instance, it will generally be moved to different underlying hardware.

Exam tip: You can stop and start an EC2 instance to move it to a different physical host if EC2 status checks are failing or there is planned maintenance on the current physical host.

You can modify the following attributes of an instance only when it is stopped:

Instance type.

User data.

Kernel.

RAM disk.

All IP addresses (IPv4 and IPv6) remain attached to the network interface when detached or reassigned to another instance.

You can attach a network interface to an instance in a different subnet if it’s within the same AZ.

You can bring part or all your publicly routable IPv4 or IPv6 address range from your on-premises network to AWS. This is called BYOIP.

You continue to own the address range, but AWS advertises it on the internet by default. After you bring the address range to AWS, it appears in your AWS account as an address pool.

BYOIP is not available in all Regions and for all resources.

The following table compares the different types of IP address available in Amazon EC2:

|  |  |
| --- | --- |
| Name | Description |
| Public IP address | Lost when the instance is stopped    Used in Public Subnets  No charge  Associated with a private IP address on the instance  Cannot be moved between instances |
| Private IP address | Retained when the instance is stopped    Used in Public and Private Subnets |
| Elastic IP address | Static Public IP address    You are charged if not used  Associated with a private IP address on the instance  Can be moved between instances and Elastic Network Adapters |

Elastic Network Interfaces

An elastic network interface (referred to as a network interface) is a logical networking component in a VPC that represents a virtual network card.

A network interface can include the following attributes:

A primary private IPv4 address from the IPv4 address range of your VPC.

One or more secondary private IPv4 addresses from the IPv4 address range of your VPC.

One Elastic IP address (IPv4) per private IPv4 address.

One public IPv4 address.

One or more IPv6 addresses.

One or more security groups.

A MAC addresses.

A source/destination check flag.

A description.

You can create and configure network interfaces in your account and attach them to instances in your VPC.

You cannot increase the network bandwidth of an instance by teaming multiple ENIs.

eth0 is the primary network interface and cannot be moved or detached.

By default, eth0 is the only Elastic Network Interface (ENI) created with an EC2 instance when launched.

You can add additional interfaces to EC2 instances (number dependent on instances family/type).

An ENI is bound to an AZ, and you can specify which subnet/AZ you want the ENI to be added in.

You can specify which IP address within the subnet to configure or leave it to be auto assigned.

You can only add one extra ENI when launching but more can be attached later.

Attaching ENIs:

ENIs can be “hot attached” to running instances.

ENIs can be “warm-attached” when the instance is stopped.

ENIs can be “cold-attached” when the instance is launched.

If you add a second interface AWS will not assign a public IP address to eth0 (you would need to add an Elastic IP).

Default interfaces are terminated with instance termination.

Manually added interfaces are not terminated by default.

You can change the termination behavior.

Enhanced Networking – Elastic Network Adapter (ENA)

Enhanced networking provides higher bandwidth, higher packet-per-second (PPS) performance, and consistently lower inter-instance latencies.

Enhanced networking is enabled using an Elastic Network Adapter (ENA).

If your packets-per-second rate appears to have reached its ceiling, you should consider moving to enhanced networking because you have likely reached the upper thresholds of the VIF driver.

AWS currently supports enhanced networking capabilities using SR-IOV.

SR-IOV provides direct access to network adapters, provides higher performance (packets-per-second) and lower latency.

Must launch an HVM AMI with the appropriate drivers.

Only available for certain instance types.

Only supported in an Amazon VPC.

Elastic Fabric Adapter (EFA)

An Elastic Fabric Adapter is an AWS [Elastic Network Adapter](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking-ena.html) (ENA) with added capabilities.

An EFA can still handle IP traffic, but also supports an important access model commonly called OS bypass.

This model allows the application (most commonly through some user-space middleware) access the network interface without having to get the operating system involved with each message.

Elastic Fabric Adapter (EFA) is a network interface for Amazon EC2 instances that enables customers to run applications requiring high levels of inter-node communications at scale on AWS.

Its custom-built operating system (OS) bypass hardware interface enhances the performance of inter-instance communications, which is critical to scaling these applications.

Common use cases for EFAs include:

High Performance Computing (HPC) applications using the Message Passing Interface (MPI).

Machine Learning (ML) applications using NVIDIA Collective Communications Library (NCCL).

With EFA you get the application performance of on-premises HPC clusters with the on-demand elasticity and flexibility of the AWS cloud.

EFA is available as an optional EC2 networking feature that you can enable on any supported EC2 instance at no additional cost.

ENI vs ENA vs EFA

When to use ENI:

This is the basic adapter type for when you don’t have any high-performance requirements.

Can use with all instance types.

When to use ENA:

Good for use cases that require higher bandwidth and lower inter-instance latency.

Supported for limited instance types (HVM only).

When to use EFA:

High Performance Computing.

MPI and ML use cases.

Tightly coupled applications.

Can use with all instance types.

Placement Groups

Placement groups are a logical grouping of instances in one of the following configurations.

Cluster – clusters instances into a low-latency group in a single AZ:

A cluster placement group is a logical grouping of instances within a single Availability Zone.

Cluster placement groups are recommended for applications that benefit from low network latency, high network throughput, or both, and if most of the network traffic is between the instances in the group.

Spread – spreads instances across underlying hardware (can span AZs):

A spread placement group is a group of instances that are each placed on distinct underlying hardware.

Spread placement groups are recommended for applications that have a small number of critical instances that should be kept separate from each other.

Partition — divides each group into logical segments called partitions:

Amazon EC2 ensures that each partition within a placement group has its own set of racks.

Each rack has its own network and power source. No two partitions within a placement group share the same racks, allowing you to isolate the impact of hardware failure within your application.

Partition placement groups can be used to deploy large distributed and replicated workloads, such as HDFS, HBase, and Cassandra, across distinct racks.

The table below describes some key differences between clustered and spread placement groups:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Clustered | Spread | Partition |
| What | Instances are placed into a low-latency group within a single AZ | Instances are spread across underlying hardware | Instances are grouped into logical segments called partitions which use distinct hardware |
| When | Need low network latency and/or high network throughput | Reduce the risk of simultaneous instance failure if underlying hardware fails | Need control and visibility into instance placement |
| Pros | Get the most out of enhanced networking Instances | Can span multiple AZs | Reduces likelihood of correlated failures for large workloads. |
| Cons | Finite capacity: recommend launching all you might need up front | Maximum of 7 instances running per group, per AZ | Partition placement groups are not supported for Dedicated Hosts |

Launching instances in a spread placement group reduces the risk of simultaneous failures that might occur when instances share the same underlying hardware.

Recommended for applications that benefit from low latency and high bandwidth.

Recommended to use an instance type that supports enhanced networking.

Instances within a placement group can communicate with each other using private or public IP addresses.

Best performance is achieved when using private IP addresses.

Using public IP addresses the performance is limited to 5Gbps or less.

Low-latency 10 Gbps or 25 Gbps network.

Recommended to keep instance types homogenous within a placement group.

Can use reserved instances at an instance level but cannot reserve capacity for the placement group.

The name you specify for a placement group must be unique within your AWS account for the Region.

You can’t merge placement groups.

An instance can be launched in one placement group at a time; it cannot span multiple placement groups.

[On-Demand Capacity Reservation](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-capacity-reservations.html#capacity-reservations-limits) and [zonal Reserved Instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/reserved-instances-scope.html) provide a capacity reservation for EC2 instances in a specific Availability Zone.

The capacity reservation can be used by instances in a placement group. However, it is not possible to explicitly reserve capacity for a placement group.

Instances with a tenancy of host cannot be launched in placement groups.

IAM Roles

IAM roles are more secure than storing access keys and secret access keys on EC2 instances

IAM roles can be used to allow EC2 to interact with several different services like S3, DynamoDB etc.

IAM roles are easier to manage and more secure than access keys.

You can attach an IAM role to an instance at launch time or at any time after by using the AWS CLI, SDK, or the EC2 console.

IAM roles can be attached, modified, or replaced at any time.

Only one IAM role can be attached to an EC2 instance at a time.

IAM roles are universal and can be used in any region.

Bastion/Jump Hosts

You can configure EC2 instances as bastion hosts (aka jump boxes) to access your VPC instances for management.

Can use the SSH or RDP protocols to connect to your bastion host.

Need to configure a security group with the relevant permissions.

Can use auto-assigned public IPs or Elastic IPs.

Can use security groups to restrict the IP addresses/CIDRs that can access the bastion host.

Use auto-scaling groups for HA (set to 1 instance to just replace if it fails).

Best practice is to deploy Linux bastion hosts in two AZs, use auto-scaling and Elastic IP addresses.

You can also use AWS Systems Manager Session Manager instead of using bastion hosts

Monitoring EC2

EC2 status checks are performed every minute, and each returns a pass or a fail status.

If all checks pass, the overall status of the instance is OK.

If one or more checks fail, the overall status is impaired.

System status checks detect (StatusCheckFailed\_System) problems with your instance that require AWS involvement to repair.

The following are examples of problems that can cause system status checks to fail:

Loss of network connectivity.

Loss of system power.

Software issues on the physical host.

Hardware issues on the physical host that impact network reachability.

Instance status checks (StatusCheckFailed\_Instance) detect problems that require your involvement to repair.

The following are examples of problems that can cause instance status checks to fail:

Failed system status checks..

Incorrect networking or startup configuration.

Exhausted memory.

Corrupted file system.

Incompatible kernel.

Status checks are built into Amazon EC2, so they cannot be disabled or deleted.

You can, however, create or delete alarms that are triggered based on the result of the status checks.

You can create Amazon CloudWatch alarms that monitor Amazon EC2 instances and automatically perform an action if the status check fails.

Actions can include:

Recover the instance (only supported on specific instance types and can be used only with StatusCheckFailed\_System).

Stop the instance (only applicable to EBS-backed volumes).

Terminate the instance (cannot terminate if termination protection is enabled).

Reboot the instance.

It is a best practice to use EC2 to reboot an instance rather than restarting through the OS.

CloudWatch Monitoring frequency:

Standard monitoring = 5 mins.

Detailed monitoring = 1 min (chargeable).

Unified CloudWatch Agent

The unified CloudWatch agent enables you to do the following:

Collect more system-level metrics from Amazon EC2 instances across operating systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances. The additional metrics that can be collected are listed in [Metrics Collected by the CloudWatch Agent](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/metrics-collected-by-CloudWatch-agent.html).

Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.

Retrieve custom metrics from your applications or services using the StatsD and collectd protocols. StatsD is supported on both Linux servers and servers running Windows Server. collectd is supported only on Linux servers.

Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.

You can download and install the CloudWatch agent manually using the command line, or you can integrate it with SSM.

Logging and Auditing

Amazon EC2 and Amazon EBS are integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon EC2 and Amazon EBS.

CloudTrail captures all API calls for Amazon EC2 and Amazon EBS as events, including calls from the console and from code calls to the APIs.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon EC2 and Amazon EBS.

A trail enables you to store records indefinitely.

If you don’t configure a trail, you can still view the most recent events in the CloudTrail console by viewing Event history (past 90 days only).

Using the information collected by CloudTrail, you can determine the request that was made to Amazon EC2 and Amazon EBS, the IP address from which the request was made, who made the request, when it was made, and additional details.

Tags

You can assign metadata to your AWS resources in the form of tags.

A tag is a label that you assign to an AWS resource.

Tags are just arbitrary name/value pairs that you can assign to virtually all AWS assets to serve as metadata.

Tags can help you manage, identify, organize, search for, and filter resources.

Each tag consists of a key and an optional value, both of which you define.

Tagging strategies can be used for cost allocation, security, automation, and many other uses. For example, you can use a tag in an IAM policy to implement access control.

Enforcing standardized tagging can be done via AWS Config rules or custom scripts. For example, EC2 instances not properly tagged are stopped or terminated daily.

Most resources can have up to 50 tags.

Resource Groups

Resource groups are mappings of AWS assets defined by tags.

Create custom consoles to consolidate metrics, alarms and config details around given tags.

High Availability Approaches For Compute

Up to date AMIs are critical for rapid fail-over.

AMIs can be copied to other regions for safety or DR staging.

Horizontally scalable architectures are preferred because risk can be spread across multiple smaller machines versus one large machine.

Reserved instances are the only way to guarantee that resources will be available when needed.

Auto Scaling and Elastic Load Balancing work together to provide automated recovery by maintaining minimum instances.

Amazon Route 53 health checks also provide “self-healing” redirection of traffic.

Migration

VM Import Export

VM Import/Export is a tool for migrating VMware, Microsoft, XEN VMs to the Cloud.

Can also be used to convert EC2 instances to VMware, Microsoft, or XEN VMs.

Supported for:

Windows and Linux.

VMware ESX VMDKs and (OVA images for export only).

Citrix XEN VHD.

Microsoft Hyper-V VHD.

Can only be used via the API or CLI (not the console).

Stop the VM before generating VMDK or VHD images.

AWS has a VM connector plugin for vCenter:

Allows migration of VMs to S3.

Then converts into a EC2 AMI.

Progress can be tracked in vCenter.

AWS Server Migration Service (SMS)

AWS Server Migration Service (SMS) is an agent-less service which makes it easier and faster for you to migrate thousands of on-premises workloads to AWS.

AWS SMS allows you to automate, schedule, and track incremental replications of live server volumes, making it easier for you to coordinate large-scale server migrations.

Automates migration of on-premises VMware vSphere or Microsoft Hyper-V/SCVMM virtual machines to AWS.

Replicates VMs to AWS, syncing volumes and creating periodic AMIs.

Minimizes cutover downtime by syncing VMs incrementally.

Supports Windows and Linux VMs only (just like AWS).

The Server Migration Connector is downloaded as a virtual appliance into your on-premises vSphere or Hyper-V environments.

Amazon EC2 instances can be managed through AWS Systems Manager.

You can also use AWS OpsWorks to manage your instances using Chef and Puppet.

AWS Config can be used to record configuration items about Amazon EC2 instances and track changes.

Amazon VPC

Amazon VPC lets you provision a logically isolated section of the Amazon Web Services (AWS) cloud where you can launch AWS resources in a virtual network that you define.

Analogous to having your own DC inside AWS.

Provides complete control over the virtual networking environment including selection of IP ranges, creation of subnets, and configuration of route tables and gateways.

A VPC is logically isolated from other VPCs on AWS.

Possible to connect the corporate data center to a VPC using a hardware VPN (site-to-site).

VPCs are region wide.

A default VPC is created in each region with a subnet in each AZ.

By default you can create up to 5 VPCs per region.

You can define dedicated tenancy for a VPC to ensure instances are launched on dedicated hardware (overrides the configuration specified at launch).

A default VPC is automatically created for each AWS account the first time Amazon EC2 resources are provisioned.

The default VPC has all-public subnets.

Public subnets are subnets that have:

“Auto-assign public IPv4 address” set to “Yes”.

The subnet route table has an attached Internet Gateway.

Instances in the default VPC always have both a public and private IP address.

AZs names are mapped to different zones for different users (i.e. the AZ “ap-southeast-2a” may map to a different physical zone for a different user).

Components of a VPC:

A Virtual Private Cloud: A logically isolated virtual network in the AWS cloud. You define a VPC’s IP address space from ranges you select.

Subnet: A segment of a VPC’s IP address range where you can place groups of isolated resources (maps to an AZ, 1:1).

Internet Gateway: The Amazon VPC side of a connection to the public Internet.

NAT Gateway: A highly available, managed Network Address Translation (NAT) service for your resources in a private subnet to access the Internet.

Hardware VPN Connection: A hardware-based VPN connection between your Amazon VPC and your datacenter, home network, or co-location facility.

Virtual Private Gateway: The Amazon VPC side of a VPN connection.

Customer Gateway: Your side of a VPN connection.

Router: Routers interconnect subnets and direct traffic between Internet gateways, virtual private gateways, NAT gateways, and subnets.

Peering Connection: A peering connection enables you to route traffic via private IP addresses between two peered VPCs.

VPC Endpoints: Enables private connectivity to services hosted in AWS, from within your VPC without using an Internet Gateway, VPN, Network Address Translation (NAT) devices, or firewall proxies.

Egress-only Internet Gateway: A stateful gateway to provide egress only access for IPv6 traffic from the VPC to the Internet.

Options for connecting to a VPC are:

Hardware based VPN.

Direct Connect.

VPN CloudHub.

Software VPN.

Want to learn how to create a VPC hands-on? In the AWS Hands-On Labs video tutorial below, we’ show you how to create a Custom Amazon Virtual Private Cloud (VPC) on AWS. You will also learn how to create subnets, route tables, and Internet Gateways. We launch some EC2 instances into the new VPC and test connectivity.

[](https://www.adplugg.com/track/click/A48221584/60492/click?toi=21584&hn=digitalcloud.training&bu=%2Famazon-vpc%2F&rf=&zn=16423&pm=8399&ct=&next=)

Routing

The VPC router performs routing between AZs within a region.

The VPC router connects different AZs together and connects the VPC to the Internet Gateway.

Each subnet has a route table the router uses to forward traffic within the VPC.

Route tables also have entries to external destinations.

Up to 200 route tables per VPC.

Up to 50 route entries per route table.

Each subnet can only be associated with one route table.

Can assign one route table to multiple subnets.

If no route table is specified a subnet will be assigned to the main route table at creation time.

Cannot delete the main route table.

You can manually set another route table to become the main route table.

There is a default rule that allows all VPC subnets to communicate with one another – this cannot be deleted or modified.

Routing between subnets is always possible because of this rule – any problems communicating is more likely to be security groups or NACLs.

Subnets and Subnet Sizing

Types of subnet:

If a subnet’s traffic is routed to an internet gateway, the subnet is known as a public subnet.

If a subnet doesn’t have a route to the internet gateway, the subnet is known as a private subnet.

If a subnet doesn’t have a route to the internet gateway, but has its traffic routed to a virtual private gateway for a VPN connection, the subnet is known as a VPN-only subnet.

The VPC is created with a master address range (CIDR block, can be anywhere from 16-28 bits), and subnet ranges are created within that range.

New subnets are always associated with the default route table.

Once the VPC is created you cannot change the CIDR block.

You cannot create additional CIDR blocks that overlap with existing CIDR blocks.

You cannot create additional CIDR blocks in a different RFC 1918 range.

Subnets with overlapping IP address ranges cannot be created.

The first 4 and last 1 IP addresses in a subnet are reserved.

Subnets are created within availability zones (AZs).

Each subnet must reside entirely within one Availability Zone and cannot span zones.

Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones.

Availability Zones are connected with low latency, high throughput, and highly redundant networking.

Can create private, public or VPN subnets.

Subnets map 1:1 to AZs and cannot span AZs.

You can only attach one Internet gateway to a custom VPC.

IPv6 addresses are all public and the range is allocated by AWS.

It is recommended these come from the private IP ranges specified in RFC 1918:

10.0.0.0 – 10.255.255.255 (10/8 prefix)

172.16.0.0 – 172.31.255.255 (172.16/12 prefix)

192.168.0.0 – 192.168.255.255 (192.168/16 prefix)

However, it is possible to create a VPC with publicly routable CIDR block.

The allowed block size is between a /28 netmask and /16 netmask.

The CIDR blocks of the subnets within a VPC cannot overlap.

The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use

For example, in a subnet with CIDR block 10.0.0.0/24, the following five IP addresses are reserved:

10.0.0.0: Network address.

10.0.0.1: Reserved by AWS for the VPC route.

10.0.0.2: Reserved by AWS.

10.0.0.3: Reserved by AWS for future use.

10.0.0.255: Network broadcast address (broadcast not supported).

For further information, check out this AWS[article](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html).

Internet Gateways

An Internet Gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the internet.

An Internet Gateway serves two purposes: .

To provide a target in your VPC route tables for internet-routable traffic.

To perform network address translation (NAT) for instances that have been assigned public IPv4 addresses.

Internet Gateways (IGW) must be created and then attached to a VPC, be added to a route table, and then associated with the relevant subnet(s).

No availability risk or bandwidth constraints.

If your subnet is associated with a route to the Internet, then it is a public subnet.

You cannot have multiple Internet Gateways in a VPC.

IGW is horizontally scaled, redundant and HA.

IGW performs NAT between private and public IPv4 addresses.

IGW supports IPv4 and IPv6.

IGWs must be detached before they can be deleted.

Can only attach 1 IGW to a VPC at a time.

Gateway terminology:

Internet gateway (IGW) – AWS VPC side of the connection to the public Internet.

Virtual private gateway (VPG) – VPC endpoint on the AWS side.

Customer gateway (CGW) – representation of the customer end of the connection.

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 (see below).

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 and security group rules allow the relevant traffic to flow to and from your instance.

Must update subnet route table to point to IGW, either:

To all destinations, e.g. 0.0.0.0/0 for IPv4 or ::/0for IPv6.

To specific public IPv4 addresses, e.g. your company’s public endpoints outside of AWS.

Egress-only Internet Gateway:

Provides outbound Internet access for IPv6 addressed instances.

Prevents inbound access to those IPv6 instances.

IPv6 addresses are globally unique and are therefore public by default.

Stateful – forwards traffic from instance to Internet and then sends back the response.

Must create a custom route for ::/0 to the Egress-Only Internet Gateway.

Use Egress-Only Internet Gateway instead of NAT for IPv6.

VPC Wizard

VPC with a Single Public Subnet:

Your instances run in a private, isolated section of the AWS cloud with direct access to the Internet.

Network access control lists and security groups can be used to provide strict control over inbound and outbound network traffic to your instances.

Creates a /16 network with a /24 subnet. Public subnet instances use Elastic IPs or Public IPs to access the Internet.

VPC with Public and Private Subnets:

In addition to containing a public subnet, this configuration adds a private subnet whose instances are not addressable from the Internet.

Instances in the private subnet can establish outbound connections to the Internet via the public subnet using Network Address Translation (NAT).

Creates a /16 network with two /24 subnets.

Public subnet instances use Elastic IPs to access the Internet.

Private subnet instances access the Internet via Network Address Translation (NAT).

VPC with Public and Private Subnets and Hardware VPN Access:

This configuration adds an IPsec Virtual Private Network (VPN) connection between your Amazon VPC and your data center – effectively extending your data center to the cloud while also providing direct access to the Internet for public subnet instances in your Amazon VPC.

Creates a /16 network with two /24 subnets.

One subnet is directly connected to the Internet while the other subnet is connected to your corporate network via an IPsec VPN tunnel.

VPC with a Private Subnet Only and Hardware VPN Access:

Your instances run in a private, isolated section of the AWS cloud with a private subnet whose instances are not addressable from the Internet.

You can connect this private subnet to your corporate data center via an IPsec Virtual Private Network (VPN) tunnel.

Creates a /16 network with a /24 subnet and provisions an IPsec VPN tunnel between your Amazon VPC and your corporate network.

NAT Instances

NAT instances are managed by you.

Used to enable private subnet instances to access the Internet.

NAT instance must live on a public subnet with a route to an Internet Gateway.

Private instances in private subnets must have a route to the NAT instance, usually the default route destination of 0.0.0.0/0.

When creating NAT instances always disable the source/destination check on the instance.

NAT instances must be in a single public subnet.

NAT instances need to be assigned to security groups.

Security groups for NAT instances must allow HTTP/HTTPS inbound from the private subnet and outbound to 0.0.0.0/0.

There needs to be a route from a private subnet to the NAT instance for it to work.

The amount of traffic a NAT instance can support is based on the instance type.

Using a NAT instance can lead to bottlenecks (not HA).

HA can be achieved by using Auto Scaling groups, multiple subnets in different AZ’s and a script to automate failover.

Performance is dependent on instance size.

Can scale up instance size or use enhanced networking.

Can scale out by using multiple NATs in multiple subnets.

Can use as a bastion (jump) host.

Can monitor traffic metrics.

Not supported for IPv6 (use Egress-Only Internet Gateway).

NAT Gateways

NAT gateways are managed for you by AWS.

Fully managed NAT service that replaces the need for NAT instances on EC2.

Must be created in a public subnet.

Uses an Elastic IP address for the public IP.

Private instances in private subnets must have a route to the NAT instance, usually the default route destination of 0.0.0.0/0.

Created in a specified AZ with redundancy in that zone.

For multi-AZ redundancy, create NAT Gateways in each AZ with routes for private subnets to use the local gateway.

Up to 5 Gbps bandwidth that can scale up to 45 Gbps.

Can’t use a NAT Gateway to access VPC peering, VPN or Direct Connect, so be sure to include specific routes to those in your route table.

NAT gateways are highly available in each AZ into which they are deployed.

They are preferred by enterprises.

No need to patch.

Not associated with any security groups.

Automatically assigned a public IP address.

Remember to update route tables and point towards your gateway.

More secure (e.g. you cannot access with SSH and there are no security groups to maintain).

No need to disable source/destination checks.

Egress only Internet gateways operate on IPv6 whereas NAT gateways operate on IPv4.

Port forwarding is not supported.

Using the NAT Gateway as a Bastion host server is not supported.

Traffic metrics are not supported.

The table below highlights the key differences between both types of gateway:

|  |  |  |
| --- | --- | --- |
|  | NAT Gateway | NAT Instance |
| Managed | Managed by AWS | Managed |
| Availability | Highly available within an AZ | Not highly available (would require scripting) |
| Bandwidth | Up to 45 GPS | Depends on the bandwidth of the EC2 instance you selected |
| Maintenance | Managed by AWS | Managed by you |
| Performance | Optimized for NAT | Amazon Linux 2 AMI configured to perform NAT |
| Public IP | Elastic IP cannot be detached | Elastic IP that can be detached |
| Security Groups | Cannot associate with a security group | Can associate with a security group |
| Bastion Host | Not supported | Can be used as a bastion host |

Security Groups

Security groups act like a firewall at the instance level.

Specifically security groups operate at the network interface level.

Can only assign permit rules in a security group, cannot assign deny rules.

There is an implicit deny rule at the end of the security group.

All rules are evaluated until a permit is encountered or continues until the implicit deny.

Can control ingress and egress traffic.

Security groups are stateful.

By default, custom security groups do not have inbound allow rules (all inbound traffic is denied by default).

By default, default security groups do have inbound allow rules (allowing traffic from within the group).

All outbound traffic is allowed by default in custom and default security groups.

You cannot delete the security group that’s created by default within a VPC.

You can use security group names as the source or destination in other security groups.

You can use the security group name as a source in its own inbound rules.

Security group members can be within any AZ or subnet within the VPC.

Security group membership can be changed whilst instances are running.

Any changes made will take effect immediately.

Up to 5 security groups can be added per EC2 instance interface.

There is no limit on the number of EC2 instances within a security group.

You cannot block specific IP addresses using security groups, use NACLs instead.

Network ACL’s

Network ACL’s function at the subnet level.

The VPC router hosts the network ACL function.

With NACLs you can have permit and deny rules.

Network ACLs contain a numbered list of rules that are evaluated in order from the lowest number until the explicit deny.

Recommended to leave spacing between network ACL numbers.

Network ACLs have separate inbound and outbound rules and each rule can allow or deny traffic.

Network ACLs are stateless, so responses are subject to the rules for the direction of traffic.

NACLs only apply to traffic that is ingress or egress to the subnet not to traffic within the subnet.

A VPC automatically comes with a default network ACL which allows all inbound/outbound traffic.

A custom NACL denies all traffic both inbound and outbound by default.

All subnets must be associated with a network ACL.

You can create custom network ACL’s. 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 do this manually it will be associated with the default network ACL.

You can associate a network ACL with multiple subnets; however a subnet can only be associated with one network ACL at a time.

Network ACLs do not filter traffic between instances in the same subnet.

NACLs are the preferred option for blocking specific IPs or ranges.

Security groups cannot be used to block specific ranges of IPs.

NACL is the first line of defense, the security group is the second line.

Also recommended to have software firewalls installed on your instances.

Changes to NACLs take effect immediately.

|  |  |
| --- | --- |
| Security Group | Network ACL |
| Operates at the instance (interface level) | Operates at the subnet level |
| Supports allow rules only | Supports allow and deny rules |
| Stateful | Stateless |
| Evaluates all rules | Processes rules in order |
| Applies to an instance only if associated with a group | Automatically applies to all instances in the subnet it is associated with |

VPC Connectivity

There are several methods of connecting to a VPC. These include:

AWS Managed VPN.

AWS Direct Connect.

AWS Direct Connect plus a VPN.

AWS VPN CloudHub.

Software VPN.

Transit VPC.

VPC Peering.

AWS PrivateLink.

VPC Endpoints.

Each of these will be further detailed below.

AWS Managed VPN

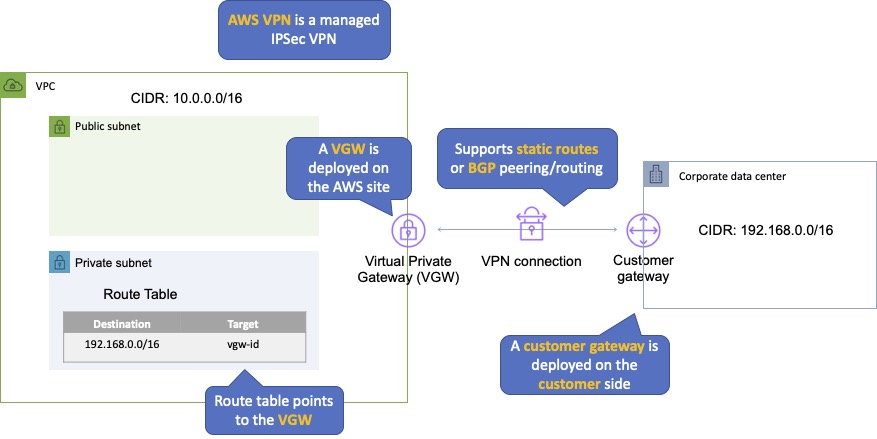
|  |  |
| --- | --- |
| What | AWS-provided network connectivity between two VPCs |
| When | Multiple VPCs need to communicate or access each other’s resources |
| Pros | Uses AWS backbone without traversing the public internet |
| Cons | Transitive peering is not supported |
| How | VPC Peering request made; acceptor accepts request (either within or across accounts) |

VPNs are quick, easy to deploy, and cost effective.

A Virtual Private Gateway (VGW) is required on the AWS side.

A Customer Gateway is required on the customer side.

The diagram below depicts an AWS S2S VPN configuration:



An Internet routable IP address is required on the customer gateway.

Two tunnels per connection must be configured for redundancy.

You cannot use a NAT gateway in AWS for clients coming in via a VPN.

For route propagation you need to point your VPN-only subnet’s route tables at the VGW.

Must define the IP prefixes that can send/receive traffic through the VGW.

VGW does not route traffic destined outside of the received BGP advertisements, static route entries, or its attached VPC CIDR.

Cannot access Elastic IPs on your VPC via the VPN – Elastic IPs can only be connected to via the Internet.

AWS Direct Connect

|  |  |
| --- | --- |
| What | Dedicated network connection over private line straight into the AWS backbone |
| When | Requires a large network link into AWS; lots of resources and services being provided on AWS to your corporate users |
| Pros | More predictable network performance; potential bandwidth cost reduction; up to 10 Gbps provisioned connections; supports BGP peering and routing |
| Cons | May require additional telecom and hosting provider relationships and/or network circuits; costly |
| How | Work with your existing data networking provider; create Virtual Interfaces (VIFs) to connect to VPCs (private VIFs) or other AWS services like S3 or Glacier (public VIFs) |

AWS Direct Connect makes it easy to establish a dedicated connection from an on-premises network to Amazon VPC.

Using AWS Direct Connect, you can establish private connectivity between AWS and your data center, office, or collocated environment.

This private connection can reduce network costs, increase bandwidth throughput, and provide a more consistent network experience than internet-based connections.

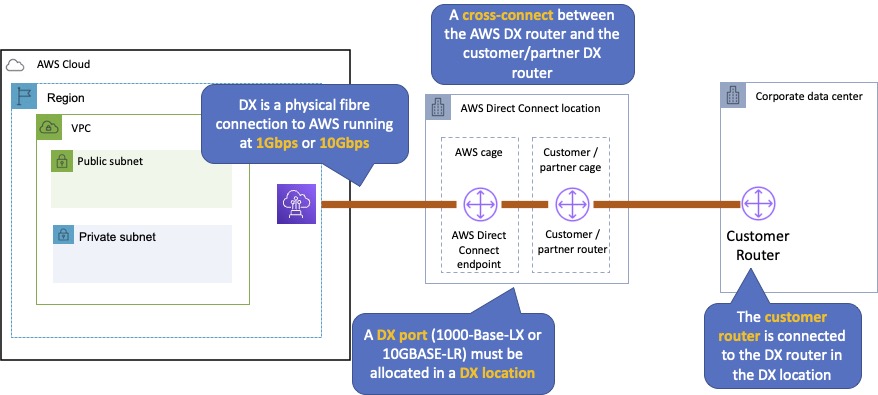
AWS Direct Connect lets you establish 1 Gbps or 10 Gbps dedicated network connections (or multiple connections) between AWS networks and one of the AWS Direct Connect locations.

It uses industry standard VLANs to access Amazon Elastic Compute Cloud (Amazon EC2) instances running within an Amazon VPC using private IP addresses.

AWS Direct Connect does not encrypt your traffic that is in transit.

You can use the encryption options for the services that traverse AWS Direct Connect.

The diagram below depicts an AWS Direct Connect configuration:



AWS Direct Connect Plus VPN

|  |  |
| --- | --- |
| What | IPSec VPN connection over private lines (Direct Connect) |
| When | Need the added security of encrypted tunnels over Direct Connect |
| Pros | More secure (in theory) than Direct Connect alone |
| Cons | More complexity introduced by VPN layer |
| How | Work with your existing data networking provider |

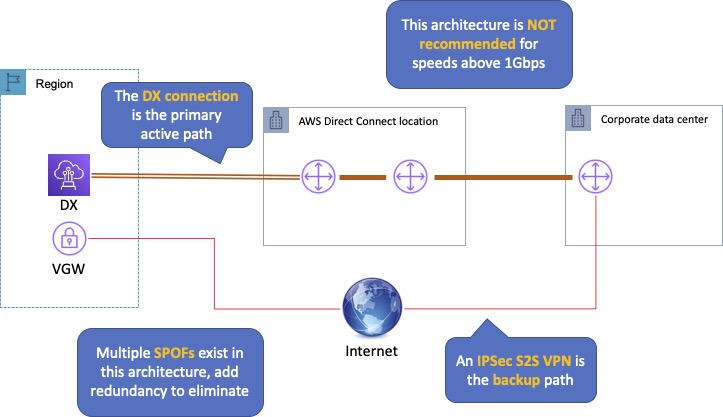
With AWS Direct Connect plus VPN, you can combine one or more AWS Direct Connect dedicated network connections with the Amazon VPC VPN.

This combination provides an IPsec-encrypted private connection that also reduces network costs, increases bandwidth throughput, and provides a more consistent network experience than internet-based VPN connections.

You can use AWS Direct Connect to establish a dedicated network connection between your network create a logical connection to public AWS resources, such as an Amazon virtual private gateway IPsec endpoint.

This solution combines the AWS managed benefits of the VPN solution with low latency, increased bandwidth, more consistent benefits of the AWS Direct Connect solution, and an end-to-end, secure IPsec connection.

The diagram below depicts an AWS Direct Connect plus VPN configuration:



AWS VPN CloudHub

|  |  |
| --- | --- |
| What | Connect locations in a hub and spoke manner using AWSs Virtual Private Gateway |
| When | Link remote offices for backup or primary WAN access to AWS resources and each other |
| Pros | Reuses existing Internet connections; supports BGP routes to direct traffic |
| Cons | Dependent on Internet connections; no inherent redundancy |
| How | Assign multiple Customer Gateways to a Virtual Private Gateway, each with their own BGP ASN and unique IP ranges |

The AWS VPN CloudHub operates on a simple hub-and-spoke model that you can use with or without a VPC.

Use this design if you have multiple branch offices and existing internet connections and would like to implement a convenient, potentially low-cost hub-and-spoke model for primary or backup connectivity between these remote offices.

VPN CloudHub is used for hardware-based VPNs and allows you to configure your branch offices to go into a VPC and then connect that to the corporate DC (hub and spoke topology with AWS as the hub).

Can have up to 10 IPSec tunnels on a VGW by default.

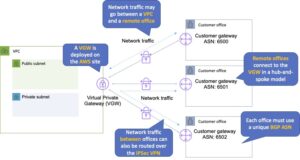
Uses eBGP.

Branches can talk to each other (and provides redundancy).

Can have Direct Connect connections.

Hourly rates plus data egress charges.

The diagram below depicts an AWS VPN CloudHub configuration:



Software VPN

|  |  |
| --- | --- |
| What | You must provide your own endpoint and software |
| When | You must manage both ends of the VPN connection for compliance reasons or you want to use a VPN option not supported by AWS |
| Pros | Ultimate flexibility and manageability |
| Cons | You must design for any needed redundancy across the whole chain |
| How | Install VPN software via Marketplace on an EC2 instance |

Amazon VPC offers you the flexibility to fully manage both sides of your Amazon VPC connectivity by creating a VPN connection between your remote network and a software VPN appliance running in your Amazon VPC network.

This option is recommended if you must manage both ends of the VPN connection either for compliance purposes or for leveraging gateway devices that are not currently supported by Amazon VPC’s VPN solution.

Transit VPC

|  |  |
| --- | --- |
| What | Common strategy for connecting geographically dispersed VPCs and locations to create a global network transit center |
| When | Locations and VPC-deployed assets across multiple regions that need to communicate with one another |
| Pros | Ultimate flexibility and manageability but also AWS-managed VPN hub-and-spoke between VPCs |
| Cons | You must design for any redundancy across the whole chain |
| How | Providers like Cisco, Juniper Networks, and Riverbed have offerings which work with their equipment and AWS VPC |

Building on the Software VPN design mentioned above, you can create a global transit network on AWS.

A transit VPC is a common strategy for connecting multiple, geographically disperse VPCs and remote networks to create a global network transit center.

A transit VPC simplifies network management and minimizes the number of connections required to connect multiple VPCs and remote networks.

VPC Peering

|  |  |
| --- | --- |
| What | AWS-provided network connectivity between VPCs |
| When | Multiple VPCs need to connect with one another and access their resources |
| Pros | Uses AWS backbone without traversing the internet |
| Cons | Transitive peering is not supported |
| How | VPC Peering request made; accepter request (either within or across accounts) |

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 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, or with a VPC in another AWS account.

The VPCs can be in different regions (also known as an inter-region VPC peering connection).

Data sent between VPCs in different regions is encrypted (traffic charges apply).

For inter-region VPC peering there are some limitations:

You cannot create a security group rule that references a peer security group.

Cannot enable DNS resolution.

Maximum MTU is 1500 bytes (no jumbo frames support).

Limited region support.

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.

A VPC peering connection helps you to facilitate the transfer of data.

Can only have one peering connection between any two VPCs at a time.

Can peer with other accounts (within or between regions).

Cannot have overlapping CIDR ranges.

A VPC peering connection is a one-to-one relationship between two VPCs.

You can create multiple VPC peering connections for each VPC that you own, but transitive peering relationships are not supported.

You do not have any peering relationship with VPCs that your VPC is not directly peered with.

Limits are 50 VPC peers per VPC, up to 125 by request.

DNS is supported.

Must update route tables to configure routing.

Must update the inbound and outbound rules for VPC security group to reference security groups in the peered VPC.

When creating a VPC peering connection with another account you need to enter the account ID and VPC ID from the other account.

Need to accept the pending access request in the peered VPC.

The VPC peering connection can be added to route tables – shows as a target starting with “pcx-“.

AWS PrivateLink

AWS PrivateLink simplifies the security of data shared with cloud-based applications by eliminating the exposure of data to the public Internet.

AWS PrivateLink provides private connectivity between VPCs, AWS services, and on-premises applications, securely on the Amazon network.

AWS PrivateLink makes it easy to connect services across different accounts and VPCs to significantly simplify the network architecture.

The table below provides more information on AWS PrivateLink and when to use it:

|  |  |
| --- | --- |
| What | AWS-provided connectivity between VPCs and/or AWS services using interface endpoints |
| When | Keep private subnets truly private by using the AWS backbone rather than using the public internet |
| Pros | Redundant; uses AWS backbone |
| Cons |  |
| How | Create endpoint for required AWS or Marketplace service in all required subnets; access via the provided DNS hostname |

EXAM TIP: Know the difference between AWS PrivateLink and ClassicLink. ClassicLink allows you to link EC2-Classic instances to a VPC in your account, within the same region. EC2-Classic is an old platform from before VPCs were introduced and is not available to accounts created after December 2013. However, ClassicLink may come up in exam questions as a possible (incorrect) answer, so you need to know what it is.

VPC Endpoints

An Interface endpoint uses AWS PrivateLink and is an elastic network interface (ENI) with a private IP address that serves as an entry point for traffic destined to a supported service.

Using PrivateLink you can connect your VPC to supported AWS services, services hosted by other AWS accounts (VPC endpoint services) and supported AWS Marketplace partner services.

AWS PrivateLink access over Inter-Region VPC Peering:

Applications in an AWS VPC can securely access AWS PrivateLink endpoints across AWS Regions using Inter-Region VPC Peering.

AWS PrivateLink allows you to privately access services hosted on AWS in a highly available and scalable manner, without using public IPs, and without requiring the traffic to traverse the Internet.

Customers can privately connect to a service even if the service endpoint resides in a different AWS Region.

Traffic using Inter-Region VPC Peering stays on the global AWS backbone and never traverses the public Internet.

A gateway endpoint is a gateway that is a target for a specified route in your route table, used for traffic destined to a supported AWS service.

An interface VPC endpoint (interface endpoint) enables you to connect to services powered by AWS PrivateLink.

The table below highlights some key information about both types of endpoints:

|  |  |  |
| --- | --- | --- |
|  | Interface Endpoint | Gateway Endpoint |
| What | Elastic Network Interface with a private IP | A gateway that is a target for a specific route |
| How | Uses DNS entries to redirect traffic | Use prefix lists in the route table to redirect traffic |
| Which Services | A large amount of AWS services, for full list follow this [link](https://docs.aws.amazon.com/vpc/latest/privatelink/integrated-services-vpce-list.html) | Amazon S3, DynamoDB |
| Security | Security Groups | VPC Endpoint Policies |

By default, IAM users do not have permission to work with endpoints.

You can create an IAM user policy that grants users the permissions to create, modify, describe, and delete endpoints.

There’s a[long list](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html) of services that are supported by interface endpoints.

Gateway endpoints are only available for:

Amazon DynamoDB

Amazon S3

EXAM TIP: Know which services use interface endpoints and gateway endpoints. The easiest way to remember this is that Gateway Endpoints are for Amazon S3 and DynamoDB only.

Shared Services VPCs

You can allow other AWS accounts to create their application resources, such as EC2 instances, Relational Database Service (RDS) databases, Redshift clusters, and Lambda functions, into shared, centrally managed Amazon Virtual Private Clouds (VPCs).

VPC sharing enables subnets to be shared with other AWS accounts within the same AWS Organization. Benefits include:

Separation of duties: centrally controlled VPC structure, routing, IP address allocation.

Application owners continue to own resources, accounts, and security groups.

VPC sharing participants can reference security group IDs of each other.

Efficiencies: higher density in subnets, efficient use of VPNs and AWS Direct Connect.

Hard limits can be avoided, for example, 50 VIFs per AWS Direct Connect connection through simplified network architecture.

Costs can be optimized through reuse of NAT gateways, VPC interface endpoints, and intra-Availability Zone traffic.

You can create separate Amazon VPCs for each account with the account owner being responsible for connectivity and security of each Amazon VPC.

With VPC sharing, your IT team can own and manage your Amazon VPCs and your application developers no longer must manage or configure Amazon VPCs, but they can access them as needed.

Can also share Amazon VPCs to leverage the implicit routing within a VPC for applications that require a high degree of interconnectivity and are within the same trust boundaries.

This reduces the number of VPCs that need to be created and managed, while you still benefit from using separate accounts for billing and access control.

Customers can further simplify network topologies by interconnecting shared Amazon VPCs using connectivity features, such as AWS PrivateLink, AWS Transit Gateway, and Amazon VPC peering.

Can also be used with AWS PrivateLink to secure access to resources shared such as applications behind a Network Load Balancer.

VPC Flow Logs

Flow Logs capture information about the IP traffic going to and from network interfaces in a VPC.

Flow log data is stored using Amazon CloudWatch Logs.

Flow logs can be created at the following levels:

VPC.

Subnet.

Network interface.

You can’t enable flow logs for VPC’s that are peered with your VPC unless the peer VPC is in your account.

You can’t tag a flow log.

You can’t change the configuration of a flow log after it’s been created.

After you’ve created a flow log, you cannot change its configuration (you need to delete and re-create).

Not all traffic is monitored, e.g. the following traffic is excluded:

Traffic that goes to Route53.

Traffic generated for Windows license activation.

Traffic to and from 169.254.169.254 (instance metadata).

Traffic to and from 169.254.169.123 for the Amazon Time Sync Service.

DHCP traffic.

Traffic to the reserved IP address for the default VPC router.

High Availability Approaches for Networking

By creating subnets in the available AZs, you create Multi-AZ presence for your VPC.

Best practice is to create at least two VPN tunnels into your Virtual Private Gateway.

Direct Connect is not HA by default, so you need to establish a secondary connection via another Direct Connect (ideally with another provider) or use a VPN.

Route 53’s health checks provide a basic level of redirecting DNS resolutions.

Elastic IPs allow you flexibility to change out backing assets without impacting name resolution.

For Multi-AZ redundancy of NAT Gateways, create gateways in each AZ with routes for private subnets to use the local gateway.

AWS Elastic Load Balancing (AWS ELB)

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses.

Network traffic can be distributed across a single or multiple Availability Zones (AZs) within an AWS Region.

There are four types of Elastic Load Balancer (ELB) on AWS:

Classic Load Balancer (CLB) – this is the oldest of the three and provides basic load balancing at both layer 4 and layer 7.

Application Load Balancer (ALB) – layer 7 load balancer that routes connections based on the content of the request.

Network Load Balancer (NLB) – layer 4 load balancer that routes connections based on IP protocol data.

Gateway Load Balancer (GLB) – layer 3/4 load balancer used in front of virtual appliances such as firewalls and IDS/IPS systems.

Note: The CLB is not covered in detail on this page as it is on old generation load balancer and is no longer featured on most AWS exams.

The following table provides a comparison of some of the key features relevant to AWS exams:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Application Load Balancer | Network Load Balancer | Classic Load Balancer | Gateway Load Balancer |
| OSI Layer | Layer 7 | Layer 4 | Layer 4/7 | Layer 3 Gateway + Layer 4 Load Balancing |
| Target Type | IP, Instance, Lambda | IP, Instance, ALB | N/A | IP, Instance |
| Protocols | HTTP, HTTPS | TCP | TCP, SSL, HTTP, HTTPS | IP |
| WebSockets |  |  |  |  |
| IP addresses as a target |  |  |  |  |
| HTTP header-based routing |  |  |  |  |
| HTTP/2/gRPC |  |  |  |  |
| Configurable idle connection timeout |  |  |  |  |
| Cross-zone load balancing |  |  |  |  |
| SSL Offloading |  |  |  |  |
| Server Name Indication (SNI) |  |  |  |  |
| Sticky sessions |  |  |  |  |
| Static / Elastic IP Address |  |  |  |  |
| Custom Security policies |  |  |  |  |

Elastic Load Balancing provides fault tolerance for applications by automatically balancing traffic across targets –

Targets can be Amazon EC2 instances, containers, IP addresses, and Lambda functions.

ELB distributes traffic across Availability Zones while ensuring only healthy targets receive traffic.

Only 1 subnet per AZ can be enabled for each ELB.

Amazon Route 53 can be used for region load balancing with ELB instances configured in each region.

ELBs can be Internet facing or internal-only.

Internet facing ELB:

ELB nodes have public IPs.

Routes traffic to the private IP addresses of the EC2 instances.

Need one public subnet in each AZ where the ELB is defined.

ELB DNS name format: <name>-<id-number>.<region>.elb.amazonaws.com.

Internal only ELB:

ELB nodes have private IPs.

Routes traffic to the private IP addresses of the EC2 instances.

ELB DNS name format: internal-<name>-<id-number>.<region>.elb.amazonaws.com.

Internal-only load balancers do not need an Internet gateway.

EC2 instances and containers can be registered against an ELB.

ELB nodes use IP addresses within your subnets, ensure at least a /27 subnet and make sure there are at least 8 IP addresses available for the ELB to scale.

An ELB forwards traffic to eth0 (primary IP address).

An ELB listener is the process that checks for connection requests:

CLB listeners support the TCP and HTTP/HTTPS protocols.

ALB listeners support the HTTP and HTTPS protocols.

NLB listeners support the TCP, UDP and TLS protocols.

GLB listeners support the IP protocol.

Deleting an ELB does not affect the instances registered against it (they won’t be deleted; they just won’t receive any more requests).

For ALB at least 2 subnets must be specified.

For NLB only one subnet must be specified (recommended to add at least 2).

ELB uses a DNS record TTL of 60 seconds to ensure new ELB node IP addresses are used to service clients.

By default the ELB has an idle connection timeout of 60 seconds, set the idle timeout for applications to at least 60 seconds.

Perfect Forward Secrecy (PFS) provides additional safeguards against the eavesdropping of encrypted data, using a unique random session key.

Server Order Preference lets you configure the load balancer to enforce cipher ordering, providing more control over the level of security used by clients to connect with your load balancer.

ELB does not support client certificate authentication (API Gateway does support this).

[](https://www.adplugg.com/track/click/A48221584/60496/click?toi=21584&hn=digitalcloud.training&bu=%2Faws-elastic-load-balancing-aws-elb%2F&rf=&zn=16428&pm=8401&ct=&next=)

ELB Security Groups

Security groups control the ports and protocols that can reach the front-end listener.

In non-default VPCs you can choose which security group to assign.

You must assign a security group for the ports and protocols on the front-end listener.

You need to also allow the ports and protocols for the health check ports and back-end listeners.

Security group configuration for ELB:

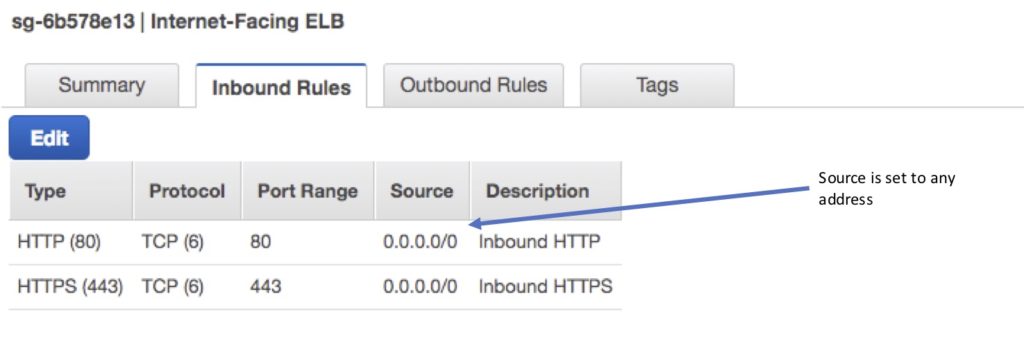
Inbound to ELB (allow).

Internet-facing ELB:

Source: 0.0.0.0/0.

Protocol: TCP.

Port: ELB listener ports.



Internal-only ELB:

Source: VPC CIDR.

Protocol: TCP.

Port: ELB Listener ports.

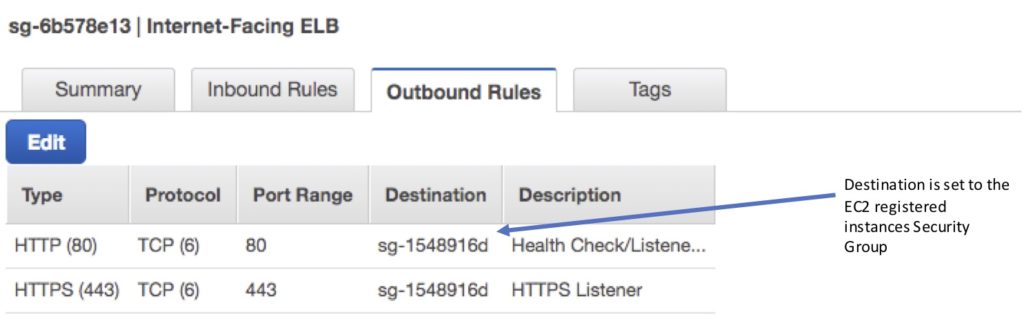


Outbound (allow, either type of ELB):

Destination: EC2 registered instances security group.

Protocol: TCP.

Port: Health Check/Listener.



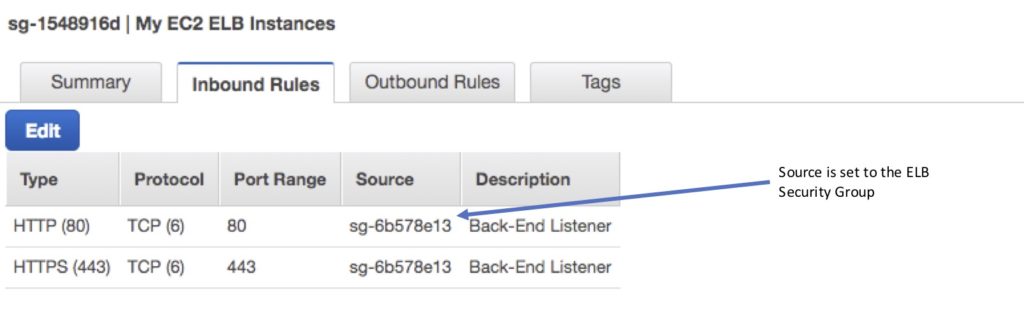
Security group configuration for registered instances:

Inbound to registered instances (Allow, either type of ELB).

Source: ELB Security Group.

Protocol: TCP.

Port: Health Check/Listener.

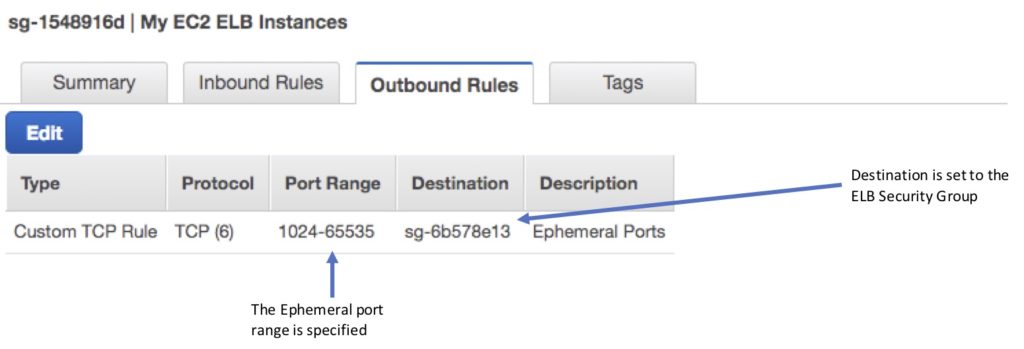


Outbound (Allow, for both types of ELB).

Destination: ELB Security Group.

Protocol: TCP.

Port: Ephemeral.



It is also important to ensure NACL settings are set correctly.

Distributed Denial of Service (DDoS) protection:

ELB automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses, and multiple Availability Zones, which minimizes the risk of overloading a single resource.

ELB only supports valid TCP requests so DDoS attacks such as UDP and SYN floods are not able to reach EC2 instances.

ELB also offers a single point of management and can serve as a line of defense between the internet and your backend, private EC2 instances.

You can also attach AWS Web Application Firewall (WAF) Web ACLs to Application Load Balancers to protect against web exploits.

ELB Monitoring

Monitoring takes place using:

CloudWatch – every 1 minute.

ELB service only sends information when requests are active.

Can be used to trigger SNS notifications.

Access Logs.

Disabled by default.

Includes information about the clients (not included in CloudWatch metrics).

Can identify requester, IP, request type etc.

Can be optionally stored and retained in S3.

CloudTrail.

Can be used to capture API calls to the ELB.

Can be stored in an S3 bucket.

Target groups

Target groups are a logical grouping of targets and are used with ALB, NLB, and GLB.

Targets are the endpoints and can be EC2 instances, ECS containers, IP addresses, Lambda functions, and other load balancers.

Target groups can exist independently from the ELB.

A single target can be in multiple target groups.

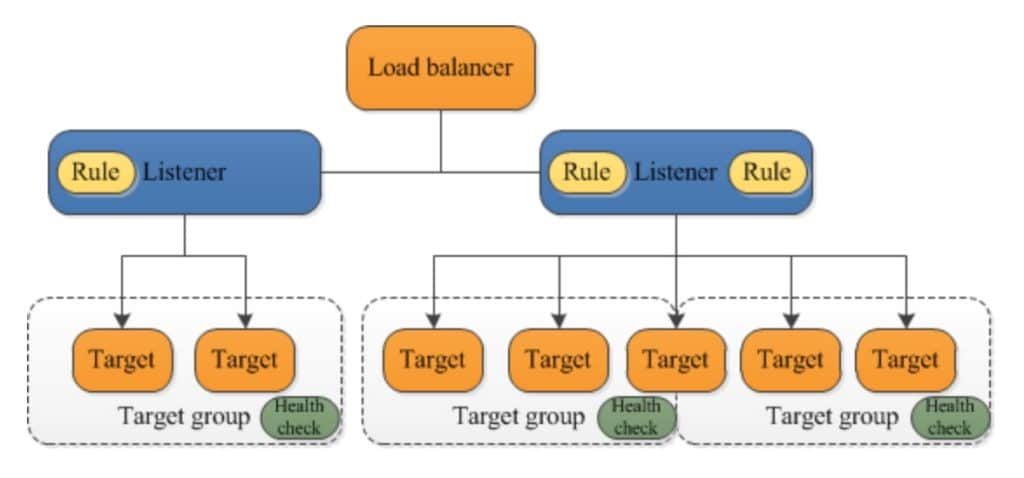
Only one protocol and one port can be defined per target group.

You cannot use public IP addresses as targets.

You cannot use instance IDs and IP address targets within the same target group.

A target group can only be associated with one load balancer.

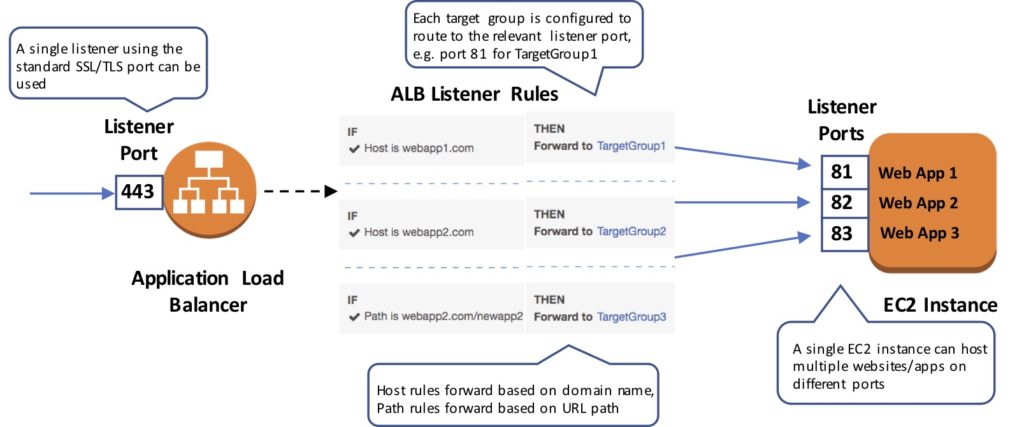
The following diagram illustrates the basic components. Notice that each listener contains a default rule, and one listener contains another rule that routes requests to a different target group. One target is registered with two target groups.



Target groups are used for registering instances against an ALB, NLB, or GLB.

Target groups are a regional construct (as are ELBs).

The following diagram shows how target groups can be used with an ALB using host-based and target-based routing to route traffic to multiple websites, running on multiple ports, on a single EC2 instance:



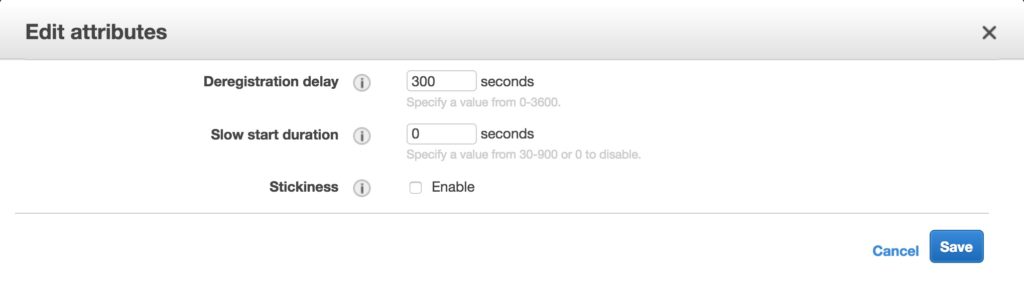
The following attributes can be defined:

Deregistration delay – the amount of time for Elastic Load Balancing to wait before deregistering a target.

Slow start duration – the time, in seconds, during which the load balancer sends a newly registered target a linearly increasing share of the traffic to the target group.

Stickiness – indicates whether sticky sessions are enabled.

The default settings for attributes are shown below:



Auto Scaling groups can scale each target group individually.

You can only use Auto Scaling with the load balancer if using instance IDs in your target group.

Health checks are defined per target group.

ALB/NLB/GLB can route to multiple target groups.

You can register the same EC2 instance or IP address with the same target group multiple times using different ports (used for routing requests to microservices).

If you register by instance ID the traffic is routed using the primary private IP address of the primary network interface.

If you register by IP address you can route traffic to an instance using any private address from one or more network interfaces.

You cannot mix different types within a target group (EC2, ECS, IP, Lambda function).

IP addresses can be used to register:

Instances in a peered VPC.

AWS resources that are addressable by IP address and port.

On-premises resources linked to AWS through Direct Connect or a VPN connection.

Application Load Balancer (ALB)

The Application Load Balancer operates at the request level (layer 7), routing traffic to targets – EC2 instances, containers and IP addresses based on the content of the request.

You can load balance HTTP/HTTPS applications and use layer 7-specific features, such as X-Forwarded-For headers.

The ALB supports HTTPS termination between the clients and the load balancer.

The ALB supports management of SSL certificates through AWS IAM and AWS Certificate Manager for predefined security policies.

The ALB supports Server Name Indication (SNI) which allows multiple secure websites to use a single secure listener.

With Server Name Indication (SNI) a client indicates the hostname to which it wants to connect.

IP addresses can be configured as targets which allows load balancing to applications hosted in AWS or on-premises using the IP addresses of the back-end instances/servers as targets.

You need at least two Availability Zones, and you can distribute incoming traffic across your targets in multiple Availability Zones.

The ALB automatically scales its request handling capacity in response to incoming application traffic.

You can configure an ALB to be Internet facing or create a load balancer without public IP addresses to serve as an internal (non-Internet-facing) load balancer.

The ALB supports content-based routing which allows the routing of requests to a service based on the content of the request. For example:

Host-based routing routes client requests based on the Host field of the HTTP header allowing you to route to multiple domains from the same load balancer.

Path-based routing routes a client request based on the URL path of the HTTP header (e.g. /images or /orders).

Support for microservices and containers with load balancing across multiple ports on a single EC2 instance.

Integration with Amazon Cognito for user authentication.

The slow start mode allows targets to “warm up” with a ramp-up period.

Health Checks:

Can have custom response codes in health checks (200-399).

Details provided in the API and management console for health check failures.

Reason codes are returned with failed health checks.

Health checks do not support WebSockets.

Fail open means that if no AZ contains a healthy target the load balancer nodes route requests to all targets.

Detailed access log information is provided and saved to an S3 bucket every 5 or 6 minutes. Deletion protection is possible. Deregistration delay is like connection draining.

Sticky Sessions:

Uses cookies to ensure a client is bound to an individual back-end instance for the duration of the cookie lifetime.

ALB supports duration-based cookies and application-based cookies.

For application-based cookies the cookie name is specified for each target group.

For duration-based cookies the name of the cookie is always AWSALB.

Sticky sessions are enabled at the target group level.

WebSockets connections are inherently sticky (following the upgrade process).

Listeners and Rules

Listeners:

Each ALB needs at least one listener and can have up to 10.

Listeners define the port and protocol to listen on.

Can add one or more listeners.

Cannot have the same port in multiple listeners.

Listener rules:

Rules determine how the load balancer routes requests to the targets in one or more target groups.

Each rule consists of a priority, one or more actions, an optional host condition, and an optional path condition.

Only one action can be configured per rule.

One or more rules are required.

Each listener has a default rule, and you can optionally define additional rules.

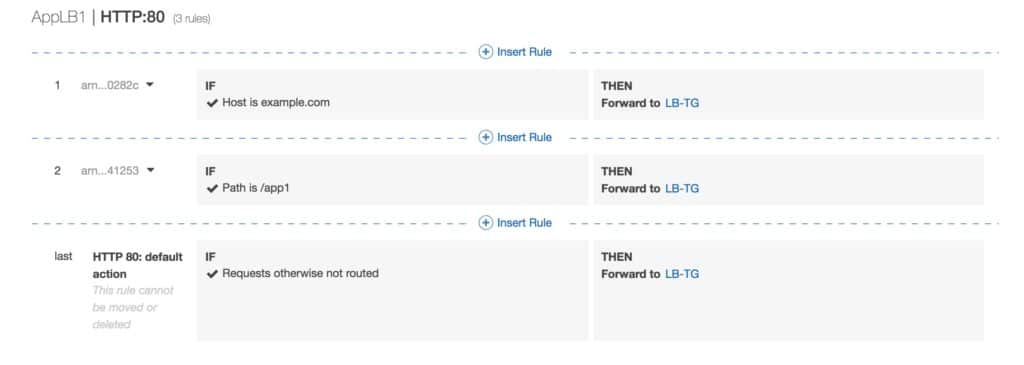
Rules determine what action is taken when the rule matches the client request.

Rules are defined on listeners.

You can add rules that specify different target groups based on the content of the request (content-based routing).

If no rules are found the default rule will be followed which directs traffic to the default target groups.

The image below shows a ruleset with a host-based and path-based entry and a default rule at the end:



Default rules:

When you create a listener you define an action for the default rule.

Default rules cannot have conditions.

You can delete the non-default rules for a listener at any time.

You cannot delete the default rule for a listener.

When you delete a listener all its rules are deleted.

If no conditions for any of a listener’s rules are met, the action for the default rule is taken.

Rule priority:

Each rule has a priority, and they are evaluated in order of lowest to highest.

The default rule is evaluated last.

You can change the value of a non-default rule at any time.

You cannot change the value of the default rule.

Rule action:

Only one target group per action.

Each rule has a type and a target group.

The only supported action type is forward, which forwards requests to the target group.

You can change the target group for a rule at any time.

Rule conditions:

There are two types of rule condition: host and path.

When the conditions for a rule are met the action is taken.

Each rule can have up to 2 conditions, 1 path condition and 1 host condition.

Optional condition is the path pattern you want the ALB to evaluate for it to route requests.

Request routing:

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 using the round robin routing algorithm.

Routing is performed independently for each target group even when a target is registered with multiple target groups.

You can configure listener rules to route requests to different target groups based on the content of the application traffic.

Content-based routing:

ALB can route requests based on the content of the request in the host field: host-based or path-based.

Host-based is domain name-based routing e.g. example.com or app1.example.com.

The host field contains the domain name and optionally the port number.

Path-based is URL based routing e.g. example.com/images, example.com/app1.

You can also create rules that combine host-based and path-based routing.

Anything that doesn’t match content routing rules will be sent to a default target group.

The ALB can also route based on [other information](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html) in the HTTP header including query string parameters, request method, and source IP addresses

ALB and ECS

ECS service maintains the “desired count” of instances.

Optionally a load balancer can distribute traffic across tasks.

All containers in a single task definition are placed on a single EC2 container instance.

ECS service can only use a single load balancer.

ALB does not support multiple listeners in a single task definition.

ALB supports dynamic host-port mapping which means that multiple ports from the same service are allowed on the same container host.

ALB supports path-based routing and priority rules.

The ALB integrates with the EC2 container service using service load balancing.

Federated authentication:

ALB supports authentication from OIDC compliant identity providers such as Google, Facebook, and Amazon.

Implemented through an authentication action on a listener rule that integrates with Amazon Cognito to create user pools.

AWS SAM can also be used with Amazon Cognito.

Network Load Balancer

Network Load Balancer operates at the connection level (Layer 4), routing connections to targets – Amazon EC2 instances, containers and IP addresses based on IP protocol data.

The NLB is designed to handle millions of requests/sec and to support sudden volatile traffic patterns at extremely low latencies.

The NLB can be configured with a single static/Elastic IP address for each Availability Zone.

You can load balance any application hosted in AWS or on-premises using IP addresses of the application back-ends as targets.

The NLB supports connections from clients to IP-based targets in peered VPCs across different AWS Regions.

NLB supports both network and application target health checks.

NLB supports long-running/lived connections (ideal for WebSocket applications).

NLB supports failover between IP addresses within and across AWS Regions (uses Amazon Route 53 health checks).

The integration with Amazon Route 53 enables the removal of a failed load balancer IP address from service and subsequent redirection of traffic to an alternate NLB in another region.

NLB support cross-zone load balancing, but it is not enabled by default when the NLB is created through the console.

Target groups for NLBs support the following protocols and ports:

Protocols: TCP, TLS, UDP, TCP\_UDP.

Ports: 1-65535.

The table below summarizes the supported listener and protocol combinations and target group settings:

|  |  |  |  |
| --- | --- | --- | --- |
| Listener Protocol | Target Group Protocol | Target Group Type | Health Check Protocol |
| TCP | TCP | TCP\_UDP | instance | ip | HTTP | HTTPS | TCP |
| TLS | TCP | TLS | Instance | ip | HTTP | HTTPS | TCP |
| UDP | UDP | TCP\_UDP | instance | HTTP | HTTPS | TCP |
| TCP\_UDP | TCP\_UDP | instance | HTTP | HTTPS | TCP |

Amazon CloudWatch reports Network Load Balancer metrics.

You can use VPC Flow Logs to record all requests sent to your load balancer.

Monitoring

AWS CloudTrail captures API calls for auditing.

You only pay for the S3 storage charges.

CloudTrail only monitors API calls.

Access logs can be used to monitor other actions such as the time the request was received, the client’s IP address, request paths etc.

Access logging is optional and disabled by default.

You are only charged for the S3 storage with access logging.

With access logging, the ALB logs requests sent to the load balancer including requests that never reached targets.

The ALB does not log health check requests.

Amazon EC2 Auto Scaling

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

AWS Auto Scaling refers to a collection of Auto Scaling capabilities across several AWS services.

The services within the AWS Auto Scaling family include:

Amazon EC2 (known as Amazon EC2 Auto Scaling).

Amazon ECS.

Amazon DynamoDB.

Amazon Aurora.

This page is specifically for Amazon EC2 Auto Scaling – Auto Scaling will also be discussed for the other services on their respective pages.

Amazon EC2 Auto Scaling Features

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.

You create collections of EC2 instances, called Auto Scaling groups.

Automatically provides horizontal scaling (scale-out) for your instances.

Triggered by an event of scaling action to either launch or terminate instances.

Availability, cost, and system metrics can all factor into scaling.

Auto Scaling is a region-specific service.

Auto Scaling can span multiple AZs within the same AWS region.

Auto Scaling can be configured from the Console, CLI, SDKs and APIs.

There is no additional cost for Auto Scaling, you just pay for the resources (EC2 instances) provisioned.

Auto Scaling works with ELB, CloudWatch and CloudTrail.

You can determine which subnets Auto Scaling will launch new instances into.

Auto Scaling will try to distribute EC2 instances evenly across AZs.

Launch configuration is the template used to create new EC2 instances and includes parameters such as instance family, instance type, AMI, key pair, and security groups.

You cannot edit a launch configuration once defined.

A launch configuration:

Can be created from the AWS console or CLI.

You can create a new launch configuration, or.

You can use an existing running EC2 instance to create the launch configuration.

The AMI must exist on EC2.

EC2 instance tags and any additional block store volumes created after the instance launch will not be considered.

If you want to change your launch configurations you have to create a new one, make the required changes, and use that with your auto scaling groups.

You can use a launch configuration with multiple Auto Scaling Groups (ASG).

Launch templates are similar to launch configurations and offer more options (more below).

An Auto Scaling Group (ASG) is a logical grouping of EC2 instances managed by an Auto Scaling Policy.

An ASG can be edited once defined.

You can attach one or more classic ELBs to your existing ASG.

You can attach one or more Target Groups to your ASG to include instances behind an ALB.

The ELBs must be in the same region.

Once you do this any EC2 instance existing or added by the ASG will be automatically registered with the ASG defined ELBs.

If adding an instance to an ASG would result in exceeding the maximum capacity of the ASG the request will fail.

You can add a running instance to an ASG if the following conditions are met:

The instance is in a running state.

The AMI used to launch the instance still exists.

The instance is not part of another ASG.

The instance is in the same AZs for the ASG.

Scaling Options

The scaling options define the triggers and when instances should be provisioned/de-provisioned.

There are four scaling options:

Maintain – keep a specific or minimum number of instances running.

Manual – use maximum, minimum, or a specific number of instances.

Scheduled – increase or decrease the number of instances based on a schedule.

Dynamic – scale based on real-time system metrics (e.g. CloudWatch metrics).

Predictive – machine learning to schedule the right number of EC2 instances in anticipation of approaching traffic changes.

The following table describes the scaling options available and when to use them:

|  |  |  |
| --- | --- | --- |
| Scaling | Description | When to use |
| Maintain | Ensures the required number of instances are running | Use when you always need a known number of instances running at all times |
| Manual | Manually change desired capacity | Use when your needs change rarely enough that you’re ok the make manual changes |
| Scheduled | Adjust min/max on specific dates/times or recurring time periods | Use when you know when your busy and quiet times are. Useful for ensuring enough instances are available before very busy times |
| Dynamic | Scale in response to system load or other triggers using metrics | Useful for changing capacity based on system utilization, e.g. CPU hits 80%. |
| Predictive | predict capacity required ahead of time using ML | Useful for when capacity, and number of instances is unknown. |

Scheduled Scaling

Scaling based on a schedule allows you to scale your application ahead of predictable load changes.

For example, you may know that traffic to your application is highest between 9am and 12pm Monday-Friday.

Dynamic Scaling

Amazon EC2 Auto Scaling enables you to follow the demand curve for your applications closely, reducing the need to manually provision Amazon EC2 capacity in advance.

For example, you can track the CPU utilization of your EC2 instances or the “Request Count Per Target” to track the number of requests coming through an Application Load Balancer.

Amazon EC2 Auto Scaling will then automatically adjust the number of EC2 instances as needed to maintain your target.

Predictive Scaling

Predictive Scaling uses machine learning to schedule the optimum number of EC2 instances in anticipation of upcoming traffic changes.

Predictive Scaling predicts future traffic, including regularly occurring spikes, and provisions the right number of EC2 instances in advance.

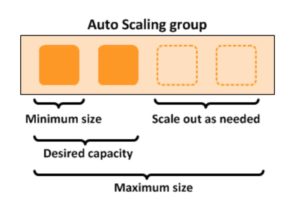
Predictive Scaling uses machine learning algorithms to detect changes in daily and weekly patterns and then automatically adjust forecasts.

You can configure the scaling options through Scaling Policies which determine when, if, and how the ASG scales out and in.

The following table describes the scaling policy types available for dynamic scaling policies and when to use them (more detail further down the page):

|  |  |  |
| --- | --- | --- |
| Scaling Policy | What it is | When to use |
| Target Tracking Policy | Adds or removes capacity as required to keep the metric at or close to the specific target value. | You want to keep the CPU usage of your ASG at 70% |
| Simple Scaling Policy | Waits for the health check and cool down periods to expire before re-evaluating | Useful when load is erratic. AWS recommends step scaling instead of simple in most cases. |
| Step Scaling Policy | Increases or decreases the configured capacity of the Auto Scaling group based on a set of scaling adjustments, known as step adjustments. | You want to vary adjustments based on the size of the alarm breach |

The diagram below depicts an Auto Scaling group with a Scaling policy set to a minimum size of 1 instance, a desired capacity of 2 instances, and a maximum size of 4 instances:



Scaling based on Amazon SQS

Can also scale based on an Amazon Simple Queue Service (SQS) queue.

This comes up as an exam question for SAA-C02.

Uses a custom metric that’s sent to Amazon CloudWatch that measures the number of messages in the queue per EC2 instance in the Auto Scaling group.

A target tracking policy configures your Auto Scaling group to scale based on the custom metric and a set target value. CloudWatch alarms invoke the scaling policy.

A custom “backlog per instance” metric is used to track the number of messages in the queue and also the number available for retrieval.

Can base the adjustments off the SQS Metric “ApproximateNumberOfMessages”.

Launch Templates vs Launch Configurations

Launch templates are like [launch configuration](https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchConfiguration.html)s in that they specify the instance configuration information.

Information includes 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.

Launch templates include additional features such as supporting 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.

EC2 Auto Scaling Lifecycle Hooks

Lifecycle pause EC2 instances as an Auto Scaling group launches or terminates them so you can perform custom actions.

Paused instances remain in a wait state either until you complete the lifecycle action using the complete-lifecycle-action command or the CompleteLifecycleAction operation, or until the timeout period ends (one hour by default).

Lifecycle hooks provide greater control over how instances launch and terminate.

You can send notifications when an instance enters a wait state using Amazon EventBridge, Amazon SNS, or Amazon SQS to receive the notifications.

High Availability

Amazon EC2 Auto Scaling offers high availability (HA) when instances are launched into at least two Availability Zones.

You can use an Amazon Elastic Load Balancer or Amazon Route 53 to direct incoming connections to your EC2 instances.

EC2 Auto Scaling cannot provide HA across multiple AWS Regions as it is a regional service.

Monitoring and Reporting

When Auto Scaling group metrics are enabled the Auto Scaling group sends sampled data to[CloudWatch](https://digitalcloud.training/amazon-cloudwatch/) every minute (no charge).

You can enable and disable Auto Scaling group metrics using the AWS Management Console, AWS CLI, or AWS SDKs.

The AWS/AutoScaling namespace includes the following metrics which are sent to CloudWatch every 1 minute:

GroupMinSize

GroupMaxSize

GroupDesiredCapacity

GroupInServiceInstances

GroupPendingInstances

GroupStandbyInstances

GroupTerminatingInstances

GroupTotalInstances

Metrics are also sent from the Amazon EC2 instances to Amazon CloudWatch:

Basic monitoring sends EC2 metrics to CloudWatch about ASG instances every 5 minutes.

Detailed monitoring can be enabled and sends metrics every 1 minute (chargeable).

If the launch configuration is created from the console basic monitoring of EC2 instances is enabled by default.

If the launch configuration is created from the CLI detailed monitoring of EC2 instances is enabled by default.

EC2 Auto Scaling uses health checks to check if instances are healthy and available.

By default Auto Scaling uses EC2 status checks.

Auto Scaling supports ELB health checks and custom health checks in addition to the EC2 status checks.

If any of the configured health checks returns an unhealthy status the instance will be terminated.

With ELB health checks enabled an instance is marked as unhealthy if the ELB reports it as OutOfService.

A healthy instance enters the InService state.

If an EC2 instance is marked as unhealthy it will be scheduled for replacement.

If connection draining is enabled, EC2 Auto Scaling will wait for any in-flight requests to complete or timeout before terminating instances.

The health check grace period is a period of time in which a new instance is allowed to warm up before health check are performed (300 seconds by default).

Note: When using Elastic Load Balancers it is an AWS best practice to enable the ELB health checks. If you don’t, EC2 status checks may show an instance as being healthy that the ELB has determined is unhealthy. In this case the instance will be removed from service by the ELB but will not be terminated by Auto Scaling.

Logging and Auditing

[AWS CloudTrail](https://docs.aws.amazon.com/autoscaling/plans/APIReference/logging-using-cloudtrail.html) captures all API calls for AWS Auto Scaling as events.

The API calls that are captured include calls from the Amazon EC2 Auto Scaling console and code calls to the AWS Auto Scaling API.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Auto Scaling.

If you don’t configure a trail, you can still view the most recent (up to 90 days) events in the CloudTrail console in the Event history.

CloudTrail events include the calls made to AWS Auto Scaling, the IP address from which the requests were made, who made the requests, when they were made, and additional details.

Authorization and Access Control

EC2 Auto Scaling support[identity-based IAM policies](https://docs.aws.amazon.com/autoscaling/ec2/userguide/control-access-using-iam.html).

Amazon EC2 Auto Scaling does not support resource-based policies.

Amazon EC2 Auto Scaling uses[service-linked roles](https://docs.aws.amazon.com/autoscaling/ec2/userguide/autoscaling-service-linked-role.html) for the permissions that it requires to call other AWS services on your behalf.

A service-linked role is a unique type of IAM role that is linked directly to an AWS service.

There is a default service-linked role for your account, named AWSServiceRoleForAutoScaling.

This role is automatically assigned to your Auto Scaling groups unless you specify a different service-linked role.

Amazon EC2 Auto Scaling also does not support Access Control Lists (ACLs).

You can apply tag-based, resource-level permissions in the identity-based policies that you create for Amazon EC2 Auto Scaling.

This offers better control over which resources a user can create, modify, use, or delete.

ASG Behavior and Configuration

EC2 Auto Scaling – Termination Policy:

Termination policies control the instances which are terminated first when a scale-in event occurs.

There is a default termination policy configured and you can create your own customized termination policies.

The default termination policy helps to ensure that EC2 instances span Availability Zones evenly for high availability.

The default policy is fairly generic and flexible to cover a range of scenarios.

You can enable Instance Protection which prevents Auto Scaling from scaling in and terminating the EC2 instances.

If Auto Scaling fails to launch instances in a specific AZ it will try other AZs until successful.

The default health check grace period is 300 seconds.

“Scaling out” is the process in which EC2 instances are launched by the scaling policy.

“Scaling in” is the process in which EC2 instances are terminated by the scaling policy.

It is recommended to create a scale-in event for every configured scale-out event.

An imbalance may occur due to:

Manually removing AZs/subnets from the configuration.

Manually terminating EC2 instances.

EC2 capacity issues.

Spot price is reached.

All Elastic IPs and EBS volumes are detached from terminated EC2 instances and will need to be manually reattached.

Using custom health checks a CLI command can be issued to set the instance’s status to unhealthy, e.g.:

aws autoscaling set–instance-health –instance-id i-123abc45d –health-status Unhealthy

Once an EC2 instance enters the terminating state it cannot be put back into service again.

However, there is a short period of time in which an AWS CLI command can be run to change an instance to healthy.

Termination of unhealthy instances happens first, then Auto Scaling attempts to launch new instances to replace terminated instances. This is different to AZ rebalancing.

You can use the AWS Console or AWS CLI to manually remove (detach) instances from an ASG.

When detaching an EC2 instance you can optionally decrement the ASG’s desired capacity (to prevent it from launching another instance).

An instance can only be attached to one Auto Scaling group at a time.

You can suspend and then resume one or more of the scaling processes for your ASG at any time.

This can be useful when if want to investigate an issue with an application and make changes without invoking the scaling processes.

You can manually move an instance from an ASG and put it in the standby state.

Instances in the standby state are still managed by Auto Scaling, are charged as normal, and do not count towards available EC2 instance for workload/application use.

Auto scaling does not perform any health checks on EC2 instances in the standby state.

Standby state can be used for performing updates/changes/troubleshooting etc. without health checks being performed or replacement instances being launched.

When you delete an Auto Scaling group all EC2 instances will be terminated.

You can select to use Spot instances in launch configurations.

The ASG treats spot instances the same as on-demand instances.

You can mix Spot instances with on-demand (when using launch templates).

The ASG can be configured to send an Amazon SNS notification when:

An instance is launched.

An instance is terminated.

An instance fails to launch.

An instance fails to terminate.

Merging ASGs.

Can merge multiple single AZ Auto Scaling Groups into a single multi-AZ ASG.

Merging can only be performed by using the CLI.

The process is to rezone one of the groups to cover/span the other AZs for the other ASGs and then delete the other ASGs.

This can be performed on ASGs with or without ELBs attached to them.

Cooldown Period:

The cooldown period is a setting you can configure for your Auto Scaling group that helps to ensure that it doesn’t launch or terminate additional instances before the previous scaling activity takes effect.

A default cooldown period of 300 seconds is applied when you create your Auto Scaling group.

You can configure the cooldown period when you create the Auto Scaling group.

You can override the default cooldown via scaling-specific cooldown.

The warm-up period is the period in which a newly launched EC2 instance in an ASG that uses step scaling is not considered toward the ASG metrics.

AWS Organizations

AWS Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources.

AWS accounts are natural boundaries for permissions, security, costs, and workloads.

Using a multi-account environment is a recommended best-practice when scaling your cloud environment.

AWS Organizations provides many features for managing multi-account environments, including:

Simplify account creation by programmatically creating new accounts using the AWS Command Line Interface (CLI), SDKs, or APIs.

Group accounts into organizational units (OUs), or groups of accounts that serve a single application or service.

Apply tag polices to classify or track resources in your organization and provide attribute-based access control for users or applications.

Delegate responsibility for supported AWS services to accounts so users can manage them on behalf of your organization.

Centrally provide tools and access for your security team to manage security needs on behalf of the organization.

Set up Amazon Single Sign-On (SSO) to provide access to AWS accounts and resources using your active directory, and customize permissions based on separate job roles.

Apply service control policies (SCPs) to users, accounts, or OUs to control access to AWS resources, services, and Regions within your organization.

Share AWS resources within your organization using AWS Resource Allocation Management (RAM).

Activate AWS CloudTrail across accounts, which creates a log of all activity in your cloud environment that cannot be turned off or modified by member accounts.

Organizations provides you with a single consolidated bill.

In addition, you can view usage from resources across accounts and track costs using AWS Cost Explorer and optimize your usage of compute resources using AWS Compute Optimizer.

AWS Organizations is available to all AWS customers at no additional charge.

The [AWS Organizations API](https://docs.aws.amazon.com/organizations/latest/APIReference/Welcome.html) enables automation for account creation and management.

AWS Organizations is available in two feature sets:

Consolidated billing.

All features.

By default, organizations support consolidated billing features.

Consolidated billing separates paying accounts and linked accounts.

You can use AWS Organizations to set up a single payment method for all the AWS accounts in your organization through consolidated billing.

With consolidated billing, you can see a combined view of charges incurred by all your accounts.

Can also take advantage of pricing benefits from aggregated usage, such as volume discounts for Amazon EC2 and Amazon S3.

Limit of 20 linked accounts for consolidated billing (default).

Policies can be assigned at different points in the hierarchy.

Can help with cost control through volume discounts.

Unused reserved EC2 instances are applied across the group.

Paying accounts should be used for billing purposes only.

Billing alerts can be setup at the paying account which shows billing for all linked accounts.

Key Concepts

Some of the key concepts you need to understand are listed here:

AWS Organization – An organization is a collection of AWS accounts that you can organize into a hierarchy and manage centrally.

AWS Account – An AWS account is a container for your AWS resources.

Management Account – A management account is the AWS account you use to create your organization.

Member Account – A member account is an AWS account, other than the management account, that is part of an organization.

Administrative Root – An administrative root is the starting point for organizing your AWS accounts. The administrative root is the top-most container in your organization’s hierarchy.

Organizational Unit (OU) – An organizational unit (OU) is a group of AWS accounts within an organization. An OU can also contain other OUs enabling you to create a hierarchy.

Policy – A policy is a “document” with one or more statements that define the controls that you want to apply to a group of AWS accounts. AWS Organizations supports a specific type of policy called a Service Control Policy (SCP). An SCP defines the AWS service actions, such as Amazon EC2 RunInstances, that are available for use in different accounts within an organization.

Best practices for the management account:

[Use the management account only for tasks that require the management account.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-use)

[Use a group email address for the management account’s root user.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_email-address)

[Use a complex password for the management account’s root user.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_complex-password)

[Enable MFA for your root user credentials.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_mfa)

[Add a phone number to the account contact information.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_phone-number)

[Review and keep track of who has access.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_review-access)

[Document the processes for using the root user credentials.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_document-processes)

[Apply controls to monitor access to the root user credentials.](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_best-practices_mgmt-acct.html#best-practices_mgmt-acct_monitor-access)

Migrating accounts between organizations

Accounts can be migrated between organizations.

You must have root or IAM access to both the member and management accounts.

Use the AWS Organizations console for just a few accounts.

Use the AWS Organizations API or AWS Command Line Interface (AWS CLI) if there are many accounts to migrate.

Billing history and billing reports for all accounts stay with the management account in an organization.

Before migration download any billing or report history for any member accounts that you want to keep.

When a member account leaves an organization, all charges incurred by the account are charged directly to the standalone account.

Even if the account move only takes a minute to process, it is likely that some charges will be incurred by the member account.

Service Control Policies (SCPs)

Service control policies (SCPs) are a type of organization policy that you can use to manage permissions in your organization.

SCPs offer central control over the maximum available permissions for all accounts in your organization. SCPs help you to ensure your accounts stay within your organization’s access control guidelines.

SCPs are available only in an organization that has all features enabled.

SCPs aren’t available if your organization has enabled only the consolidated billing features.

SCPs alone are not sufficient to granting permissions to the accounts in your organization.

No permissions are granted by an SCP. An SCP defines a guardrail, or sets limits, on the actions that the account’s administrator can delegate to the IAM users and roles in the affected accounts.

The administrator must still attach identity-based or resource-based policies to IAM users or roles, or to the resources in your accounts to grant permissions.

The effective permissions are the logical intersection between what is allowed by the SCP and what is allowed by the IAM and resource-based policies.

SCP Inheritance:

SCPs affect only IAM users and roles that are managed by accounts that are part of the organization. SCPs don’t affect resource-based policies directly. They also don’t affect users or roles from accounts outside the organization.

An SCP restricts permissions for IAM users and roles in member accounts, including the member account’s root user.

Any account has only those permissions permitted by every parent above it.

If a permission is blocked at any level above the account, either implicitly (by not being included in an Allow policy statement) or explicitly (by being included in a Deny policy statement), a user or role in the affected account can’t use that permission, even if the account administrator attaches the AdministratorAccess IAM policy with \*/\* permissions to the user.

SCPs affect only member accounts in the organization. They have no effect on users or roles in the management account.

Users and roles must still be granted permissions with appropriate IAM permission policies. A user without any IAM permission policies has no access, even if the applicable SCPs allow all services and all actions.

If a user or role has an IAM permission policy that grants access to an action that is also allowed by the applicable SCPs, the user or role can perform that action.

If a user or role has an IAM permission policy that grants access to an action that is either not allowed or explicitly denied by the applicable SCPs, the user or role can’t perform that action.

SCPs affect all users and roles in attached accounts, including the root user. The only exceptions are those described in [Tasks and entities not restricted by SCPs](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html#not-restricted-by-scp).

SCPs do not affect any service-linked role. Service-linked roles enable other AWS services to integrate with AWS Organizations and can’t be restricted by SCPs.

When you disable the SCP policy type in a root, all SCPs are automatically detached from all AWS Organizations entities in that root. AWS Organizations entities include organizational units, organizations, and accounts.

If you reenable SCPs in a root, that root reverts to only the default FullAWSAccess policy automatically attached to all entities in the root.

Any attachments of SCPs to AWS Organizations entities from before SCPs were disabled are lost and aren’t automatically recoverable, although you can manually reattach them.

If both a permissions boundary (an advanced IAM feature) and an SCP are present, then the boundary, the SCP, and the identity-based policy must all allow the action.

You can’t use SCPs to restrict the following tasks:

Any action performed by the management account.

Any action performed using permissions that are attached to a service-linked role.

Register for the Enterprise support plan as the root user.

Change the AWS support level as the root user.

Provide trusted signer functionality for CloudFront private content.

Configure reverse DNS for an Amazon Lightsail email server as the root user.

Tasks on some AWS-related services:

Alexa Top Sites.

Alexa Web Information Service.

Amazon Mechanical Turk.

Amazon Product Marketing API.

Resource Groups

You can use resource groups to organize your AWS resources.

In AWS, a resource is an entity that you can work with.

Resource groups make it easier to manage and automate tasks on large numbers of resources at one time.

Resource groups allow you to group resources and then tag them.

The Tag Editor assists with finding resources and adding tags.

You can access Resource Groups through any of the following entry points:

On the navigation bar of the AWS Management Console.

In the AWS Systems Manager console, from the left navigation pane entry for Resource Groups.

By using the Resource Groups API, in AWS CLI commands or AWS SDK programming languages.

A resource group is a collection of AWS resources that are all in the same AWS region, and that match criteria provided in a query.

In Resource Groups, there are two types of queries on which you can build a group.

Both query types include resources that are specified in the format AWS::service::resource.

Tag-based – Tag-based queries include lists of resources and tags. Tags are keys that help identify and sort your resources within your organization. Optionally, tags include values for keys.

AWS CloudFormation stack-based – In an AWS CloudFormation stack-based query, you choose an AWS CloudFormation stack in your account in the current region, and then choose resource types within the stack that you want to be in the group. You can base your query on only one AWS CloudFormation stack.

Resource groups can be nested; a resource group can contain existing resource groups in the same region.

AWS Direct Connect

AWS Direct Connect is a network service that provides an alternative to using the Internet to connect a customer’s on-premises sites to AWS.

Data is transmitted through a private network connection between AWS and a customer’s datacenter or corporate network.

Benefits:

Reduce cost when using large volumes of traffic.

Increase reliability (predictable performance).

Increase bandwidth (predictable bandwidth).

Decrease latency.

Each AWS Direct Connect connection can be configured with one or more virtual interfaces (VIFs).

Public VIFs allow access to public services such as S3, EC2, and DynamoDB.

Private VIFs allow access to your VPC.

Must use public IP addresses on public VIFs.

Must use private IP addresses on private VIFs.

Cannot do layer 2 over Direct Connect (L3 only).

From Direct Connect you can connect to all AZs within the region.

You can establish IPSec connections over public VIFs to remote regions.

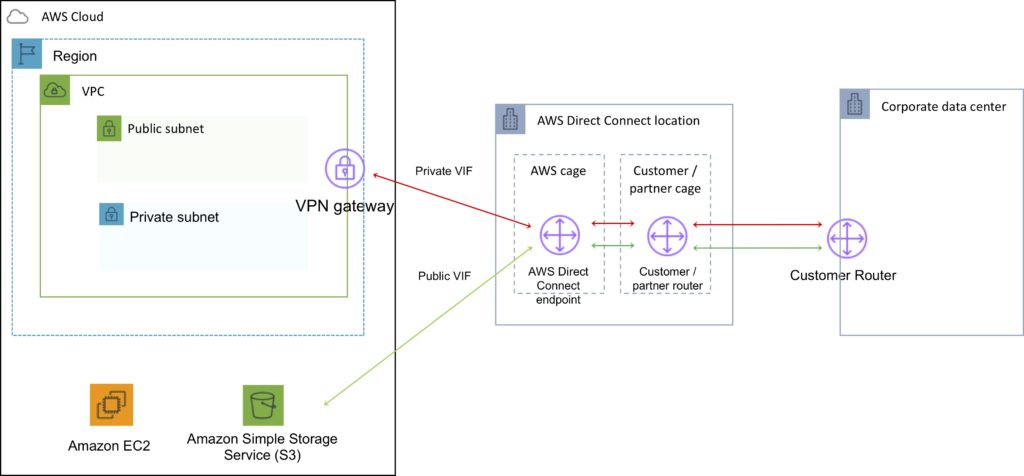
Route propagation can be used to send customer side routes to the VPC.

You can only have one 0.0.0.0/0 (all IP addresses) entry per route table.

You can bind multiple ports for higher bandwidth.

Virtual interfaces are configured to connect to either AWS public services (e.g. EC2/S3) or private services (e.g. VPC based resources).

The diagram below shoes the components of AWS Direct Connect:



Direct Connect is charged by port hours and data transfer.

Available in 1 Gbps, 10 Gbps, and 100 Gbps (limited Regions).

Speeds of 50 Mbps, 100 Mbps, 200 Mbps, 300 Mbps, 400 Mbps, and 500 Mbps can be purchased through AWS Direct Connect Partners (APN Partners).

Uses 802.1q VLANs.

Each connection consists of a single dedicated connection between ports on the customer router and an Amazon router.

for HA you must have 2x DX connections – can be active/active or active/standby.

Route tables need to be updated to point to a Direct Connect connection.

VPN can be maintained as a backup with a higher BGP priority.

Recommended to enable Bidirectional Forwarding Detection (BFD) for faster detection and failover.

You cannot extend your on-premises VLANs into the AWS cloud using Direct Connect.

Can aggregate up to 4 Direct Connect ports into a single connection using Link Aggregation Groups (LAG).

AWS Direct Connect supports both single (IPv4) and dual stack (IPv4/IPv6) configurations on public and private VIFs.

Technical requirements for connecting virtual interfaces:

A public or private ASN. If you are using a public ASN you must own it. If you are using a private ASN, it must be in the 64512 to 65535 range.

A new unused VLAN tag that you select.

Private Connection (VPC) – The VPC Virtual Private Gateway (VGW) ID.

Public Connection – Public IPs (/30) allocated by you for the BGP session.

AWS Direct Connect does not offer encryption.

You can encrypting data sent over DX by establishing a VPN tunnel over the DX connection.

Running an AWS VPN connection over a DX connection provides consistent levels of throughput and encryption algorithms that protect your data.

Though a private VIF is typically used to connect to a VPC, in the case of running an IPSec VPN over the top of a DX connection it is necessary to use a public VIF.

AWS Direct Connect Gateway

Grouping of Virtual Private Gateways (VGWs) and Private Virtual Interfaces (VIFs) that belong to the same AWS account.

Direct Connect Gateway enables you to interface with VPCs in any AWS Region (except AWS China Region).

You associate an AWS Direct Connect gateway with either of the following gateways:

A transit gateway when you have multiple VPCs in the same Region.

A virtual private gateway.

Can share private virtual interface to interface with more than one Virtual Private Clouds (VPCs) reducing the number of BGP sessions.

A Direct Connect gateway is a globally available resource.

You can create the Direct Connect gateway in any public Region and access it from all other public Regions.

The diagram below depicts the components of an AWS Direct Connect Gateway configuration:



Amazon S3 and Glacier

Amazon S3 is object storage built to store and retrieve any amount of data from anywhere on the Internet.

It’s a simple storage service that offers an extremely durable, highly available, and infinitely scalable data storage infrastructure at very low costs.

Amazon S3 is a distributed architecture and objects are redundantly stored on multiple devices across multiple facilities (AZs) in an Amazon S3 region.

Amazon S3 is a simple key-based object store.

Keys can be any string, and they can be constructed to mimic hierarchical attributes.

Alternatively, you can use S3 Object Tagging to organize your data across all your S3 buckets and/or prefixes.

Amazon S3 provides a simple, standards-based REST web services interface that is designed to work with any Internet-development toolkit.

Files can be from 0 bytes to 5TB.

The largest object that can be uploaded in a single PUT is 5 gigabytes.

For objects larger than 100 megabytes use the Multipart Upload capability.

Updates to an object are atomic – when reading an updated object you will either get the new object or the old one, you will never get partial or corrupt data.

There is unlimited storage available.

It is recommended to access S3 through SDKs and APIs (the console uses APIs).

Event notifications for specific actions, can send alerts or trigger actions.

Notifications can be sent to:

SNS Topics.

SQS Queue.

Lambda functions.

Need to configure SNS/SQS/Lambda before S3.

No extra charges from S3 but you pay for SNS, SQS and Lambda.

Requester pays function causes the requester to pay (removes anonymous access).

Can provide time-limited access to objects.

Provides read after write consistency for PUTS of new objects.

Provides eventual consistency for overwrite PUTS and DELETES (takes time to propagate).

You can only store files (objects) on S3.

HTTP 200 code indicates a successful write to S3.

S3 data is made up of:

Key (name).

Value (data).

Version ID.

Metadata.

Access Control Lists.

Amazon S3 automatically scales to high request rates.

For example, your application can achieve at least 3,500 PUT/POST/DELETE and 5,500 GET requests per second per prefix in a bucket.

There are no limits to the number of prefixes in a bucket.

For read intensive requests you can also use CloudFront edge locations to offload from S3.

[](https://www.adplugg.com/track/click/A48221584/60488/click?toi=21584&hn=digitalcloud.training&bu=%2Famazon-s3-and-glacier%2F&rf=&zn=16420&pm=8397&ct=&next=)

Additional Capabilities

Additional capabilities offered by Amazon S3 include:

|  |  |
| --- | --- |
| Additional S3 Capability | How it works |
| Transfer Acceleration | Speeds up data uploads using CloudFront in reverse |
| Requester Pays | The requester rather than the bucket owner pays for requests and data transfer |
| Tags | Assign tags to objects to use in hosting, billing, security etc. |
| Events | Trigger notifications to SNS, SQS, or Lambda when certain events happen in your bucket |
| Static Web Hosting | Simple and massively scalable static web hosting |
| BitTorrent | Use the BitTorrent protocol to retrieve any publicly available object by automatically generating a .torrent file. |
| Storage Class Analysis | Analyzes storage access patterns to help you decide when to transition the right data to the right storage class. |
| Storage Lens | Delivers organization-wide visibility into object storage usage, activity trends, and makes actionable recommendations to improve cost-efficiency and apply data protection best practices. |
| S3 Object Lambda | Add your own code to S3 GET requests to modify and process data as it is returned to an application. |

Use Cases

Typical use cases include:

Backup and Storage – Provide data backup and storage services for others.

Application Hosting – Provide services that deploy, install, and manage web applications.

Media Hosting – Build a redundant, scalable, and highly available infrastructure that hosts video, photo, or music uploads and downloads.

Software Delivery – Host your software applications that customers can download.

Static Website – you can configure a static website to run from an S3 bucket.

S3 is a persistent, highly durable data store.

Persistent data stores are non-volatile storage systems that retain data when powered off.

This contrasts with transient data stores and ephemeral data stores which lose the data when powered off.

The following table provides a description of persistent, transient, and ephemeral data stores and which AWS service to use:

|  |  |  |
| --- | --- | --- |
| Storage Type | Description | Examples |
| Persistent data store | Data is durable and sticks around after reboots, restarts, or power cycles | S3, Glacier, EBS, EFS |
| Transient Data Store | Data is just temporarily stored and passed along to another process or persistent store | SQS, SNS |
| Ephemeral Data Store | Data is lost when the system is stopped | EC2 Instance Store, Memcached (Elasticache) |

Buckets

Files are stored in buckets:

A bucket can be viewed as a container for objects.

A bucket is a flat container of objects.

It does not provide a hierarchy of objects.

You can use an object key name (prefix) to mimic folders.

100 buckets per account by default.

You can store unlimited objects in your buckets.

You can create folders in your buckets (only available through the Console).

You cannot create nested buckets.

Bucket ownership is not transferable.

Bucket names cannot be changed after they have been created.

If a bucket is deleted its name becomes available again.

Bucket names are part of the URL used to access the bucket.

An S3 bucket is region specific.

S3 is a universal namespace so names must be unique globally.

URL is in this format: https://s3-eu-west-1.amazonaws.com/<bucketname>.

Can backup a bucket to another bucket in another account.

Can enable logging to a bucket.

Bucket naming:

Bucket names must be at least 3 and no more than 63 characters in length.

Bucket names must start and end with a lowercase character or a number.

Bucket names must be a series of one or more labels which are separated by a period.

Bucket names can contain lowercase letters, numbers, and hyphens.

Bucket names cannot be formatted as an IP address.

For better performance, lower latency, and lower cost, create the bucket closer to your clients.

Objects

Each object is stored and retrieved by a unique key (ID or name).

An object in S3 is uniquely identified and addressed through:

Service endpoint.

Bucket name.

Object key (name).

Optionally, an object version.

Objects stored in a bucket will never leave the region in which they are stored unless you move them to another region or enable cross-region replication.

You can define permissions on objects when uploading and at any time afterwards using the AWS Management Console.

Subresources

Sub-resources are subordinate to objects, they do not exist independently but are always associated with another entity such as an object or bucket.

Sub-resources (configuration containers) associated with buckets include:

Lifecycle – define an object’s lifecycle.

Website – configuration for hosting static websites.

Versioning – retain multiple versions of objects as they are changed.

Access Control Lists (ACLs) – control permissions access to the bucket.

Bucket Policies – control access to the bucket.

Cross Origin Resource Sharing (CORS).

Logging.

Sub-resources associated with objects include:

ACLs – define permissions to access the object.

Restore – restoring an archive.

Cross-origin-resource-sharing (CORS)

Used to allow requests to a different origin when connected to the main origin.

The request will fail unless the origin allows the requests using CORS headers (e.g. Access-Control-Allow-Origin).

Must enable the correct CORS headers.

Specify a CORS configuration on the S3 bucket.

Storage Classes

There are six S3 storage classes.

S3 Standard (durable, immediately available, frequently accessed).

S3 Intelligent-Tiering (automatically moves data to the most cost-effective tier).

S3 Standard-IA (durable, immediately available, infrequently accessed).

S3 One Zone-IA (lower cost for infrequently accessed data with less resilience).

S3 Glacier (archived data, retrieval times in minutes or hours).

S3 Glacier Deep Archive (lowest cost storage class for long term retention).

The table below provides the details of each Amazon S3 storage class:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | S3 Standard | S3 Intelligent Tiering | S3 Standard-IA | S3 One Zone-IA | S3 Glacier Instant Retrieval | S3 Glacier Flexible Retrieval | S3 Glacier Deep Archive |
| Designed for durability | 99.999999999%    (11 9’s) | 99.999999999%    (11 9’s) | 99.999999999%    (11 9’s) | 99.999999999%    (11 9’s) | 99.999999999%    (11 9’s) | 99.999999999%    (11 9’s) | 99.999999999%    (11 9’s) |
| Designed for availability | 99.99% | 99.9% | 99.9% | 99.5% | 99.9% | 99.99% | 99.99% |
| Availability SLA | 99.9% | 99% | 99% | 99% | 99% | 99.% | 99.9% |
| Availability Zones | ≥3 | ≥3 | ≥3 | 1 | ≥3 | ≥3 | ≥3 |
| Minimum capacity charge per object | N/A | N/A | 128 KB | 128 KB | 128 KB | 40 KB | 40 KB |
| Minimum storage duration charge | N/A | N/A | 30 days | 30 days | 90 days | 90 days | 180 days |
| Retrieval charge | N/A | N/A | per GB retrieved | per GB retrieved | per GB retrieved | per GB retrieved | per GB retrieved |
| First byte latency | milliseconds | milliseconds | milliseconds | milliseconds | milliseconds | minutes or hours | hours |
| Storage type | Object | Object | Object | Object | Object | Object | Object |
| Lifecycle transitions | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

Objects stored in the S3 One Zone-IA storage class are stored redundantly within a single Availability Zone in the AWS Region you select.

Access and Access Policies

There are four mechanisms for controlling access to Amazon S3 resources:

IAM policies.

Bucket policies.

Access Control Lists (ACLs).

Query string authentication (URL to an Amazon S3 object which is only valid for a limited time).

Access auditing can be configured by configuring an Amazon S3 bucket to create access log records for all requests made against it.

For capturing IAM/user identity information in logs configure AWS CloudTrail Data Events.

By default a bucket, its objects, and related sub-resources are all private.

By default only a resource owner can access a bucket.

The resource owner refers to the AWS account that creates the resource.

With IAM the account owner rather than the IAM user is the owner.

Within an IAM policy you can grant either programmatic access or AWS Management Console access to Amazon S3 resources.

Amazon Resource Names (ARN) are used for specifying resources in a policy.

The format for any resource on AWS is:

arn:partition:service:region:namespace:relative-id.

For S3 resources:

aws is a common partition name.

s3 is the service.

You don’t specify Region and namespace.

For Amazon S3, it can be a bucket-name or a bucket-name/object-key. You can use wild card.

The format for S3 resources is:

arn:aws:s3:::bucket\_name.

arn:aws:s3:::bucket\_name/key\_name.

A bucket owner can grant cross-account permissions to another AWS account (or users in an account) to upload objects.

The AWS account that uploads the objects owns them.

The bucket owner does not have permissions on objects that other accounts own, however:

The bucket owner pays the charges.

The bucket owner can deny access to any objects regardless of ownership.

The bucket owner can archive any objects or restore archived objects regardless of ownership.

Access to buckets and objects can be granted to:

Individual users.

AWS accounts.

Everyone (public/anonymous).

All authenticated users (AWS users).

Access policies define access to resources and can be associated with resources (buckets and objects) and users.

You can use the AWS Policy Generator to create a bucket policy for your Amazon S3 bucket.

The categories of policy are resource-based policies and user policies.

Resource-based policies:

Attached to buckets and objects.

ACL-based policies define permissions.

ACLs can be used to grant read/write permissions to other accounts.

Bucket policies can be used to grant other AWS accounts or IAM users’ permission to the bucket and objects.

User policies:

Can use IAM to manage access to S3 resources.

Using IAM you can create users, groups and roles and attach access policies to them granting them access to resources.

You cannot grant anonymous permissions in an IAM user policy as the policy is attached to a user.

User policies can grant permissions to a bucket and the objects in it.

ACLs:

S3 ACLs enable you to manage access to buckets and objects.

Each bucket and object has an ACL attached to it as a subresource.

Bucket and object permissions are independent of each other.

The ACL defines which AWS accounts (grantees) or pre-defined S3 groups are granted access and the type of access.

A grantee can be an AWS account or one of the predefined Amazon S3 groups.

When you create a bucket or an object, S3 creates a default ACL that grants the resource owner full control over the resource.

Cross account access:

You grant permission to another AWS account using the email address or the canonical user ID.

However, if you provide an email address in your grant request, Amazon S3 finds the canonical user ID for that account and adds it to the ACL.

Grantee accounts can then then delegate the access provided by other accounts to their individual users.

Pre-defined Groups

Authenticated Users group:

This group represents all AWS accounts.

Access permission to this group allows any AWS account access to the resource.

All requests must be signed (authenticated).

Any authenticated user can access the resource.

All Users group:

Access permission to this group allows anyone in the world access to the resource.

The requests can be signed (authenticated) or unsigned (anonymous).

Unsigned requests omit the authentication header in the request.

AWS recommends that you never grant the All Users group WRITE, WRITE\_ACP, or FULL\_CONTROL permissions.

Log Delivery group:

Providing WRITE permission to this group on a bucket enables S3 to write server access logs.

Not applicable to objects.

The following table lists the set of permissions that Amazon S3 supports in an ACL.

The set of ACL permissions is the same for an object ACL and a bucket ACL.

Depending on the context (bucket ACL or object ACL), these ACL permissions grant permissions for specific buckets or object operations.

The table lists the permissions and describes what they mean in the context of objects and buckets.

|  |  |  |
| --- | --- | --- |
| Permission | When granted on a bucket | When granted on an object |
| READ | Allows grantee to list the objects in the bucket | Allows grantee to read the object data and its metadata |
| WRITE | Allows grantee to create, overwrite and delete any object in the bucket | N/A |
| READ\_ACP | Allows grantee to read the bucket ACL | Allows grantee to read the object ACL |
| WRITE\_ACP | Allows grantee to write the ACL for the applicable buckets | Allows grantee to write the ACL for the applicable object |
| FULL\_CONTROL | Allows grantee the READ, WRITE, READ\_ACP, WRITE\_ACP permissions on the bucket | Allows grantee the READ, WRITE, READ\_ACP, WRITE\_ACP permissions on the object |

Note the following:

Permissions are assigned at the account level for authenticated users.

You cannot assign permissions to individual IAM users.

When Read is granted on a bucket it only provides the ability to list the objects in the bucket.

When Read is granted on an object the data can be read.

ACP means access control permissions and READ\_ACP/WRITE\_ACP control who can read/write the ACLs themselves.

WRITE is only applicable to the bucket level (except for ACP).

Bucket policies are limited to 20 KB in size.

Object ACLs are limited to 100 granted permissions per ACL.

The only recommended use case for the bucket ACL is to grant write permissions to the S3 Log Delivery group.

There are limits to managing permissions using ACLs:

You cannot grant permissions to individual users.

You cannot grant conditional permissions.

You cannot explicitly deny access.

When granting other AWS accounts the permissions to upload objects, permissions to these objects can only be managed by the object owner using object ACLs.

You can use bucket policies for:

Granting users permissions to a bucket owned by your account.

Managing object permissions (where the object owner is the same account as the bucket owner).

Managing cross-account permissions for all Amazon S3 permissions.

You can use user policies for:

Granting permissions for all Amazon S3 operations.

Managing permissions for users in your account.

Granting object permissions to users within the account.

For an IAM user to access resources in another account the following must be provided:

Permission from the parent account through a user policy.

Permission from the resource owner to the IAM user through a bucket policy, or the parent account through a bucket policy, bucket ACL or object ACL.

If an AWS account owns a resource it can grant permissions to another account, that account can then delegate those permissions or a subset of them to users in the account (permissions delegation).

An account that receives permissions from another account cannot delegate permissions cross-account to a third AWS account.

Charges

No charge for data transferred between EC2 and S3 in the same region.

Data transfer into S3 is free of charge.

Data transferred to other regions is charged.

Data Retrieval (applies to S3 Standard-IA and S3 One Zone-IA, S3 Glacier and S3 Glacier Deep Archive).

Charges are:

Per GB/month storage fee.

Data transfer out of S3.

Upload requests (PUT and GET).

Retrieval requests (S3-IA or Glacier).

Requester pays:

The bucket owner will only pay for object storage fees.

The requester will pay for requests (uploads/downloads) and data transfers.

Can only be enabled at the bucket level.

Multipart upload

Can be used to speed up uploads to S3.

Multipart upload uploads objects in parts independently, in parallel and in any order.

Performed using the S3 Multipart upload API.

It is recommended for objects of 100MB or larger.

Can be used for objects from 5MB up to 5TB.

Must be used for objects larger than 5GB.

If transmission of any part fails it can be retransmitted.

Improves throughput.

Can pause and resume object uploads.

Can begin upload before you know the final object size.

S3 Copy

You can create a copy of objects up to 5GB in size in a single atomic operation.

For files larger than 5GB you must use the multipart upload API.

Can be performed using the AWS SDKs or REST API.

The copy operation can be used to:

Generate additional copies of objects.

Renaming objects.

Changing the copy’s storage class or encryption at rest status.

Move objects across AWS locations/regions.

Change object metadata.

Once uploaded to S3 some object metadata cannot be changed, copying the object can allow you to modify this information.

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.

S3 Transfer Acceleration leverages Amazon CloudFront’s globally distributed AWS Edge Locations.

Used to accelerate object uploads to S3 over long distances (latency).

Transfer acceleration is as secure as a direct upload to S3.

You are charged only if there was a benefit in transfer times.

Need to enable transfer acceleration on the S3 bucket.

Cannot be disabled, can only be suspended.

May take up to 30 minutes to implement.

URL is: <bucketname>.s3-accelerate.amazonaws.com.

Bucket names must be DNS compliance and cannot have periods between labels.

Now HIPAA compliant.

You can use multipart uploads with transfer acceleration.

Must use one of the following endpoints:

.s3-accelerate.amazonaws.com.

.s3-accelerate.dualstack.amazonaws.com (dual-stack option).

S3 Transfer Acceleration supports all bucket level features including multipart uploads.

Static Websites

S3 can be used to host static websites.

Cannot use dynamic content such as PHP, .Net etc.

Automatically scales.

You can use a custom domain name with S3 using a Route 53 Alias record.

When using a custom domain name the bucket name must be the same as the domain name.

Can enable redirection for the whole domain, pages, or specific objects.

URL is: <bucketname>.s3-website-.amazonaws.com.

Requester pays does not work with website endpoints.

Does not support HTTPS/SSL.

Returns an HTML document.

Supports object and bucket level redirects.

Only supports GET and HEAD requests on objects.

Supports publicly readable content only.

To enable website hosting on a bucket, specify:

An Index document (default web page).

Error document (optional).

|  |  |  |
| --- | --- | --- |
| Key Difference | REST API Endpoint | Website Endpoint |
| Access Control | Supports both public and private content | Supports only publicly readable content |
| Error message handling | Returns an XML-formatted error response | Returns an HTML document |
| Redirection support | Not applicable | Supports both object-level and bucket-level redirects |
| Requests support | Supports all bucket and object operations | Supports only GET and HEAD requests on objects |
| Responses to GET and HEAD requests at the root of the bucket | Returns a list of the object keys in the bucket | Returns the Index document that is specified in the website configuration |
| SSL support | Supports SSL connections | Does not support SSL connections |

Pre-Signed URLs

Pre-signed URLs can be used to provide temporary access to a specific object to those who do not have AWS credentials.

By default all objects are private and can only be accessed by the owner.

To share an object you can either make it public or generate a pre-signed URL.

Expiration date and time must be configured.

These can be generated using SDKs for Java and .Net and AWS explorer for Visual Studio.

Can be used for downloading and uploading S3 objects.

Versioning

Versioning stores all versions of an object (including all writes and even if an object is deleted).

Versioning protects against accidental object/data deletion or overwrites.

Enables “roll-back” and “un-delete” capabilities.

Versioning can also be used for data retention and archive.

Old versions count as billable size until they are permanently deleted.

Enabling versioning does not replicate existing objects.

Can be used for backup.

Once enabled versioning cannot be disabled only suspended.

Can be integrated with lifecycle rules.

Multi-factor authentication (MFA) delete can be enabled.

MFA delete can also be applied to changing versioning settings.

MFA delete applies to:

Changing the bucket’s versioning state.

Permanently deleting an object.

Cross Region Replication requires versioning to be enabled on the source and destination buckets.

Reverting to previous versions isn’t replicated.

By default a HTTP GET retrieves the most recent version.

Only the S3 bucket owner can permanently delete objects once versioning is enabled.

When you try to delete an object with versioning enabled a DELETE marker is placed on the object.

You can delete the DELETE marker and the object will be available again.

Deletion with versioning replicates the delete marker. But deleting the delete marker is not replicated.

Bucket versioning states:

Enabled.

Versioned.

Un-versioned.

Objects that existed before enabling versioning will have a version ID of NULL.

Suspension:

If you suspend versioning the existing objects remain as they are however new versions will not be created.

While versioning is suspended new objects will have a version ID of NULL and uploaded objects of the same name will overwrite the existing object.

Object Lifecycle Management

Used to optimize storage costs, adhere to data retention policies and to keep S3 volumes well-maintained.

A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. There are two types of actions:

Transition actions—Define when objects transition to another [storage class](https://docs.aws.amazon.com/AmazonS3/latest/dev/storage-class-intro.html). For example, you might choose to transition objects to the STANDARD\_IA storage class 30 days after you created them, or archive objects to the GLACIER storage class one year after creating them.

There are costs associated with the lifecycle transition requests. For pricing information, see [Amazon S3 Pricing](https://aws.amazon.com/s3/pricing/).

Expiration actions—Define when objects expire. Amazon S3 deletes expired objects on your behalf.

Lifecycle configuration is an XML file applied at the bucket level as a subresource.

Can be used in conjunction with versioning or independently.

Can be applied to current and previous versions.

Can be applied to specific objects within a bucket: objects with a specific tag or objects with a specific prefix.

Supported Transitions and Related Constraints

Amazon S3 supports the following lifecycle transitions between storage classes using a lifecycle configuration:

You can transition from the STANDARD storage class to any other storage class.

You can transition from any storage class to the GLACIER or DEEP\_ARCHIVE storage classes.

You can transition from the STANDARD\_IA storage class to the INTELLIGENT\_TIERING or ONEZONE\_IA storage classes.

You can transition from the INTELLIGENT\_TIERING storage class to the ONEZONE\_IA storage class.

You can transition from the GLACIER storage class to the DEEP\_ARCHIVE storage class.

The following lifecycle transitions are not supported:

You can’t transition from any storage class to the STANDARD storage class.

You can’t transition from any storage class to the REDUCED\_REDUNDANCY storage class.

You can’t transition from the INTELLIGENT\_TIERING storage class to the STANDARD\_IA storage class.

You can’t transition from the ONEZONE\_IA storage class to the STANDARD\_IA or INTELLIGENT\_TIERING storage classes.

You can transition from the GLACIER storage class to the DEEP\_ARCHIVE storage class only.

You can’t transition from the DEEP\_ARCHIVE storage class to any other storage class.

The lifecycle storage class transitions have the following constraints:

From the STANDARD or STANDARD\_IA storage class to INTELLIGENT\_TIERING. The following constraints apply:

For larger objects, there is a cost benefit for transitioning to INTELLIGENT\_TIERING. Amazon S3 does not transition objects that are smaller than 128 KB to the INTELLIGENT\_TIERING storage class because it’s not cost effective.

From the STANDARD storage classes to STANDARD\_IA or ONEZONE\_IA. The following constraints apply:

For larger objects, there is a cost benefit for transitioning to STANDARD\_IA or ONEZONE\_IA. Amazon S3 does not transition objects that are smaller than 128 KB to the STANDARD\_IA or ONEZONE\_IA storage classes because it’s not cost effective.

Objects must be stored at least 30 days in the current storage class before you can transition them to STANDARD\_IA or ONEZONE\_IA. For example, you cannot create a lifecycle rule to transition objects to the STANDARD\_IA storage class one day after you create them.

Amazon 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 STANDARD\_IA or ONEZONE\_IA storage.

If you are transitioning noncurrent objects (in versioned buckets), you can transition only objects that are at least 30 days noncurrent to STANDARD\_IA or ONEZONE\_IA storage.

From the STANDARD\_IA storage class to ONEZONE\_IA. The following constraints apply:

Objects must be stored at least 30 days in the STANDARD\_IA storage class before you can transition them to the ONEZONE\_IA class.

Encryption

You can securely upload/download your data to Amazon S3 via SSL endpoints using the HTTPS protocol (In Transit – SSL/TLS).

Encryption options:

|  |  |
| --- | --- |
| Encryption Option | How It Works |
| SSE-S3 | Use S3’s existing encryption key for AES-256 |
| SSE-C | Upload your own AES-256 encryption key which S3 uses when it writes objects |
| SSE-KMS | Use a key generated and managed by AWS KMS |
| Client Side | Encrypt objects using your own local encryption process before uploading to S3 |

Server-side encryption options

Server-side encryption protects data at rest.

Amazon S3 encrypts each object with a unique key.

As an additional safeguard, it encrypts the key itself with a master key that it rotates regularly.

Amazon S3 server-side encryption uses one of the strongest block ciphers available to encrypt your data, 256-bit Advanced Encryption Standard (AES-256).

If you need server-side encryption for all the objects that are stored in a bucket, use a bucket policy.

To request server-side encryption using the object creation REST APIs, provide the x-amz-server-side-encryption request header.

Note: You need the kms:Decrypt permission when you upload or download an Amazon S3 object encrypted with an AWS Key Management Service (AWS KMS) customer master key (CMK), and that is in addition to kms:ReEncrypt, kms:GenerateDataKey, and kms:DescribeKey permissions.

There are three options for using server-side encryption: SSE-S3, SSE-KMS and SSE-C. These are detailed below,

SSE-S3 – Server-Side Encryption with S3 managed keys

When you use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), each object is encrypted with a unique key.

As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates.

Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.

Each object is encrypted with a unique key.

Encryption key is encrypted with a master key.

AWS regularly rotate the master key.

Uses AES 256.

SSE-KMS – Server-Side Encryption with AWS KMS keys

Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS) is like SSE-S3, but with some additional benefits and charges for using this service.

There are separate permissions for the use of a CMK that provides added protection against unauthorized access of your objects in Amazon S3.

SSE-KMS also provides you with an audit trail that shows when your CMK was used and by whom.

Additionally, you can create and manage customer managed CMKs or use AWS managed CMKs that are unique to you, your service, and your Region.

KMS uses Customer Master Keys (CMKs) to encrypt.

Can use the automatically created CMK key.

OR you can select your own key (gives you control for management of keys).

An envelope key protects your keys.

Chargeable.

SSE-C – Server-Side Encryption with client provided keys

With Server-Side Encryption with Customer-Provided Keys (SSE-C), you manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption, when you access your objects.

Client manages the keys, S3 manages encryption.

AWS does not store the encryption keys.

If keys are lost data cannot be decrypted.

When using server-side encryption with customer-provided encryption keys (SSE-C), you must provide encryption key information using the following request headers:

x-amz-server-side​-encryption​-customer-algorithm – Use this header to specify the encryption algorithm. The header value must be “AES256”.

x-amz-server-side​-encryption​-customer-key – Use this header to provide the 256-bit, base64-encoded encryption key for Amazon S3 to use to encrypt or decrypt your data.

x-amz-server-side​-encryption​-customer-key-MD5 – Use this header to provide the base64-encoded 128-bit MD5 digest of the encryption key according to [RFC 1321](http://tools.ietf.org/html/rfc1321). Amazon S3 uses this header for a message integrity check to ensure that the encryption key was transmitted without error.

Client-side encryption

This is the act of encrypting data before sending it to Amazon S3.

To enable client-side encryption, you have the following options:

Use a customer master key (CMK) stored in AWS Key Management Service (AWS KMS).

Use a master key you store within your application.

Option 1. Use a customer master key (CMK) stored in AWS Key Management Service (AWS KMS)

When uploading an object—Using the customer master key (CMK) ID, the client first sends a request to AWS KMS for a CMK that it can use to encrypt your object data. AWS KMS returns two versions of a randomly generated data key:

A plaintext version of the data key that the client uses to encrypt the object data.

A cipher blob of the same data key that the client uploads to Amazon S3 as object metadata.

When downloading an object—The client downloads the encrypted object from Amazon S3 along with the cipher blob version of the data key stored as object metadata. The client then sends the cipher blob to AWS KMS to get the plaintext version of the data key so that it can decrypt the object data.

Option 2. Use a master key you store within your application

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:

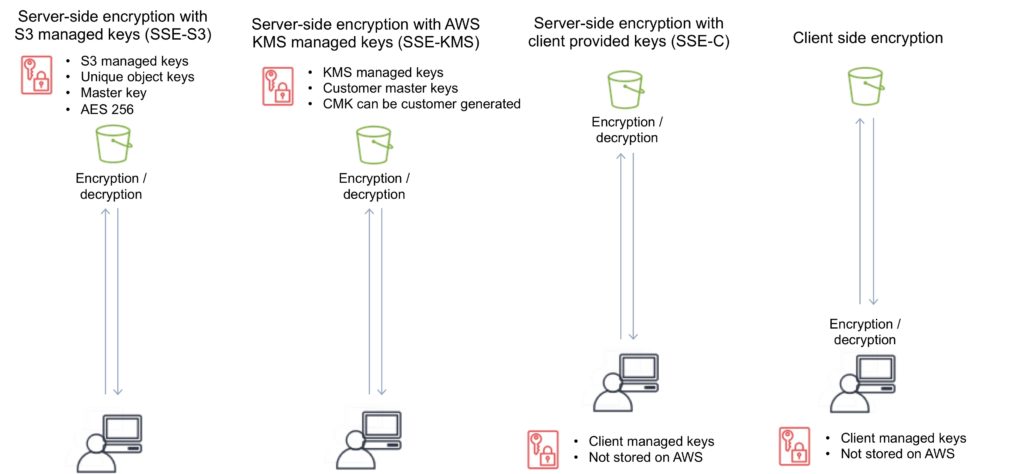
The Amazon S3 encryption client generates a one-time-use symmetric key (also known as a data encryption key or data key) locally. It uses the data key to encrypt the data of a single Amazon S3 object. The client generates a separate data key for each object.

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.

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.

The following diagram depicts the options for enabling encryption and shows you where the encryption is applied and where the keys are managed:



Event Notifications

Amazon S3 event notifications can be sent in response to actions in Amazon S3 like PUTs, POSTs, COPYs, or DELETEs.

Amazon S3 event notifications enable you to run workflows, send alerts, or perform other actions in response to changes in your objects stored in S3.

To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications.

You can configure notifications to be filtered by the prefix and suffix of the key name of objects.

Amazon S3 can publish notifications for the following events:

New object created events.

Object removal events.

Restore object events.

Reduced Redundancy Storage (RRS) object lost events.

Replication events.

Amazon S3 can send event notification messages to the following destinations:

Publish event messages to an Amazon Simple Notification Service (Amazon SNS) topic.

Publish event messages to an Amazon Simple Queue Service (Amazon SQS) queue.

Publish event messages to AWS Lambda by invoking a Lambda function and providing the event message as an argument.

Need to grant Amazon S3 permissions to post messages to an Amazon SNS topic or an Amazon SQS queue.

Need to also grant Amazon S3 permission to invoke an AWS Lambda function on your behalf. For information about granting these permissions.

Object Tags

S3 object tags are key-value pairs applied to S3 objects which can be created, updated, or deleted at any time during the lifetime of the object.

Allow you to create Identity and Access Management (IAM) policies, setup S3 Lifecycle policies, and customize storage metrics.

Up to ten tags can be added to each S3 object and you can use either the AWS Management Console, the REST API, the AWS CLI, or the AWS SDKs to add object tags.

Amazon S3 CloudWatch Metrics

You can use the AWS Management Console to enable the generation of 1-minute CloudWatch request metrics for your S3 bucket or configure filters for the metrics using a prefix or object tag.

Alternatively, you can call the S3 PUT Bucket Metrics API to enable and configure publication of S3 storage metrics.

CloudWatch Request Metrics will be available in CloudWatch within 15 minutes after they are enabled.

CloudWatch Storage Metrics are enabled by default for all buckets and reported once per day.

The S3 metrics that can be monitored include:

S3 requests.

Bucket storage.

Bucket size.

All requests.

HTTP 4XX/5XX errors.

Cross Region Replication

CRR is an Amazon S3 feature that automatically replicates data across AWS Regions.

With CRR, every object uploaded to an S3 bucket is automatically replicated to a destination bucket in a different AWS Region that you choose.

Provides automatic, asynchronous copying of objects between buckets in different regions.

CRR is configured at the S3 bucket level.

You enable a CRR configuration on your source bucket by specifying a destination bucket in a different Region for replication.

You can use either the AWS Management Console, the REST API, the AWS CLI, or the AWS SDKs to enable CRR.

Versioning must be enabled for both the source and destination buckets .

Source and destination buckets must be in different regions.

With CRR you can only replication between regions, not within a region (see SRR below for single region replication).

Replication is 1:1 (one source bucket, to one destination bucket).

You can configure separate S3 Lifecycle rules on the source and destination buckets.

You can replicate KMS-encrypted objects by providing a destination KMS key in your replication configuration.

You can set up CRR across AWS accounts to store your replicated data in a different account in the target region.

Provides low latency access for data by copying objects to buckets that are closer to users.

To activate CRR you need to configure the replication on the source bucket:

Define the bucket in the other region to replicate to.

Specify to replicate all objects or a subset of objects with specific key name prefixes.

The replicas will be exact replicas and share the same key names and metadata.

You can specify a different storage class (by default the source storage class will be used).

AWS S3 will encrypt data in-transit with SSL.

AWS S3 must have permission to replicate objects.

Bucket owners must have permission to read the object and object ACL.

Can be used across accounts but the source bucket owner must have permission to replicate objects into the destination bucket.

Triggers for replication are:

Uploading objects to the source bucket.

DELETE of objects in the source bucket.

Changes to the object, its metadata, or ACL.

What is replicated:

New objects created after enabling replication.

Changes to objects.

Objects created using SSE-S3 using the AWS managed key.

Object ACL updates.

What isn’t replicated:

Objects that existed before enabling replication (can use the copy API).

Objects created with SSE-C and SSE-KMS.

Objects to which the bucket owner does not have permissions.

Updates to bucket-level subresources.

Actions from lifecycle rules are not replicated.

Objects in the source bucket that are replicated from another region are not replicated.

Deletion behavior:

If a DELETE request is made without specifying an object version ID a delete marker will be added and replicated.

If a DELETE request is made specifying an object version ID the object is deleted but the delete marker is not replicated.

Charges:

Requests for upload.

Inter-region transfer.

S3 storage in both regions.

Same Region replication (SRR)

As the name implies you can use SRR to replication objects to a destination bucket within the same region as the source bucket.

This feature was released in September 2018.

Replication is automatic and asynchronous.

New objects uploaded to an Amazon S3 bucket are configured for replication at the bucket, prefix, or object tag levels.

Replicated objects can be owned by the same AWS account as the original copy or by different accounts, to protect from accidental deletion.

Replication can be to any Amazon S3 storage class, including S3 Glacier and S3 Glacier Deep Archive to create backups and long-term archives.

When an S3 object is replicated using SRR, the metadata, Access Control Lists (ACL), and object tags associated with the object are also part of the replication.

Once SRR is configured on a source bucket, any changes to the object, metadata, ACLs, or object tags trigger a new replication to the destination bucket.

S3 Analytics

Can run analytics on data stored on Amazon S3.

This includes data lakes, IoT streaming data, machine learning, and artificial intelligence.

The following strategies can be used:

|  |  |
| --- | --- |
| S3 Analytics Strategies | Service Used |
| Data Lake Concept | Athena, Redshift Spectrum, QuickSight |
| IoT Streaming Data Repository | Kinesis Firehose |
| ML and AI Storage | Rekognition, Lex, MXNet |
| Storage Class Analysis | S3 Management Analytics |

S3 Inventory

You can use S3 Inventory to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs.

Amazon S3 inventory provides comma-separated values (CSV), [Apache optimized row columnar (ORC)](https://orc.apache.org/) or [Apache Parquet (Parquet)](https://parquet.apache.org/) output files that list your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or a shared prefix (that is, objects that have names that begin with a common string).

Monitoring and Reporting

Amazon[CloudWatch](https://digitalcloud.training/amazon-cloudwatch/) metrics for Amazon S3 can help you understand and improve the performance of applications that use Amazon S3. There are several ways that you can use CloudWatch with Amazon S3.

Daily storage metrics for buckets ‐ Monitor bucket storage using CloudWatch, which collects and processes storage data from Amazon S3 into readable, daily metrics. These storage metrics for Amazon S3 are reported once per day and are provided to all customers at no additional cost.

Request metrics ‐ Monitor Amazon S3 requests to quickly identify and act on operational issues. The metrics are available at 1-minute intervals after some latency to process. These CloudWatch metrics are billed at the same rate as the Amazon CloudWatch custom metrics.

Replication metrics ‐ Monitor the total number of S3 API operations that are pending replication, the total size of objects pending replication, and the maximum replication time to the destination Region. Only replication rules that have S3 Replication Time Control (S3 RTC) enabled will publish replication metrics.

Logging and Auditing

You can record the actions that are taken by users, roles, or AWS services on Amazon S3 resources and maintain log records for auditing and compliance purposes.

To do this, you can use [Amazon S3 server access logging](https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html), [AWS CloudTrail logs](https://docs.aws.amazon.com/AmazonS3/latest/dev/cloudtrail-logging.html), or a combination of both.

AWS recommend that you use AWS CloudTrail for[logging bucket and object-level actions](https://docs.aws.amazon.com/AmazonS3/latest/dev/logging-with-S3.html) for your Amazon S3 resources.

Server access logging provides detailed records for the requests that are made to a bucket.

This information can be used for auditing.

You must not set the bucket being logged to be the destination for the logs as this creates a logging loop and the bucket will grow exponentially.

S3 performance guidelines

AWS provide some[performance guidelines](https://docs.aws.amazon.com/AmazonS3/latest/dev/optimizing-performance-guidelines.html) for Amazon S3. These are summarized here:

Measure Performance – When optimizing performance, look at network throughput, CPU, and DRAM requirements. Depending on the mix of demands for these different resources, it might be worth evaluating different Amazon EC2 instance types.

Scale Storage Connections Horizontally – You can achieve the best performance by issuing multiple concurrent requests to Amazon S3. Spread these requests over separate connections to maximize the accessible bandwidth from Amazon S3.

Use Byte-Range Fetches – Using the Range HTTP header in a GET Object request, you can fetch a byte-range from an object, transferring only the specified portion. You can use concurrent connections to Amazon S3 to fetch different byte ranges from within the same object. This helps you achieve higher aggregate throughput versus a single whole-object request. Fetching smaller ranges of a large object also allows your application to improve retry times when requests are interrupted.

Retry Requests for Latency-Sensitive Applications – Aggressive timeouts and retries help drive consistent latency. Given the large scale of Amazon S3, if the first request is slow, a retried request is likely to take a different path and quickly succeed. The AWS SDKs have configurable timeout and retry values that you can tune to the tolerances of your specific application.

Combine Amazon S3 (Storage) and Amazon EC2 (Compute) in the Same AWS Region – Although S3 bucket names are globally unique, each bucket is stored in a Region that you select when you create the bucket. To optimize performance, we recommend that you access the bucket from Amazon EC2 instances in the same AWS Region when possible. This helps reduce network latency and data transfer costs.

Use Amazon S3 Transfer Acceleration to Minimize Latency Caused by Distance – Amazon S3 Transfer Acceleration manages fast, easy, and secure transfers of files over long geographic distances between the client and an S3 bucket. Transfer Acceleration takes advantage of the globally distributed edge locations in Amazon CloudFront. As the data arrives at an edge location, it is routed to Amazon S3 over an optimized network path. Transfer Acceleration is ideal for transferring gigabytes to terabytes of data regularly across continents. It’s also useful for clients that upload to a centralized bucket from all over the world.

Glacier

Glacier is an archiving storage solution for infrequently accessed data.

There are three storage tiers:

S3 Glacier Instant Retrieval

Data retrieval in milliseconds with the same performance as S3 Standard

Designed for durability of 99.999999999% of objects across multiple Availability Zones

Data is resilient in the event of the destruction of one entire Availability Zone

Designed for 99.9% data availability each year

128 KB minimum object size

Backed with the[Amazon S3 Service Level Agreement](https://aws.amazon.com/s3/sla/) for availability

S3 PUT API for direct uploads to S3 Glacier Instant Retrieval, and S3 Lifecycle management for automatic migration of objects

S3 Glacier Flexible Retrieval (Formerly S3 Glacier)

Designed for durability of 99.999999999% of objects across multiple Availability Zones

Data is resilient in the event of one entire Availability Zone destruction

Supports SSL for data in transit and encryption of data at rest

Ideal for backup and disaster recovery use cases when large sets of data occasionally need to be retrieved in minutes, without concern for costs

Configurable retrieval times, from minutes to hours, with free bulk retrievals

S3 PUT API for direct uploads to S3 Glacier Flexible Retrieval, and S3 Lifecycle management for automatic migration of objects

Amazon S3 Glacier Deep Archive (S3 Glacier Deep Archive)

Designed for durability of 99.999999999% of objects across multiple Availability Zones

Lowest cost storage class designed for long-term retention of data that will be retained for 7-10 years

Ideal alternative to magnetic tape libraries

Retrieval time within 12 hours

S3 PUT API for direct uploads to S3 Glacier Deep Archive, and S3 Lifecycle management for automatic migration of objects

The key difference between the top tiers is that Deep Archive is lower cost, but retrieval times are much longer (12 hours).

The S3 Glacier tier has configurable retrieval times from minutes to hours (you pay accordingly).

Archived objects are not available for real time access and you need to submit a retrieval request.

Glacier must complete a job before you can get its output.

Requested archival data is copied to S3 One Zone-IA.

Following retrieval you have 24 hours to download your data.

You cannot specify Glacier as the storage class at the time you create an object.

Glacier is designed to sustain the loss of two facilities.

Glacier automatically encrypts data at rest using AES 256 symmetric keys and supports secure transfer of data over SSL.

Glacier may not be available in all AWS regions.

Glacier objects are visible through S3 only (not Glacier directly).

Glacier does not archive object metadata; you need to maintain a client-side database to maintain this information.

Archives can be 1 byte up to 40TB.

Glacier file archives of 1 byte – 4 GB can be performed in a single operation.

Glacier file archives from 100MB up to 40TB can be uploaded to Glacier using the multipart upload API.

Uploading archives is synchronous.

Downloading archives is asynchronous.

The contents of an archive that has been uploaded cannot be modified.

You can upload data to Glacier using the CLI, SDKs or APIs – you cannot use the AWS Console.

Glacier adds 32-40KB (indexing and archive metadata) to each object when transitioning from other classes using lifecycle policies.

AWS recommends that if you have lots of small objects they are combined in an archive (e.g. zip file) before uploading.

A description can be added to archives, no other metadata can be added.

Glacier archive IDs are added upon upload and are unique for each upload.

Archive retrieval:

Expedited is 1-5 minutes retrieval (most expensive).

Standard is 3.5 hours retrieval (cheaper, 10GB data retrieval free per month).

Bulk retrieval is 5-12 hours (cheapest, use for large quantities of data).

You can retrieve parts of an archive.

When data is retrieved it is copied to S3 and the archive remains in Glacier and the storage class therefore does not change.

AWS SNS can send notifications when retrieval jobs are complete.

Retrieved data is available for 24 hours by default (can be changed).

To retrieve specific objects within an archive you can specify the byte range (Range) in the HTTP GET request (need to maintain a DB of byte ranges).

Glacier Charges:

There is no charge for data transfer between EC2 and Glacier in the same region.

There is a charge if you delete data within 90 days.

When you restore you pay for:

The Glacier archive.

The requests.

The restored data on S3.

Amazon EBS

EBS is the Amazon Elastic Block Store.

EBS volumes are network attached storage that can be attached to EC2 instances.

EBS volume data persists independently of the life of the instance.

EBS volumes do not need to be attached to an instance.

You can attach multiple EBS volumes to an instance.

You can attach an EBS volume to multiple instances with specific constraints.

For most use cases where you need a shared volume across EC2 instances use Amazon EFS.

EBS volume data is replicated across multiple servers in an AZ.

EBS volumes must be in the same AZ as the instances they are attached to.

EBS is designed for an annual failure rate of 0.1%-0.2% & an SLA of 99.95%.

Termination protection is turned off by default and must be manually enabled (keeps the volume/data when the instance is terminated).

Root EBS volumes are deleted on termination by default.

Extra non-boot volumes are not deleted on termination by default.

The behavior can be changed by altering the “DeleteOnTermination” attribute.

You can now create AMIs with encrypted root/boot volumes as well as data volumes (you can also use separate CMKs per volume).

Volume sizes and types can be upgraded without downtime (except for magnetic standard).

Elastic Volumes allow you to increase volume size, adjust performance, or change the volume type while the volume is in use.

To migrate volumes between AZ’s create a snapshot then create a volume in another AZ from the snapshot (possible to change size and type).

Auto-enable IO setting prevents the stopping of IO to a disk when AWS detects inconsistencies.

The root device is created under /dev/sda1 or /dev/xvda.

Magnetic EBS is for workloads that need throughput rather than IOPS.

Throughput optimized EBS volumes cannot be a boot volume.

Each instance that you launch has an associated root device volume, either an Amazon EBS volume or an instance store volume.

You can use block device mapping to specify additional EBS volumes or instance store volumes to attach to an instance when it’s launched.

You can also attach additional EBS volumes to a running instance.

You cannot decrease an EBS volume size.

When changing volumes the new volume must be at least the size of the current volume’s snapshot.

Images can be made public but not if they’re encrypted.

AMIs can be shared with other accounts.

You can have up to 5,000 EBS volumes by default.

You can have up to 10,000 snapshots by default.

Instance Store

An instance store provides temporary (non-persistent) block-level storage for your instance.

This is different to EBS which provides persistent storage but is also a block storage service that can be a root or additional volume.

Instance store 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, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers.

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.

The instance type determines the size of the instance store available, and the type of hardware used for the instance store volumes.

Instance store volumes are included as part of the instance’s usage cost.

Some instance types use NVMe or SATA-based solid-state drives (SSD) to deliver high random I/O performance.

This is a good option when you need storage with very low latency, but you don’t need the data to persist when the instance terminates, or you can take advantage of fault-tolerant architectures.

EXAM TIP: Instance stores offer very high performance and low latency. If you can afford to lose an instance, i.e. you are replicating your data, these can be a good solution for high performance/low latency requirements. Look out for questions that mention distributed or replicated databases that need high I/O. Also, remember that the cost of instance stores is included in the instance charges so it can also be more cost-effective than EBS Provisioned IOPS.

EBS vs Instance Store

EBS-backed means the root volume is an EBS volume and storage is persistent.

Instance store-backed means the root volume is an instance store volume and storage is not persistent.

On an EBS-backed instance, the default action is for the root EBS volume to be deleted upon termination.

Instance store volumes are sometimes called Ephemeral storage (non-persistent).

Instance store backed instances cannot be stopped. If the underlying host fails the data will be lost.

Instance store volume root devices are created from AMI templates stored on S3.

EBS backed instances can be stopped. You will not lose the data on this instance if it is stopped (persistent).

EBS volumes can be detached and reattached to other EC2 instances.

EBS volume root devices are launched from AMI’s that are backed by EBS snapshots.

Instance store volumes cannot be detached/reattached.

When rebooting the instances for both types data will not be lost.

By default, both root volumes will be deleted on termination unless you configured otherwise.

EBS Volume Types

SSD, General Purpose – gp2/gp3:

Volume size from 1 GiB to 16 TiB.

Up to 16,000 IOPS per volume.

Performance:

3 IOPS/GiB for gp2.

Up to 500 IOPS/GiB for gp3.

Can be a boot volume.

EBS multi-attach not supported.

Use cases:

Low-latency interactive apps.

Development and test environments.

SSD, Provisioned IOPS – io1/io2:

More than 16,000 IOPS.

Up to 64,000 IOPS per volume (Nitro instances).

Up to 32,000 IOPS per volume for other instance types.

Performance:

Up to 50 IOPS/GiB for io1.

Up to 500 IOPS/Gib for io2.

Can be a boot volume.

EBS multi-attach is supported.

Use cases:

Workloads that require sustained IOPS performance or more than 16,000 IOPS.

I/O-intensive database workloads.

HDD, Throughput Optimized – (st1):

Frequently accessed, throughput intensive workloads with large datasets and large I/O sizes, such as MapReduce, Kafka, log processing, data warehouse, and ETL workloads.

Throughput measured in MiB/s and includes the ability to burst up to 250 MiB/s per TB, with a baseline throughput of 40 MB/s per TB and a maximum throughput of 500 MiB/s per volume.

Cannot be a boot volume.

EBS multi-attach not supported.

HDD, Cold – (sc1):

Lowest cost storage – cannot be a boot volume.

Less frequently accessed workloads with large, cold datasets.

These volumes can burst up to 80 MiB/s per TiB, with a baseline throughput of 12 MiB/s.

Cannot be a boot volume.

EBS multi-attach not supported.

EBS optimized instances:

Dedicated capacity for Amazon EBS I/O.

EBS-optimized instances are designed for use with all EBS volume types.

Max bandwidth: 400 Mbps – 12000 Mbps.

IOPS: 3000 – 65000.

GP-SSD within 10% of baseline and burst performance 99.9% of the time.

PIOPS within 10% of baseline and burst performance 99.9% of the time.

Additional hourly fee.

Available for select instance types.

Some instance types have EBS-optimized enabled by default.

The following EBS volumes appear most often on the AWS exams:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Volume Type | EBS Provisioned IOPS SSD (io1/io2) | EBS General Purpose SSD (gp2/gp3) | Throughput Optimized HDD (st1) | Cold HDD (sc1) |
| Short Description | Highest performance SSD volume designed for latency-sensitive transactional workloads | General Purpose SSD volume that balances price performance for a wide variety of transactional workloads | Low-cost HDD volume, designed for frequently accessed. Throughput intensive workloads | Lowest cost HDD volume designed for less frequently accessed workloads |
| Use Cases | I/O-intensive NoSQL and relational databases | Boot volumes, low-latency interactive apps, dev & test | Big-data, data warehouses, log processing | Colder data requiring fewer scans per day |
| Volume Size | 4 GiB – 16 TiB | 1 GiB – 16 TiB | 125 GB – 16 TiB | 125 GB – 16 TiB |
| Max IOPS\*\* / Volume | 64,000 | 16,000 | 500 | 250 |
| Max Throughput\*\*\*Volume | 1,000 MiB/s | 250 MiB/s (gp2)    1000 MiB/s (gp3) | 500 MiB/s | 250 MiB/s |
| Can be boot volume? | Yes | Yes | No | No |
| EBS Multi-attach | Supported | Not Supported | Not Supported | Not Supported |

Amazon EBS Snapshots

Snapshots capture a point-in-time state of an instance.

Cost-effective and easy backup strategy.

Share data sets with other users or accounts.

Can be used to migrate a system to a new AZ or region.

Can be used to convert an unencrypted volume to an encrypted volume.

Snapshots are stored on Amazon S3.

Does not provide granular backup (not a replacement for backup software).

If you make periodic snapshots of a volume, the snapshots are incremental, which means that only the blocks on the device that have changed after your last snapshot are saved in the new snapshot.

Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot to restore the volume.

Snapshots can only be accessed through the EC2 APIs.

EBS volumes are AZ specific, but snapshots are region specific.

Volumes can be created from EBS snapshots that are the same size or larger.

Snapshots can be taken of non-root EBS volumes while running.

To take a consistent snapshot, writes must be stopped (paused) until the snapshot is complete. if this is not possible the volume needs to be detached; or if it’s an EBS root volume the instance must be stopped.

To lower storage costs on S3 a full snapshot and subsequent incremental updates can be created.

You are charged for data traffic to S3 and storage costs on S3.

You are billed only for the changed blocks.

Deleting a snapshot removes only the data not needed by any other snapshot.

You can resize volumes through restoring snapshots with different sizes (configured when taking the snapshot).

Snapshots can be copied between regions (and be encrypted). Images are then created from the snapshot in the other region which creates an AMI that can be used to boot an instance.

You can create volumes from snapshots and choose the availability zone within the region.

Encryption

You can encrypt both the boot and data volumes of an EC2 instance. 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 created from the volume.

All volumes created from those snapshots.

Encryption is supported by all EBS volume types.

Expect the same IOPS performance on encrypted volumes as on unencrypted volumes.

All instance families support encryption.

Amazon EBS encryption is available on the instance types listed below:

General purpose: A1, M3, M4, M5, M5a, M5ad, M5d, T2, T3, and T3a.

Compute optimized: C3, C4, C5, C5d, and C5n.

Memory optimized: cr1.8xlarge, R3, R4, R5, R5a, R5ad, R5d, u-6tb1.metal, u-9tb1.metal, u-12tb1.metal, X1, X1e, and z1d.

Storage optimized: D2, h1.2xlarge, h1.4xlarge, I2, and I3.

Accelerated computing: F1, G2, G3, G4, P2, and P3.

EBS encrypts your volume with a data key using the industry-standard AES-256 algorithm.

Your data key is stored on-disk with your encrypted data, but not before EBS encrypts it with your CMK. Your data key never appears on disk in plaintext. .

The same data key is shared by snapshots of the volume and any subsequent volumes created from those snapshots.

Snapshots of encrypted volumes are encrypted automatically.

EBS volumes restored from encrypted snapshots are encrypted automatically.

EBS volumes created from encrypted snapshots are also encrypted.

You can share snapshots, but if they’re encrypted it must be with a custom CMK key.

You can check the encryption status of your EBS volumes with AWS Config.

There is no direct way to change the encryption state of a volume.

Either create an encrypted volume and copy data to it or take a snapshot, encrypt it, and create a new encrypted volume from the snapshot.

To encrypt a volume or snapshot you need an encryption key, these are customer managed keys (CMK), and they are managed by the AWS Key Management Service (KMS).

A default CMK key is generated for the first encrypted volumes.

Subsequent encrypted volumes will use their own unique key (AES 256 bit).

The CMK used to encrypt a volume is used by any snapshots and volumes created from snapshots.

You cannot share encrypted volumes created using a default CMK key.

You cannot change the CMK key that is used to encrypt a volume.

You must create a copy of the snapshot and change encryption keys as part of the copy.

This is required to be able to share the encrypted volume.

By default only the account owner can create volumes from snapshots.

You can share unencrypted snapshots with the AWS community by making them public.

You can also share unencrypted snapshots with other AWS accounts by making them private and selecting the accounts to share them with.

You cannot make encrypted snapshots public.

You can share encrypted snapshots with other AWS accounts using a non-default CMK key and configuring cross-account permissions to give the account access to the key, mark as private and configure the account to share with.

The receiving account must copy the snapshot before they can then create volumes from the snapshot.

It is recommended that the receiving account re-encrypt the shared and encrypted snapshot using their own CMK key.

The following information applies to snapshots:

Snapshots are created asynchronously and are incremental.

You can copy unencrypted snapshots (optionally encrypt).

You can copy an encrypted snapshot (optionally re-encrypt with a different key).

Snapshot copies receive a new unique ID.

You can copy within or between regions.

You cannot move snapshots, only copy them.

You cannot take a copy of a snapshot when it is in a “pending” state, it must be “complete”.

S3 Server Side Encryption (SSE) protects data in transit while copying.

User defined tags are not copied.

You can have up to 5 snapshot copy requests running in a single destination per account.

You can copy Import/Export service, AWS Marketplace, and AWS Storage Gateway snapshots.

If you try to copy an encrypted snapshot without having access to the encryption keys it will fail silently (cross-account permissions are required).

Copying snapshots may be required for:

Creating services in other regions.

DR – the ability to restore from snapshot in another region.

Migration to another region.

Applying encryption.

Data retention.

To take application-consistent snapshots of RAID arrays:

Stop the application from writing to disk.

Flush all caches to the disk.

Freeze the filesystem.

Unmount the RAID array.

Shut down the associated EC2 instance.

AMIs

An Amazon Machine Image (AMI) is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud (“EC2”).

An AMI includes the following:

A template for the root volume for the instance (for example, an operating system, an application server, and applications).

Launch permissions that control which AWS accounts can use the AMI to launch instances.

A block device mapping that specifies the volumes to attach to the instance when it’s launched.

AMIs are either instance store-backed or EBS-backed.

Instance store-backed:

Launch an EC2 instance from an AWS instance store-backed AMI.

Update the root volume as required.

Create the AMI which will upload to a user specified S3 bucket (user bucket).

Register the AMI with EC2 (creates another EC2 controlled S3 image).

To make changes update the source then deregister and reregister.

Upon launch the image is copied to the EC2 host.

Deregister an image when the AMI is not needed anymore (does not affect existing instances created from the AMI).

Instance store-backed volumes can only be created at launch time.

EBS-backed:

Must stop the instance to create a consistent image and then create the AMI.

AWS registers the AMIs automatically.

During creation AWS creates snapshots of all attached volumes – there is no need to specify a bucket, but you will be charged for storage on S3.

You cannot delete the snapshot of the root volume if the AMI is registered (deregister and delete).

You can now create AMIs with encrypted root/boot volumes as well as data volumes (can also use separate CMKs per volume).

Copying AMIs:

You can copy an Amazon Machine Image (AMI) within or across an AWS region using the AWS Management Console, the AWS Command Line Interface 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.

Deployment and Provisioning

Termination protection is turned off by default and must be manually enabled (keeps the volume/data when the instance is terminated).

Root EBS volumes are deleted on termination by default.

Extra non-boot volumes are not deleted on termination by default.

The behavior can be changed by altering the “DeleteOnTermination” attribute.

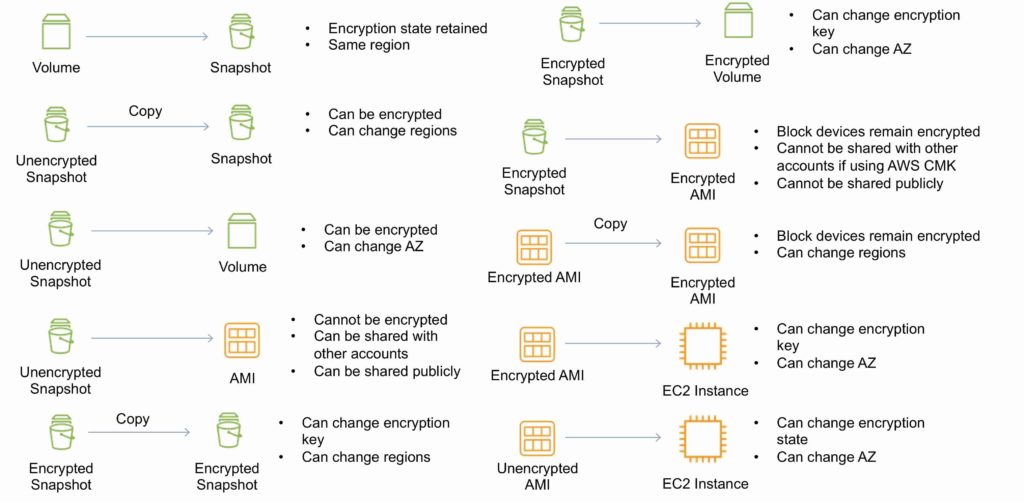
Volume sizes and types can be upgraded without downtime (except for magnetic standard).

Elastic Volumes allow you to increase volume size, adjust performance, or change the volume type while the volume is in use.

To migrate volumes between AZ’s create a snapshot then create a volume in another AZ from the snapshot (possible to change size and type).

EBS Copying, Sharing and Encryption Methods

The following diagram aims to articulate the various possible options for copying EBS volumes, sharing AMIs and snapshots and applying encryption:



RAID

RAID can be used to increase IOPS.

RAID 0 = 0 striping – data is written across multiple disks and increases performance but no redundancy.

RAID 1 = 1 mirroring – creates 2 copies of the data but does not increase performance, only redundancy.

RAID 10 = 10 combination of RAID 1 and 2 resulting in increased performance and redundancy (at the cost of additional disks).

You can configure multiple striped gp2 or standard volumes (typically RAID 0).

You can configure multiple striped PIOPS volumes (typically RAID 0).

RAID is configured through the guest OS.

EBS optimized EC2 instances are another way of increasing performance.

Ensure the EC2 instance can handle the bandwidth required for the increased performance.

Use EBS optimized instances or instances with a 10 Gbps network interface.

Not recommended to use RAID for root/boot volumes.

Monitoring and Reporting

Amazon Elastic Block Store (Amazon EBS) sends data points to CloudWatch for[several metrics](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using_cloudwatch_ebs.html).

A few specific metrics to understand for the exam:

DiskReadBytes / DiskWriteBytes:

Relates to Instance Store volumes NOT to EBS.

Included in the AWS/EC2 namespace.

VolumeReadBytes / VolumeWriteBytes:

Relates to the EBS volume.

Included in the AWS/EBS namespace.

There are two types of Amazon CloudWatch monitoring available for Amazon EBS volumes:

Basic – Data is available automatically in 5-minute periods at no charge. This includes data for the root device volumes for EBS-backed instances.

Detailed – Provisioned IOPS SSD (io1) volumes automatically send one-minute metrics to CloudWatch.

Amazon EBS General Purpose SSD (gp2), Throughput Optimized HDD (st1) , Cold HDD (sc1), and Magnetic (standard) volumes automatically send five-minute metrics to CloudWatch.

Provisioned IOPS SSD (io1) volumes automatically send one-minute metrics to CloudWatch. Data is only reported to CloudWatch when the volume is attached to an instance.

Volume[status checks](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring-volume-status.html) enable you to better understand, track, and manage potential inconsistencies in the data on an Amazon EBS volume.

|  |  |  |
| --- | --- | --- |
| Volume Status | I/O Enabled Status | I/O performance status (only available for Provisioned IOPS volumes) |
| ok | Enabled (I/O Enabled or I/O Auto-Enabled) | Normal (Volume performance is expected) |
| warning | Enabled (I/O Enabled or I/O Auto-Enabled)    Disabled (Volume is offline and pending recovery or is waiting for the user to enable I/O). | Degraded (Volume performance is below expectations)    Severely Degraded (Volume performance is well below expectations) |
| impaired | Enabled (I/O Enabled or I/O Auto-Enabled)    Disabled (Volume is offline and pending recovery, or is waiting for the user to enable I/O) | Stalled (Volume performance is severely impacted)    Not Available (Unable to determine I/O performance because I/O is disabled) |
| insufficient-data | Enabled (I/O Enabled or I/O Auto-Enabled)    Insufficient Data | Insufficient Data |

Logging and Auditing

Amazon EC2 and Amazon EBS are integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon EC2 and Amazon EBS.

CloudTrail captures all API calls for Amazon EC2 and Amazon EBS as events, including calls from the console and from code calls to the APIs.

Amazon Data Lifecycle Manager (DLM)

Automates the creation, retention, and deletion of EBS snapshots and EBS-backed AMIs.

Protect valuable data by enforcing a regular backup schedule.

Create standardized AMIs that can be refreshed at regular intervals.

Retain backups as required by auditors or internal compliance.

Reduce storage costs by deleting outdated backups.

Create disaster recovery backup policies that back up data to isolated accounts.

EBS Limits (per region)

|  |  |
| --- | --- |
| Name | Default Limit |
| Provisioned IOPS | 300,000 |
| Provisioned IOPS (SSD) volume storage (TiB) | 300 |
| General Purpose (SSD) volume storage (TiB) | 300 |
| Magnetic volume storage (TiB) | 300 |
| Max Cold HDD (sc1) Storage in (TiB) | 300 |
| Max Throughput Optimized HDD (st1) Storage (TiB) | 300 |

Amazon EFS

Amazon EFS is a fully managed service for hosting Network File System (NFS) filesystems in the cloud.

It is an implementation of a NFS file share and is accessed using the NFS protocol.

It provides elastic storage capacity and pay for what you use (in contrast to Amazon EBS with which you pay for what you provision).

You can configure mount-points in one, or many, AZs.

You can mount an AWS EFS filesystem from on-premises systems ONLY if you are using AWS Direct Connect or a VPN connection.

Typical use cases include big data and analytics, media processing workflows, content management, web serving, home directories etc.

Uses a pay for what you use model with no pre-provisioning required.

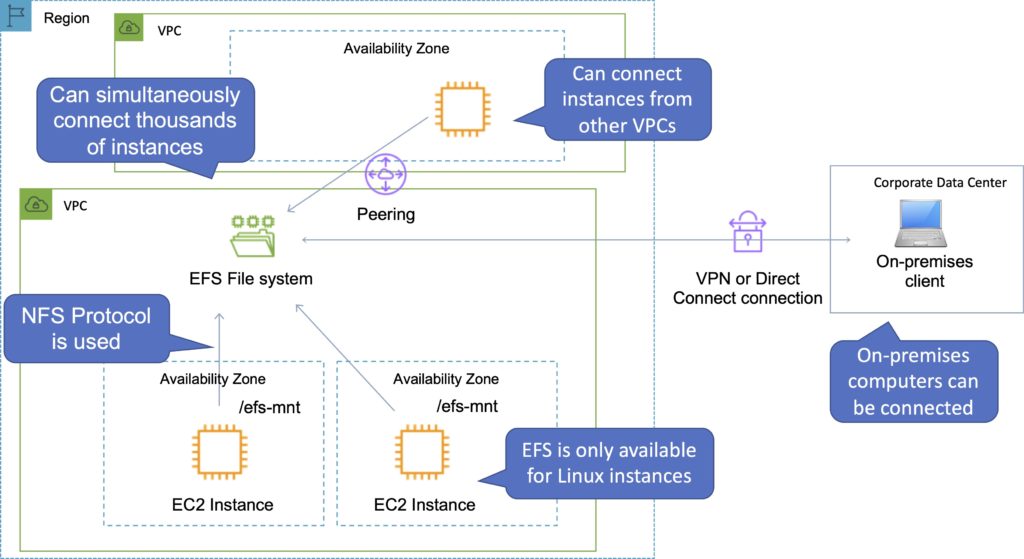
AWS EFS can scale up to petabytes.

AWS EFS is elastic and grows and shrinks as you add and remove data.

You can concurrently connect up to thousands of Amazon EC2 instances, from multiple AZs.

A file system can be accessed concurrently from all AZs in the region where it is located.

The following diagram depicts the various options for mounting an EFS filesystem:



Access to AWS EFS file systems from on-premises servers can be enabled via AWS Direct Connect or AWS VPN.

You mount an AWS EFS file system on your on-premises Linux server using the standard Linux mount command for mounting a file system via the NFS protocol.

The Amazon VPC of the connecting instance must have DNS hostnames enabled.

EFS provides a file system interface, file system access semantics (such as strong consistency and file locking).

Data is stored across multiple AZs within a region.

Read after write consistency.

Need to create mount targets and choose AZs to include (recommended to include all AZ’s).

Instances can be behind an Elastic Load Balancer (ELB).

Amazon EFS is compatible with all Linux-based AMIs for Amazon EC2.

Using the EFS-to-EFS Backup solution, you can schedule automatic incremental backups of your Amazon EFS file system.

The following table provides a comparison of the storage characteristics of EFS vs EBS:

|  |  |  |
| --- | --- | --- |
|  | Amazon EFS | Amazon EBS Provisioned IOPS |
| Availability and durability | Data is stored redundantly across multiple AZs | Data is stored redundantly in a single AZ |
| Access | Up to thousands of Amazon EC2 instances, from multiple AZs, can connect concurrently to a file system | A single Amazon EC2 instance in a single AZ can connect to a file system |
| Use cases | Big data and analytics, media processing and workflows, content management, web serving and home directories | Boot volumes, transactional and NoSQL databases, data warehousing and ETL |

Backups and Lifecycle Management

Automatic backups are enabled by default and use AWS Backup.

Lifecycle management moves files that have not been accessed for a period of time to the EFS Infrequent Access Storage class.

Amazon EFS Performance

There are two performance modes:

“General Purpose” performance mode is appropriate for most file systems.

“Max I/O” performance mode is optimized for applications where tens, hundreds, or thousands of EC2 instances are accessing the file system.

Amazon EFS is designed to burst to allow high throughput levels for periods of time.

There are two throughput modes:

“Bursting” – throughput scales with file system size.

“Provisioned” – Throughput is fixed at the specified amount.

Amazon EFS file systems are distributed across an unconstrained number of storage servers, enabling file systems to grow elastically to petabyte scale and allowing massively parallel access from Amazon EC2 instances to your data.

This distributed data storage design means that multithreaded applications and applications that concurrently access data from multiple Amazon EC2 instances can drive substantial levels of aggregate throughput and IOPS.

The table below compares high-level performance and storage characteristics for AWS’s file  (EFS) and block (EBS) cloud storage offerings:

|  |  |  |
| --- | --- | --- |
|  | Amazon EFS | Amazon EBS Provisioned IOPS |
| Per-operation latency | Low, consistent latency | Lowest, consistent latency |
| Throughput scale | 10+ GB per second | Up to 2 GB per second |

Amazon EFS Encryption

EFS offers the ability to encrypt data at rest and in transit.

Encryption keys are managed by the AWS Key Management Service (KMS).

Encryption in transit:

Data encryption in transit uses Transport Layer Security (TLS) 1.2 to encrypt data sent between your clients and EFS file systems.

Encryption in transit is enabled when mounting the file system.

Encryption at rest:

Enable encryption at rest in the EFS console or by using the AWS CLI or SDKs.

Encryption at rest MUST be enabled at file system creation time.

Data encrypted at rest is transparently encrypted while being written, and transparently decrypted while being read.

Encryption of data at rest and of data in transit can be configured together or separately.

Amazon EFS Access Control

When you create a file system, you create endpoints in your VPC called “mount targets”.

When mounting from an EC2 instance, your file system’s DNS name, which you provide in your mount command, resolves to a mount target’s IP address.

You can control who can administer your file system using IAM (user-based and resource-based policies)

You can control the NFS clients that can access your file systems (resource-based policies).

You can control access to files and directories with POSIX-compliant user and group-level permissions.

POSIX permissions allow you to restrict access from hosts by user and group.

EFS Security Groups act as a firewall, and the rules you add define the traffic flow.

Monitoring and Reporting

The Amazon EFS console shows the following monitoring information for your file systems:

The current metered size.

The number of mount targets.

The lifecycle state.

Amazon EFS reports metrics for Amazon CloudWatch.  A few useful metrics are:

TotalIOBytes – use the daily Sum statistic to determine throughput.

ClientConnections – use the daily Sum statistic to track the number of connections from EC2 instances.

BurstCreditBalance – monitor the burst credit balance.

Logging and Auditing

Amazon EFS is integrated with AWS CloudTrail.

CloudTrail captures all API calls for Amazon EFS as events, including calls from the Amazon EFS console and from code calls to Amazon EFS API operations.

AWS Lambda

AWS Lambda lets you run code as functions without provisioning or managing servers.

Lambda-based applications are composed of functions triggered by events.

With serverless computing, your application still runs on servers, but all the server management is done by AWS.

You cannot log in to the compute instances that run Lambda functions or customize the operating system or language runtime.

Lambda functions:

Consist of code and any associated dependencies.

Configuration information is associated with the function.

You specify the configuration information when you create the function.

API provided for updating configuration data.

You specify the amount of memory you need allocated to your Lambda functions.

AWS Lambda allocates CPU power proportional to the memory you specify using the same ratio as a general purpose EC2 instance type.

Functions can access:

AWS services or non-AWS services.

AWS services running in VPCs (e.g. RedShift, Elasticache, RDS instances).

Non-AWS services running on EC2 instances in an AWS VPC.

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.

You can request additional memory in 1 MB increments from 128 MB to 10240 MB.

There is a maximum execution timeout.

Max is 15 minutes (900 seconds), default is 3 seconds.

You pay for the time it runs.

Lambda terminates the function at the timeout.

Code is invoked using API calls made using AWS SDKs.

Lambda assumes an IAM role when it executes the function.

AWS Lambda stores code in Amazon S3 and encrypts it at rest.

Lambda provides continuous scaling – scales out not up.

Lambda scales concurrently executing functions up to your default limit (1000).

Lambda can scale up to tens of thousands of concurrent executions.

Lambda functions are serverless and independent, 1 event = 1 function.

Functions can trigger other functions so 1 event can trigger multiple functions.

Use cases fall within the following categories:

Using Lambda functions with AWS services as event sources.

On-demand Lambda function invocation over HTTPS using Amazon API Gateway (custom REST API and endpoint).

On-demand Lambda function invocation using custom applications (mobile, web apps, clients) and AWS SDKs, AWS Mobile SDKs, and the AWS Mobile SDK for Android.

Scheduled events can be configured to run code on a scheduled basis through the AWS Lambda Console.

Invoking Lambda Functions

You can invoke Lambda functions directly with the Lambda console, the Lambda API, the AWS SDK, the AWS CLI, and AWS toolkits.

You can also configure other AWS services to invoke your function, or you can configure Lambda to read from a stream or queue and invoke your function.

When you invoke a function, you can choose to invoke it synchronously or asynchronously.

Other AWS services and resources invoke your function directly.

For example, you can configure CloudWatch Events to invoke your function on a timer, or you can configure Amazon S3 to invoke your function when an object is created.

Each service varies in the method it uses to invoke your function, the structure of the event, and how you configure it.

Synchronous invocation

You wait for the function to process the event and return a response.

When you invoke a function synchronously, Lambda runs the function and waits for a response.

When the function execution ends, Lambda returns the response from the function’s code with additional data, such as the version of the function that was executed. To invoke a function synchronously with the AWS CLI, use the invoke command.

$ aws lambda invoke –function-name my-function –payload ‘{ “key”: “value” }’ response.json { “ExecutedVersion”: “$LATEST”, “StatusCode”: 200 }

Asynchronous invocation

When you invoke a function asynchronously, you don’t wait for a response from the function code.

For asynchronous invocation, Lambda handles retries and can send invocation records to a destination.

For asynchronous invocation, Lambda places the event in a queue and returns a success response without additional information. A separate process reads events from the queue and sends them to your function. To invoke a function asynchronously, set the invocation type parameter to Event.

$ aws lambda invoke --function-name my-function --invocation-type Event --payload '{ "key": "value" }' response.json { "StatusCode": 202 }

The output file (response.json) doesn’t contain any information but is still created when you run this command. If Lambda can’t add the event to the queue, the error message appears in the command output.

Event source mappings

Lambda is an event-driven compute service where AWS Lambda runs code in response to events such as changes to data in an S3 bucket or a DynamoDB table.

An event source is an AWS service or developer-created application that produces events that trigger an AWS Lambda function to run.

You can use event source mappings to process items from a stream or queue in services that don’t invoke Lambda functions directly.

Supported AWS event sources include:

[Amazon S3](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-s3).

[Amazon DynamoDB](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-dynamo-db).

[Amazon Kinesis Data Streams](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-kinesis-streams).

[Amazon Simple Notification Service](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-sns).

[Amazon Simple Email Service](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-ses).

[Amazon Simple Queue Service](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-sqs).

[Amazon Cognito](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cognito).

[AWS CloudFormation](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudformation).

[Amazon CloudWatch Logs](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudwatch-logs).

[Amazon CloudWatch Events](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudwatch-events).

[AWS CodeCommit](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-codecommit).

[AWS Config](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-config).

[Amazon Alexa](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-echo).

[Amazon Lex](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-lex).

[Amazon API Gateway](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-api-gateway).

[AWS IoT Button](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-iot-button).

[Amazon CloudFront](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudfront).

[Amazon Kinesis Data Firehose](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-kinesis-firehose).

[Other Event Sources: Invoking a Lambda Function On Demand](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#api-gateway-with-lambda).

Other event sources can invoke Lambda functions on-demand.

Applications need permissions to invoke Lambda functions.

Lambda can run code in response to HTTP requests using Amazon API gateway or API calls made using the AWS SDKs.

Services that Lambda reads events from:

[Amazon Kinesis](https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html)

[Amazon DynamoDB](https://docs.aws.amazon.com/lambda/latest/dg/with-ddb.html)

[Amazon Simple Queue Service](https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html)

An event source mapping uses permissions in the function’s [execution role](https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.html) to read and manage items in the event source.

Permissions, event structure, settings, and polling behavior vary by event source.

To process items from a stream or queue, you can create an event source mapping.

Each event that your function processes can contain hundreds or thousands of items.

The configuration of the event source mapping for stream-based services (DynamoDB, Kinesis), and Amazon SQS, is made on the Lambda side.

Note: for other services such as Amazon S3 and SNS, the function is invoked asynchronously, and the configuration is made on the source (S3/SNS) rather than Lambda.

Lambda Versions

Versioning means you can have multiple versions of your function.

You can use versions to manage the deployment of your AWS Lambda functions.  For example, you can publish a new version of a function for beta testing without affecting users of the stable production version.

The function version includes the following information:

The function code and all associated dependencies.

The Lambda runtime that executes the function.

All the function settings, including the environment variables.

A unique Amazon Resource Name (ARN) to identify this version of the function.

You work on $LATEST which is the latest version of the code – this is mutable (changeable).

When you’re ready to publish a Lambda function you create a version (these are numbered).

Numbered versions are assigned a number starting with 1 and subsequent versions are incremented by 1.

Versions are immutable (code cannot be edited).

Each version has its own ARN.

Because different versions have unique ARNs this allows you to effectively manage them for different environments like Production, Staging or Development.

A qualified ARN has a version suffix.

An unqualified ARN does not have a version suffix.

You cannot create an alias from an unqualified ARN.

Lambda Aliases

Lambda aliases are pointers to a specific Lambda version.

Using an alias you can invoke a function without having to know which version of the function is being referenced.

Aliases are mutable.

Aliases enable stable configuration of event triggers / destinations.

Aliases also have static ARNs but can point to any version of the same function.

Aliases can also be used to split traffic between Lambda versions (blue/green).

Aliases enable blue / green deployment by assigning weights to Lambda version (doesn’t work for $LATEST, you need to create an alias for $LATEST).

Traffic Shifting

With the introduction of alias traffic shifting, it is now possible to trivially implement canary deployments of Lambda functions.

By updating additional version weights on an alias, invocation traffic is routed to the new function versions based on the weight specified.

Detailed CloudWatch metrics for the alias and version can be analyzed during the deployment, or other health checks performed, to ensure that the new version is healthy before proceeding.

The following example AWS CLI command points an alias to a new version, weighted at 5% (original version at 95% of traffic):

aws lambda update-alias --function-name myfunction --name myalias --routing-config '{"AdditionalVersionWeights" : {"2" : 0.05} }'

Lambda Handler

A handler is a function which Lambda will invoke to execute your code – it is an entry point.

When you create a Lambda function, you specify a handler that AWS Lambda can invoke when the service executes the function on your behalf.

You define a Lambda function handler as an instance or static method in a class.

Function Dependencies

If your Lambda function depends on external libraries such as AWS X-Ray SDK, database clients etc. you need to install the packages with the code and zip it all up.

For Node.js use npm & “node modules” directory.

For Python use pip — target options.

For Java include the relevant .jar files.

Upload the zip file straight to Lambda if it’s less than 50MB, otherwise upload to S3.

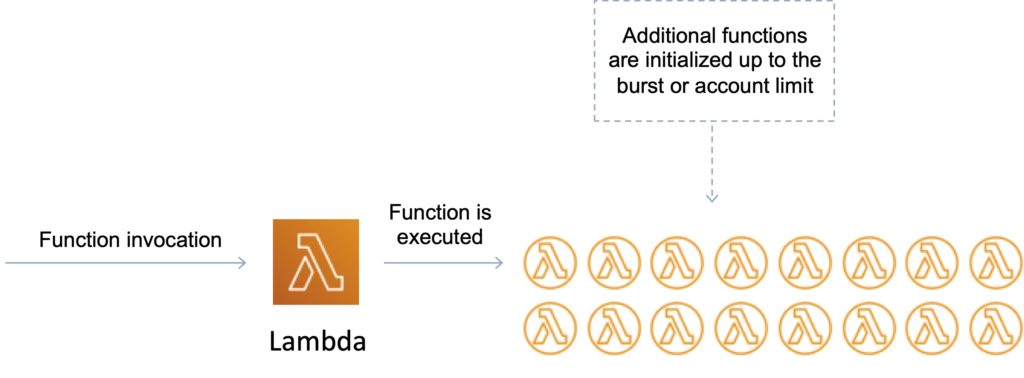
Native libraries work they need to be compiled on Amazon Linux.

AWS SDK comes with every Lambda function by default.

Concurrent executions

Managing Concurrency

The first time you invoke your function, AWS Lambda creates an instance of the function and runs its handler method to process the event. When the function returns a response, it stays active and waits to process additional events. If you invoke the function again while the first event is being processed, Lambda initializes another instance, and the function processes the two events concurrently.



Your functions’ concurrency is the number of instances that serve requests at a given time. For an initial burst of traffic, your functions’ cumulative concurrency in a Region can reach an initial level of between 500 and 3000, which varies per Region.

Burst Concurrency Limits:

3000 – US West (Oregon), US East (N. Virginia), Europe (Ireland).

1000 – Asia Pacific (Tokyo), Europe (Frankfurt).

500 – Other Regions.

After the initial burst, your functions’ concurrency can scale by an additional 500 instances each minute. This continues until there are enough instances to serve all requests, or until a concurrency limit is reached.

The default account limit is up to 1000 executions per second, per region (can be increased).

This is a safety feature to limit the number of concurrent executions across all functions in each region per account.

Each invocation over the concurrency limit will trigger a throttle.

TooManyRequestsExeception may be experienced if the concurrent execution limit is exceeded.

You may receive a HTTP status code: 429 and the message is “Request throughput limit exceeded”.

Throttle behavior:

For synchronous invocations returns throttle error 429.

For asynchronous invocations retries automatically (twice) then goes to a Dead Letter Queue (DLQ).

A DLQ can be an SNS topic or SQS queue.

The original event payload is sent to the DLQ.

The Lambda function needs an IAM role with permissions to SNS / SQS.

Lambda also integrates with X-Ray for debugging.

Can trace Lambda with X-Ray.

Need to enable in the Lambda configuration and it will run the X-Ray daemon.

Use AWS SDK in your code.

Reserved Concurrency

You can set a reserved concurrency at the function level to guarantee a set number of concurrent executions will be available for a critical function.

You can reserve up to the Unreserved account concurrency value that is shown in the console, minus 100 for functions that don’t have reserved concurrency.

To throttle a function, set the reserved concurrency to zero. This stops any events from being processed until you remove the limit.

To reserve concurrency for a function

Open the Lambda console [Functions page](https://console.aws.amazon.com/lambda/home#/functions).

Choose a function.

Under Concurrency, choose Reserve concurrency.

Enter the amount of concurrency to reserve for the function.

Choose Save.

Provisioned Concurrency

When provisioned concurrency is allocated, the function scales with the same burst behavior as standard concurrency.

After it’s allocated, provisioned concurrency serves incoming requests with very low latency.

When all provisioned concurrency is in use, the function scales up normally to handle any additional requests.

Application Auto Scaling takes this a step further by providing autoscaling for provisioned concurrency.

With Application Auto Scaling, you can create a target tracking scaling policy that adjusts provisioned concurrency levels automatically, based on the utilization metric that Lambda emits.

[Use the Application Auto Scaling API](https://docs.aws.amazon.com/lambda/latest/dg/configuration-concurrency.html#configuration-concurrency-api) to register an alias as a scalable target and create a scaling policy.

Provisioned concurrency runs continually and is billed in addition to standard invocation costs.

Success and Failure Destinations

Lambda asynchronous invocations can put an event or message on Amazon Simple Notification Service (SNS), Amazon Simple Queue Service (SQS), or Amazon EventBridge for further processing.

With Destinations, you can route asynchronous function results as an execution record to a destination resource without writing additional code.

An execution record contains details about the request and response in JSON format including version, timestamp, request context, request payload, response context, and response payload.

For each execution status such as Success or Failure you can choose one of four destinations: another Lambda function, SNS, SQS, or EventBridge. Lambda can also be configured to route different execution results to different destinations.

On-Success:

When a function is invoked successfully, Lambda routes the record to the destination resource for every successful invocation.

You can use this to monitor the health of your serverless applications via execution status or build workflows based on the invocation result.

On-Failure:

Destinations gives you the ability to handle the Failure of function invocations along with their Success.

When a function invocation fails, such as when retries are exhausted or the event age has been exceeded (hitting its TTL),

Destinations routes the record to the destination resource for every failed invocation for further investigation or processing.

Destinations provide more useful capabilities than Dead Letter Queues (DLQs) by passing additional function execution information, including code exception stack traces, to more destination services.

Destinations and DLQs can be used together and at the same time although Destinations should be considered a more preferred solution.

Dead Letter Queue (DLQ)

You can configure a dead letter queue (DLQ) on AWS Lambda to give you more control over message handling for all asynchronous invocations, including those delivered via AWS events (S3, SNS, IoT, etc)..

A dead-letter queue saves discarded events for further processing. A dead-letter queue acts the same as an on-failure destination in that it is used when an event fails all processing attempts or expires without being processed.

However, a dead-letter queue is part of a function’s version-specific configuration, so it is locked in when you publish a version. On-failure destinations also support additional targets and include details about the function’s response in the invocation record.

You can setup a DLQ by configuring the ‘DeadLetterConfig’ property when creating or updating your Lambda function.

You can provide an SQS queue or an SNS topic as the ‘TargetArn’ for your DLQ, and AWS Lambda will write the event object invoking the Lambda function to this endpoint after the standard retry policy (2 additional retries on failure) is exhausted.

Lambda Layers

You can configure your Lambda function to pull in additional code and content in the form of layers.

A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies.

With layers, you can use libraries in your function without needing to include them in your deployment package.

A function can use up to 5 layers at a time.

Layers are extracted to the /opt directory in the function execution environment.

Each runtime looks for libraries in a different location under /opt, depending on the language.

Lambda@Edge

Lambda@Edge allows you to run code across AWS locations globally without provisioning or managing servers, responding to end users at the lowest network latency.

Lambda@Edge lets you run Node.js and Python Lambda functions to customize 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:

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).

You just upload your Node.js code to AWS Lambda and configure your function to be triggered in response to an Amazon CloudFront request.

The code is then ready to execute across AWS locations globally when a request for content is received, and scales with the volume of CloudFront requests globally.

Lambda and Amazon VPC

You can connect a Lambda function to private subnets in a VPC.

Lambda needs the following VPC configuration information so that it can connect to the VPC:

Private subnet ID.

Security Group ID (with required access).

Lambda uses this information to setup an Elastic Network Interface (ENI) using an available IP address from your private subnet.

Lambda functions provide access only to a single VPC. If multiple subnets are specified, they must all be in the same VPC.

Lambda functions configured to access resources in a particular VPC will not have access to the Internet as a default configuration.

If you need access to the internet, you will need to create a NAT in your VPC to forward this traffic and configure your security group to allow this outbound traffic.

Careful with DNS resolution of public hostnames as it could add to function running time (cost).

Cannot connect to a dedicated tenancy VPC.

Exam tip: If a Lambda function needs to connect to a VPC and needs Internet access, make sure you connect to a private subnet that has a route to a NAT Gateway (the NAT Gateway will be in a public subnet).

Lambda uses your function’s permissions to create and manage network interfaces. To connect to a VPC, your function’s execution role must have the following permissions:

ec2:CreateNetworkInterface

ec2:DescribeNetworkInterfaces

ec2:DeleteNetworkInterface

These permissions are included in the AWSLambdaVPCAccessExecutionRole managed policy.

Only connect to a VPC if you need to as it can slow down function execution.

Building Lambda Apps

You can deploy and manage your serverless applications using the AWS Serverless Application Model (AWS SAM).

AWS SAM is a specification that prescribes the rules for expressing serverless applications on AWS.

This specification aligns with the syntax used by AWS CloudFormation today and is supported natively within AWS CloudFormation as a set of resource types (referred to as “serverless resources”).

You can automate your serverless application’s release process using AWS CodePipeline and AWS CodeDeploy.

You can enable your Lambda function for tracing with AWS X-Ray.

Elastic Load Balancing

Application Load Balancers (ALBs) support AWS Lambda functions as targets.

You can register your Lambda functions as targets and configure a listener rule to forward requests to the target group for your Lambda function.

Exam tip: Functions can be registered to target groups using the API, AWS Management Console or the CLI.

When the load balancer forwards the request to a target group with a Lambda function as a target, it invokes your Lambda function and passes the content of the request to the Lambda function, in JSON format.

Limits:

The Lambda function and target group must be in the same account and in the same Region.

The maximum size of the request body that you can send to a Lambda function is 1 MB.

The maximum size of the response JSON that the Lambda function can send is 1 MB.

WebSockets are not supported. Upgrade requests are rejected with an HTTP 400 code.

By default, health checks are disabled for target groups of type lambda.

You can enable health checks to implement DNS failover with Amazon Route 53. The Lambda function can check the health of a downstream service before responding to the health check request.

If you create the target group and register the Lambda function using the AWS Management Console, the console adds the required permissions to your Lambda function policy on your behalf.

Otherwise, after you create the target group and register the function using the AWS CLI, you must use the add-permission command to grant Elastic Load Balancing permission to invoke your Lambda function.

Lambda Limits

Memory – minimum 128 MB, maximum 10,240 MB in 1 MB increments.

Ephemeral disk capacity (/tmp space) per invocation – 512 MB.

Size of environment variables maximum 4 KB.

Number of file descriptors – 1024.

Number of processes and threads (combined) – 1024.

Maximum execution duration per request – 900 seconds.

Concurrent executions per account – 1000 (soft limit).

Function burst concurrency 500 -3000 (region dependent).

Invocation payload:

Synchronous 6 MB.

Asynchronous 256 KB

Lambda function deployment size is 50 MB (zipped), 250 MB unzipped.

Operations and Monitoring

Lambda automatically monitors Lambda functions and reports metrics through CloudWatch.

Lambda tracks the number of requests, the latency per request, and the number of requests resulting in an error.

You can view the request rates and error rates using the AWS Lambda Console, the CloudWatch console, and other AWS resources.

You can use AWS X-Ray to visualize the components of your application, identify performance bottlenecks, and troubleshoot requests that resulted in an error.

Your Lambda functions send trace data to X-Ray, and X-Ray processes the data to generate a service map and searchable trace summaries.

The AWS X-Ray Daemon is a software application that gathers raw segment data and relays it to the AWS X-Ray service.

The daemon works in conjunction with the AWS X-Ray SDKs so that data sent by the SDKs can reach the X-Ray service.

When you trace your Lambda function, the X-Ray daemon automatically runs in the Lambda environment to gather trace data and send it to X-Ray.

Must have permissions to write to X-Ray in the execution role.

Development Best Practices

Perform one-off time-consuming tasks outside of the function handler, e.g.:

Connect to databases.

Initialize the AWS SDK.

Pull in dependencies or datasets.

Use environment variables for:

Connection strings, S3 bucket etc.

Passwords and other sensitive data (can be encrypted with KMS).

Minimize deployment packages size to runtime necessities.

Break down the function if required.

Remember the Lambda limits.

Avoid using recursive code, never have a Lambda function call itself.

Don’t put you Lambda function in a VPC unless you need to (can take longer to initialize).

Charges

Priced based on:

Number of requests.

Duration of the request calculated from the time your code begins execution until it returns or terminates.

The amount of memory allocated to the function.

Amazon API Gateway

API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale.

API Gateway supports the following:

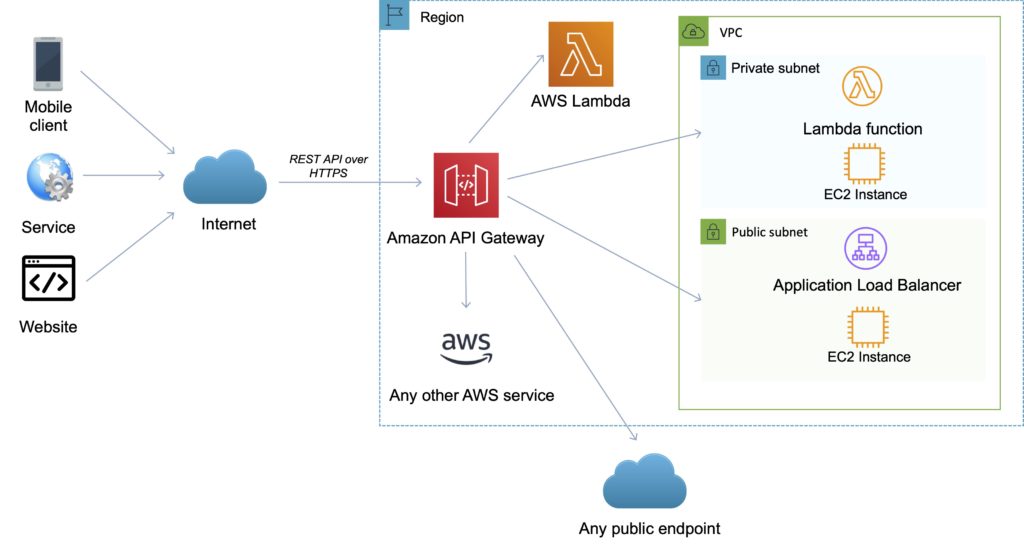
Creating, deploying, and managing a [REST](https://en.wikipedia.org/wiki/Representational_state_transfer) application programming interface (API) to expose backend HTTP endpoints, AWS Lambda functions, or other AWS services.

Creating, deploying, and managing a [WebSocket](https://tools.ietf.org/html/rfc6455) API to expose AWS Lambda functions or other AWS services.

Invoking exposed API methods through the frontend HTTP and WebSocket endpoints.

Together with Lambda, API Gateway forms the app-facing part of the AWS serverless infrastructure.

Back-end services include Amazon EC2, AWS Lambda or any web application (public or private endpoints).



API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.

API calls include traffic management, authorization and access control, monitoring, and API version management.

API Gateway provides a REST API that uses JSON.

API Gateway exposes HTTPS endpoints to define a RESTful API.

All the APIs created with Amazon API Gateway expose HTTPS endpoints only (does not support unencrypted endpoints).

An API can present a certificate to be authenticated by the back end.

Can send each API endpoint to a different target.

CloudFront is used as the public endpoint for API Gateway.

Using CloudFront behind the scenes, custom domains, and SNI are supported.

Supports API keys and Usage Plans for user identification, throttling or quota management.

Permissions to invoke a method are granted using IAM roles and policies or API Gateway custom authorizers.

By default API Gateway assigns an internal domain that automatically uses the API Gateway certificates.

When configuring your APIs to run under a custom domain name you can provide your own certificate.

Amazon API Gateway Features

The following table describes some of the core features of Amazon API Gateway.

|  |  |
| --- | --- |
| API Gateway Feature | Benefit |
| Support for RESTful APIs and WebSocket APIs | With API Gateway, you can create RESTful APIs using either HTTP APIs or REST APIs |
| Private integrations with AWS ELB & AWS Cloud Map | With API Gateway, you can route requests to private resources in your VPC. Using HTTP APIs, you can build APIs for services behind private ALBs, private NLBs, and IP-based services registered in AWS Cloud Map, such as ECS tasks. |
| Metering | Define plans that meter and restrict third-party developer access to APIs. |
| Security | API Gateway provides multiple tools to authorize access to APIs and control service operation access. |
| Resiliency | Manage traffic with throttling so that backend operations can withstand traffic spikes. |
| Operations Monitoring | API Gateway provides a metrics dashboard to monitor calls to services. |
| Lifecycle Management | Operate multiple API versions and multiple stages for each version simultaneously so that existing applications can continue to call previous versions after new API versions are published. |
| AWS Authorization | Support for signature version 4 for REST APIs and WebSocket APIs, IAM access policies, and authorization with bearer tokens (e.g., JWT, SAML) using Lambda functions. |

Endpoints

An [API endpoint](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-basic-concept.html#apigateway-definition-api-endpoints) type is a hostname for an API in API Gateway that is deployed to a specific region.

The hostname is of the form {api-id}.execute-api.{region}.amazonaws.com.

The API endpoint type can be edge-optimized, regional, or private, depending on where most of your API traffic originates from.

Edge-Optimized Endpoint

An edge-optimized API endpoint is best for geographically distributed clients. API requests are routed to the nearest CloudFront Point of Presence (POP). This is the default endpoint type for API Gateway REST APIs.

Edge-optimized APIs capitalize the names of HTTP headers (for example, Cookie).

CloudFront sorts HTTP cookies in natural order by cookie name before forwarding the request to your origin. For more information about the way CloudFront processes cookies, see Caching Content Based on Cookies.

Any custom domain name that you use for an edge-optimized API applies across all regions.

Regional Endpoint

A regional API endpoint is intended for clients in the same region.

When a client running on an EC2 instance calls an API in the same region, or when an API is intended to serve a small number of clients with high demands, a regional API reduces connection overhead.

For a regional API, any custom domain name that you use is specific to the region where the API is deployed.

If you deploy a regional API in multiple regions, it can have the same custom domain name in all regions.

You can use custom domains together with Amazon Route 53 to perform tasks such as latency-based routing.

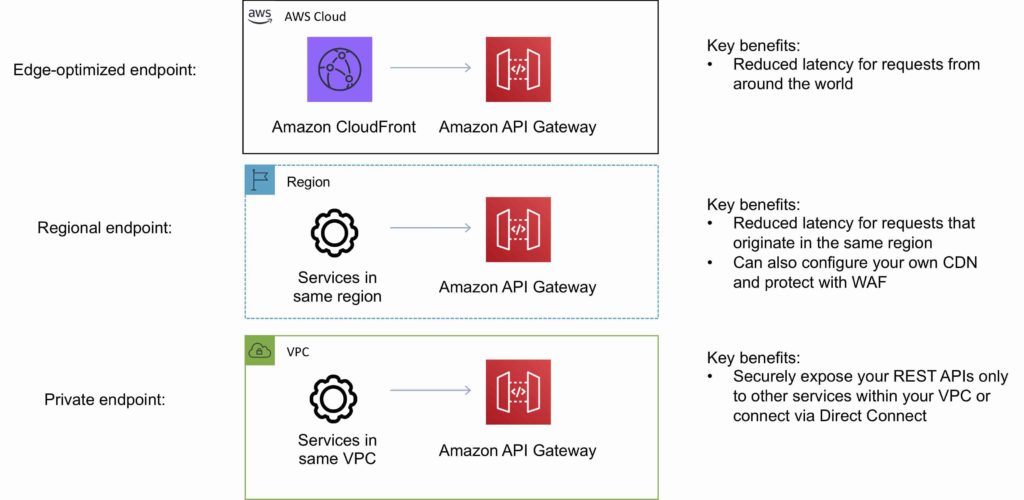
Regional API endpoints pass all header names through as-is.

Private Endpoint

A private API endpoint is an API endpoint that can only be accessed from your Amazon Virtual Private Cloud (VPC) using an interface VPC endpoint, which is an endpoint network interface (ENI) that you create in your VPC.

Private API endpoints pass all header names through as-is.

The following diagram depicts the three different Amazon API Gateway endpoint types:



Amazon API Gateway API’s

API Gateway REST API

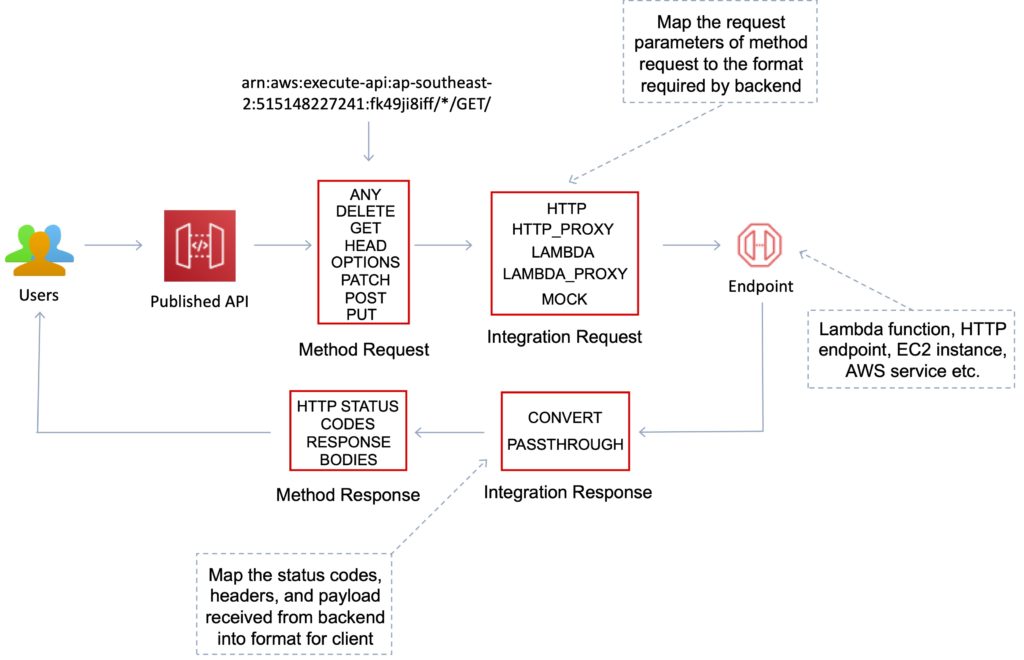
A collection of HTTP resources and methods that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services.

This collection can be deployed in one or more stages.

Typically, API resources are organized in a resource tree according to the application logic.

Each API resource can expose one or more API methods that have unique HTTP verbs supported by API Gateway.

The following diagram depicts the structure of an API:



API Gateway WebSocket API

A collection of WebSocket routes and route keys that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services.

The collection can be deployed in one or more stages.

API methods are invoked through frontend WebSocket connections that you can associate with a registered custom domain name.

Methods

API Gateway Methods are HTTP methods associated with an API Gateway resource.

Each resource URL can have HTTP methods such as: GET, PUT, POST and DELETE.

AWS also offers the “ANY” method as a catch-all.

Deployments

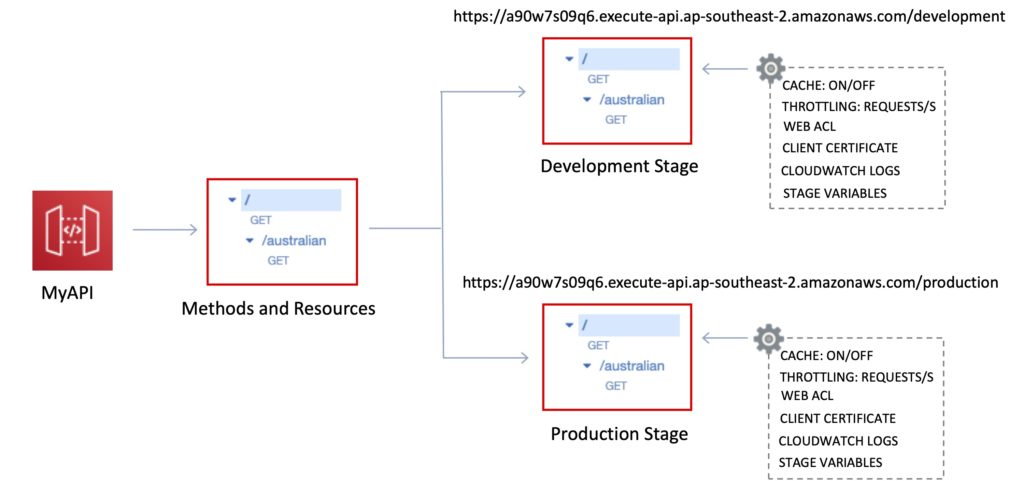
Deployments are a snapshot of the APIs resources and methods.

Deployments must be created and associated with a stage for anyone to access the API.

Stages and Stage variables

A stage is a logical reference to a lifecycle state of your REST or WebSocket API (for example, ‘dev’, ‘prod’, ‘beta’, ‘v2’).

API stages are identified by API ID and stage name.



Stage variables are like environment variables for API Gateway.

Stage variables can be used in:

Lambda function ARN.

HTTP endpoint.

Parameter mapping templates.

Use cases for stage variables:

Configure HTTP endpoints your stages talk to (dev, test, prod etc.).

Pass configuration parameters to AWS Lambda through mapping templates.

Stage variables are passed to the “context” object in Lambda.

Stage variables are used with Lambda aliases.

You can create a stage variable to indicate the corresponding Lambda alias.

You can create canary deployments for any stage – choose the % of traffic the canary channel receives.

Mapping templates

Mapping templates can be used to modify request / responses.

Rename parameters.

Modify body content.

Add headers.

Map JSON to XML for sending to backend or back to client.

Uses Velocity Template Language (VTL).

Filter output results (remove unnecessary data).

Caching

You can add caching to API calls by provisioning an Amazon API Gateway cache and specifying its size in gigabytes.

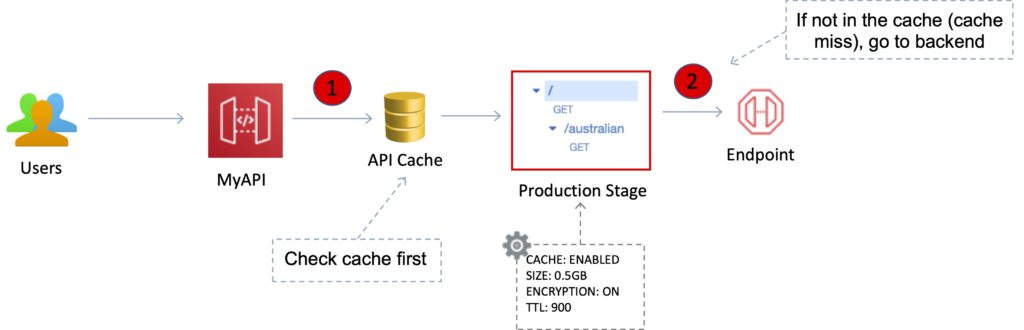
Caching allows you to cache the endpoint’s response.

Caching can reduce the number of calls to the backend and improve the latency of requests to the API.

API Gateway caches responses for a specific amount of time (time to live or TTL).

The default TTL is 300 seconds (min 0, max 3600).

Caches are defined per stage.



You can encrypt caches.

The cache capacity is between 0.5GB to 237GB.

It is possible to override cache settings for specific methods.

You can flush the entire cache (invalidate it) immediately if required.

Clients can invalidate the cache with the header: Cache-Control: max-age=0 .

API throttling

API Gateway sets a limit on a steady-state rate and a burst of request submissions against all APIs in your account.

Limits:

By default API Gateway limits the steady-state request rate to 10,000 requests per second.

The maximum concurrent requests is 5,000 requests across all APIs within an AWS account.

If you go over 10,000 requests per second or 5,000 concurrent requests you will receive a 429 Too Many Requests error response.

Upon catching such exceptions, the client can resubmit the failed requests in a way that is rate-limiting, while complying with the API Gateway throttling limits.

Amazon API Gateway provides two basic types of throttling-related settings:

Server-side throttling limits are applied across all clients. These limit settings exist to prevent your API—and your account—from being overwhelmed by too many requests.

Per-client throttling limits are applied to clients that use API keys associated with your usage policy as a client identifier.

API Gateway throttling-related settings are applied in the following order:

Per-client per-method throttling limits that you set for an API stage in a usage plan.

Per-client throttling limits that you set in a usage plan.

Default per-method limits and individual per-method limits that you set in API stage settings.

Account-level throttling.

Integrations and Method Requests / Responses

A method represents a client-facing interface by which the client calls the API to access back-end resources.

A Method resource is integrated with an Integration resource.

API methods are integrated with backend endpoints using API integrations.

Backend endpoints are known as “integration endpoints”.

Integration request

The internal interface of a WebSocket API route or REST API method in API Gateway, in which you map the body of a route request or the parameters and body of a method request to the formats required by the backend.

Integration response

The internal interface of a WebSocket API route or REST API method in API Gateway, in which you map the status codes, headers, and payload that are received from the backend to the response format that is returned to a client app.

Method request

The public interface of a REST API method in API Gateway that defines the parameters and body that an app developer must send in requests to access the backend through the API

Method response

The public interface of a REST API that defines the status codes, headers, and body models that an app developer should expect in responses from the API.

Mapping template

A script in [Velocity Template Language (VTL)](http://velocity.apache.org/engine/devel/vtl-reference.html) that transforms a request body from the frontend data format to the backend data format, or that transforms a response body from the backend data format to the frontend data format.

Mapping templates can be specified in the integration request or in the integration response. They can reference data made available at run time as context and stage variables.

The mapping can be as simple as an [identity transform](https://en.wikipedia.org/wiki/Identity_transform) that passes the headers or body through the integration as-is from the client to the backend for a request.

The same is true for a response, in which the payload is passed from the backend to the client.

Integration Type

You choose an API integration type according to the types of integration endpoint you work with and how you want data to pass to and from the integration endpoint.

For a Lambda function, you can have the Lambda proxy integration, or the Lambda custom integration.

For an HTTP endpoint, you can have the HTTP proxy integration or the HTTP custom integration.

For an AWS service action, you have the AWS integration of the non-proxy type only. API Gateway also supports the mock integration, where API Gateway serves as an integration endpoint to respond to a method request.

The Lambda custom integration is a special case of the AWS integration, where the integration endpoint corresponds to the [function-invoking action](https://docs.aws.amazon.com/lambda/latest/dg/API_Invoke.html)of the Lambda service.

Programmatically, you choose an integration type by setting the [type](https://docs.aws.amazon.com/apigateway/api-reference/resource/integration/#type) property on the [Integration](https://docs.aws.amazon.com/apigateway/api-reference/resource/integration/) resource.

For the Lambda proxy integration, the value is AWS\_PROXY.

For the Lambda custom integration and all other AWS integrations, it is AWS.

For the HTTP proxy integration and HTTP integration, the value is HTTP\_PROXY and HTTP, respectively.

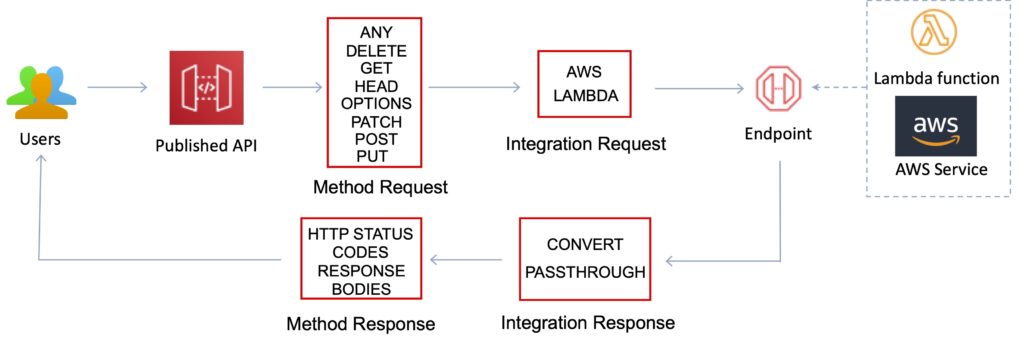
For the mock integration, the type value is MOCK.

The following list summarizes the supported integration types:

AWS Integration

This type of integration lets an API expose AWS service actions.

In AWS integration, you must configure both the integration request and integration response and set up necessary data mappings from the method request to the integration request, and from the integration response to the method response.



AWS\_PROXY Integration

This type of integration lets an API method be integrated with the Lambda function invocation action with a flexible, versatile, and streamlined integration setup.

This integration relies on direct interactions between the client and the integrated Lambda function.

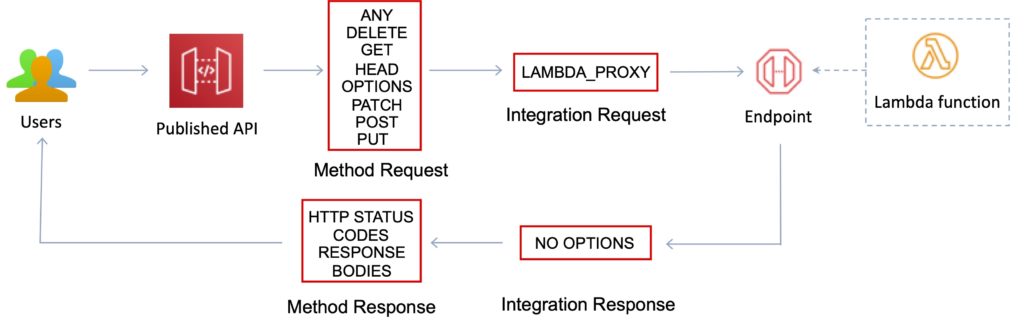
With this type of integration, also known as the Lambda proxy integration, you do not set the integration request or the integration response.

API Gateway passes the incoming request from the client as the input to the backend Lambda function.

The integrated Lambda function takes the [input of this format](https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format) and parses the input from all available sources, including request headers, URL path variables, query string parameters, and applicable body.

The function returns the result following this [output format](https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-output-format).

This is the preferred integration type to call a Lambda function through API Gateway and is not applicable to any other AWS service actions, including Lambda actions other than the function-invoking action.



HTTP Integration

This type of integration lets an API expose HTTP endpoints in the backend.

With the HTTP integration, also known as the HTTP custom integration, you must configure both the integration request and integration response.

You must set up necessary data mappings from the method request to the integration request, and from the integration response to the method response.

HTTP\_PROXY Integration

The HTTP proxy integration allows a client to access the backend HTTP endpoints with a streamlined integration setup on single API method.

You do not set the integration request or the integration response.

API Gateway passes the incoming request from the client to the HTTP endpoint and passes the outgoing response from the HTTP endpoint to the client.

MOCK Integration

This type of integration lets API Gateway return a response without sending the request further to the backend.

This is useful for API testing because it can be used to test the integration set up without incurring charges for using the backend and to enable collaborative development of an API.

In collaborative development, a team can isolate their development effort by setting up simulations of API components owned by other teams by using the MOCK integrations.

It is also used to return CORS-related headers to ensure that the API method permits CORS access.

In fact, the API Gateway console integrates the OPTIONS method to support CORS with a mock integration.

[Gateway responses](https://docs.aws.amazon.com/apigateway/latest/developerguide/customize-gateway-responses.html) are other examples of mock integrations.

Usage plans and API keys

A usage plan specifies who can access one or more deployed API stages and methods — and how much and how fast they can access them.

You can use a usage plan to configure throttling and quota limits, which are enforced on individual client API keys.

The plan uses API keys to identify API clients and meters access to the associated API stages for each key.

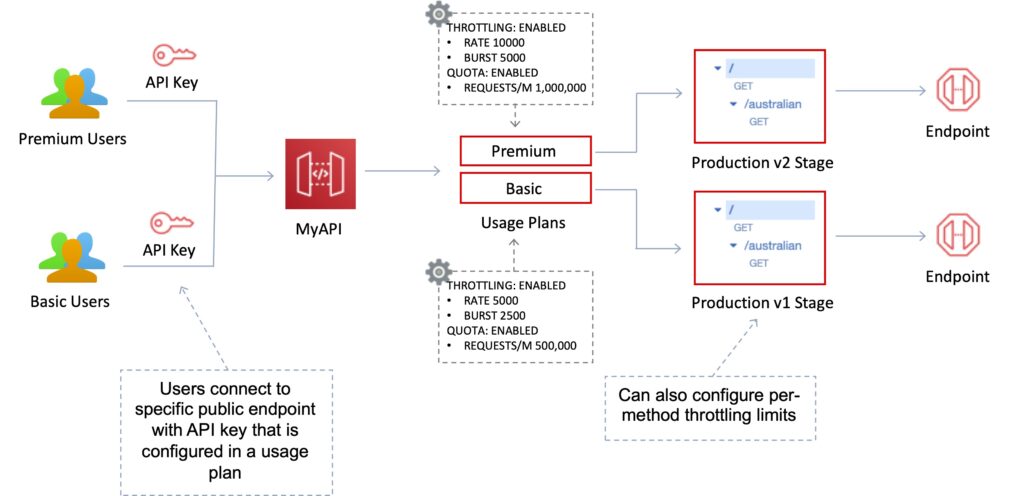
It also lets you configure throttling limits and quota limits that are enforced on individual client API keys.

API keys are alphanumeric string values that you distribute to app developer customers to grant access to your API.

You can use API keys together with [usage plans](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-usage-plans.html) or [Lambda authorizers](https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html) to control access to your APIs.

API Gateway can generate API keys on your behalf, or you can import them from a [CSV file](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-key-file-format.html).

You can generate an API key in API Gateway or import it into API Gateway from an external source.



Same origin policy

Used to prevent cross-site scripting attacks.

Web browser permits scripts in a first web page to access data in a second web page but only if the web pages have the same origin.

This is enforced by web browsers.

Cross-origin resource sharing (CORS)

Can enable Cross Origin Resource Sharing (CORS) for multiple domain use with JavaScript/AJAX.

CORS is one way that the server at the other end (not the client code in the browser) can relax the same-origin policy.

CORS allows restricted resources (e.g. fonts) on a web page to be requested from another domain outside the domain from which the first resource was shared.

Using CORS:

Can enable CORS on API Gateway if using JavaScript / AJAX.

Can be used to enable requests from domains other than the APIs domain.

Allows the sharing of resources between different domains.

The method (GET, PUT, POST etc.) for which you will enable CORS must be available in the API Gateway API before you enable CORS.

If CORS is not enabled and an API resource received requests from another domain the request will be blocked.

Enable CORS on the API resources using the selected methods under the API Gateway.

Identity and Access Management

There are several mechanisms for controlling and managing access to an API.

These mechanisms include:

Resource-based policies.

Standard IAM Roles and Policies (identity-based policies).

IAM Tags.

Endpoint policies for interface VPC endpoints.

Lambda authorizers.

Amazon Cognito user pools.

IAM Resource-Based Policies

Amazon API Gateway resource policies are JSON policy documents that you attach to an API to control whether a specified principal (typically an IAM user or role) can invoke the API.

You can use API Gateway resource policies to allow your API to be securely invoked by:

Users from a specified AWS account.

Specified source IP address ranges or CIDR blocks.

Specified virtual private clouds (VPCs) or VPC endpoints (in any account).

You can use resource policies for all API endpoint types in API Gateway: private, edge-optimized, and Regional.



IAM Identity-Based Policies

You need to create IAM policies and attach to users / roles.

API Gateway verifies IAM permissions passed by the calling application.

Leverages sigv4 capability where IAM credentials are passed in headers.

Handles authentication and authorization.

Great for user / roles within your AWS account.

Lambda Authorizer

Use AWS Lambda to validate the token in the header being passed.

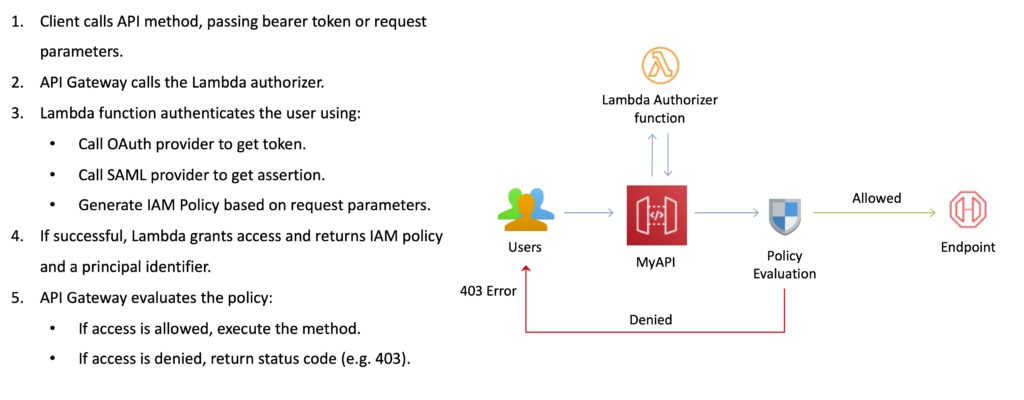
Option to cache the result of the authentication.

Lambda must return an IAM policy for the user.

You pay per Lambda invocation.

Handles authentication and authorization.

Good for using OAuth, SAML or 3rd party authentication.



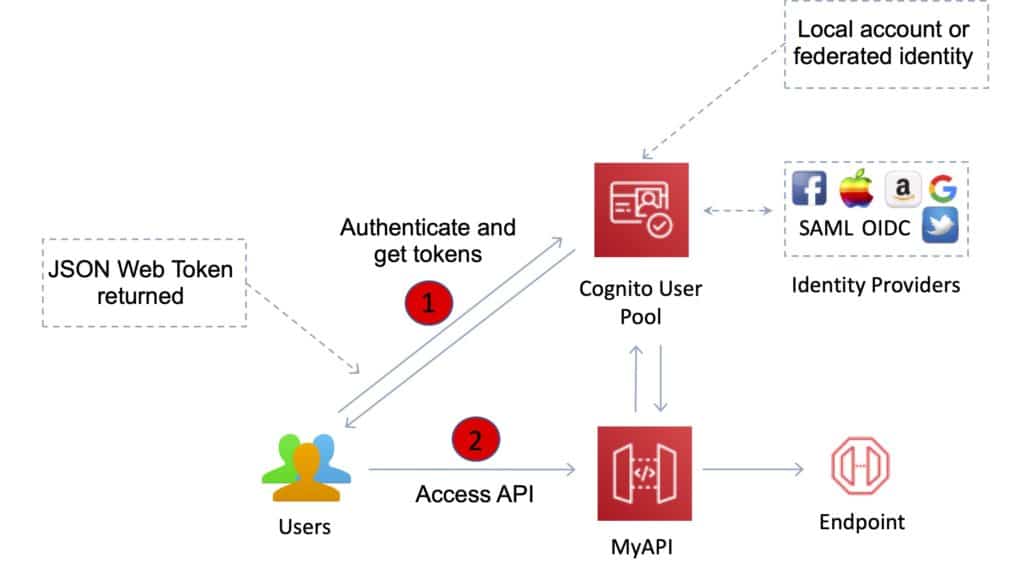
Cognito User Pools

A user pool is a user directory in Amazon Cognito.

With a user pool, users can sign in to a web or mobile app through Amazon Cognito.

Users can also sign in through social identity providers like Google, Facebook, Amazon, or Apple, and through SAML identity providers.

Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through a Software Development Kit (SDK).



User pools provide:

Sign-up and sign-in services.

A built-in, customizable web UI to sign in users.

Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple, as well as sign-in with SAML identity providers from your user pool.

User directory management and user profiles.

Security features such as multi-factor authentication (MFA).

Additional Features and Benefits

API Gateway provides several features that assist with creating and managing APIs:

Metering – Define plans that meter and restrict third-party developer access to APIs.

Security – API Gateway provides multiple tools to authorize access to APIs and control service operation access.

Resiliency – Manage traffic with throttling so that backend operations can withstand traffic spikes.

Operations Monitoring – API Gateway provides a metrics dashboard to monitor calls to services.

Lifecycle Management – Operate multiple API versions and multiple stages for each version simultaneously so that existing applications can continue to call previous versions after new API versions are published.

API Gateway provides robust, secure, and scalable access to backend APIs and hosts multiple versions and release stages for your APIs.

You can create and distribute API Keys to developers.

Option to use AWS Sig-v4 to authorize access to APIs.

You can throttle and monitor requests to protect your backend.

API Gateway allows you to maintain a cache to store API responses.

SDK Generation for iOS, Android, and JavaScript.

Reduced latency and distributed denial of service protection using CloudFront.

Request/response data transformation and API mocking.

Resiliency through throttling rules based on the number of requests per second for each HTTP method (GET, PUT).

Throttling can be configured at multiple levels including Global and Service Call.

A cache can be created and specified in gigabytes (not enabled by default).

Caches are provisioned for a specific stage of your APIs.

Caching features include customizable keys and time-to-live (TTL) in seconds for your API data which enhances response times and reduces load on back-end services.

API Gateway can scale to any level of traffic received by an API.

Logging and Monitoring

The Amazon API Gateway logs (near real time) back-end performance metrics such as API calls, latency, and error rates to CloudWatch.

You can monitor through the API Gateway dashboard (REST API) allowing you to visually monitor calls to the services.

API Gateway also meters utilization by third-party developers and the data is available in the API Gateway console and through APIs.

Amazon API Gateway is integrated with AWS CloudTrail to give a full auditable history of the changes to your REST APIs.

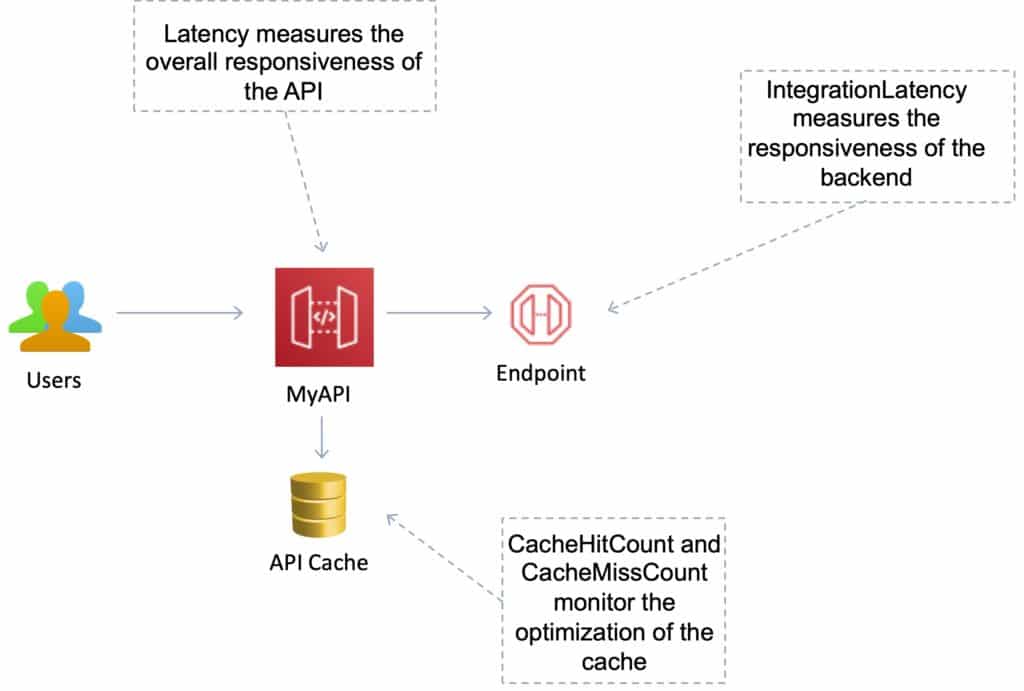
All API calls made to the Amazon API Gateway APIs to create, modify, delete, or deploy REST APIs are logged to CloudTrail.

Understanding the following metrics is useful for the exam:

Monitor the IntegrationLatency metrics to measure the responsiveness of the backend.

Monitor the Latency metrics to measure the overall responsiveness of your API calls.

Monitor the CacheHitCount and CacheMissCount metrics to optimize cache capacities to achieve a desired performance.



Open API / Swagger

Can import existing Swagger / Open API 3.0 definitions (written in YAML or JSON) to API Gateway.

This is a common way of defining REST APIs using API definition as code.

Can also export current APIs as Swagger / Open API 3.0 definition.

Uses the API Gateway  Import API feature to import an API from an external definition.

With the import API you can either create a new API by submitting a POST request that includes a Swagger definition in the payload and endpoint configuration or you can update an existing API by using a PUT request that contains a Swagger definition or merge a definition with an existing API.

You specify the options using a mode query parameter in the request URL.

Charges

With Amazon API Gateway, you only pay when your APIs are in use.

There are no minimum fees or upfront commitments.

You pay only for the API calls you receive, and the amount of data transferred out.

There are no data transfer out charges for Private APIs (however, AWS PrivateLink charges apply when using Private APIs in Amazon API Gateway).

Amazon API Gateway also provides optional data caching charged at an hourly rate that varies based on the cache size you select.

The API Gateway free tier includes one million API calls per month for up to 12 months.

AWS Application Integration Services

The AWS Application Integration Services are a suite of services that can be used to enable communication between application components in a decoupled architecture. This includes microservices application, distributed systems, and serverless applications.

AWS Application Integration Services

The AWS application integration services are a family of services that enable decoupled communication between applications.

These services provide decoupling for microservices, distributed systems, and serverless applications.

AWS application integration services allow you to connect apps, without needing to write custom code to enable interoperability.

Decoupled applications can interoperate whilst being resilient to the failure or overload of any individual component.

The following services are involved with application integration:

|  |  |  |
| --- | --- | --- |
| Service | What it does | Example use cases |
| Simple Queue Service (SQS) | Messaging queue; store and forward patterns | Building distributed / decoupled applications |
| Simple Notification Service (SNS) | Set up, operate, and send notifications from the cloud | Send email notification when CloudWatch alarm is triggered |
| Step Functions | Out-of-the-box coordination of AWS service components with visual workflow | Order processing workflow |
| Simple Workflow Service (SWF) | Need to support external processes or specialized execution logic | Human-enabled workflows like an order fulfilment system or for procedural requests    Note: AWS recommends that for new applications customers consider Step Functions instead of SWF |
| Amazon MQ | Message broker service for Apache Active MQ and RabbitMQ | Need a message queue that supports industry standard APIs and protocols; migrate queues to AWS |
| Amazon Kinesis | Collect, process, and analyze streaming data. | Collect data from IoT devices for later processing |

Amazon SNS

Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication.

The pub/sub functionality provides messaging for high-throughput, push-based, many-to-many use cases.

Amazon SNS is used for sending notifications between distributed systems, microservices, and event-driven serverless applications.

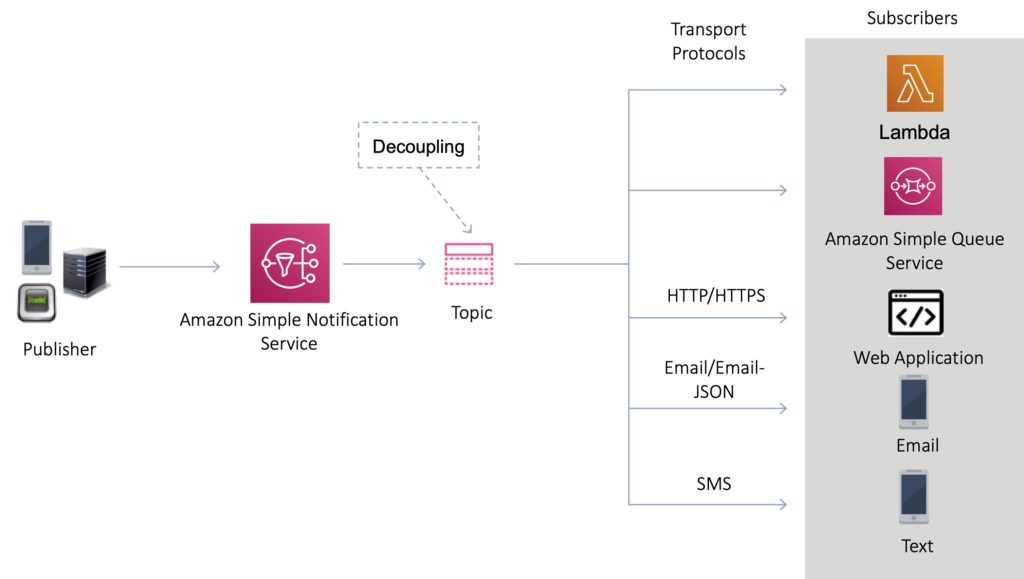
Push notifications can go to Apple, Google, Fire OS, and Windows devices as well as Android devices in China with Baidu Cloud Push.

Amazon SNS can also send notifications via SMS text message, email, SQS queues or to any HTTP endpoint.

Amazon SNS notifications can also trigger Lambda functions.

Amazon SNS is inexpensive and based on a pay-as-you-go model with no upfront costs.

SNS uses a pub-sub model whereby users or applications subscribe to SNS topics.



Amazon SNS provides decoupling of your applications so messages can be processed asynchronously.

SNS Topics

Multiple recipients can be grouped using Topics.

A topic is an “access point” for allowing recipients to dynamically subscribe for identical copies of the same notification.

One topic can support deliveries to multiple endpoint types.

All messages are stored redundantly across multiple availability zones.

Provides instantaneous, push-based delivery.

Flexible message delivery is provided over multiple transport protocols.

SNS Subscribers and Endpoints

When subscribing to an SNS topic the following endpoint types are supported:

HTTP/HTTPS.

Email/Email-JSON.

Amazon Kinesis Data Firehose.

Amazon SQS.

AWS Lambda.

Platform application endpoint (mobile push).

SMS.

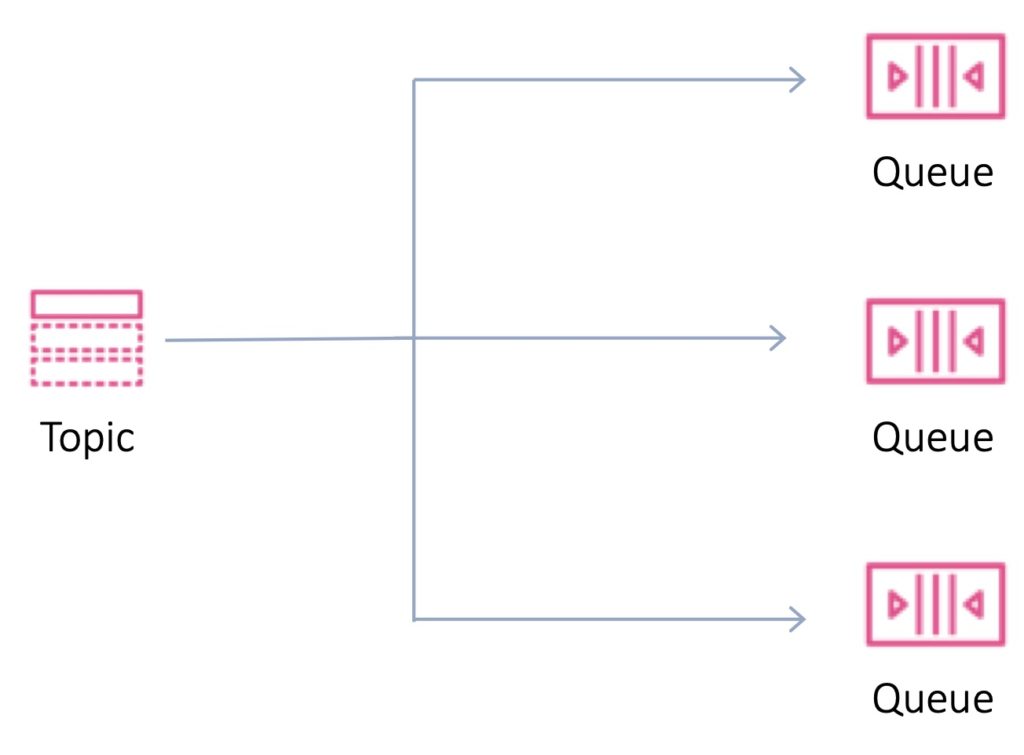
SNS Fanout

Your publisher systems can fanout messages to many subscriber systems including Amazon SQS queues, AWS Lambda functions and HTTPS endpoints, for parallel processing, and Amazon Kinesis Data Firehose.

You can subscribe one or more Amazon SQS queues to an Amazon SNS topic from a list of topics available for the selected queue.

Amazon SQS manages the subscription and any necessary permissions.

When you publish a message to a topic, Amazon SNS sends the message to every subscribed queue.



Fanout is supported for application-to-application (A2A) messaging:

[Fanout to Kinesis Data Firehose delivery streams](https://docs.aws.amazon.com/sns/latest/dg/sns-firehose-as-subscriber.html).

[Fanout to Lambda functions](https://docs.aws.amazon.com/sns/latest/dg/sns-lambda-as-subscriber.html).

[Fanout to Amazon SQS queues](https://docs.aws.amazon.com/sns/latest/dg/sns-sqs-as-subscriber.html).

[Fanout to HTTP/S endpoints](https://docs.aws.amazon.com/sns/latest/dg/sns-http-https-endpoint-as-subscriber.html).

[Fanout to AWS Event Fork Pipelines](https://docs.aws.amazon.com/sns/latest/dg/sns-fork-pipeline-as-subscriber.html).

Amazon SQS

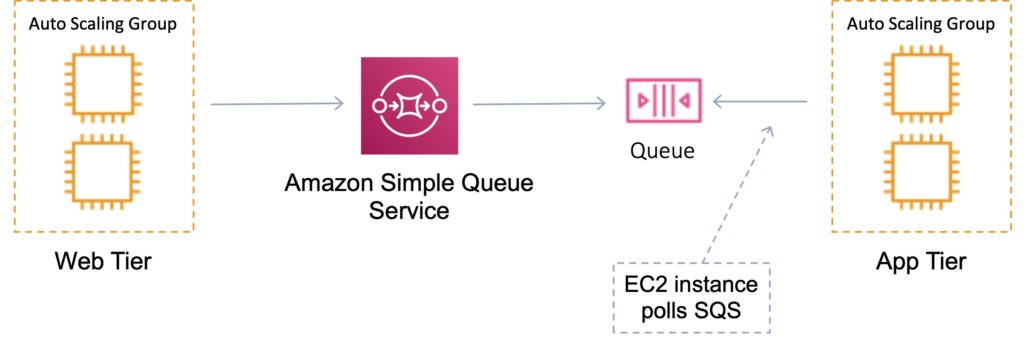
Amazon Simple Queue Service (SQS) is a distributed queue system that enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component.

Amazon SQS enables you to send, store, and receive messages between software components.

An Amazon SQS queue is a temporary repository for messages that are awaiting processing.

The SQS queue acts as a buffer between the component producing and saving data, and the component receiving the data for processing.

The SQS queue resolves issues that arise if the producer is producing work faster than the consumer can process it, or if the producer or consumer are only intermittently connected to the network.



This is known as decoupling / loose coupling and helps to enable elasticity for your application.

Amazon SQS is pull-based, not push-based (like Amazon SNS).

Messages are up to 256KB in size.

Messages can be kept in the queue from 1 minute to 14 days.

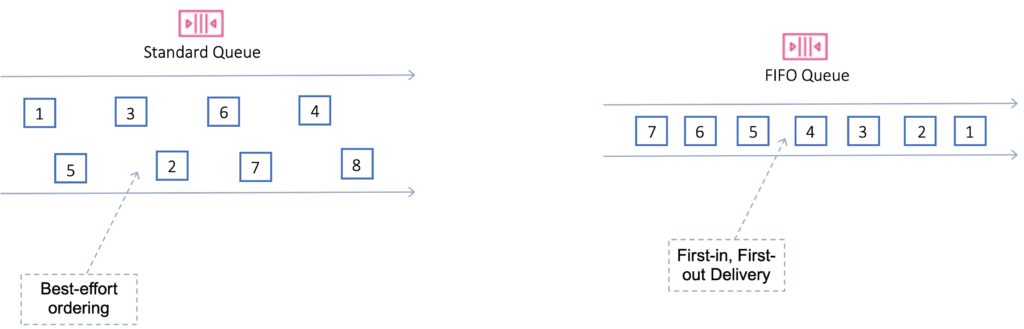
The default retention period is 4 days.

SQS guarantees that your messages will be processed at least once.

SQS Queues

Queue names must be unique within a region.

There are two types of queue – standard queues and FIFO queues.



Standard

Default queue type.

Nearly unlimited transactions per second.

Guarantee that a message is delivered at least once.

Occasionally more than one copy of a message might be delivered out of order.

Provides best-effort ordering which ensures that messages are generally delivered in the same order as they are sent.

First in First Out (FIFO)

Delivers exactly-once processing.

The order in which messages are sent and received is strictly preserved and a message is delivered once and remains available until a consumer processes and deletes it.

Duplicates are not introduced into the queue.

FIFO queues also support message groups that allow multiple ordered message groups within a single queue.

Limited to 300 transactions per second (TPS) but have all the capabilities of standard queues.

Deduplication with FIFO queues:

Provide a MessageDeduplicationId with the message.

The de-duplication interval is 5 minutes.

Content based duplication – the MessageDeduplicationId is generated as the SHA-256 with the message body.

Sequencing with FIFO queues:

To ensure strict ordering between messages, specify a MessageGroupId.

Messages with a different Group ID may be received out of order.

Messages with the same Group ID are delivered to one consumer at a time.

FIFO queues require the Message Group ID and Message Deduplication ID parameters to be added to messages.

Message Group ID:

The tag that specifies that a message belongs to a specific message group. Messages that belong to the same message group are guaranteed to be processed in a FIFO manner.

Message Deduplication ID:

The token used for deduplication of messages within the deduplication interval.

The following table provides a side-by-side comparison of standard and FIFO queues:

|  |  |
| --- | --- |
| Standard Queue | FIFO Queue |
| Unlimited Throughput: Standard queues support a nearly unlimited number of transactions per second (TPS) per API action. | High Throughput: FIFO queues support up to 300 messages per second (300 send, receive, or delete operations per second). When you batch 10 messages per operation (maximum), FIFO queues can support up to 3,000 messages per second |
| Best-Effort Ordering: Occasionally, messages might be delivered in an order different from which they were sent | First-ln-First-out Delivery: The order in which messages are sent and received is strictly preserved |
| At-Least-Once Delivery: A message is delivered at least once, but occasionally more than one copy of a message is delivered | Exactly-Once Processing: A message is delivered once and remains available until a consumer processes and deletes it. Duplicates are not introduced into the queue |

Scalability and Durability

You can have multiple queues with different priorities.

Scaling is performed by creating more queues.

SQS stores all message queues and messages within a single, highly available AWS region with multiple redundant AZs.

Visibility timeout

The amount of time a message is invisible in the queue after a reader picks it up.

Provided the job is processed before the visibility timeout expires, the message will then be deleted from the queue.

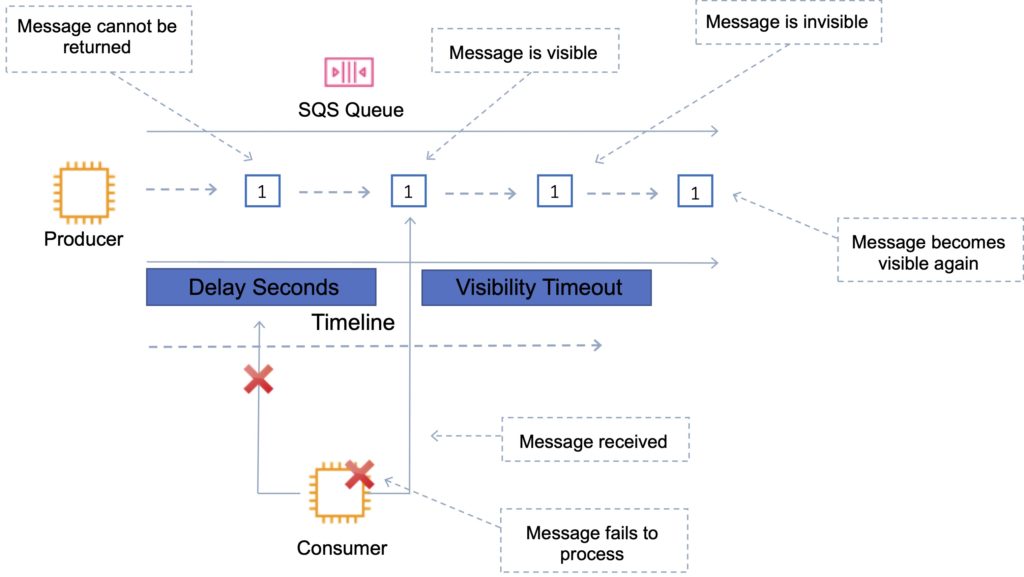
If the job is not processed within the visibility timeout, the message will become visible again and another reader will process it.

This could result in the same message being delivered twice.

The default visibility timeout is 30 seconds.

Increase it if your task takes >30 seconds.

The maximum is 12 hours.



SQS Polling

SQS uses short polling and long polling.

Amazon SQS long polling is a way to retrieve messages from SQS queues.

Short polling returns immediately (even if the message queue is empty).

Long polling doesn’t return a response until a message arrives in the message queue or the long poll times out.

Long polling can be enabled at the queue level or at the API level using WaitTimeSeconds.

Long polling is in effect when the Receive Message Wait Time is a value greater than 0 seconds and up to 20 seconds.

Short polling:

Does not wait for messages to appear in the queue.

It queries only a subset of the available servers for messages (based on weighted random execution).

Short polling is the default.

ReceiveMessageWaitTime is set to 0.

More requests are used, which implies higher cost.

Long polling:

Uses fewer requests and reduces cost.

Eliminates false empty responses by querying all servers.

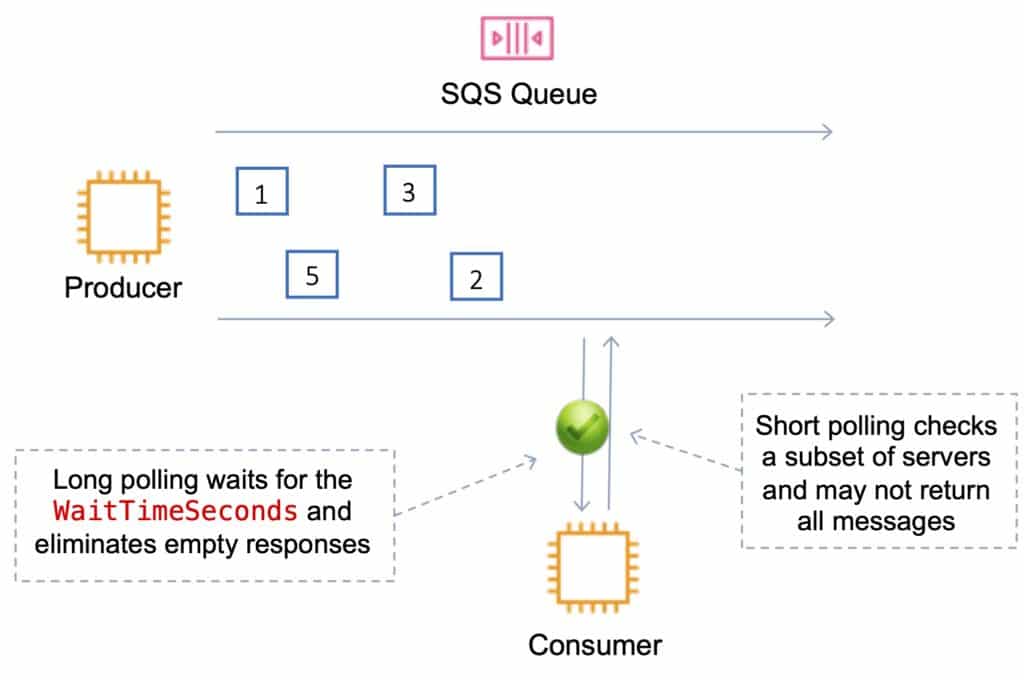
SQS waits until a message is available in the queue before sending a response.

Requests contain at least one of the available messages up to the maximum number of messages specified in the ReceiveMessage action.

Shouldn’t be used if your application expects an immediate response to receive message calls.

ReceiveMessageWaitTime is set to a non-zero value (up to 20 seconds).

Same charge per million requests as short polling.

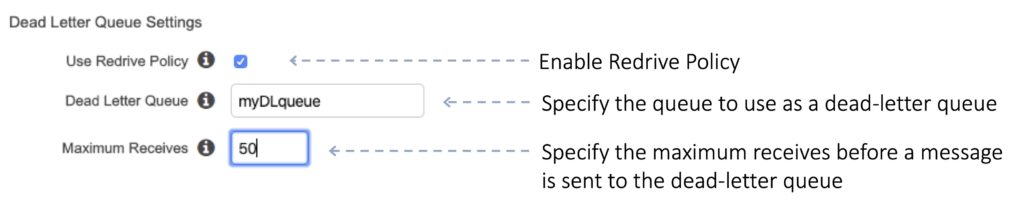


Dead-Letter Queue

The main task of a dead-letter queue is handling message failure.

A dead-letter queue lets you set aside and isolate messages that can’t be processed correctly to determine why their processing didn’t succeed.

It is not a queue type; it is a standard or FIFO queue that has been specified as a dead-letter queue in the configuration of another standard or FIFO queue.



Messages are moved to the dead-letter queue when the ReceiveCount for a message exceeds the maxReceiveCount for a queue.

Dead-letter queues should not be used with standard queues when your application will keep retrying transmission.

Dead-letter queues will break the order of messages in FIFO queues.



SQS Delay Queues

A delay queue postpones delivery of new messages to a queue for a number of seconds.

Messages sent to the Delay Queue remain invisible to consumers for the duration of the delay period.

Default delay is 0 seconds, maximum is 900 seconds (15 minutes).

For standard SQS queues, changing this setting doesn’t affect the delay of messages already in the queue, only new messages.

For FIFO queues, this affects the delay of messages already in the queue.

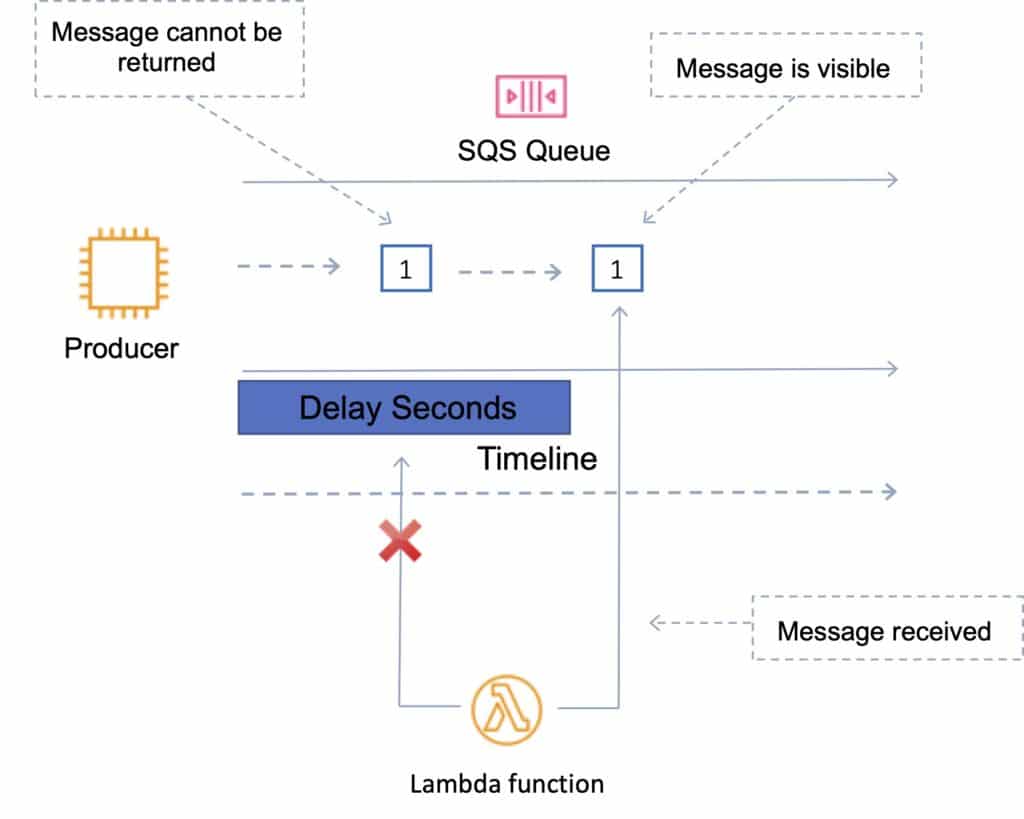
When to use a delay queue:

Large distributed applications which may need to introduce a delay in processing.

You need to apply a delay to an entire queue of messages.

For example, adding a delay of a few seconds to allow updates to sales or stock control databases before sending a notification to a customer confirming an online transaction.

DLQs must be the same type as the source.



Lambda with Amazon SQS

SQS can be configured as an event source for AWS Lambda functions.

Lambda with SQS:

You can use an AWS Lambda function to process messages in an Amazon Simple Queue Service (Amazon SQS) queue.

Lambda event source mappings support standard queues and first-in, first-out (FIFO) queues.

With Amazon SQS, you can offload tasks from one component of your application by sending them to a queue and processing them asynchronously.

Lambda polls the queue and invokes your Lambda function synchronously with an event that contains queue messages.

Lambda reads messages in batches and invokes your function once for each batch.

When your function successfully processes a batch, Lambda deletes its messages from the queue.

SQS Extended Client

The maximum message size in SQS is 256 KB.

You can use Amazon S3 and the Amazon SQS Extended Client Library for Java to manage Amazon SQS messages.

Useful for storing and consuming messages up to 2 GB in size.

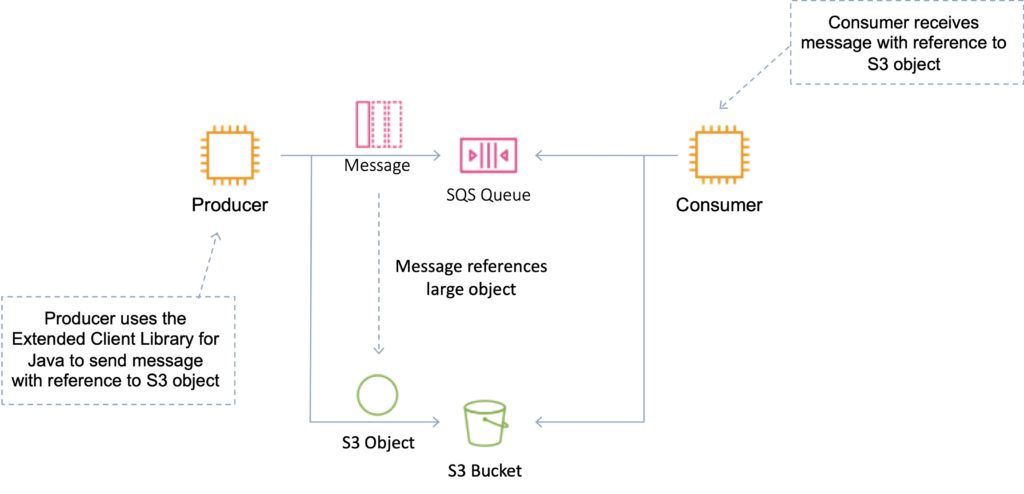
You can use the Amazon SQS Extended Client Library for Java library to do the following:

Specify whether messages are always stored in Amazon S3 or only when the size of a message exceeds 256 KB.

Send a message that references a single message object stored in an Amazon S3 bucket.

Get the corresponding message object from an Amazon S3 bucket.

Delete the corresponding message object from an Amazon S3 bucket.



Monitoring

CloudWatch is integrated with SQS and you can view and monitor queue metrics.

CloudWatch metrics are automatically collected every 5 minutes.

CloudWatch considers a queue to be active for up to 6 hours if it contains any messages or if any API action accesses it.

No charge for CloudWatch (no detailed monitoring).

CloudTrail captures API calls from SQS and logs to a specified S3 bucket.

Security

You can use IAM policies to control who can read/write messages.

Authentication can be used to secure messages within queues (who can send and receive).

SQS is PCI DSS level 1 compliant and HIPAA eligible.

In-flight security is provided using HTTPS.

Can enable server-side encryption (SSE) using KMS.

Can set the CMK you want to use.

Can set the data key reuse period.

SSE only encrypts the message body not the message attributes.

IAM policy must allow usage of SQS.

You can also specify permissions in an SQS queue access policy:

Providers finer grained control.

Control over the requests that come in.

Amazon SQS API’s

APIs you should know for the exam:

CreateQueue (aws sqs create-queue):

Creates a new standard or FIFO queue.  You can pass one or more attributes in the request.

DeleteQueue (aws sqs delete-queue):

Deletes the queue specified by the QueueUrl , regardless of the queue’s contents.

If the specified queue doesn’t exist, Amazon  SQS  returns a successful response.

PurgeQueue (aws sqs purge-queue):

Deletes the messages in a queue specified by the QueueURL parameter.

SendMessage (aws sqs send-message):

Delivers a message to the specified queue.

ReceiveMessage (aws sqs receive-messsage):

Retrieves one or more messages (up to 10), from the specified queue.

Using the WaitTimeSeconds parameter enables long-poll support.

DeleteMessage (aws sqs delete-message):

Deletes the specified message from the specified queue.

To select the message to delete, use the ReceiptHandle of the message (not the MessageId which you receive when you send the message).

ChangeMessageVisibility (aws sqs change-message-visibility):

Changes the visibility timeout of a specified message in a queue to a new value.

The default visibility timeout for a message is 30  seconds.

The minimum is 0 seconds.

The maximum is 12 hours.

To reduce costs or manipulate up to 10 messages with a single action, you can use the following actions:

SendMessageBatch

DeleteMessageBatch

ChangeMessageVisibilityBatch

Amazon SWF

Amazon Simple Workflow Service (SWF) is a web service that makes it easy to coordinate work across distributed application components.

Amazon SWF is used for processing background jobs that have parallel or sequential steps.

You can think of Amazon SWF as a fully managed state tracker and task coordinator.

Use Amazon SWF if your app’s steps take more than 500 milliseconds to complete, you need to track the state of processing, or you need to recover or retry if a task fails.

With SWF you can create distributed asynchronous systems as workflows.

Tracks the state of your workflow which you interact and update via API.

Best suited for human-enabled workflows like an order fulfilment system or for procedural requests.

AWS recommends that for new applications customers consider AWS Step Functions instead of SWF.

SWF enables applications for a range of use cases, including media processing, web application back-ends, business process workflows, and analytics pipelines.

Registration is a one-time step that you perform for each different type of workflow and activity.

SWF has a completion time of up to 1 year for workflow executions.

SWF uses a task-oriented API.

SWF ensures a task is assigned once and never duplicated.

SWF keeps track of all the tasks and events in an application.

A domain is a logical container for application resources such as workflows, activities, and executions.

Workers are programs that interact with Amazon SWF to get tasks, process received tasks, and return the results.

The decider is a program that controls the coordination of tasks, i.e. their ordering, concurrency, and scheduling according to the application logic.

SWF applications include the following logical components:

Domains.

Workflows.

Activities.

Task Lists.

Workers.

Workflow Execution.

Amazon MQ

Amazon MQ is a managed message broker service for ActiveMQ..

Amazon MQ supports industry-standard APIs and protocols so you can migrate messaging and applications without rewriting code.

Amazon MQ provides cost-efficient and flexible messaging capacity – you pay for broker instance and storage usage as you go.

Amazon MQ manages the administration and maintenance of ActiveMQ brokers and automatically provisions infrastructure for high availability.

With Amazon MQ, you can use the AWS Management Console, AWS CloudFormation, the Command Line Interface (CLI), or simple API calls to launch a production-ready message broker in minutes.

Amazon MQ is fully managed and highly available within a region.

Amazon MQ stores your messages redundantly across multiple Availability Zones (AZs).

Active/standby brokers are designed for high availability.

In the event of a failure of the broker, or even a full AZ outage, Amazon MQ automatically fails over to the standby broker so you can continue sending and receiving messages.

ActiveMQ API and support for JMS, NMS, MQTT, and WebSockets.

It is designed as a drop-in replacement for on-premises message brokers.

Use SQS if you’re creating a new application from scratch.

Use MQ if you want an easy low-hassle path to migrate from existing message brokers to AWS.

Amazon MQ provides encryption of your messages at rest and in transit.

It’s easy to ensure that your messages are securely stored in an encrypted format. Connections to the broker use SSL, and access can be restricted to a private endpoint within your Amazon VPC, which allows you to isolate your broker in your own virtual network.

You can configure security groups to control network access to your broker.

Amazon MQ is integrated with Amazon CloudWatch and AWS CloudTrail. With CloudWatch you can monitor metrics on your brokers, queues, and topics.

AWS Step Functions

AWS Step Functions can be used to coordinate the components of distributed applications as a series of steps in a visual workflow.

You can quickly build and run state machines to execute the steps of your application in a reliable and scalable fashion.

How it works:

Define the steps of your workflow in the JSON-based Amazon States Language. The visual console automatically graphs each step in the order of execution.

Start an execution to visualize and verify the steps of your application are operating as intended. The console highlights the real-time status of each step and provides a detailed history of every execution.

AWS Step Functions operates and scales the steps of your application and underlying compute for you to help ensure your application executes reliably under increasing demand.

It is a managed workflow and orchestration platform.

It is scalable and highly available.

You define your app as a state machine.

Create tasks, sequential steps, parallel steps, branching paths or timers.

Uses Amazon State Language declarative JSON.

Apps can interact and update the stream via Step Function API.

Provides a visual interface which describes flow and real-time status.

Provides detailed logs of each step execution.

AWS Step Functions features:

Built-in error handling – AWS Step Functions tracks the state of each step, so you can automatically retry failed or timed-out tasks, catch specific errors, and recover gracefully, whether the task takes seconds or months to complete.

Automatic Scaling – AWS Step Functions automatically scales the operations and underlying compute to run the steps of your application for you in response to changing workloads. Step Functions scales automatically to help ensure the performance of your application workflow remains consistently high as the frequency of requests increases.

Pay per use – With AWS Step Functions, you pay only for the transition from one step of your application workflow to the next, called a state transition. Billing is metered by state transition, regardless of how long each state persists (up to one year).

Execution event history – AWS Step Functions creates a detailed event log for every execution, so when things do go wrong, you can quickly identify not only where, but why. All of the execution history is available visually and programmatically to quickly troubleshoot and remediate failures.

High availability – AWS Step Functions has built-in fault tolerance. Step Functions maintains service capacity across multiple Availability Zones in each region to help protect application workflows against individual machine or data center facility failures. There are no maintenance windows or scheduled downtimes.

Administrative security – AWS Step Functions is integrated with AWS Identity and Access Management (IAM). IAM policies can be used to control access to the Step Functions APIs.

Amazon CloudWatch

Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS.

CloudWatch is used to collect and track metrics, collect, and monitor log files, and set alarms.

With CloudWatch you can:

Gain system-wide visibility into resource utilization.

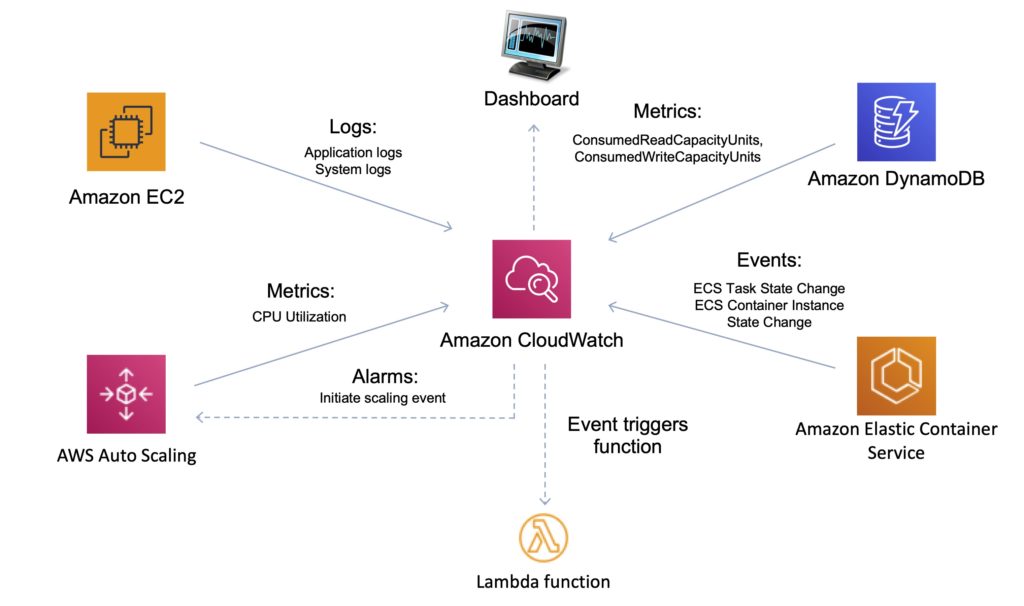
Monitor application performance.

Monitor operational health.

CloudWatch alarms monitor metrics and can be configured to automatically initiate actions.

CloudWatch Logs centralizes logs from systems, applications, and AWS services.

CloudWatch Events delivers a stream of system events that describe changes in AWS resources.



CloudWatch is accessed via API, command-line interface, AWS SDKs, and the AWS Management Console.

CloudWatch integrates with AWS IAM.

CloudWatch can automatically react to changes in your AWS resources.

With CloudWatch you can monitor resources such as:

EC2 instances.

DynamoDB tables.

RDS DB instances.

Custom metrics generated by applications and services.

Any log files generated by your applications.

CloudWatch retains metric data as follows:

Data points with a period of less than 60 seconds are available for 3 hours. These data points are high-resolution custom metrics.

Data points with a period of 60 seconds (1 minute) are available for 15 days.

Data points with a period of 300 seconds (5 minute) are available for 63 days.

Data points with a period of 3600 seconds (1 hour) are available for 455 days (15 months).

Amazon CloudWatch vs AWS CloudTrail:

|  |  |
| --- | --- |
| CloudWatch | CloudTrail |
| Performance monitoring | Auditing |
| Log events across AWS Services – think operations | Log API activity across AWS services – think activities, or who to blame |
| Higher-level comprehensive monitoring and event service | More low-level, granular |
| Log from multiple accounts | Log from multiple accounts |
| Logs stored indefinitely | Logs stored to S3 or CloudWatch indefinitely |
| Alarms history for 14 days | No native alarming; can use CloudWatch alarms |

Metrics

Metrics are the fundamental concept in CloudWatch.

A metric represents a time-ordered set of data points that are published to CloudWatch.

AWS services send metrics to CloudWatch.

You can also send your own custom metrics to CloudWatch.

Metrics exist within a region.

Metrics cannot be deleted but automatically expire after 15 months.

Metrics are uniquely defined by a name, a namespace, and zero or more dimensions.

CloudWatch retains metric data as follows:

Data points with a period of less than 60 seconds are available for 3 hours. These data points are high-resolution custom metrics.

Data points with a period of 60 seconds (1 minute) are available for 15 days.

Data points with a period of 300 seconds (5 minute) are available for 63 days.

Data points with a period of 3600 seconds (1 hour) are available for 455 days (15 months).

Custom Metrics

You can publish your own metrics to CloudWatch using the AWS CLI or an API.

You can view statistical graphs of your published metrics with the AWS Management Console.

CloudWatch stores data about a metric as a series of data points.

Each data point has an associated time stamp.

You can even publish an aggregated set of data points called a statistic set.

High-Resolution Metrics

Each metric is one of the following:

Standard resolution, with data having a one-minute granularity

High resolution, with data at a granularity of one second

Metrics produced by AWS services are standard resolution by default.

When you publish a custom metric, you can define it as either standard resolution or high resolution.

When you publish a high-resolution metric, CloudWatch stores it with a resolution of 1 second, and you can read and retrieve it with a period of 1 second, 5 seconds, 10 seconds, 30 seconds, or any multiple of 60 seconds.

High-resolution metrics can give you more immediate insight into your application’s sub-minute activity.

Keep in mind that every PutMetricData call for a custom metric is charged, so calling PutMetricData more often on a high-resolution metric can lead to higher charges.

If you set an alarm on a high-resolution metric, you can specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds.

There is a higher charge for high-resolution alarms with a period of 10 or 30 seconds.

Namespace

A namespace is a container for CloudWatch metrics.

Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics.

The following table provides some examples of namespaces for several AWS services:

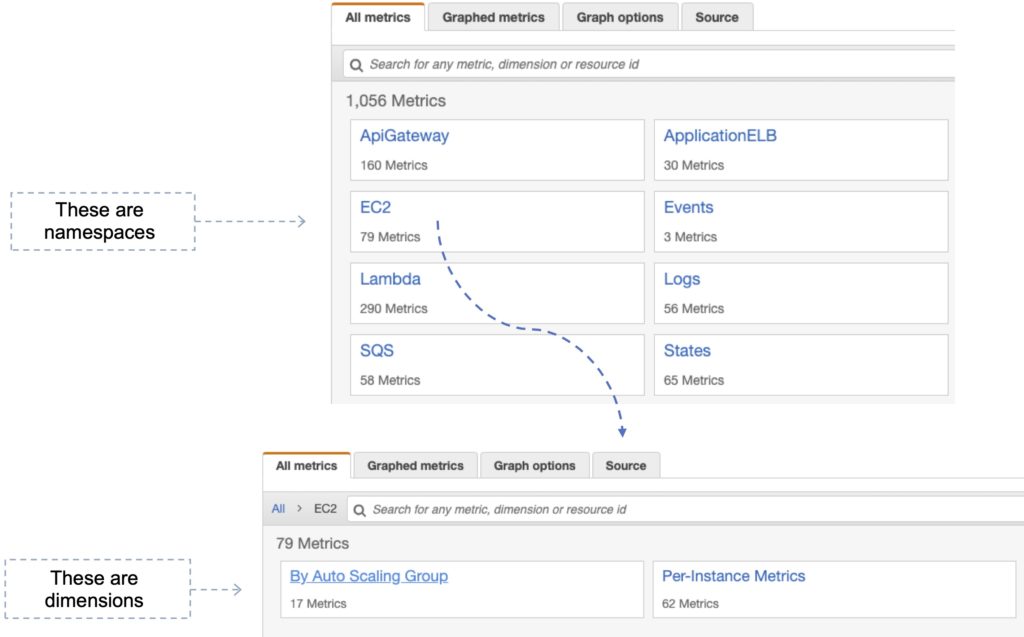
|  |  |
| --- | --- |
| Service | Namespace |
| Amazon API Gateway | AWS/ApiGateway |
| Amazon CloudFront | AWS/CloudFront |
| AWS CloudHSM | AWS/CloudHSM |
| Amazon CloudWatch Logs | AWS/Logs |
| AWS CodeBuild | AWS/CodeBuild |
| Amazon Cognito | AWS/Cognito |
| Amazon DynamoDB | AWS/DynamoDB |
| Amazon EC2 | AWS/EC2 |
| AWS Elastic Beanstalk | AWS/ElasticBeanstalk |

Dimensions

In custom metrics, the –dimensions parameter is common.

A dimension further clarifies what the metric is and what data it stores.

You can have up to 10 dimensions in one metric, and each dimension is defined by a name and value pair.



How you specify a dimension is different when you use different commands.

With [put-metric-data](https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/put-metric-data.html), you specify each dimension as MyName=MyValue, and with [get-metric-statistics](https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/get-metric-statistics.html) or [put-metric-alarm](https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/put-metric-alarm.html) you use the format Name=MyName, Value=MyValue.

For example, the following command publishes a Buffers metric with two dimensions named InstanceId and InstanceType.

aws cloudwatch put-metric-data –metric-name Buffers –namespace MyNameSpace –unit Bytes –value 231434333 –dimensions InstanceId=1-23456789,InstanceType=m1.small

This command retrieves statistics for that same metric. Separate the Name and Value parts of a single dimension with commas, but if you have multiple dimensions, use a space between one dimension and the next.

aws cloudwatch get-metric-statistics –metric-name Buffers –namespace MyNameSpace –dimensions Name=InstanceId,Value=1-23456789 Name=InstanceType,Value=m1.small –start-time 2016-10-15T04:00:00Z –end-time 2016-10-19T07:00:00Z –statistics Average –period 60

If a single metric includes multiple dimensions, you must specify a value for every defined dimension when you use [get-metric-statistics](https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/get-metric-statistics.html). For example, the Amazon S3 metric BucketSizeBytes includes the dimensions BucketName and StorageType, so you must specify both dimensions with [get-metric-statistics](https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/get-metric-statistics.html).

aws cloudwatch get-metric-statistics –metric-name BucketSizeBytes –start-time 2017-01-23T14:23:00Z –end-time 2017-01-26T19:30:00Z –period 3600 –namespace AWS/S3 –statistics Maximum –dimensions Name=BucketName,Value=MyBucketName Name=StorageType,Value=StandardStorage –output table

Publishing Single Data Points

To publish a single data point for a new or existing metric, use the [put-metric-data](https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/put-metric-data.html) command with one value and time stamp.

For example, the following actions each publish one data point.

aws cloudwatch put-metric-data –metric-name PageViewCount –namespace MyService –value 2 –timestamp 2016-10-20T12:00:00.000Z

aws cloudwatch put-metric-data –metric-name PageViewCount –namespace MyService –value 4 –timestamp 2016-10-20T12:00:01.000Z

aws cloudwatch put-metric-data –metric-name PageViewCount –namespace MyService –value 5 –timestamp 2016-10-20T12:00:02.000Z

Statistics

Statistics are metric data aggregations over specified periods of time.

CloudWatch provides statistics based on the metric data points provided by your custom data or provided by other AWS services to CloudWatch.

|  |  |
| --- | --- |
| Statistic | Description |
| Minimum | The lowest value observed during the specified period. You can use this value to determine low volumes of activity for your application. |
| Maximum | The highest value observed during the specified period. You can use this value to determine high volumes of activity for your application. |
| Sum | All values submitted for the matching metric added together. This statistic can be useful for determining the total volume of a metric. |
| Average | The value Sum/SampleCount during the specific period. By comparing this statistic with the Minimum and Maximum you can determine the full scope of a metric and how close the average is to the Minimum and Maximum. This comparison helps you know when to increase or decrease your resources as needed. |
| SampleCount | The count (number) of data points used for the statistical calculation. |
| pNN.NN | The value of the specified percentile. You can specify and percentile using up to two decimal places (e.. p45.45). Percentile statistics are not available for metric that include negative values. For more information see [Percentiles](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_concepts.html#Percentiles). |

CloudWatch Alarms

You can use an alarm to automatically initiate actions on your behalf.

An alarm watches a single metric over a specified time period, and performs one or more specified actions, based on the value of the metric relative to a threshold over time.

The action is a notification sent to an Amazon SNS topic or an Auto Scaling policy.

You can also add alarms to dashboards.

Alarms invoke actions for sustained state changes only.

CloudWatch alarms do not invoke actions simply because they are in a particular state.

The state must have changed and been maintained for a specified number of periods.

CloudWatch Logs

Amazon CloudWatch Logs lets you monitor and troubleshoot your systems and applications using your existing system, application, and custom log files.

You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS CloudTrail, Route 53, and other sources.

Features:

Monitor logs from Amazon EC2 instances – monitors application and system logs and can trigger notifications.

Monitor CloudTrail Logged Events – alarms can be created in CloudWatch based on API activity captured by CloudTrail.

Log retention – by default, logs are retained indefinitely. Configurable per log group from 1 day to 10 years.

CloudWatch Logs can be used for real time application and system monitoring as well as long term log retention.

CloudTrail logs can be sent to CloudWatch Logs for real-time monitoring.

CloudWatch Logs metric filters can evaluate CloudTrail logs for specific terms, phrases, or values.

Export to S3 possible with CreateExportTask – takes 21 hours.

For near real-time or persistent logs use Kinesis Firehose.

Use Firehose for any Firehose supported destinations.

For real-time use Lambda or Kinesis Data Stream with KCL consumers.

Use a metric filter for scanning log data which generates a CloudWatch metric.

Subscription filters can be created for sending data to a subscriber.

CloudWatch Logs Agent

The CloudWatch Logs agent provides an automated way to send log data to CloudWatch Logs from Amazon EC2 instances.

There is now a unified CloudWatch agent that collects both logs and metrics.

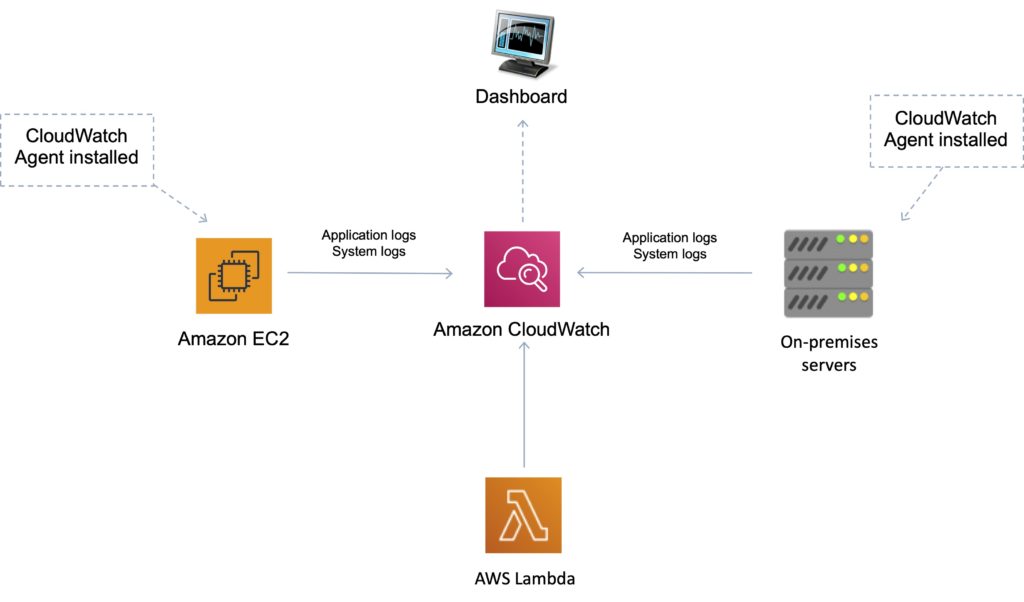
The unified CloudWatch agent includes metrics such as memory and disk utilization.

The unified CloudWatch agent enables you to do the following:

Collect more system-level metrics from Amazon EC2 instances across operating systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances.

Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.

Retrieve custom metrics from your applications or services using the StatsD and collectd protocols.



CloudWatch Events

Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.

Can use CloudWatch Events to schedule automated actions that self-trigger at certain times using cron or rate expressions

Can match events and route them to one or more target functions or streams.

Targets include:

Amazon EC2 instances.

AWS Lambda functions.

Streams in Amazon Kinesis Data Streams.

Delivery streams in Amazon Kinesis Data Firehose.

Log groups in Amazon CloudWatch Logs.

Amazon ECS tasks.

Systems Manager Run Command.

Systems Manager Automation.

AWS Batch jobs.

Step Functions state machines.

Pipelines in CodePipeline.

CodeBuild projects.

Amazon Inspector assessment templates.

Amazon SNS topics.

Amazon SQS queues.

In the following example, an EC2 instance changes state (terminated) and the event is sent to CloudWatch Events which forwards the event to the target (SQS queue).



Useful API Actions

It is useful to understand the following API actions for the Developer Associate exam. You should check these out and other API actions on the AWS website as well prior to your exam.

GetMetricData

Retrieve as many as 500 different metrics in a single request.

PutMetricData

Publishes metric data points to Amazon CloudWatch.

CloudWatch associates the data points with the specified metric.

If the specified metric does not exist, CloudWatch creates the metric.

GetMetricStatistics

Gets statistics for the specified metric.

CloudWatch aggregates data points based on the length of the period that you specify.

Maximum number of data points returned from a single call is 1,440.

PutMetricAlarm

Creates or updates an alarm and associates it with the specified metric, metric math expression, or anomaly detection model.

Alarms based on anomaly detection models cannot have Auto Scaling actions.

AWS CloudTrail

AWS CloudTrail is a web service that records activity made on your account.

A CloudTrail trail can be created which delivers log files to an Amazon S3 bucket.

CloudTrail is about logging and saves a history of API calls for your AWS account.

CloudTrail enables governance, compliance, and operational and risk auditing of your AWS account.

Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.

CloudTrail provides visibility into user activity by recording actions taken on your account.

API history enables security analysis, resource change tracking, and compliance auditing.

Logs API calls made via:

AWS Management Console.

AWS SDKs.

Command line tools.

Higher-level AWS services (such as CloudFormation).

CloudTrail records account activity and service events from most AWS services and logs the following records:

The identity of the API caller.

The time of the API call.

The source IP address of the API caller.

The request parameters.

The response elements returned by the AWS service.

CloudTrail is enabled on your AWS account when you create it.

CloudTrail is per AWS account.

[](https://www.adplugg.com/track/click/A48221584/60493/click?toi=21584&hn=digitalcloud.training&bu=%2Faws-cloudtrail%2F&rf=&zn=16423&pm=8399&ct=&next=)

You can create two types of trails for an AWS account:

A trail that applies to all regions – records events in all regions and delivers to an S3 bucket.

A trail that applies to a single region – records events in a single region and delivers to an S3 bucket. Additional single trails can use the same or a different S3 bucket.

Trails can be configured to log data events and management events:

Data events: These events provide insight into the resource operations performed on or within a resource. These are also known as data plane operations.

Management events: Management events provide insight into management operations that are performed on resources in your AWS account. These are also known as control plane operations. Management events can also include non-API events that occur in your account.

Example data events include:

Amazon S3 object-level API activity (for example, GetObject, DeleteObject, and PutObject API operations).

AWS Lambda function execution activity (the Invoke API).

Example management events include:

Configuring security (for example, IAM AttachRolePolicy API operations).

Registering devices (for example, Amazon EC2 CreateDefaultVpc API operations).

Configuring rules for routing data (for example, Amazon EC2 CreateSubnet API operations).

Setting up logging (for example, AWS CloudTrail CreateTrail API operations).

CloudTrail log files are encrypted using S3 Server Side Encryption (SSE).

You can also enable encryption using SSE KMS for additional security.

A single KMS key can be used to encrypt log files for trails applied to all regions.

[](https://www.adplugg.com/track/click/A48221584/60494/click?toi=21584&hn=digitalcloud.training&bu=%2Faws-cloudtrail%2F&rf=&zn=16425&pm=8400&ct=&next=)

You can consolidate logs from multiple accounts using an S3 bucket:

Turn on CloudTrail in the paying account.

Create a bucket policy that allows cross-account access.

Turn on CloudTrail in the other accounts and use the bucket in the paying account.

You can integrate CloudTrail with CloudWatch Logs to deliver data events captured by CloudTrail to a CloudWatch Logs log stream.

CloudTrail log file integrity validation feature allows you to determine whether a CloudTrail log file was unchanged, deleted, or modified since CloudTrail delivered it to the specified Amazon S3 bucket.

CloudWatch vs CloudTrail:

|  |  |
| --- | --- |
| CloudWatch | CloudTrail |
| Performance monitoring | Auditing |
| Log events across AWS services – think operations | Log API activity across AWS services – think activities |
| Higher-level comprehensive monitoring and events | More low-level granular |
| Log from multiple accounts | Log from multiple accounts |
| Logs stored indefinitely | Logs stored to S3 or CloudWatch indefinitely |
| Alarms history for 14 days | No native alarming; can use CloudWatch alarms |

Amazon RDS

Amazon Relational Database Service (Amazon RDS) is a managed service that you can use to launch and manage relational databases on AWS.

Amazon RDS is an Online Transaction Processing (OLTP) type of database.

The primary use case is a transactional database (rather than an analytical database).

It is best suited to structured, relational data store requirements.

It aims to be drop-in replacement for existing on-premises instances of the same databases.

Automated backups and patching are applied in customer-defined maintenance windows.

Push-button scaling, replication, and redundancy.

Amazon RDS supports the following database engines:

Amazon Aurora.

MySQL.

MariaDB.

Oracle.

SQL Server.

PostgreSQL.

RDS is a managed service and you do not have access to the underlying EC2 instance (no root access).

The exception to the above rule is Amazon RDS Custom which allows access to the underlying operating system. This is new, available for limited DB engines, and does not appear on the exam yet.

The Amazon RDS managed service includes the following:

Security and patching of the DB instances.

Automated backup for the DB instances.

Software updates for the DB engine.

Easy scaling for storage and compute.

Multi-AZ option with synchronous replication.

Automatic failover for Multi-AZ option.

Read replicas option for read heavy workloads.

A DB instance is a database environment in the cloud with the compute and storage resources you specify.

Database instances are accessed via endpoints.

Endpoints can be retrieved via the DB instance description in the AWS Management Console, DescribeDBInstances API or describe-db-instances command.

By default, customers are allowed to have up to a total of 40 Amazon RDS DB instances (only 10 of these can be Oracle or MS SQL unless you have your own licenses).

Maintenance windows are configured to allow DB instances modifications to take place such as scaling and software patching (some operations require the DB instance to be taken offline briefly).

You can define the maintenance window or AWS will schedule a 30-minute window.

Windows integrated authentication for SQL only works with domains created using the AWS directory service – need to establish a trust with an on-premises AD directory.

Events and Notifications:

Amazon RDS uses AWS SNS to send RDS events via SNS notifications.

You can use API calls to the Amazon RDS service to list the RDS events in the last 14 days (DescribeEvents API).

You can view events from the last 14 days using the CLI.

Using the AWS Console you can only view RDS events for the last 1 day.

Use Cases, Alternatives and Anti-Patterns

The table below provides guidance on when best to use RDS and several other AWS database/data store services:

|  |  |
| --- | --- |
| Data Store | When to Use |
| Database on EC2 | Ultimate control over database  Preferred DB not available under RDS |
| Amazon RDS | Need traditional relational database for OLTP  Your data is well formed and structured  Existing apps requiring RDBMS |
| Amazon DynamoDB | Name/value pair data or unpredictable data structure  In-memory performance with persistence  High I/O needs  Scale dynamically |
| Amazon RedShift | Massive amounts of data  Primarily OLAP workloads |
| Amazon Neptune | Relationships between objects a major portion of data value |
| Amazon Elasticache | Fast temporary storage for small amounts of data  Highly volatile data |
| Amazon S3 | BLOBs  Static Websites |

Alternative to Amazon RDS:

If your use case isn’t supported on RDS, you can run databases on Amazon EC2.

Consider the following points when considering a DB on EC2:

You can run any database you like with full control and ultimate flexibility.

You must manage everything like backups, redundancy, patching and scaling.

Good option if you require a database not yet supported by RDS, such as IBM DB2 or SAP HANA.

Good option if it is not feasible to migrate to AWS-managed database.

Anti-Patterns:

Anti-patterns are certain patterns in architecture or development that are considered bad, or sub-optimal practices – i.e. there may be a better service of method to produce the best result.

The following table describes requirements that are not a good fit for RDS:

|  |  |
| --- | --- |
| Requirement | More Suitable Service |
| Lots of large binary objects (BLOBs) | S3 |
| Automated Scalability | DynamoDB |
| Name/Value Data Structure | DynamoDB |
| Data is not well structured or unpredictable | DynamoDB |
| Other database platforms like IBM DB2 or SAP HANA | EC2 |
| Complete control over the database | EC2 |

Encryption

You can encrypt your Amazon RDS instances and snapshots at rest by enabling the encryption option for your Amazon RDS DB instance.

Encryption at rest is supported for all DB types and uses AWS KMS.

When using encryption at rest the following elements are also encrypted:

All DB snapshots.

Backups.

DB instance storage.

Read Replicas.

You cannot encrypt an existing DB, you need to create a snapshot, copy it, encrypt the copy, then build an encrypted DB from the snapshot.

Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, Read Replicas, and snapshots.

A Read Replica of an Amazon RDS encrypted instance is also encrypted using the same key as the master instance when both are in the same region.

If the master and Read Replica are in different regions, you encrypt using the encryption key for that region.

You can’t have an encrypted Read Replica of an unencrypted DB instance or an unencrypted Read Replica of an encrypted DB instance.

Encryption/decryption is handled transparently.

RDS supports SSL encryption between applications and RDS DB instances.

RDS generates a certificate for the instance.

DB Subnet Groups

A DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances.

Each DB subnet group should have subnets in at least two Availability Zones in a given region.

It is recommended to configure a subnet group with subnets in each AZ (even for standalone instances).

During the creation of an RDS instance you can select the DB subnet group and the AZ within the group to place the RDS DB instance in.

You cannot pick the IP within the subnet that is allocated.

Billing and Provisioning

AWS Charge for:

DB instance hours (partial hours are charged as full hours).

Storage GB/month.

I/O requests/month – for magnetic storage.

Provisioned IOPS/month – for RDS provisioned IOPS SSD.

Egress data transfer.

Backup storage (DB backups and manual snapshots).

Backup storage for the automated RDS backup is free of charge up to the provisioned EBS volume size.

However, AWS replicate data across multiple AZs and so you are charged for the extra storage space on S3.

For multi-AZ you are charged for:

Multi-AZ DB hours.

Provisioned storage.

Double write I/Os.

For multi-AZ you are not charged for DB data transfer during replication from primary to standby.

Oracle and Microsoft SQL licenses are included, or you can bring your own (BYO).

On-demand and reserved instance pricing available.

Reserved instances are defined based on the following attributes which must not be changed:

DB engine.

DB instance class.

Deployment type (standalone, multi-AZ\_.

License model.

Region.

Reserved instances:

Can be moved between AZs in the same region.

Are available for multi-AZ deployments.

Can be applied to Read Replicas if DB instance class and region are the same.

Scaling is achieved through changing the instance class for compute and modifying storage capacity for additional storage allocation.

Scalability

You can only scale RDS up (compute and storage).

You cannot decrease the allocated storage for an RDS instance.

You can scale storage and change the storage type for all DB engines except MS SQL.

For MS SQL the workaround is to create a new instance from a snapshot with the new configuration.

Scaling storage can happen while the RDS instance is running without outage however there may be performance degradation.

Scaling compute will cause downtime.

You can choose to have changes take effect immediately, however the default is within the maintenance window.

Scaling requests are applied during the specified maintenance window unless “apply immediately” is used.

All RDS DB types support a maximum DB size of 64 TiB except for Microsoft SQL Server (16 TiB)

Performance

Amazon RDS uses EBS volumes (never uses instance store) for DB and log storage.

There are three storage types available: General Purpose (SSD), Provisioned IOPS (SSD), and Magnetic.

General Purpose (SSD):

Use for Database workloads with moderate I/O requirement.

Cost effective.

Also called gp2.

3 IOPS/GB.

Burst up to 3000 IOPS.

Provisioned IOPS (SSD):

Use for I/O intensive workloads.

Low latency and consistent I/O.

User specified IOPS (see table below).

For provisioned IOPS storage the table below shows the range of Provisioned IOPS and storage size range for each database engine.

|  |  |  |
| --- | --- | --- |
| Database Engine | Range of Provisioned IOPS | Range of Storage |
| MariaDB | 1,000-80,000 IOPS | 100 GiB-64TiB |
| SQL Server | 1,000-64,000 IOPS | 20 GiB-16TiB |
| MySQL | 1,000-80,000 IOPS | 100 GiB-64TiB |
| Oracle | 1,000-256,000 IOPS | 100 GiB-64TiB |
| PostgreSQL | 1,000-80,000 IOPS | 100 GiB-64TiB |

Magnetic:

Not recommended anymore, available for backwards compatibility.

Doesn’t allow you to scale storage when using the SQL Server database engine.

Doesn’t support elastic volumes.

Limited to a maximum size of 4 TiB.

Limited to a maximum of 1,000 IOPS.

Multi-AZ and Read Replicas

Multi-AZ and Read Replicas are used for high availability, fault tolerance and performance scaling.

The table below compares multi-AZ deployments to Read Replicas:

|  |  |
| --- | --- |
| Multi-AZ Deployments | Read Replicas |
| Synchronous Replication – highly durable | Asynchronous replication – highly scalable |
| Only database engine on primary instance is active | All read replicas are accessible and can be used for read scaling |
| Automated backups are taken from standby | No backups configured by default |
| Always span two availability zones within a single region | Can be within an Availability Zone, Cross-AZ, or Cross-Region |
| Database engine version upgrades happen on primary | Database engine version upgrade is independent from source instance |
| Automatic failover to standby when a problem is detected | Can be manually promoted to a standalone database instance |

Multi-AZ

Multi-AZ RDS creates a replica in another AZ and synchronously replicates to it (DR only).

There is an option to choose multi-AZ during the launch wizard.

AWS recommends the use of provisioned IOPS storage for multi-AZ RDS DB instances.

Each AZ runs on its own physically distinct, independent infrastructure, and is engineered to be highly reliable.

You cannot choose which AZ in the region will be chosen to create the standby DB instance.

You can view which AZ the standby DB instance is created in.

A failover may be triggered in the following circumstances:

Loss of primary AZ or primary DB instance failure.

Loss of network connectivity on primary.

Compute (EC2) unit failure on primary.

Storage (EBS) unit failure on primary.

The primary DB instance is changed.

Patching of the OS on the primary DB instance.

Manual failover (reboot with failover selected on primary).

During failover RDS automatically updates configuration (including DNS endpoint) to use the second node.

Depending on the instance class it can take 1 to a few minutes to failover to a standby DB instance.

It is recommended to implement DB connection retries in your application.

Recommended to use the endpoint rather than the IP address to point applications to the RDS DB.

The method to initiate a manual RDS DB instance failover is to reboot selecting the option to failover.

A DB instance reboot is required for changes to take effect when you change the DB parameter group or when you change a static DB parameter.

The DB parameter group is a configuration container for the DB engine configuration.

You will be alerted by a DB instance event when a failover occurs.

The secondary DB in a multi-AZ configuration cannot be used as an independent read node (read or write).

There is no charge for data transfer between primary and secondary RDS instances.

System upgrades like OS patching, DB Instance scaling and system upgrades, are applied first on the standby, before failing over and modifying the other DB Instance.

In multi-AZ configurations snapshots and automated backups are performed on the standby to avoid I/O suspension on the primary instance.

Read Replica Support for Multi-AZ:

Amazon RDS Read Replicas for MySQL, MariaDB, PostgreSQL, and Oracle support Multi-AZ deployments.

Combining Read Replicas with Multi-AZ enables you to build a resilient disaster recovery strategy and simplify your database engine upgrade process.

A Read Replica in a different region than the source database can be used as a standby database and promoted to become the new production database in case of a regional disruption.

This allows you to scale reads whilst also having multi-AZ for DR.

The process for implementing maintenance activities is as follows:

Perform operations on standby.

Promote standby to primary.

Perform operations on new standby (demoted primary).

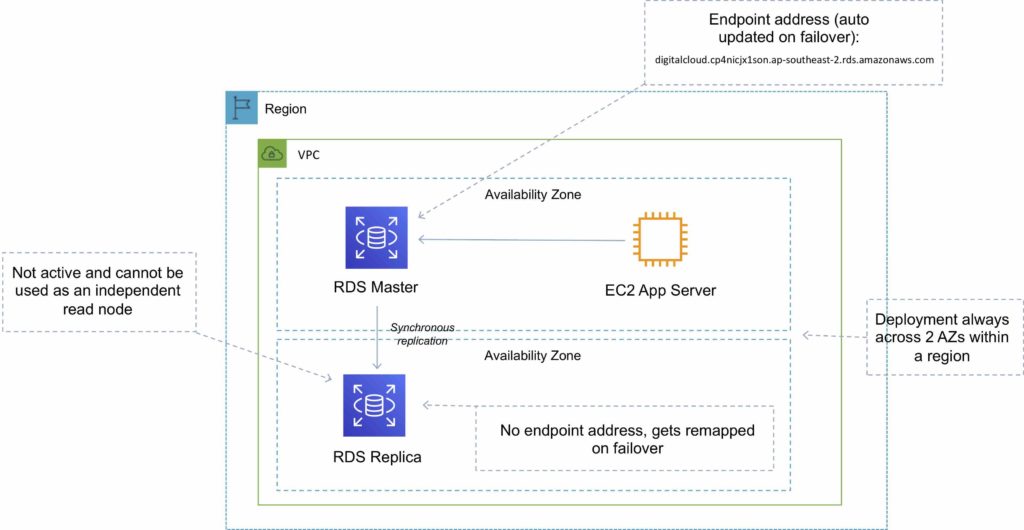
You can manually upgrade a DB instance to a supported DB engine version from the AWS Console.

By default upgrades will take effect during the next maintenance window.

You can optionally force an immediate upgrade.

In multi-AZ deployments version upgrades will be conducted on both the primary and standby at the same time causing an outage of both DB instance.

Ensure security groups and NACLs will allow your application servers to communicate with both the primary and standby instances.



Read Replicas

Read replicas are used for read-heavy DBs and replication is asynchronous.

Read replicas are for workload sharing and offloading.

Read replicas provide read-only DR.

Read replicas are created from a snapshot of the master instance.

Must have automated backups enabled on the primary (retention period > 0).

Only supported for transactional database storage engines (InnoDB not InnoDB).

Read replicas are available for MySQL, PostgreSQL, MariaDB, Oracle, Aurora, and SQL Server.

For the MySQL, MariaDB, PostgreSQL, and Oracle database engines, Amazon RDS creates a second DB instance using a snapshot of the source DB instance.

It then uses the engines’ native asynchronous replication to update the read replica whenever there is a change to the source DB instance.

[Amazon Aurora](https://aws.amazon.com/rds/aurora/) employs an SSD-backed virtualized storage layer purpose-built for database workloads.

You can take snapshots of PostgreSQL read replicas but cannot enable automated backups.

You can enable automatic backups on MySQL and MariaDB read replicas.

You can enable writes to the MySQL and MariaDB Read Replicas.

You can have 5 read replicas of a production DB.

You cannot have more than four instances involved in a replication chain.

You can have read replicas of read replicas for MySQL and MariaDB but not for PostgreSQL.

Read replicas can be configured from the AWS Console or the API.

You can specify the AZ the read replica is deployed in.

The read replicas storage type and instance class can be different from the source but the compute should be at least the performance of the source.

You cannot change the DB engine.

In a multi-AZ failover the read replicas are switched to the new primary.

Read replicas must be explicitly deleted.

If a source DB instance is deleted without deleting the replicas each replica becomes a standalone single-AZ DB instance.

You can promote a read replica to primary.

Promotion of read replicas takes several minutes.

Promoted read replicas retain:

Backup retention window.

Backup window.

DB parameter group.

Existing read replicas continue to function as normal.

Each read replica has its own DNS endpoint.

Read replicas can have multi-AZ enabled and you can create read replicas of multi-AZ source DBs.

Read replicas can be in another region (uses asynchronous replication).

This configuration can be used for centralizing data from across different regions for analytics.



DB Snapshots

DB Snapshots are user-initiated and enable you to back up your DB instance in a known state as frequently as you wish, and then restore to that specific state.

Cannot be used for point-in-time recovery.

Snapshots are stored on S3.

Snapshots remain on S3 until manually deleted.

Backups are taken within a defined window.

I/O is briefly suspended while backups initialize and may increase latency (applicable to single-AZ RDS).

DB snapshots that are performed manually will be stored even after the RDS instance is deleted.

Restored DBs will always be a new RDS instance with a new DNS endpoint.

Can restore up to the last 5 minutes.

Only default DB parameters and security groups are restored – you must manually associate all other DB parameters and SGs.

It is recommended to take a final snapshot before deleting an RDS instance.

Snapshots can be shared with other AWS accounts.

High Availability Approaches for Databases

If possible, choose DynamoDB over RDS because of inherent fault tolerance.

If DynamoDB can’t be used, choose Aurora because of redundancy and automatic recovery features.

If Aurora can’t be used, choose Multi-AZ RDS.

Frequent RDS snapshots can protect against data corruption or failure, and they won’t impact performance of Multi-AZ deployment.

Regional replication is also an option but will not be strongly consistent.

If the database runs on EC2, you must design the HA yourself.

Migration

AWS Database Migration Service helps you migrate databases to AWS quickly and securely.

Use along with the Schema Conversion Tool (SCT) to migrate databases to AWS RDS or EC2-based databases.

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.

Schema Conversion Tool can copy database schemas for homogenous migrations (same database) and convert schemas for heterogeneous migrations (different database).

DMS is used for smaller, simpler conversions and supports MongoDB and DynamoDB.

SCT is used for larger, more complex datasets like data warehouses.

DMS has replication functions for on-premises to AWS or to Snowball or S3.

Monitoring, Logging and Reporting

You can use the following automated monitoring tools to watch Amazon RDS and report when something is wrong:

Amazon RDS Events – Subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB snapshot, DB parameter group, or DB security group.

Database log files – View, download, or watch database log files using the Amazon RDS console or Amazon RDS API operations. You can also query some database log files that are loaded into database tables.

Amazon RDS Enhanced Monitoring — Look at metrics in real time for the operating system.

Amazon RDS Performance Insights — Assess the load on your database and determine when and where to act.

Amazon RDS Recommendations — Look at automated recommendations for database resources, such as DB instances, read replicas, and DB parameter groups.

In addition, Amazon RDS integrates with Amazon CloudWatch, Amazon EventBridge, and AWS CloudTrail for additional monitoring capabilities:

Amazon CloudWatch Metrics – Amazon RDS automatically sends metrics to CloudWatch every minute for each active database. You don’t get additional charges for Amazon RDS metrics in CloudWatch.

Amazon CloudWatch Alarms – You can watch a single Amazon RDS metric over a specific time period. You can then perform one or more actions based on the value of the metric relative to a threshold that you set.

Amazon CloudWatch Logs – Most DB engines enable you to monitor, store, and access your database log files in CloudWatch Logs.

Amazon CloudWatch Events and Amazon EventBridge – You can automate AWS services and respond to system events such as application availability issues or resource changes. Events from AWS services are delivered to CloudWatch Events and EventBridge nearly in real time. You can write simple rules to indicate which events interest you and what automated actions to take when an event matches a rule

AWS CloudTrail – You can view a record of actions taken by a user, role, or an AWS service in Amazon RDS. CloudTrail captures all API calls for Amazon RDS as events. These captures include calls from the Amazon RDS console and from code calls to the Amazon RDS API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon RDS. If you don’t configure a trail, you can still view the most recent events in the CloudTrail console in Event history.

Authorization and Access Control

Amazon RDS supports[identity-based policies](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/security_iam_service-with-iam.html#security_iam_service-with-iam-resource-based-policies).

RDS does not support resource-based policies.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon RDS:

AmazonRDSReadOnlyAccess – Grants read-only access to all Amazon RDS resources for the AWS account specified.

AmazonRDSFullAccess – Grants full access to all Amazon RDS resources for the AWS account specified.

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.

IAM database authentication provides the following benefits:

Network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).

You can use IAM to centrally manage access to your database resources, instead of managing access individually on each DB instance.

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.

Amazon Aurora

Amazon Aurora is a relational database service that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases.

Aurora is an AWS proprietary database.

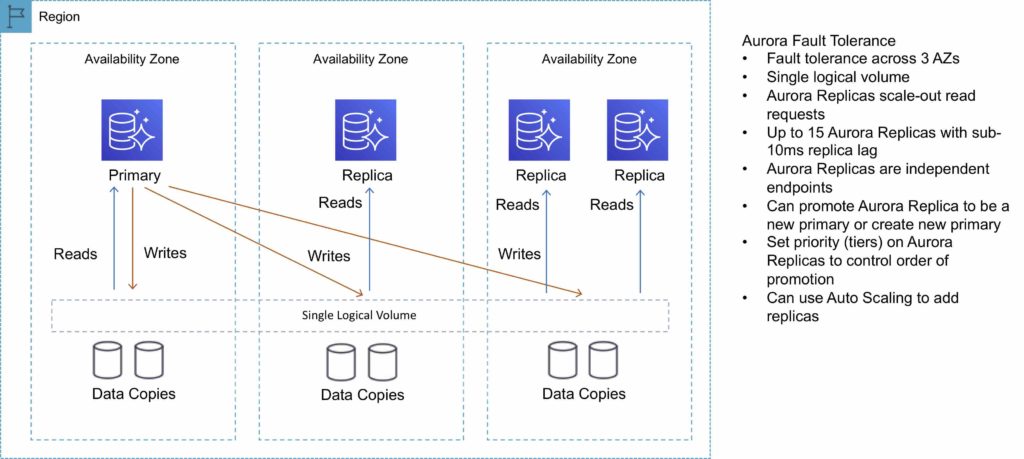
Fully managed service.

High performance, low price.

2 copies of data are kept in each AZ with a minimum of 3 AZ’s (6 copies).

Can handle the loss of up to two copies of data without affecting DB write availability and up to three copies without affecting read availability.

The following diagram depicts how Aurora Fault Tolerance and Replicas work:



Aurora Replicas

There are two types of replication: Aurora replica (up to 15), MySQL Read Replica (up to 5).

The table below describes the differences between the two replica options:

|  |  |  |
| --- | --- | --- |
| Feature | Aurora Replica | MySQL Replica |
| Number of replicas | Up to 15 | Up to 5 |
| Replication type | Asynchronous (milliseconds) | Asynchronous (seconds) |
| Performance impact on primary | Low | High |
| Replica location | In-region | Cross-region |
| Act as failover target | Yes (no data loss) | Yes (potentially minutes of data loss) |
| Automated failover | Yes | No |
| Support for user-defined replication delay | No | Yes |
| Support for different data or schema vs. primary | No | Yes |

You can create read replicas for an Amazon Aurora database in up to five AWS regions. This capability is available for Amazon Aurora with MySQL compatibility.

Cross-Region Read Replicas

Cross-region read replicas allow you to improve your disaster recovery posture, scale read operations in regions closer to your application users, and easily migrate from one region to another.

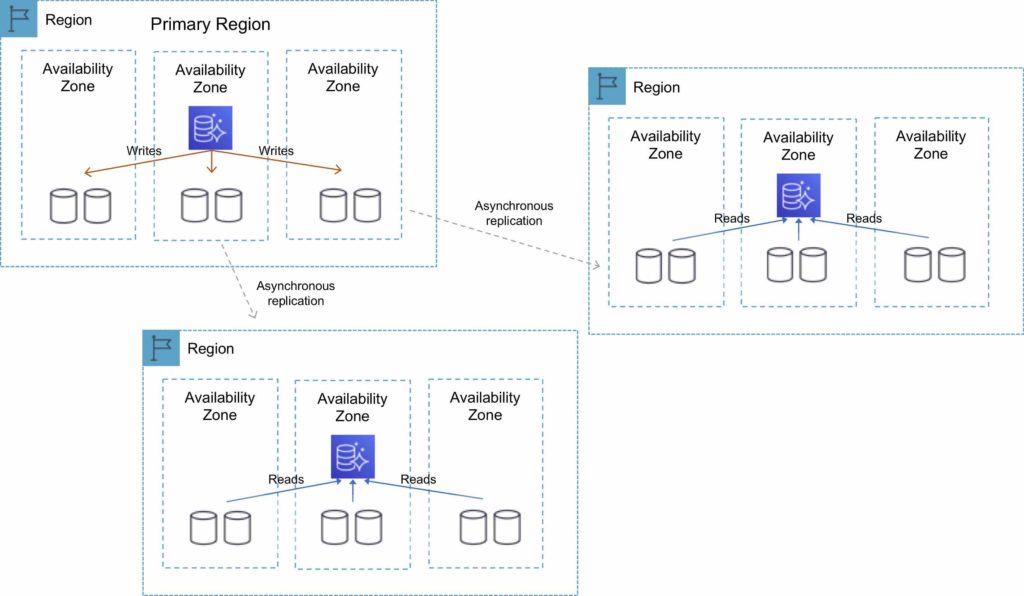
Cross-region replicas provide fast local reads to your users.

Each region can have an additional 15 Aurora replicas to further scale local reads.

You can choose between [Global Database](https://aws.amazon.com/rds/aurora/global-database/), which provides the best replication performance, and traditional binlog-based replication.

You can also set up your own binlog replication with external MySQL databases.

The following diagram depicts the Cross-Region Read Replica topology:



Global Database

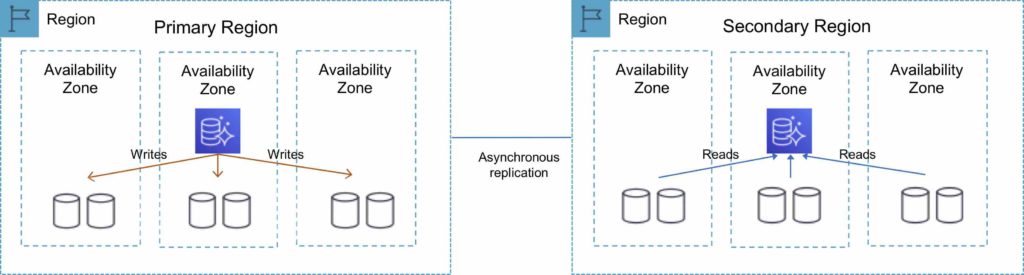
For globally distributed applications you can use [Global Database](https://aws.amazon.com/rds/aurora/global-database/), where a single Aurora database can span multiple AWS regions to enable fast local reads and quick disaster recovery.

Global Database uses storage-based replication to replicate a database across multiple AWS Regions, with typical latency of less than 1 second.

You can use a secondary region as a backup option in case you need to recover quickly from a regional degradation or outage.

A database in a secondary region can be promoted to full read/write capabilities in less than 1 minute.

The following table depicts the Aurora Global Database topology:



Multi-Master

Amazon Aurora Multi-Master is a new feature of the Aurora MySQL-compatible edition that adds the ability to scale out write performance across multiple Availability Zones, allowing applications to direct read/write workloads to multiple instances in a database cluster and operate with higher availability.

Aurora Multi-Master is designed to achieve high availability and ACID transactions across a cluster of database nodes with configurable read after write consistency.

Architecture

An Aurora cluster consists of a set of compute (database) nodes and a shared storage volume.

The storage volume consists of six storage nodes placed in three Availability Zones for high availability and durability of user data.

Every database node in the cluster is a writer node that can run read and write statements.

There is no single point of failure in the cluster.

Applications can use any writer node for their read/write and DDL needs.

A database change made by a writer node is written to six storage nodes in three Availability Zones, providing data durability and resiliency against storage node and Availability Zone failures.

The writer nodes are all functionally equal, and a failure of one writer node does not affect the availability of the other writer nodes in the cluster.

High Availability

Aurora Multi-Master improves upon the high availability of the single-master version of Amazon Aurora because all the nodes in the cluster are read/write nodes.

With single-master Aurora, a failure of the single writer node requires the promotion of a read replica to be the new writer.

In the case of Aurora Multi-Master, the failure of a writer node merely requires the application using the writer to open connections to another writer.

Aurora Serverless

Amazon Aurora Serverless is an on-demand, auto-scaling configuration for Amazon Aurora.

Available for MySQL-compatible and PostgreSQL-compatible editions.

The database automatically starts up, shuts down, and scales capacity up or down based on application needs.

It enables you to run a database in the cloud without managing any database instances. It’s a simple, cost-effective option for infrequent, intermittent, or unpredictable workloads.

You simply create a database endpoint and optionally specify the desired database capacity range and connect applications.

With Aurora Serverless, you only pay for database storage and the database capacity and I/O your database consumes while it is active.

Pay on a per-second basis for the database capacity you use when the database is active.

Can migrate between standard and serverless configurations with a few clicks in the Amazon RDS Management Console.

The table below provides a few example use cases for Amazon Aurora Serverless:

|  |  |
| --- | --- |
| Use Case | Example |
| Infrequently Used Applications | Application that is only used for a few minutes several times per day or week. Need a cost-effective database that only requires you to pay when it’s active. With Aurora Serverless, you only pay for the database resources you consume. |
| New Applications | Deploying a new application and are unsure which instance size you need. With Aurora Serverless, you simply create an endpoint and let the database auto-scale to the capacity requirements of your application. |
| Variable Workloads | Running a lightly used application, with peaks of 30 minutes to several hours a few times each day or several times per year. Now you only pay for what the resources needed based on load – avoiding paying for unused resources or risking poor performance. |
| Unpredictable Workloads | Running workloads where there is database usage throughout the day, and peaks of activity that are hard to predict. With Aurora Serverless, your database will auto-scale capacity to meet the needs of the application’s peak load and scale back down when the surge of activity is over. |
| Development and Test Databases | Software development and QA teams are using databases during work hours, but don’t need them on nights or weekends. With Aurora Serverless, your database automatically shuts down when not in use, and starts up much more quickly when work starts the next day. |
| Multitenant Applications | Web-based application with a database for each of your customers. Now you don’t have to manage database capacity individually for each application in your fleet. Aurora manages individual database capacity for you, saving you valuable time. |

Fault-Tolerant and Self-Healing Storage

Each 10GB chunk of your database volume is replicated 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.

Amazon Aurora storage is also self-healing; data blocks and disks are continuously scanned for errors and replaced automatically.

Aurora Auto Scaling

Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas provisioned for an Aurora DB cluster using single-master replication.

Aurora Auto Scaling is available for both Aurora MySQL and Aurora PostgreSQL.

Aurora Auto Scaling enables your Aurora DB cluster to handle sudden increases in connectivity or workload.

When the connectivity or workload decreases, Aurora Auto Scaling removes unnecessary Aurora Replicas so that you don’t pay for unused provisioned DB instances.

Backup and Restore

Amazon Aurora’s backup capability enables point-in-time recovery for your instance.

This allows you to restore your database to any second during your retention period, up to the last five minutes.

Your automatic backup retention period can be configured up to thirty-five days.

Automated backups are stored in [Amazon S3](https://aws.amazon.com/s3/), which is designed for 99.999999999% durability. Amazon Aurora backups are automatic, incremental, and continuous and have no impact on database performance.

When automated backups are turned on for your DB Instance, Amazon RDS automatically performs a full daily snapshot of your data (during your preferred backup window) and captures transaction logs (as updates to your DB Instance are made).

Automated backups are enabled by default and data is stored on S3 and is equal to the size of the DB.

Amazon RDS retains backups of a DB Instance for a limited, user-specified period called the retention period, which by default is 7 days but can be up to 35 days.

There are two methods to backup and restore RDS DB instances:

Amazon RDS automated backups.

User initiated manual backups.

Both options back up the entire DB instance and not just the individual DBs.

Both options create a storage volume snapshot of the entire DB instance.

You can make copies of automated backups and manual snapshots.

Automated backups backup data to multiple AZs to provide for data durability.

Multi-AZ backups are taken from the standby instance (for MariaDB, MySQL, Oracle, and PostgreSQL).

The DB instance must be in an Active state for automated backups to happen.

Only automated backups can be used for point-in-time DB instance recovery.

The granularity of point-in-time recovery is 5 minutes.

Amazon RDS creates a daily full storage volume snapshot and captures transaction logs regularly.

You can choose the backup window.

There is no additional charge for backups, but you will pay for storage costs on S3.

You can disable automated backups by setting the retention period to zero (0).

An outage occurs if you change the backup retention period from zero to a non-zero value or the other way around.

The retention period is the period AWS keeps the automated backups before deleting them.

Retention periods:

By default the retention period is 7 days if configured from the console for all DB engines except Aurora.

The default retention period is 1 day if configured from the API or CLI.

The retention period for Aurora is 1 day regardless of how it is configured.

You can increase the retention period up to 35 days.

During the backup window I/O may be suspended.

Automated backups are deleted when you delete the RDS DB instance.

Automated backups are only supported for InnoDB storage engine for MySQL (not for myISAM).

When you restore a DB instance the default DB parameters and security groups are applied – you must then apply the custom DB parameters and security groups.

You cannot restore from a DB snapshot into an existing DB instance.

Following a restore the new DB instance will have a new endpoint.

The storage type can be changed when restoring a snapshot.

Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

Amazon DynamoDB stores three geographically distributed replicas of each table to enable high availability and data durability.

Data is synchronously replicated across 3 facilities (AZs) in a region.

DynamoDB is schema-less.

It is a non-relational, key-value type of database.

DynamoDB is a serverless service – there are no instances to provision or manage.

Push button scaling means that you can scale the DB at any time without incurring downtime.

DynamoDB can be used for storing session state data.

Provides very low latency.

Data is stored on SSD storage.

Multi-AZ redundancy and Cross-Region Replication option.

The underlying hardware storing data is spread across 3 geographically distinct data centers.

DynamoDB is made up of:

Tables.

Items.

Attributes.

Tables are a collection of items and items are made up of attributes (columns).

Attributes consists of a name and a value or set of values.

Attributes in DynamoDB are like fields or columns in other database systems.

The aggregate size of an item cannot exceed 400KB including keys and all attributes.

Can store pointers to objects in S3, including items over 400KB.

Supports key value and document structures.

A key-value database stores data as a collection of key-value pairs in which a key serves as a unique identifier.

Key = the name of the data; Value = the data itself.

Documents can be written in JSON, HTML, or XML.

Some of the features and benefits of Amazon DynamoDB are summarized in the following table:

|  |  |
| --- | --- |
| DynamoDB Feature | Benefit |
| Serverless | Fully managed, fault tolerant service |
| Highly available | 99.99% Availability SLA – 99.999% for Global tables |
| NoSQL type of database with Name / Value structure | Flexible Schema, good for when data is not well structured or unpredictable |
| Horizontal scaling | Seamless scalability to any scale with push button scaling or Auto scaling |
| DynamoDB Streams | Captures a time-ordered sequence of item-level modifications in a DynamoDB table and durably stores the information for 24 hours. Often used with Lambda and the Kinesis Client Library (KCL). |
| DynamoDB Accelerator (DAX) | Fully managed in-memory cache for DynamoDB that increases performance (microsecond latency) |
| Transaction options | Strongly consistent or eventually consistent reads, support for ACID transactions |
| Backup | Point-in-time recovery down to the second in last 35 days; On-demand backup and restore |
| Global Tables | Fully managed multi-region, multi-master solution |

Anti-Patterns

Amazon DynamoDB is not ideal for the following situations:

Traditional RDS apps.

Joins and/or complex transactions.

BLOB data.

Large data with low I/O rate.

Access control

All authentication and access control is managed using IAM.

DynamoDB supports identity-based policies:

Attach a permissions policy to a user or a group in your account.

Attach a permissions policy to a role (grant cross-account permissions).

DynamoDB doesn’t support resource-based policies.

You can use a special IAM condition to restrict user access to only their own records.

In DynamoDB, the primary resources are tables.

DynamoDB also supports additional resource types, indexes, and streams.

You can create indexes and streams only in the context of an existing DynamoDB table (subresources).

These resources and subresources have unique Amazon Resource Names (ARNs) associated with them, as shown in the following table.

|  |  |
| --- | --- |
| Resource Type | ARN Format |
| Table | arn:aws:dynamodb:region:account-id:table:table/name-name |
| Index | arn:aws:dynamodb:region:account-id:table:table/name-name/index/index-name |
| Stream | arn:aws:dynamodb:region:account-id:table:table/name-name/stream/stream-label |

Partitions

Amazon DynamoDB stores data in partitions.

A partition is an allocation of storage for a table that is automatically replicated across multiple AZs within an AWS Region.

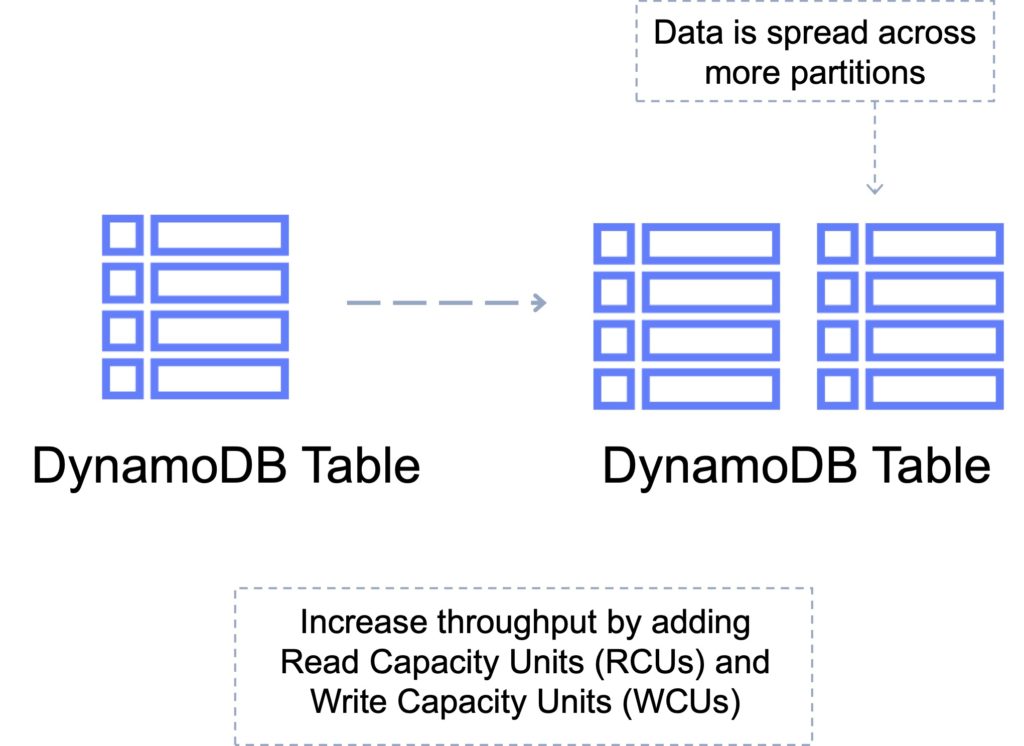
Partition management is handled entirely by DynamoDB—you never have to manage partitions yourself.

DynamoDB allocates sufficient partitions to your table so that it can handle your provisioned throughput requirements.

DynamoDB allocates additional partitions to a table in the following situations:

If you increase the table’s provisioned throughput settings beyond what the existing partitions can support.

If an existing partition fills to capacity and more storage space is required.



Primary Keys

DynamoDB stores and retrieves data based on a Primary key.

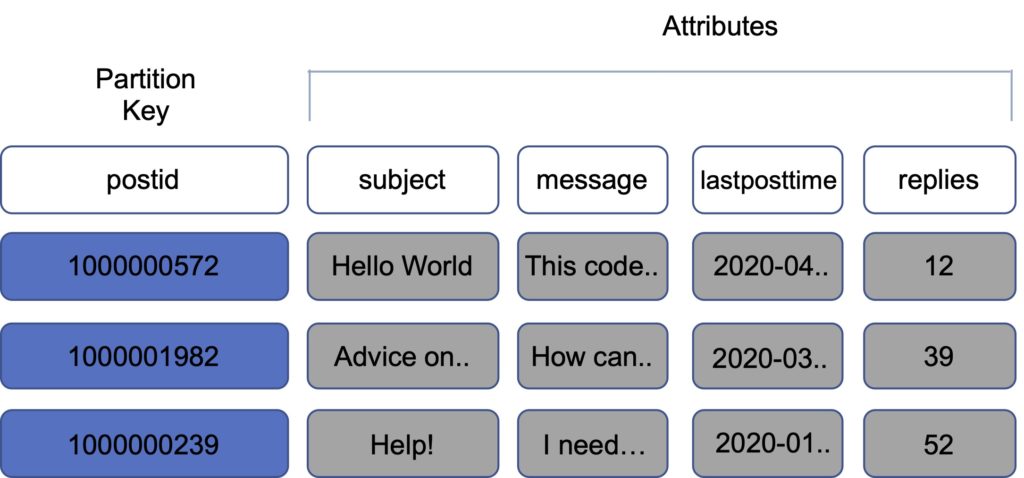
There are two types of Primary key:

Partition key – unique attribute (e.g. user ID).

Value of the Partition key is input to an internal hash function which determines the partition or physical location on which the data is stored.

If you are using the Partition key as your Primary key, then no two items can have the same partition key.

The image below depicts a table with a partition key:



Composite key – Partition key + Sort key in combination.

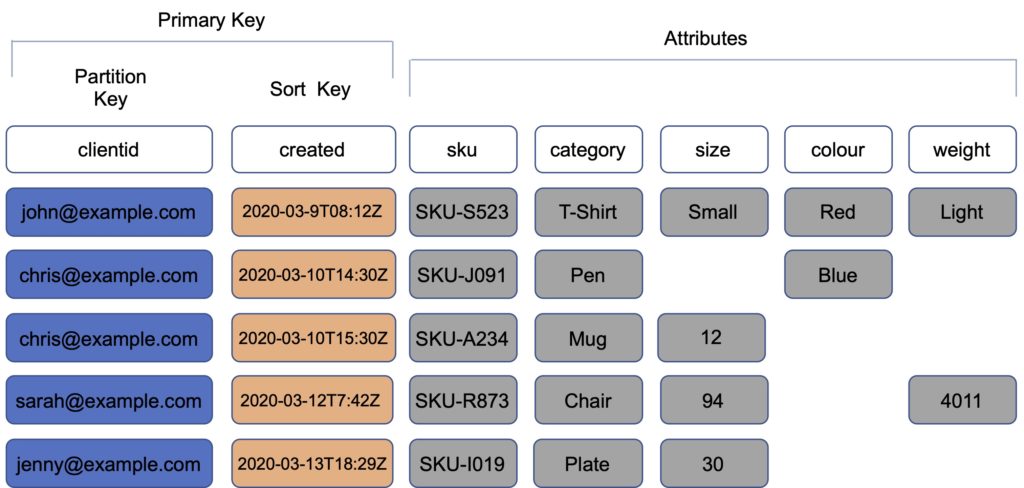
Example is user posting to a forum. Partition key would be the user ID, Sort key would be the timestamp of the post.

2 items may have the same Partition key, but they must have a different Sort key.

All items with the same Partition key are stored together, then sorted according to the Sort key value.

Allows you to store multiple items with the same partition key.

The image below depicts a table with a composite key:



Partitions and Performance

DynamoDB evenly distributes provisioned throughput—read capacity units (RCUs) and write capacity units (WCUs) among partitions

If your access pattern exceeds 3000 RCU or 1000 WCU for a single partition key value, your requests might be throttled.

Reading or writing above the limit can be caused by these issues:

Uneven distribution of data due to the wrong choice of partition key.

Frequent access of the same key in a partition (the most popular item, also known as a hot key).

A request rate greater than the provisioned throughput.

Best practices for partition keys:

Use high-cardinality attributes – e.g. e-mailid, employee\_no, customerid, sessionid, orderid, and so on.

Use composite attributes – e.g. customerid+productid+countrycode as the partition key and order\_date as the sort key.

Cache popular items – use DynamoDB accelerator (DAX) for caching reads.

Add random numbers or digits from a predetermined range for write-heavy use cases – e.g. add a random suffix to an invoice number such as INV00023-04593

Consistency Models

DynamoDB supports eventually consistent and strongly consistent reads.

Eventually consistent reads:

When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation.

The response might include some stale data.

If you repeat your read request after a short time, the response should return the latest data.

Strongly consistent reads:

When you request a strongly consistent read, DynamoDB returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful.

A strongly consistent read might not be available if there is a network delay or outage. In this case, DynamoDB may return a server error (HTTP 500).

Strongly consistent reads may have higher latency than eventually consistent reads.

Strongly consistent reads are not supported on global secondary indexes.

Strongly consistent reads use more throughput capacity than eventually consistent reads.

DynamoDB uses eventually consistent reads by default.

You can configure strongly consistent reads with the GetItem, Query and Scan APIs by setting the –consistent-read (or ConsistentRead) parameter to “true”.

DynamoDB Transactions

Amazon DynamoDB transactions simplify the developer experience of making coordinated, all-or-nothing changes to multiple items both within and across tables.

Transactions provide atomicity, consistency, isolation, and durability (ACID) in DynamoDB.

Enables reading and writing of multiple items across multiple tables as an all or nothing operation.

Checks for a pre-requisite condition before writing to a table.

With the transaction write API, you can group multiple Put, Update, Delete, and ConditionCheck actions.

You can then submit the actions as a single TransactWriteItems operation that either succeeds or fails as a unit.

The same is true for multiple Get actions, which you can group and submit as a single TransactGetItems operation.

There is no additional cost to enable transactions for DynamoDB tables.

You pay only for the reads or writes that are part of your transaction.

DynamoDB performs two underlying reads or writes of every item in the transaction: one to prepare the transaction and one to commit the transaction.

These two underlying read/write operations are visible in your Amazon CloudWatch metrics.

The following diagram depicts a failed write using DynamoDB Transactions:



The following diagram depicts a successful write using DynamoDB Transactions:



Scan and Query API calls

Scan

The Scan operation returns one or more items and item attributes by accessing every item in a table or a secondary index.

A single Scan operation reads up to the maximum number of items set (if using the Limit parameter) or a maximum of 1 MB.

Scan API calls can use a lot of RCUs as they access every item in the table.

You can use the ProjectionExpression parameter so that Scan only returns some of the attributes, rather than all of them.

If you need to further refine the Scan results, you can optionally provide a filter expression.

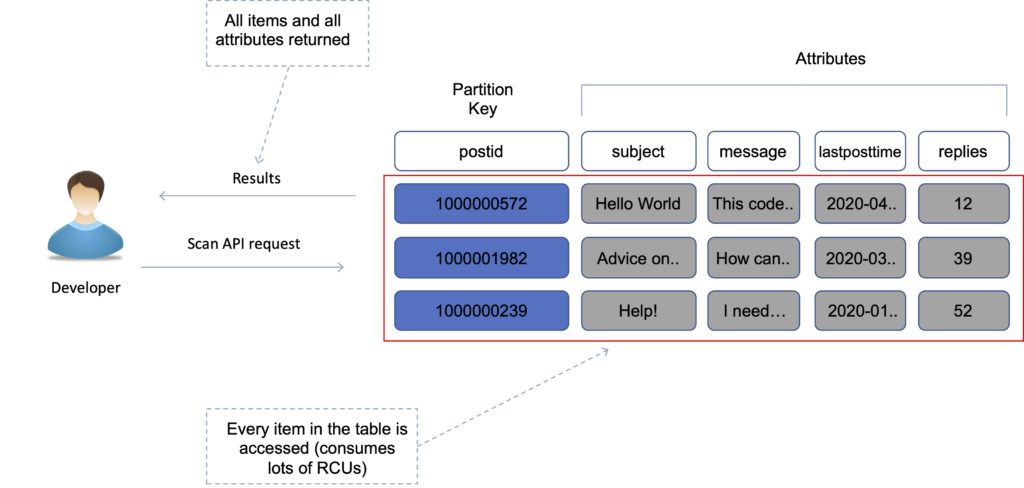
A filter expression is applied after a Scan finishes but before the results are returned.

Scan operations proceed sequentially.

For faster performance on a large table or secondary index, applications can request a parallel Scan operation by providing the Segment and TotalSegments parameters.

Scan uses eventually consistent reads when accessing the data in a table.

If you need a consistent copy of the data, as of the time that the Scan begins, you can set the ConsistentRead parameter to true.



Query

A query operation finds items in your table based on the primary key attribute and a distinct value to search for.

For example, you might search for a user ID value and all attributes related to that item would be returned.

You can use an optional sort key name and value to refine the results.

For example, if your sort key is a timestamp, you can refine the query to only select items with a timestamp of the last 7 days.

By default, a query returns all the attributes for the items, but you can use the ProjectionExpression parameter if you want the query to only return the attributes you want to see.

Results are always sorted by the sort key.

Numeric order is used – by default in ascending order (e.g. 1,2,3,4).

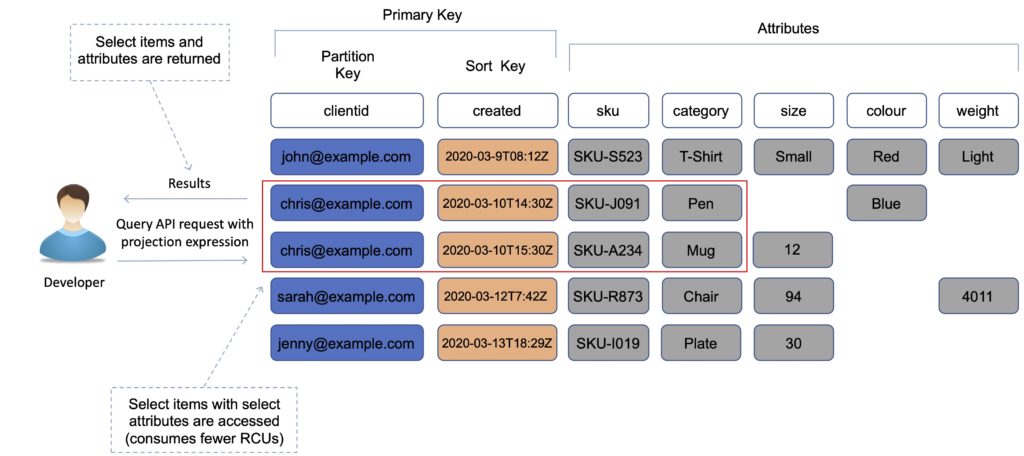
ASCII character code values are used.

You can reverse the order by setting the ScanIndexForward (yes, it’s a query, not a scan) parameter to false.

By default, queries are eventually consistent.

To use strongly consistent you need to explicitly set this in the query.

The following diagram shows a query API call with a projection expression limiting the attributes that are returned:



Scan vs Query

Query is more efficient than Scan.

Scan dumps the entire table, then filters out the values that provide the desired result (removing unwanted data).

This adds an extra step of removing the data you don’t want.

As the table grows, the scan operation takes longer.

A Scan operation on a large table can use up the provisioned throughput for a large table in just a single operation.

Performance optimization

You can reduce the impact of a query or scan by setting a smaller page size which uses fewer read operations.

A larger number of smaller operations will allow other requests to succeed without throttling.

Avoid using scan operations if you can: design tables in a way that you can use the Query, Get, or BatchGetItem APIs.

Scan performance optimization:

By default, a scan operation processes data sequentially and returns data in 1MB increments before moving on to retrieve the next 1MB of data. It can only scan 1 partition at a time.

You can configure DynamoDB to use Parallel scans instead by logically dividing a table or index into segments and scanning each segment in parallel.

Note: best to avoid parallel scans if your table or index is already incurring heavy read / write activity from other applications.

Indexes

An index is a data structure which allows you to perform fast queries on specific columns in a table.

You select columns that you want included in the index and run your searches on the index instead of the entire dataset.

There are 2 types of index supported for speeding up queries in DynamoDB:

Local Secondary Index.

Global Secondary Index.

Local Secondary Index (LSI)

An LSI provides an alternative sort key to use for scans and queries.

It provides an alternative range key for your table, local to the hash key.

You can have up to five LSIs per table.

The sort key consists of exactly one scalar attribute.

The attribute that you choose must be a scalar String, Number, or Binary.

An LSI must be created at table creation time.

It can only be created when you are creating your table.

You cannot add, remove, or modify it later.

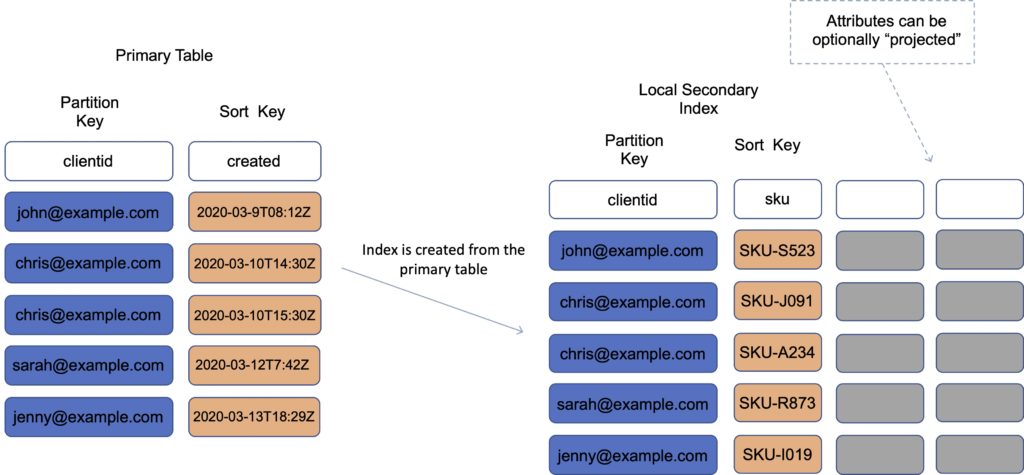
It has the same partition key as your original table (different sort key).

It gives you a different view of your data, organized by an alternative sort key.

Any queries based on this sort key are much faster using the index than the main table.

An example might be having a user ID as a partition key and account creation date as the sort key.

The key benefit of an LSI is that you can query on additional values in the table other than the partition key / sort key.



Global Secondary Index (GSI)

A GSI is used to speed up queries on non-key attributes use a GSI

You can create when you create your table or at any time later.

A GSI has a different partition key as well as a different sort key.

It gives a completely different view of the data.

It speeds up any queries relating to this alternative partition and sort key.

An example might be an email address as the partition key, and last login date as the sort key.

With a GSI the index is a new “table”, and you can project attributes on it.

The partition key and sort key of the original table are always projected (KEYS\_ONLY).

Can specify extra attributes to project (INCLUDE).

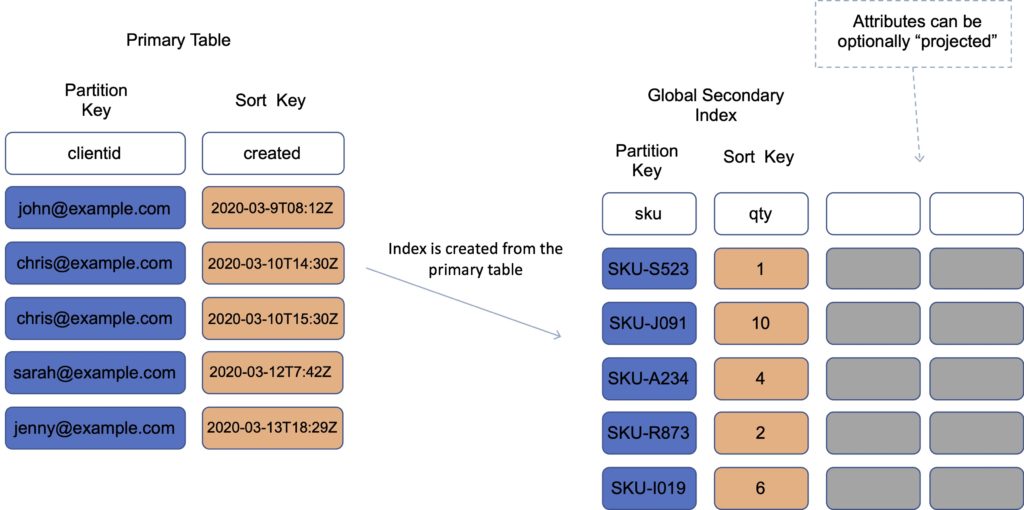
Can use all attributes from main table (ALL).

You must define RCU / WCU for the index

It is possible to add / modify GSI at any time.

If writes are throttled on the GSI, the main table will be throttled (even if there’s enough WCUs on the main table). LSIs do not cause any special throttling considerations.

Exam tip: You typically need to ensure that you have at least the same, or more, RCU/WCU specified in your GSI as in your main table to avoid throttling on your main table.



Performance and Optimization

DynamoDB Provisioned Capacity

With provisioned capacity mode you specify the number of data reads and writes per second that you require for your application.

You can use auto scaling to automatically adjust your table’s capacity based on the specified utilization rate to ensure application performance while reducing costs.

When you create your table you specify your requirements using Read Capacity Units (RCUs) and Write Capacity Units (WCUs).

Note: WCUs and RCUs are spread between partitions evenly.

You can also use Auto Scaling with provisioned capacity.

DynamoDB auto scaling uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to 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.

Read capacity unit (RCU):

Each API call to read data from your table is a read request.

Read requests can be strongly consistent, eventually consistent, or transactional.

For items up to 4 KB in size, one RCU can perform one strongly consistent read request per second.

Items larger than 4 KB require additional RCUs.

For items up to 4 KB in size, one RCU can perform two eventually consistent read requests per second.

Transactional read requests require two RCUs to perform one read per second for items up to 4 KB.

For example, a strongly consistent read of an 8 KB item would require two RCUs, an eventually consistent read of an 8 KB item would require one RCU, and a transactional read of an 8 KB item would require four RCUs.

Write capacity unit (WCU):

Each API call to write data to your table is a write request.

For items up to 1 KB in size, one WCU can perform one standard write request per second.

Items larger than 1 KB require additional WCUs.

Transactional write requests require two WCUs to perform one write per second for items up to 1 KB.

For example, a standard write request of a 1 KB item would require one WCU, a standard write request of a 3 KB item would require three WCUs, and a transactional write request of a 3 KB item would require six WCUs.

Replicated write capacity unit (rWCU):

When using DynamoDB global tables, your data is written automatically to multiple AWS Regions of your choice.

Each write occurs in the local Region as well as the replicated Regions.

Streams read request unit:

Each GetRecords API call to DynamoDB Streams is a streams read request unit.

Each streams read request unit can return up to 1 MB of data.

Transactional read/write requests:

In DynamoDB, a transactional read or write differs from a standard read or write because it guarantees that all operations contained in a single transaction set succeed or fail as a set.

DynamoDB On-Demand Capacity

With on-demand, you don’t need to specify your requirements.

DynamoDB instantly scales up and down based on the activity of your application.

Great for unpredictable / spikey workloads or new workloads that aren’t well understood.

You pay for what you use (pay per request).

You can switch between the provisioned capacity and on-demand pricing models once per day.

Performance and Throttling

Throttling occurs when the configured RCU or WCU are exceeded.

May receive the ProvisionedThroughputExceededException error.

This error indicates that your request rate is too high for the read / write capacity provisioned for the table.

The AWS SDKs for DynamoDB automatically retry requests that receive this exception.

Your request is eventually successful, unless your retry queue is too large to finish.

Possible causes of performance issues:

Hot keys – one partition key is being read too often.

Hot partitions – when data access is imbalanced, a “hot” partition can receive a higher volume of read and write traffic compared to other partitions.

Large items – large items consume more RCUs and WCUs.

Resolution:

Reduce the frequency of requests and use exponential backoff.

Try to design your application for uniform activity across all logical partition keys in the table and its secondary indexes.

Use burst capacity effectively – DynamoDB currently retains up to 5 minutes (300 seconds) of unused read and write capacity which can be consumed quickly.

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.

Improves performance from milliseconds to microseconds, even at millions of requests per second.

DAX is a managed service that provides in-memory acceleration for DynamoDB tables.

Provides managed cache invalidation, data population, and cluster management.

DAX is used to improve READ performance (not writes).

You do not need to modify application logic, since DAX is compatible with existing DynamoDB API calls.

Ideal for read-heavy and bursty workloads such as auction applications, gaming, and retail sites when running special sales / promotions.

You can enable DAX with just a few clicks in the AWS Management Console or using the AWS SDK.

Just as with DynamoDB, you only pay for the capacity you provision.

Provisioned through clusters and charged by the node (runs on EC2 instances).

Pricing is per node-hour consumed and is dependent on the instance type you select.

How it works:

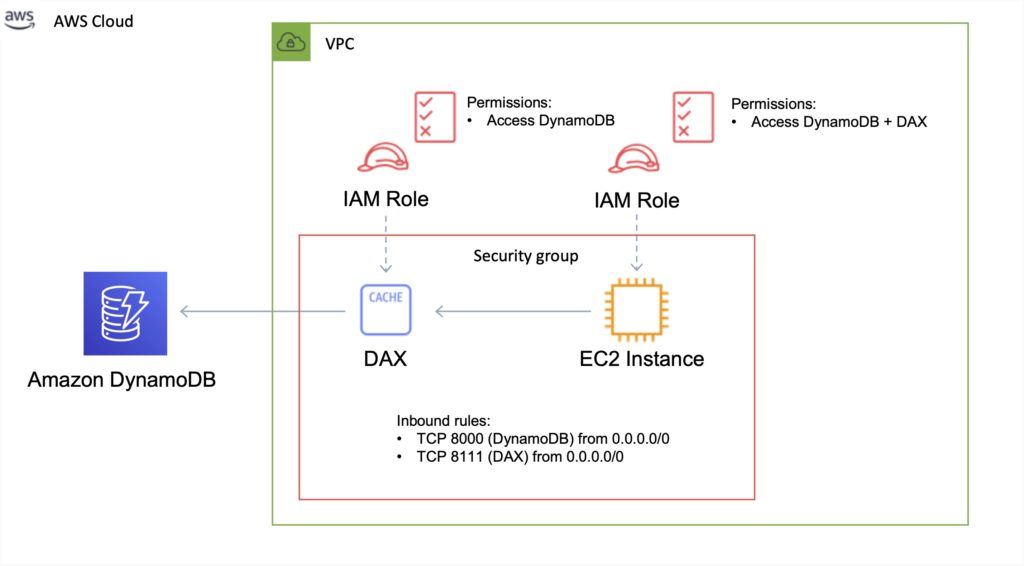
DAX is a write-through caching service – this means the data is written to the cache as well as the back-end store at the same time.

Allows you to point your DynamoDB API calls at the DAX cluster and if the item is in the cache (cache hit), DAX returns the result to the application.

If the item requested is not in the cache (cache miss) then DAX performs an Eventually Consistent GetItem operation against DynamoDB

Retrieval of data from DAX reduces the read load on DynamoDB tables.

This may result in being able to reduce the provisioned read capacity on the table.



DAX vs ElastiCache

DAX is optimized for DynamoDB.

DAX does not support lazy loading (uses write-through caching).

With ElastiCache you have more management overhead (e.g. invalidation).

With ElastiCache you need to modify application code to point to cache.

ElastiCache supports more datastores.

DynamoDB Time To Live (TTL)

Automatically deletes an item after an expiry date / time.

Expired items are marked for deletion.

Great for removing irrelevant or old data such as:

Session data.

Event logs.

Temporary data.

No extra cost and does not use WCU / RCU.

TTL is a background task operated by DynamoDB.

A TTL helps reduce storage and manage the table size over time.

The TTL is enabled per row (you define a TTL column and add the expiry date / time there).

DynamoDB typically deletes expired items within 48 hours of expiration.

Deleted items are also deleted from the LSI / GSI.

DynamoDB streams can help recover expired items.



Exponential backoff

Many components in a network can generate errors when overloaded.

In addition to simple retries all AWS SDKs use Exponential Backoff.

Progressively longer waits will occur between retries for improved flow control.

If after 1 minute this does not work, your request size may be exceeding the throughput for your read/write capacity.

If your workload is mainly reads consider offloading using DAX or ElastiCache.

If your workload is mainly writes consider increasing the WCUs for the table.

DynamoDB Streams

DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table and stores this information in a log for up to 24 hours.

Applications can access this log and view the data items as they appeared before and after they were modified, in near-real time.

You can also use the CreateTable or UpdateTable API operations to enable or modify a stream.

Logs are encrypted at rest and stored for 24 hours.

Accessed using a dedicated endpoint.

By default, just the Primary key is recorded.

Before and after images can be captured.

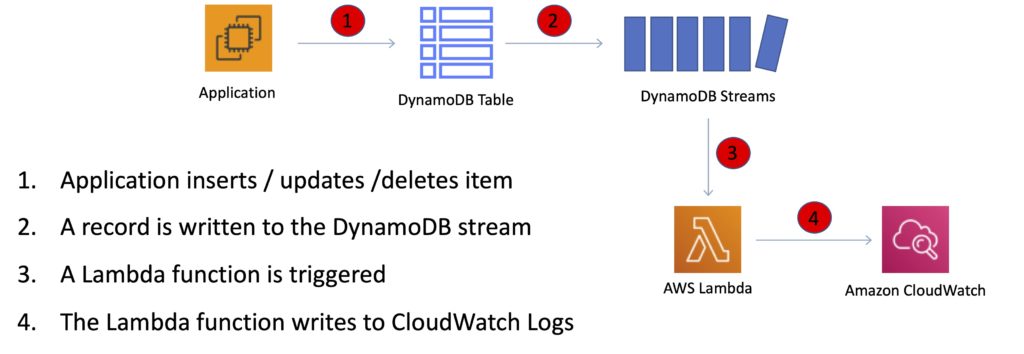
Events are recorded in near real-time.

Applications can take actions based on contents.

A stream can be an event source for Lambda.

Lambda polls the DynamoDB stream and executes code based on a DynamoDB streams event.

Data is stored in stream for 24 hours only.



The StreamSpecification parameter determines how the stream is configured:

StreamEnabled — Specifies whether a stream is enabled (true) or disabled (false) for the table.

StreamViewType — Specifies the information that will be written to the stream whenever data in the table is modified:

KEYS\_ONLY — Only the key attributes of the modified item.

NEW\_IMAGE — The entire item, as it appears after it was modified.

OLD\_IMAGE — The entire item, as it appeared before it was modified.

NEW\_AND\_OLD\_IMAGES — Both the new and the old images of the item.

API calls

Writing data

PutItem – create data or full replacement (consumes WCU).

UpdateItem – update data, partial update of attributes (can use atomic counters).

Conditional writes – accept a write / update only if conditions are met.

DeleteItem – delete an individual row (can perform conditional delete).

DeleteTable – delete a whole table (quicker than using DeleteItem on all items).

BatchWriteItem – can put or delete up to 25 items in one call (max 16MB write / 400KB per item).

Batching allows you to save in latency by reducing the number of API calls.

Operations are done in parallel for better efficiency.

Reading data

GetItem – read based on primary key (eventually consistent by default, can request strongly consistent read). Projection expression can be specified to include only certain attributes.

BatchGetItem – up to 100 items, up to 16MB per item. Items are retrieved in parallel to minimize latency.

Query – return items based on PartitionKey value and optionally a sort key. FilterExpression can be used for filtering. Returns up to 1MB of data or number of items specified in Limit. Can do pagination on results. Can query table, local secondary index, or a global secondary index.

Scan – scans the entire table (inefficient). Returns up to 1MB of data – use pagination to view more results. Consumes a lot of RCU. Can use a ProjectionExpression + FilterExpression.

Optimistic Locking

Optimistic locking is a strategy to ensure that the client-side item that you are updating (or deleting) is the same as the item in Amazon DynamoDB.

Protects database writes from being overwritten by the writes of others, and vice versa.

Conditional Updates

To manipulate data in an Amazon DynamoDB table, you use the PutItem, UpdateItem, and DeleteItem operations.

You can optionally specify a condition expression to determine which items should be modified.

If the condition expression evaluates to true, the operation succeeds; otherwise, the operation fails.

Security

VPC endpoints are available for DynamoDB.

Encryption at rest can be enabled using AWS KMS.

Encryption in transit uses SSL / TLS.

Best practices

Keep item sizes small.

If you are storing serial data in DynamoDB that will require actions based on date/time use separate tables for days, weeks, months.

Store more frequently and less frequently accessed data in separate tables.

If possible compress larger attribute values.

Store objects larger than 400KB in S3 and use pointers (S3 Object ID) in DynamoDB.

Integrations

ElastiCache can be used in front of DynamoDB for performance of reads on infrequently changed data.

Triggers integrate with AWS Lambda to respond to triggers.

Integration with RedShift:

RedShift complements DynamoDB with advanced business intelligence.

When copying data from a DynamoDB table into RedShift you can perform complex data analysis queries including joins with other tables.

A copy operation from a DynamoDB table counts against the table’s read capacity.

After data is copied, SQL queries do not affect the data in DynamoDB.

DynamoDB is integrated with Apache Hive on EMR. Hive can allow you to:

Read and write data in DynamoDB tables allowing you to query DynamoDB data using a SQL-like language (HiveQL).

Copy data from a DynamoDB table to an S3 bucket and vice versa.

Copy data from a DynamoDB table into HDFS and vice versa.

Perform join operations on DynamoDB tables.

Scalability

Push button scaling without downtime.

You can scale down only 4 times per calendar day.

AWS places default limits on the throughput you can provision.

DynamoDB can throttle requests that exceed the provisioned throughput for a table.

DynamoDB can also throttle read requests for an Index to prevent your application from consuming too many capacity units.

When a request is throttled it fails with an HTTP 400 code (Bad Request) and a ProvisionedThroughputExceeded exception.

Cross Region Replication with Global Tables

Amazon DynamoDB global tables provide a fully managed solution for deploying a multi-region, multi-master database.

When you create a global table, you specify the AWS regions where you want the table to be available.

DynamoDB performs all the necessary tasks to create identical tables in these regions and propagate ongoing data changes to all of them.

DynamoDB global tables are ideal for massively scaled applications, with globally dispersed users.

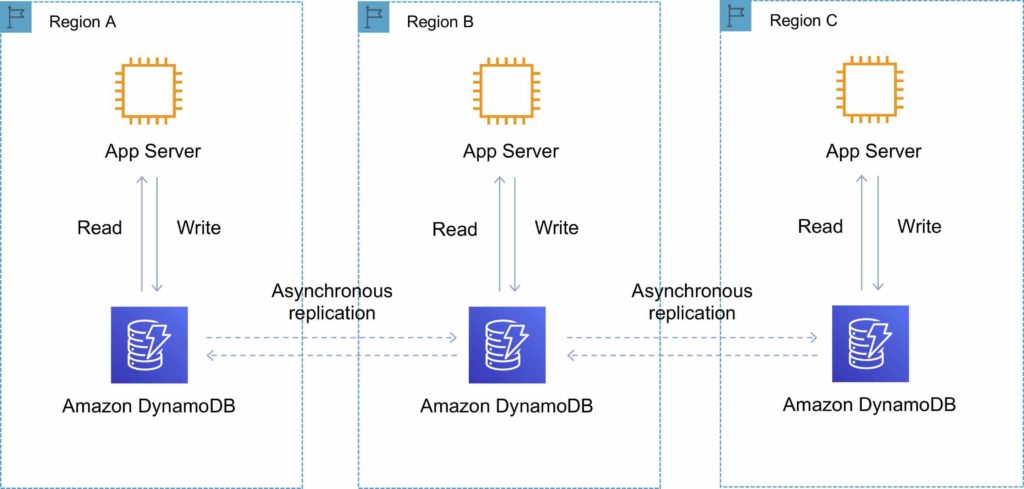
Global tables provide automatic multi-master replication to AWS regions world-wide, so you can deliver low-latency data access to your users no matter where they are located.

Definitions:

A global table is a collection of one or more replica tables, all owned by a single AWS account.

A replica table (or replica, for short) is a single DynamoDB table that functions as a part of a global table. Each replica stores the same set of data items. Any given global table can only have one replica table per region.

The following diagram depicts the Amazon DynamoDB Global Tables topology:



You can add replica tables to the global table, so that it can be available in additional AWS regions.

With a global table, each replica table stores the same set of data items. DynamoDB does not support partial replication of only some of the items.

An application can read and write data to any replica table. If your application only uses eventually consistent reads, and only issues reads against one AWS region, then it will work without any modification.

However, if your application requires strongly consistent reads, then it must perform all its strongly consistent reads and writes in the same region. DynamoDB does not support strongly consistent reads across AWS regions.

It is important that each replica table and secondary index in your global table has identical write capacity settings to ensure proper replication of data.

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.

How Application Auto Scaling works:

You create a scaling policy for a table or a global secondary index.

The scaling policy specifies whether you want to scale read capacity or write capacity (or both), and the minimum and maximum provisioned capacity unit settings for the table or index.

The scaling policy also contains a target utilization—the percentage of consumed provisioned throughput at a point in time.

Uses a target tracking algorithm to adjust the provisioned throughput of the table (or index) upward or downward in response to actual workloads, so that the actual capacity utilization remains at or near your target utilization.

Currently, Auto Scaling does not scale down your provisioned capacity if your table’s consumed capacity becomes zero.

If you use the AWS Management Console to create a table or a global secondary index, DynamoDB auto scaling is enabled by default.

High Availability Approaches for Databases

If possible, choose DynamoDB over RDS because of inherent fault tolerance.

If DynamoDB can’t be used, choose Aurora because of redundancy and automatic recovery features.

If Aurora can’t be used, choose Multi-AZ RDS.

Frequent RDS snapshots can protect against data corruption or failure, and they won’t impact performance of Multi-AZ deployment.

Regional replication is also an option but will not be strongly consistent.

If the database runs on EC2, you must design the HA yourself.

Amazon ElastiCache

Fully managed implementations of two popular in-memory data stores – Redis and Memcached.

ElastiCache is a web service that makes it easy to deploy and run Memcached or Redis protocol-compliant server nodes in the cloud.

The in-memory caching provided by ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads or compute-intensive workloads.

Best for scenarios where the DB load is based on Online Analytics Processing (OLAP) transactions.

Push-button scalability for memory, writes and reads.

In-memory key/value store – not persistent in the traditional sense.

Billed by node size and hours of use.

ElastiCache EC2 nodes cannot be accessed from the Internet, nor can they be accessed by EC2 instances in other VPCs.

Cached information may include the results of I/O-intensive database queries or the results of computationally intensive calculations.

Can be on-demand or reserved instances too (but not Spot instances).

ElastiCache can be used for storing session state.

A node is a fixed-sized chunk of secure, network-attached RAM and is the smallest building block.

Each node runs an instance of the Memcached or Redis protocol-compliant service and has its own DNS name and port.

Failed nodes are automatically replaced.

Access to ElastiCache nodes is controlled by VPC security groups and subnet groups (when deployed in a VPC).

Subnet groups are a collection of subnets designated for your Amazon ElastiCache Cluster.

You cannot move an existing Amazon ElastiCache Cluster from outside VPC into a VPC.

You need to configure subnet groups for ElastiCache for the VPC that hosts the EC2 instances and the ElastiCache cluster.

When not using a VPC, Amazon ElastiCache allows you to control access to your clusters through Cache Security Groups (you need to link the corresponding EC2 Security Groups).

ElastiCache nodes are deployed in clusters and can span more than one subnet of the same subnet group.

A cluster is a collection of one or more nodes using the same caching engine.

Applications connect to ElastiCache clusters using endpoints.

An endpoint is a node or cluster’s unique address.

Maintenance windows can be defined and allow software patching to occur.

There are two types of ElastiCache engine:

Memcached – simplest model, can run large nodes with multiple cores/threads, can be scaled in and out, can cache objects such as DBs.

Redis – complex model, supports encryption, master / slave replication, cross AZ (HA), automatic failover and backup/restore.

Use Cases

The following table describes a few typical use cases for ElastiCache:

|  |  |
| --- | --- |
| Use Case | Benefit |
| Web session store | In cases with load-balanced web servers, store web session information in Redis so if a server is lost, the session info is not lost, and another web server can pick it up |
| Database caching | Use Memcached in front of AWS RDS to cache popular queries to offload work from RDS and return results faster to users |
| Leaderboards | Use Redis to provide a live leaderboard for millions of users of your mobile app |
| Streaming data dashboards | Provide a landing spot for streaming sensor data on the factory floor, providing live real-time dashboard displays |

Exam tip: the key use cases for ElastiCache are offloading reads from a database and storing the results of computations and session state. Also, remember that ElastiCache is an in-memory database and it’s a managed service (so you can’t run it on EC2).

The table below describes the requirements that would determine whether to use the Memcached or Redis engine:

|  |  |
| --- | --- |
| Memcached | Redis |
| Simple, no-frills | You need encryption |
| You need to elasticity (scale out and in) | You need HIPAA compliance |
| You need to run multiple CPU cores and threads | Support for clustering |
| You need to cache objects (e.g. database queries) | You need complex data types |
|  | You need HA (replication |
|  | Pub/Sub capability |

Memcached

Simplest model and can run large nodes.

It can be scaled in and out and cache objects such as DBs.

Widely adopted memory object caching system.

Multi-threaded.

Scales out and in, by adding and removing nodes.

Ideal front-end for data stores (RDS, Dynamo DB etc.).

Use cases:

Cache the contents of a DB.

Cache data from dynamically generated web pages.

Transient session data.

High frequency counters for admission control in high volume web apps.

Max 100 nodes per region, 1-20 nodes per cluster (soft limits).

Can integrate with SNS for node failure/recovery notification.

Supports auto-discovery for nodes added/removed from the cluster.

Scales out/in (horizontally) by adding/removing nodes.

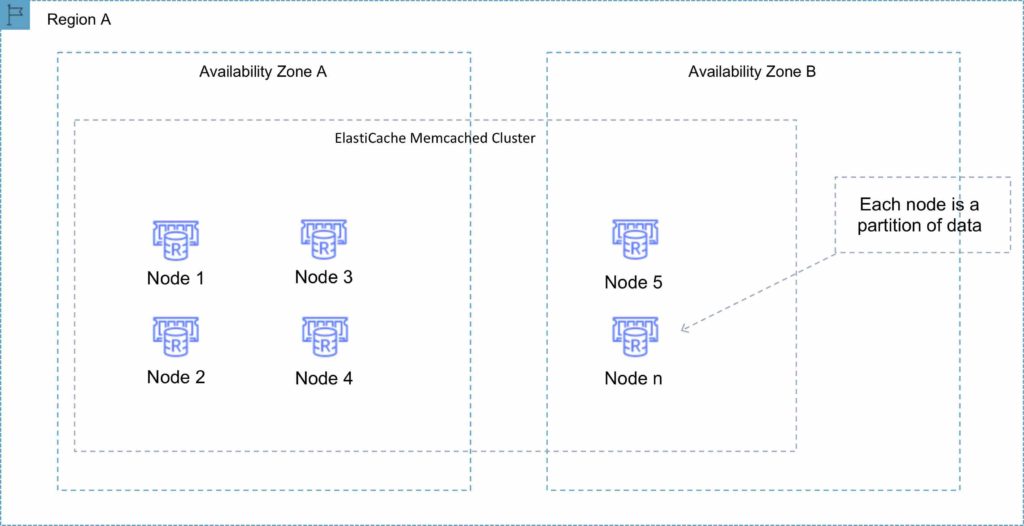
Scales up/down (vertically) by changing the node family/type.

Does not support multi-AZ failover or replication.

Does not support snapshots.

You can place nodes in different AZs.

With ElastiCache Memcached each node represents a partition of data and nodes in a cluster can span availability zones:



Redis

Open-source in-memory key-value store.

Supports more complex data structures: sorted sets and lists.

Data is persistent and it can be used as a datastore.

Redis is not multi-threaded.

Scales by adding shards, not nodes.

Supports master / slave replication and multi-AZ for cross-AZ redundancy.

Supports automatic failover and backup/restore.

A Redis shard is a subset of the cluster’s keyspace, that can include a primary node and zero or more read replicas.

Supports automatic and manual snapshots (S3).

Backups include cluster data and metadata.

You can restore your data by creating a new Redis cluster and populating it from a backup.

During backup you cannot perform CLI or API operations on the cluster.

Automated backups are enabled by default (automatically deleted with Redis deletion).

You can only move snapshots between regions by exporting them from ElastiCache before moving between regions (can then populate a new cluster with data).

Clustering mode disabled

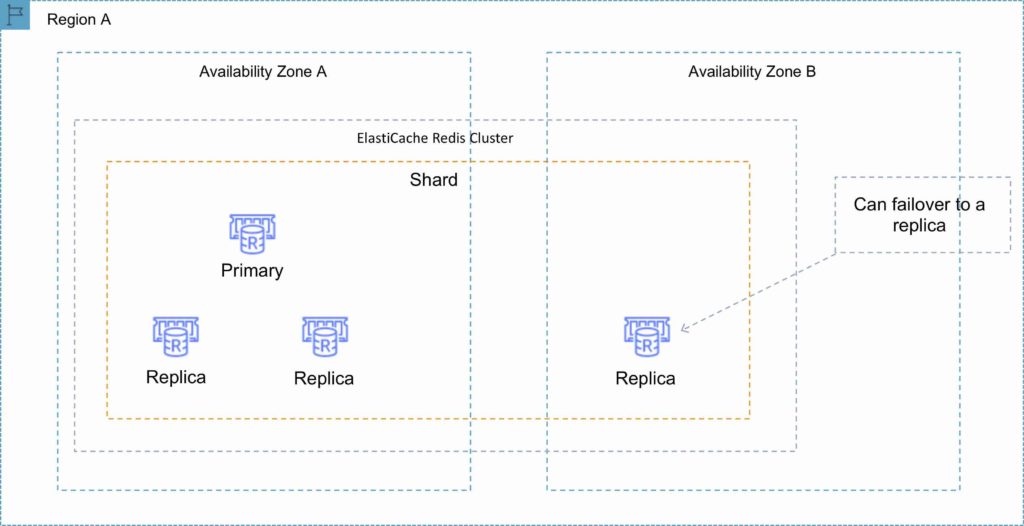
You can only have one shard.

One shard can have one read/write primary node and 0-5 read only replicas.

You can distribute the replicas over multiple AZs in the same region.

Replication from the primary node is asynchronous.

A Redis cluster with cluster mode disabled is represented in the diagram below:



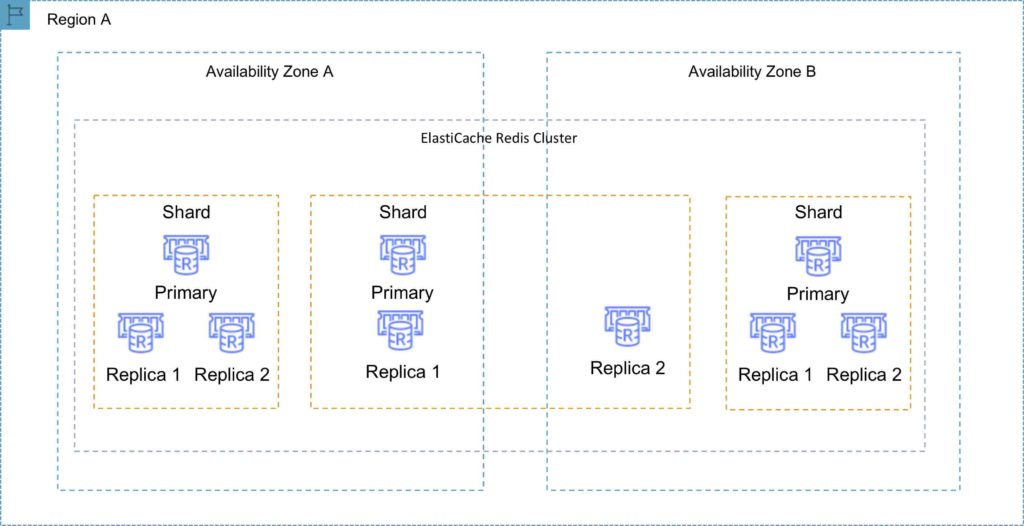
Clustering mode enabled

Can have up to 15 shards.

Each shard can have one primary node and 0-5 read only replicas.

Taking snapshots can slow down nodes, best to take from the read replicas.

A Redis cluster with cluster mode enabled is represented in the diagram below:



Multi-AZ failover

Failures are detected by ElastiCache.

ElastiCache automatically promotes the replica that has the lowest replica lag.

DNS records remain the same but point to the IP of the new primary.

Other replicas start to sync with the new primary.

You can have a fully automated, fault tolerant ElastiCache-Redis implementation by enabling both cluster mode and multi-AZ failover.

The following table compares the Memcached and Redis engines:

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Memcached | Redis (cluster mode disabled) | Redis (cluster mode enabled) |
| Data persistence | No | Yes | Yes |
| Data types | Simple | Complex | Complex |
| Data partitioning | Yes | No | Yes |
| Encryption | No | Yes | Yes |
| High availability (replication) | No | Yes | Yes |
| Multi-AZ | Yes, place nodes in multiple AZs. No failover or replication | Yes, with auto-failover. Uses read replicas (0-5 per shard) | Yes, with auto-failover. Uses read replicas (0-5 per shard) |
| Scaling | Up (node type); out (add nodes) | Single shard (can add replicas) | Add shards |
| Multithreaded | Yes | No | No |
| Backup and restore | No (and no snapshots) | Yes, automatic and manual snapshots | Yes, automatic and manual snapshots |

Caching strategies

There are two caching strategies available: Lazy Loading and Write-Through:

Lazy Loading

Loads the data into the cache only when necessary (if a cache miss occurs).

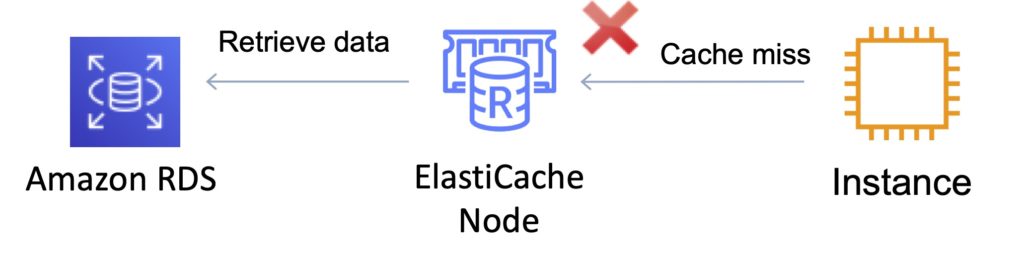
Lazy loading avoids filling up the cache with data that won’t be requested.

If requested data is in the cache, ElastiCache returns the data to the application.

If the data is not in the cache or has expired, ElastiCache returns a null.

The application then fetches the data from the database and writes the data received into the cache so that it is available for next time.

Data in the cache can become stale if Lazy Loading is implemented without other strategies (such as TTL).



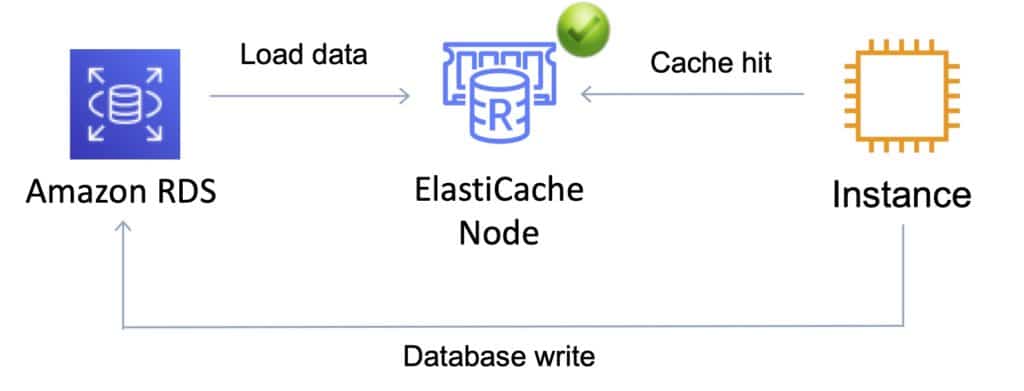
Write Through

When using a write-through strategy, the cache is updated whenever a new write or update is made to the underlying database.

Allows cache data to remain up to date.

This can add wait time to write operations in your application.

Without a TTL you can end up with a lot of cached data that is never read.



Dealing with stale data – Time to Live (TTL)

The drawbacks of lazy loading and write through techniques can be mitigated by a TTL.

The TTL specifies the number of seconds until the key (data) expires to avoid keeping stale data in the cache.

When reading an expired key, the application checks the value in the underlying database.

Lazy Loading treats an expired key as a cache miss and causes the application to retrieve the data from the database and subsequently write the data into the cache with a new TTL.

Depending on the frequency with which data changes this strategy may not eliminate stale data – but helps to avoid it.

Exam tip: Compared to DynamoDB Accelerator (DAX) remember that DAX is optimized for DymamoDB specifically and only supports the write-through caching strategy (does not use lazy loading).

Monitoring and Reporting

Memcached Metrics

The following [CloudWatch](https://digitalcloud.training/amazon-cloudwatch/) metrics offer good insight into ElastiCache Memcached performance:

CPUUtilization – This is a host-level metric reported as a percent. Because Memcached is multi-threaded, this metric can be as high as 90%. If you exceed this threshold, scale your cache cluster up by using a larger cache node type, or scale out by adding more cache nodes.

SwapUsage – This is a host-level metric reported in bytes. This metric should not exceed 50 MB. If it does, we recommend that you increase the ConnectionOverhead parameter value.

Evictions – This is a cache engine metric. If you exceed your chosen threshold, scale your cluster up by using a larger node type, or scale out by adding more nodes.

CurrConnections – This is a cache engine metric. An increasing number of CurrConnections might indicate a problem with your application; you will need to investigate the application behavior to address this issue.

Redis Metrics

The following CloudWatch metrics offer good insight into ElastiCache Redis performance:

EngineCPUUtilization – Provides CPU utilization of the Redis engine thread. Since Redis is single threaded, you can use this metric to analyze the load of the Redis process itself.

MemoryFragmentationRatio – Indicates the efficiency in the allocation of memory of the Redis engine. Certain threshold will signify different behaviors. The recommended value is to have fragmentation above 1.0.

CacheHits – The number of successful read-only key lookups in the main dictionary.

CacheMisses – The number of unsuccessful read-only key lookups in the main dictionary.

CacheHitRate – Indicates the usage efficiency of the Redis instance. If the cache ratio is lower than ~0.8, it means that a significant number of keys are evicted, expired, or do not exist.

CurrConnections – The number of client connections, excluding connections from read replicas. ElastiCache uses two to four of the connections to monitor the cluster in each case.

Logging and Auditing

All Amazon ElastiCache actions are logged by AWS CloudTrail.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

Whether the request was made with root or IAM user credentials.

Whether the request was made with temporary security credentials for a role or federated user.

Whether the request was made by another AWS service.

Authorization and Access Control

Access to Amazon ElastiCache requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an ElastiCache cache cluster or an Amazon Elastic Compute Cloud (Amazon EC2) instance.

You can use [identity-based policies](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/IAM.IdentityBasedPolicies.html) with Amazon ElastiCache to provide the necessary access.

You can use [Redis Auth](https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/auth.html) to require a token with ElastiCache Redis.

The Redis authentication tokens enable Redis to require a token (password) before allowing clients to run commands, thereby improving data security.

Charges

Pricing is per Node-hour consumed for each Node Type.

Partial Node-hours consumed are billed as full hours.

There is no charge for data transfer between Amazon EC2 and Amazon ElastiCache within the same Availability Zone.

High Availability for ElastiCache

Memcached:

Because Memcached does not support replication, a node failure will result in data loss.

Use multiple nodes to minimize data loss on node failure.

Launch multiple nodes across available AZs to minimize data loss on AZ failure.

Redis:

Use multiple nodes in each shard and distribute the nodes across multiple AZs.

Enable Multi-AZ on the replication group to permit automatic failover if the primary nodes fail.

Schedule regular backups of your Redis cluster.

Amazon RedShift

Amazon RedShift is a fast, fully managed data warehouse that makes it simple and cost-effective to analyze all your data using standard SQL and existing Business Intelligence (BI) tools. Amazon RedShift is a clustered peta-byte scale data warehouse and is an SQL based data warehouse used for analytics applications.

Amazon RedShift is an Online Analytics Processing (OLAP) type of Database which can be used for running complex analytic queries against petabytes of structured data, using sophisticated query optimization, columnar storage on high-performance local disks, and massively parallel query execution. Amazon RedShift is also ideal for processing large amounts of data for business intelligence.

Advantages of Amazon RedShift

The benefits of Amazon RedShift are as follows:

Amazon RedShift is extremely cost-effective as compared to some other on-premises data warehouse platforms.

Amazon RedShift is PostgreSQL compatible with JDBC and ODBC drivers available; compatible with most Business Intelligence tools out of the box.

Features parallel processing and columnar data stores which are optimized for complex queries.

Option to query directly from data files on S3 via Amazon RedShift Spectrum.

Amazon RedShift is 10x faster than a traditional SQL DB.

Amazon RedShift can store huge amounts of data but cannot ingest huge amounts of data in real time.

Amazon RedShift uses columnar data storage:

Data is stored sequentially in columns instead of rows.

Columnar based DB is ideal for data warehousing and analytics.

Requires fewer I/Os which greatly enhances performance.

Amazon RedShift provides advanced compression:

Data is stored sequentially in columns which allows for much better performance and less storage space.

Amazon RedShift automatically selects the compression scheme.

Amazon RedShift provides good query performance and compression.

Amazon RedShift provides Massively Parallel Processing (MPP) by distributing data and queries across all nodes.

Availability and Durability

Amazon RedShift uses replication and continuous backups to enhance availability and improve durability and can automatically recover from component and node failures.

Only available in one AZ but you can restore snapshots into another AZ.

Alternatively, you can run data warehouse clusters in multiple AZ’s by loading data into two Amazon RedShift data warehouse clusters in separate AZs from the same set of Amazon S3 input files.

Amazon RedShift replicates your data within your data warehouse cluster and continuously backs up your data to Amazon S3.

Amazon RedShift always keeps three copies of your data:

The original.

A replica of compute nodes (within the cluster).

A backup copy on S3.

Amazon RedShift provides continuous/incremental backups:

Multiple copies within a cluster.

Continuous and incremental backups to S3.

Continuous and incremental backups across regions.

Streaming restore.

Amazon RedShift provides fault tolerance for the following failures:

Disk failures.

Node failures.

Network failures.

AZ/region level disasters.

For node failures the data warehouse cluster will be unavailable for queries and updates until a replacement node is provisioned and added to the DB.

High availability for Amazon RedShift:

Currently, Amazon RedShift does not support Multi-AZ deployments.

The best HA option is to use a multi-node cluster which supports data replication and node recovery.

A single node Amazon RedShift cluster does not support data replication and you’ll have to restore from a snapshot on S3 if a drive fails.

Amazon RedShift can asynchronously replicate your snapshots to S3 in another region for DR.

Single-node clusters do not support data replication (in a failure scenario you would need to restore from a snapshot).

Scaling requires a period of unavailability of a few minutes (typically during the maintenance window).

During scaling operations Amazon RedShift moves data in parallel from the compute nodes in your existing data warehouse cluster to the compute nodes in your new cluster.

By default, Amazon RedShift retains backups for 1 day. You can configure this to be up to 35 days.

If you delete the cluster, you can choose to have a final snapshot taken and retained.

Manual backups are not automatically deleted when you delete a cluster.

Security

You can load encrypted data from S3.

Supports SSL Encryption in-transit between client applications and Amazon RedShift data warehouse cluster.

VPC for network isolation.

Encryption for data at rest (AES 256).

Audit logging and AWS CloudTrail integration.

Amazon RedShift takes care of key management, or you can manage your own through HSM or KMS.

Charges

Charged for compute nodes hours, 1 unit per hour (only compute node, not leader node).

Backup storage – storage on S3.

Data transfer – no charge for data transfer between Amazon RedShift and S3 within a region but for other scenarios you may pay charges.

HDD and SSD storage options.

The size of a single node is 160GB and clusters can be created up to a petabyte or more.

Multi-node consists of:

Leader node:

Manages client connections and receives queries.

Simple SQL endpoint.

Stores metadata.

Optimizes query plan.

Coordinates query execution.

Compute nodes:

Stores data and performs queries and computations.

Local columnar storage.

Parallel/distributed execution of all queries, loads, backups, restores, resizes.

Up to 128 compute nodes.

Amazon RedShift Spectrum is a feature of Amazon RedShift that enables you to run queries against exabytes of unstructured data in Amazon S3, with no loading or ETL required.

Use Cases of Amazon RedShift

A data warehouse for enterprise operations: Many organizations work with data from multiple sources, such as advertising, customer relationship management, and customer support.

As a centralized repository, Redshift can be used to store data from multiple sources in a unified schema and structure. This can then feed enterprise-wide reporting and analytics.

In business intelligence and analytics, Redshift’s fast query execution against terabyte-scale data makes it an excellent selection. BI tools such as Tableau often use Redshift as the underlying database (which would otherwise struggle to perform queries and joins of large datasets).

Organizations may choose to monetize their data by exposing it to their customers through embedded analytics and analytics as a service. In these scenarios, Redshift’s data sharing, search, and aggregation capabilities make it ideal, as it allows customers to access only relevant subsets of data while keeping other databases, tables, or rows confidential.

As long as the cluster is adequately resourced, Redshift’s performance is consistent and predictable. It is therefore a popular choice for data-driven applications, such as reporting and calculations.

Database migration and change data capture: AWS Database Migration Service (DMS) can be used to replicate changes in an operational data store into Amazon Redshift. It is typically done to provide more flexibility in analysis, or when migrating from legacy data warehouses.

Amazon EMR

Amazon EMR is a web service that enables businesses, researchers, data analysts, and developers to process vast amounts of data.

EMR utilizes a hosted Hadoop framework running on Amazon EC2 and Amazon S3.

With EMR you can run petabyte-scale analysis at less than half of the cost of traditional on-premises solutions and over 3x faster than standard Apache Spark.

You can run workloads on Amazon EC2 instances, on Amazon Elastic Kubernetes Service (EKS) clusters, or on-premises using EMR on AWS Outposts.

Runs in one Availability Zone within an Amazon VPC.

Supports Apache Spark, HBase, Presto and Flink.

Most used for log analysis, financial analysis, or extract, translate and loading (ETL) activities.

A Step is a programmatic task for performing some process on the data (e.g. count words).

A cluster is a collection of EC2 instances provisioned by EMR to run your Steps.

EMR is a good place to deploy Apache Spark, an open-source distributed processing used for big data workloads which utilizes in-memory caching and optimized query execution.

You can also launch Presto clusters. Presto is an open source distributed SQL query engine designed for fast analytic queries against large datasets.

EMR launches all nodes for a given cluster in the same Amazon EC2 Availability Zone.

You can access Amazon EMR by using the AWS Management Console, Command Line Tools, SDKs, or the EMR API.

With EMR you have access to the underlying operating system (you can SSH in).

Amazon Kinesis

Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information.

Kinesis is a collection of services for processing streams of various data.

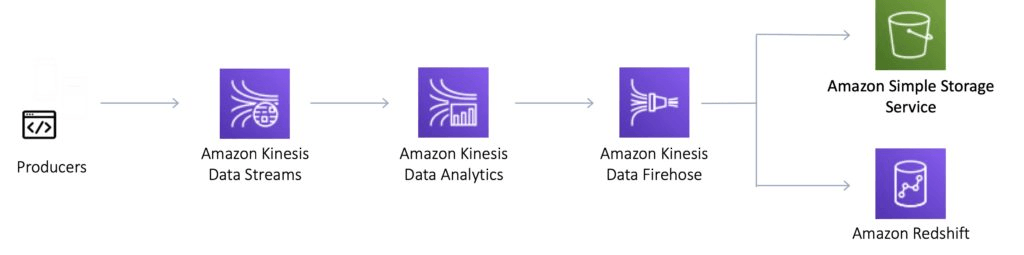
Data is processed in “shards” – with each shard able to ingest 1000 records per second.

There is a default limit of 500 shards, but you can request an increase to unlimited shards.

A record consists of a partition key, sequence number, and data blob (up to 1 MB).

Transient data store – default retention of 24 hours but can be configured for up to 7 days.

There are four types of Kinesis service, and these are detailed below.



Kinesis Video Streams

Kinesis Video Streams makes it easy to securely stream video from connected devices to AWS for analytics, machine learning (ML), and other processing.

Durably stores, encrypts, and indexes video data streams, and allows access to data through easy-to-use APIs.

Producers provide data streams.

Stores data for 24 hours by default, up to 7 days.

Stores data in shards – 5 transaction per second for reads, up to a max read rate of 2MB per second and 1000 records per second for writes up to a max of 1MB per second.

Consumers receive and process data.

Can have multiple shards in a stream.

Supports encryption at rest with server-side encryption (KMS) with a customer master key.

Kinesis Video Streams does not appear much on AWS exams.

Kinesis Data Streams

Kinesis Data Streams enables you to build custom applications that process or analyze streaming data for specialized needs.

Kinesis Data Streams enables real-time processing of streaming big data.

Kinesis Data Streams is useful for rapidly moving data off data producers and then continuously processing the data.

Kinesis Data Streams stores data for later processing by applications (key difference with Firehose which delivers data directly to AWS services).

Common use cases include:

Accelerated log and data feed intake.

Real-time metrics and reporting.

Real-time data analytics.

Complex stream processing.

The diagram below illustrates the high-level architecture of Kinesis Data Streams.

Producers continually push data to Kinesis Data Streams.

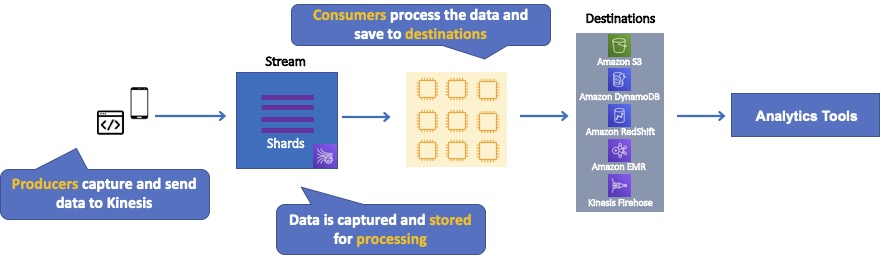
Consumers process the data in real time.

Consumers can store their results using an AWS service such as Amazon DynamoDB, Amazon Redshift, or Amazon S3.

Kinesis Streams applications are consumers that run on EC2 instances.

Shards are uniquely identified groups or data records in a stream.

Records are the data units stored in a Kinesis Stream.



A producer creates the data that makes up the stream.

Producers can be used through the following:

Kinesis Streams API.

Kinesis Producer Library (KPL).

Kinesis Agent.

A record is the unit of data stored in a Amazon Kinesis data stream.

A record is composed of a sequence number, partition key, and data blob.

By default, records of a stream are accessible for up to 24 hours from the time they are added to the stream (can be raised to 7 days by enabling extended data retention).

A data blob is the data of interest your data producer adds to a data stream.

The maximum size of a data blob (the data payload before Base64-encoding) within one record is 1 megabyte (MB).

A shard is the base throughput unit of an Amazon Kinesis data stream.

One shard provides a capacity of 1MB/sec data input and 2MB/sec data output.

Each shard can support up to 1000 PUT records per second.

A stream is composed of one or more shards.

The total capacity of the stream is the sum of the capacities of its shards.

Kinesis Data Streams supports resharding, which lets you adjust the number of shards in your stream to adapt to changes in the rate of data flow through the stream.

There are two types of resharding operations: shard split and shard merge.

In a shard split, you divide a single shard into two shards.

In a shard merge, you combine two shards into a single shard.



Splitting increases the number of shards in your stream and therefore increases the data capacity of the stream.

Splitting increases the cost of your stream (you pay per-shard).

Merging reduces the number of shards in your stream and therefore decreases the data capacity—and cost—of the stream.

Consumers are the EC2 instances that analyze the data received from a stream.

Consumers are known as Amazon Kinesis Streams Applications.

When the data rate increases, add more shards to increase the size of the stream.

Remove shards when the data rate decreases.

Partition keys are used to group data by shard within a stream.

Kinesis Streams uses KMS master keys for encryption.

To read from or write to an encrypted stream the producer and consumer applications must have permission to access the master key.

Kinesis Data Streams replicates synchronously across three AZs.

Kinesis Data Firehose

Kinesis Data Firehose is the easiest way to load streaming data into data stores and analytics tools.

Captures, transforms, and loads streaming data.

Enables near real-time analytics with existing business intelligence tools and dashboards.

Kinesis Data Streams can be used as the source(s) to Kinesis Data Firehose.

You can configure Kinesis Data Firehose to transform your data before delivering it.

With Kinesis Data Firehose you don’t need to write an application or manage resources.

Firehose can batch, compress, and encrypt data before loading it.

Firehose synchronously replicates data across three AZs as it is transported to destinations.

Each delivery stream stores data records for up to 24 hours.

A source is where your streaming data is continuously generated and captured.

A delivery stream is the underlying entity of Amazon Kinesis Data Firehose.

A record is the data of interest your data producer sends to a delivery stream.

The maximum size of a record (before Base64-encoding) is 1000 KB.

A destination is the data store where your data will be delivered.

Firehose Destinations include:

Amazon S3.

Amazon Redshift.

Amazon Elasticsearch Service.

Splunk.

Producers provide data streams.

No shards, totally automated.

Can encrypt data with an existing AWS Key Management Service (KMS) key.

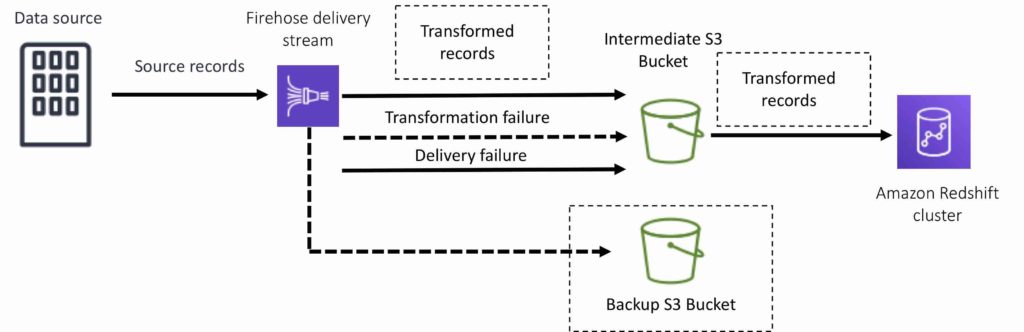
Server-side-encryption can be used if Kinesis Streams is used as the data source.

Firehose can invoke an AWS Lambda function to transform incoming data before delivering it to a destination.

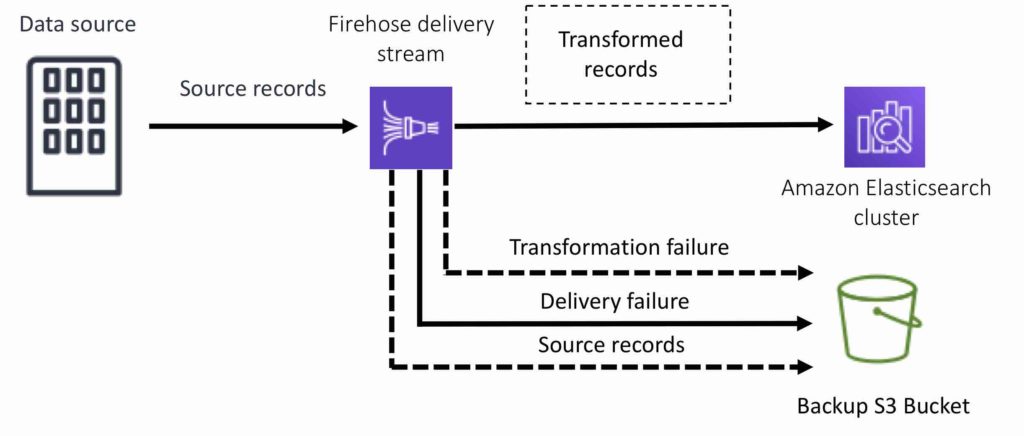
For Amazon S3 destinations, streaming data is delivered to your S3 bucket. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket:



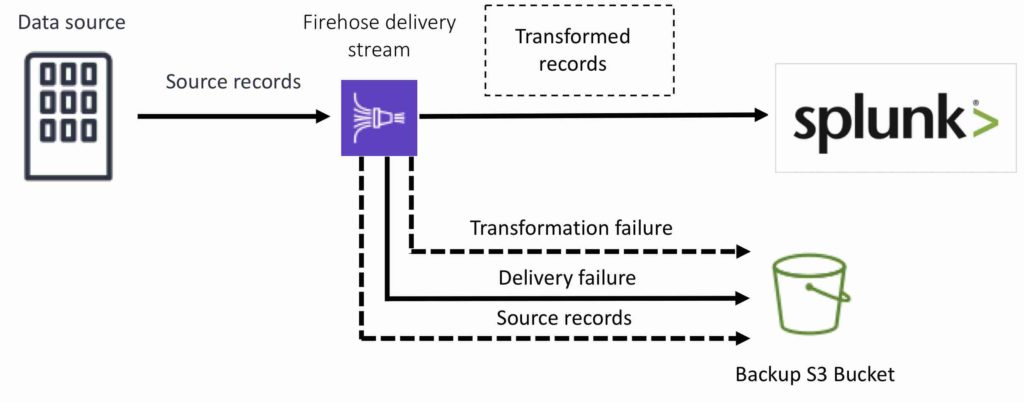
For Amazon Redshift destinations, streaming data is delivered to your S3 bucket first. Kinesis Data Firehose then issues an Amazon Redshift COPY command to load data from your S3 bucket to your Amazon Redshift cluster. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket:



For Amazon Elasticsearch destinations, streaming data is delivered to your Amazon ES cluster, and it can optionally be backed up to your S3 bucket concurrently:



For Splunk destinations, streaming data is delivered to Splunk, and it can optionally be backed up to your S3 bucket concurrently:



Kinesis Data Analytics

Amazon Kinesis Data Analytics is the easiest way to process and analyze real-time, streaming data.

Can use standard SQL queries to process Kinesis data streams.

Provides real-time analysis.

Use cases:

Generate time-series analytics.

Feed real-time dashboards.

Create real-time alerts and notifications.

Quickly author and run powerful SQL code against streaming sources.

Can ingest data from Kinesis Streams and Kinesis Firehose.

Output to S3, RedShift, Elasticsearch and Kinesis Data Streams.

Sits over Kinesis Data Streams and Kinesis Data Firehose.

A Kinesis Data Analytics application consists of three components:

Input – the streaming source for your application.

Application code – a series of SQL statements that process input and produce output.

Output – one or more in-application streams to hold intermediate results.

Kinesis Data Analytics supports two types of inputs: streaming data sources and reference data sources:

A streaming data source is continuously generated data that is read into your application for processing.

A reference data source is static data that your application uses to enrich data coming in from streaming sources.

Can configure destinations to persist the results.

Supports Kinesis Streams and Kinesis Firehose (S3, RedShift, Elasticsearch) as destinations.

IAM can be used to provide Kinesis Analytics with permissions to read records from sources and write to destinations.

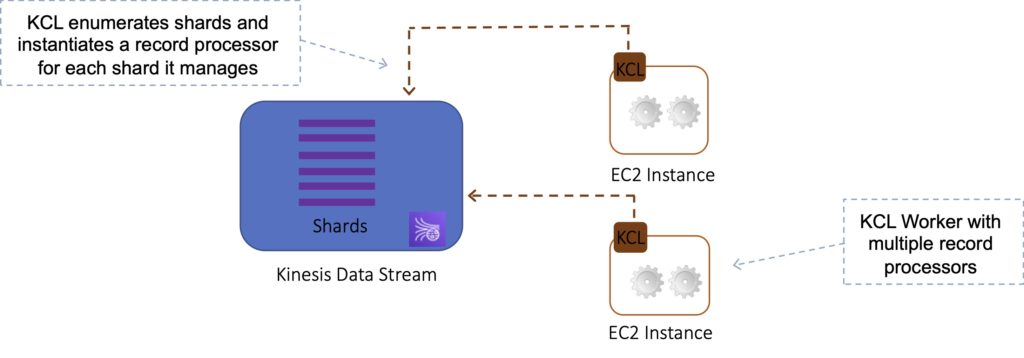
Kinesis client library

Kinesis Client Library is a Java library that helps read records from a Kinesis Stream with distributed applications sharing the read workload.

The KCL is different from the Kinesis Data Streams API that is available in the AWS SDKs.

The Kinesis Data Streams API helps you manage many aspects of Kinesis Data Streams (including creating streams, resharding, and putting and getting records).

The KCL provides a layer of abstraction specifically for processing data in a consumer role.



The KCL acts as an intermediary between your record processing logic and Kinesis Data Streams.

When you start a KCL application, it calls the KCL to instantiate a worker. The KCL performs the following tasks:

Connects to the stream.

Enumerates the shards.

Coordinates shard associations with other workers (if any).

Instantiates a record processor for every shard it manages.

Pulls data records from the stream.

Pushes the records to the corresponding record processor.

Checkpoints processed records.

Balances shard-worker associations when the worker instance count changes.

Balances shard-worker associations when shards are split or merged.

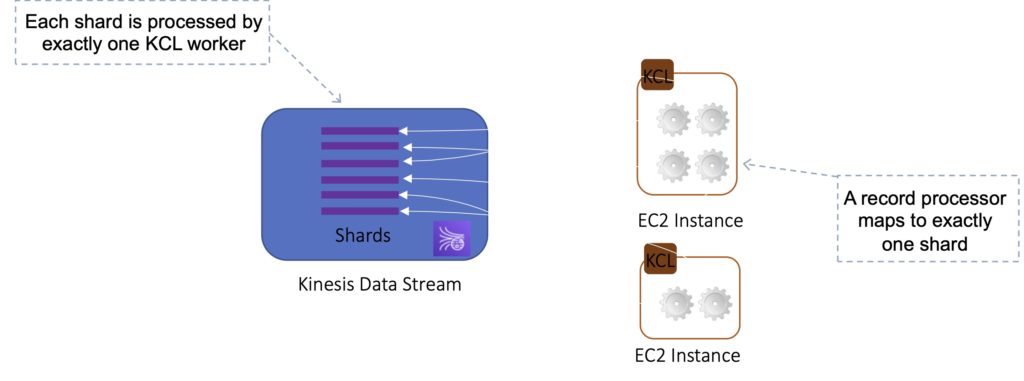
The KCL ensures that for every shard there is a record processor.

Manages the number of record processors relative to the number of shards & consumers.

If you have only one consumer, the KCL will create all the record processors on a single consumer.

Each shard is processed by exactly one KCL worker and has exactly one corresponding record processor, so you never need multiple instances to process one shard.

However, one worker can process any number of shards, so it’s fine if the number of shards exceeds the number of instances.



If you have two consumers it will load balance and create half the processors on one instance and half on another.

Scaling out consumers:

With KCL, generally you should ensure that the number of instances does not exceed the number of shards (except for failure or standby purposes).

Each shard can be read by only one KCL instance.

You never need multiple instances to handle the processing of one shard.

However, one worker can process multiple shards.

Example:

4 shards = max 4 KCL instances.

6 shards = max 6 KCL instances.

Progress is checkpointed into DynamoDB (IAM access required).

KCL can run on EC2, Elastic Beanstalk, and on-premises servers.

Records are read in order at the shard level.

Security

Control access / authorization using IAM policies.

Encryption in flight using HTTPS endpoints.

Encryption at rest using KMS.

Possible to encrypt / decrypt data on the client side.

VPC endpoints available for Kinesis to access within a VPC.

SQS vs SNS vs Kinesis

SQS:

Consumers pull data.

Data is deleted after being consumed.

Can have as many workers (consumers) as you need.

No need to provision throughput.

No ordering guarantee (except with FIFO queues).

Individual message delay.

SNS:

Push data to many subscribers.

Up to 10,000,000 subscribers.

Data is not persisted (lost if not deleted).

Pub/sub.

Up to 10,000,000 topics.

No need to provision throughput.

Integrates with SQS for fan-out architecture pattern.

Kinesis:

Consumers pull data.

As many consumers as you need.

Possible to replay data.

Meant for real-time big data, analytics, and ETL.

Ordering at the shard level.

Data expires after X days.

Must provision throughput.

Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.

Athena is a serverless product so with Amazon Athena AWS  manages all the infrastructure for you, and you pay only for the queries that you run.

With Amazon Athena AWS makes it extremely easy to use – simply point to your data in Amazon S3, define the schema, and start querying using standard SQL.

Amazon Athena uses Presto with full standard SQL support and works with a variety of standard data formats, including CSV, JSON, ORC, Apache Parquet and Avro.

While Amazon Athena is ideal for quick, ad-hoc querying and integrates with Amazon QuickSight for easy visualization, it can also handle complex analysis, including large joins, window functions, and arrays.

Amazon Athena uses a managed Data Catalog to store information and schemas about the databases and tables that you create for your data stored in Amazon S3.

With Amazon Athena, you don’t have to worry about managing or tuning clusters to get fast performance.

Amazon Athena is optimized for fast performance with Amazon S3.

Amazon Athena automatically executes queries in parallel, so that you get query results in seconds, even on large datasets.

Most results are delivered within seconds.

With Amazon Athena, there’s no need for complex ETL jobs to prepare data for analysis.

This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

Amazon Athena is out-of-the-box integrated with AWS Glue Data Catalog, allowing you to create a unified metadata repository across various services, crawl data sources to discover schemas and populate your Catalog with new and modified table and partition definitions, and maintain schema versioning.

You can also use Glue’s fully managed ETL capabilities to transform data or convert it into columnar formats to optimize cost and improve performance.

Amazon Athena Features

You can query Amazon Athena using Standard SQL

Amazon Athena uses Presto, an open source, distributed SQL query engine optimized for low latency, interactive data analysis.

Athena supports a wide variety of data formats such as CSV, JSON, ORC, Avro, or Parquet.

With Athena’s federated data source connectors, you can query additional data stores and join the data with data stored in Amazon S3.

You can access Athena and run queries from the Athena console, API, CLI, AWS SDK,

On Amazon Athena AWS supports the integration with business intelligence and SQL development applications through Athena’s JDBC and ODBC drivers.

With Amazon Athena AWS use the highly redundant Global infrastructure to maintain high availability.

Whenever a particular facility is unavailable, Amazon Athena routes queries to another facility based on the available compute resources.

In Amazon Athena AWS uses Amazon S3 as an origin, making your data highly available and durable.

In addition to providing durable infrastructure for keeping important data, Amazon S3 is designed for 99.999999999% durability on a per object basis.

Amazon Athena integrates directly with Identity and Access Management and you can leverage the use of bucket policies within S3 also.

By granting IAM policies to IAM users, you can easily control S3 buckets for your users. By controlling access to data in S3, you can restrict users from querying it using Athena.

Additionally, Athena allows you to query encrypted data stored in Amazon S3 and write encrypted results back to your bucket.

Athena provides connectors for enterprise data sources including Amazon DynamoDB, Amazon Redshift, Amazon OpenSearch, MySQL, PostgreSQL, Redis, and other popular third-party data stores.

Amazon Athena Limitations

In terms of optimization capabilities, AWS Athena doesn’t offer much in the way of capabilities.

Although Amazon Athena is a highly scalable and reliable service, it is hosted in a muti-tenant environment.  
  
This multi-tenancy approach might trigger throttling from time to time, leading to potential issues.

Amazon Athena doesn’t include any kind of Data Manipulation interface for inserting, deleting, and updating data.

If you intend to run your SQL queries efficiently, you might want to partition the data sets stored in Amazon S3. The number of partitions that you manage to create will substantially affect the speed and performance.

Indexing is not natively supported within Amazon Athena, which is built upon AWS.

Use Cases

Query services like Amazon Athena, data warehouses like Amazon Redshift, and sophisticated data processing frameworks like Amazon EMR, all address different needs and use cases.

Amazon Redshift provides the fastest query performance for enterprise reporting and business intelligence workloads, particularly those involving extremely complex SQL with multiple joins and sub-queries.

Amazon EMR makes it simple and cost effective to run highly distributed processing frameworks such as Hadoop, Spark, and Presto when compared to on-premises deployments. Amazon EMR is flexible – you can run custom applications and code, and define specific compute, memory, storage, and application parameters to optimize your analytic requirements.

Amazon Athena provides the easiest way to run ad-hoc queries for data in S3 without the need to setup or manage any servers.

With Athena, you can analyze unstructured, semistructured, and structured data stored in Amazon S3. Examples include CSV, JSON, or columnar data formats such as Apache Parquet and Apache ORC.

In Athena, ANSI SQL queries can be run ad-hoc without aggregating or loading the data.

Data visualization is made simple with Athena’s integration with Amazon QuickSight.

Using Athena, you will be able to generate reports or to explore your data with the help of business intelligence tools or SQL clients that are connected to a JDBC or ODBC driver to generate reports or to explore data.

With AWS Glue Data Catalog, AWHena integrates with your data to create a persistent metadata store in Amazon S3, which will allow you to access your data from anywhere.

With AWS Glue, you can create tables and query data in Athena using a central metadata store that is accessible throughout your Amazon Web Services account.

In order to troubleshoot a performance issue on your website, Athena is useful for running a quick query on web logs. A table for your data can be defined quickly with Athena, and you can start querying using the standard SQL language.

You should use Amazon Athena if you want to interact with your Cost and Usage Report stored in S3 to gain extremely specific information on how your AWS bill is being calculated. This can be done natively within the console.

The table below shows the primary use case and situations for using a few AWS query and analytics services:

|  |  |  |
| --- | --- | --- |
| AWS Service | Primary Use Case | When to use |
| Amazon Athena | Query | Run interactive queries against data directly in Amazon S3 without worrying about formatting data or managing infrastructure. Can use with other services such as Amazon RedShift |
| Amazon RedShift | Data Warehouse | Pull data from many sources, format and organize it, store it, and support complex, high speed queries that produce business reports. |
| Amazon EMR | Data Processing | Highly distributed processing frameworks such as Hadoop, Spark, and Presto. Run a wide variety of scale-out data processing tasks for applications such as machine learning, graph analytics, data transformation, streaming data. |
| AWS Glue | ETL Service | Transform and move data to various destinations. Used to prepare and load data for analytics. Data source can be S3, RedShift or another database. Glue Data Catalog can be queried by Athena, EMR and RedShift Spectrum |

Best Practices

[Best practices](https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena/) for performance with Athena:

Partition your data – Partition the table into parts and keeps the related data together based on column values such as date, country, region, etc. Athena supports Hive partitioning.

Bucket your data – Partition your data is to bucket the data within a single partition.

Use Compression – AWS recommend using either Apache Parquet or Apache ORC.

Optimize file sizes – Queries run more efficiently when reading data can be parallelized and when blocks of data can be read sequentially.

Optimize columnar data store generation – Apache Parquet and Apache ORC are popular columnar data stores.

Optimize ORDER BY – The ORDER BY clause returns the results of a query in sort order.

Optimize GROUP BY – The GROUP BY operator distributes rows based on the GROUP BY columns to worker nodes, which hold the GROUP BY values in memory.

Use approximate functions – For exploring large datasets, a common use case is to find the count of distinct values for a certain column using COUNT(DISTINCT column).

Only include the columns that you need – When running your queries, limit the final SELECT statement to only the columns that you need instead of selecting all columns.

Pricing

With Amazon Athena, you pay only for the queries that you run.

You are charged based on the amount of data scanned by each query.

You can get significant cost savings and performance gains by compressing, partitioning, or converting your data to a columnar format, because each of those operations reduces the amount of data that Athena needs to scan to execute a query.

Amazon Athena Frequently Asked Questions (FAQs)

Q: What can I do with Amazon Athena?

With Amazon Athena, you can analyze data stored in Amazon S3. Athena supports interactive analytics using ANSI SQL without aggregation or loading data. Data sets can be unstructured, semistructured, or structured using Amazon Athena. You can use Amazon Athena with Amazon QuickSight to visualize CSV, JSON, Avro or columnar data formats like Apache Parquet and Apache ORC.

Q: How do I get started with Amazon Athena?

Using Amazon Athena is as simple as logging into the AWS Management Console for Athena and writing your DDL statements or using a create table wizard. A built-in query editor lets you query data.

Q: What is the difference between Amazon Athena, Amazon EMR, and Amazon Redshift?

The Amazon Redshift platform delivers the fastest query performance for enterprise reporting and business intelligence workloads, especially for SQL that has multiple joins.

Comparatively to on-premises deployments, Amazon EMR makes highly distributed processing frameworks like Hadoop, Spark, and Presto simple and cost-effective.

Amazon EMR allows you to run custom applications and code, and define specific compute, memory, storage, and application parameters.

Without setting up or managing any servers, Amazon Athena offers interactive queries for S3 data.

Q: When should you use a full featured enterprise data warehouse, like Amazon Redshift vs. a query service like Amazon Athena?

‘Q: How do I create tables and schemas for my data on Amazon S3?

AWS Glue

AWS Glue is a fully-managed, pay-as-you-go, extract, transform, and load (ETL) service that automates the time-consuming steps of data preparation for analytics.

AWS Glue automatically discovers and profiles data via the Glue Data Catalog, recommends and generates ETL code to transform your source data into target schemas.

AWS Glue runs the ETL jobs on a fully managed, scale-out Apache Spark environment to load your data into its destination.

AWS Glue also allows you to setup, orchestrate, and monitor complex data flows.

You can create and run an ETL job with a few clicks in the AWS Management Console.

Simply point AWS Glue to your data stored on AWS, and AWS Glue discovers data and stores the associated metadata (e.g. table definition and schema) in the AWS Glue Data Catalog.

Once cataloged, data is immediately searchable, queryable, and available for ETL.

AWS Glue consists of a Data Catalog which is a central metadata repository, an ETL engine that can automatically generate Scala or Python code, and a flexible scheduler that handles dependency resolution, job monitoring, and retries.

Together, these automate much of the undifferentiated heavy lifting involved with discovering, categorizing, cleaning, enriching, and moving data, so you can spend more time analyzing your data.

AWS Glue Crawlers

You can use a crawler to populate the AWS Glue Data Catalog with tables.

This is the primary method used by most AWS Glue users.

A crawler can crawl multiple data stores in a single run.

Upon completion, the crawler creates or updates one or more tables in your Data Catalog. Extract, transform, and load (ETL) jobs that you define in AWS Glue use these Data Catalog tables as sources and targets.

The ETL job reads from and writes to the data stores that are specified in the source and target Data Catalog tables.

AWS Glue crawlers connect to a source or target data store, progress through a prioritized list of classifiers to determine the schema for the data, and then creates metadata in the AWS Glue Data Catalog.

The metadata is stored in tables in a data catalog and used in the authoring process of ETL jobs.

You can run crawlers on a schedule, on-demand, or trigger them based on an event to ensure that your metadata is up-to-date.

AWS Glue automatically generates the code to extract, transform, and load data.

Simply point AWS Glue to a source and target, and AWS Glue creates ETL scripts to transform, flatten, and enrich the data.

The code is generated in Scala or Python and written for Apache Spark.

AWS Glue helps clean and prepare data for analysis by providing a Machine Learning Transform called FindMatches for deduplication and finding matching records.

Use Cases

Use AWS Glue to discover properties of data, transform it, and prepare it for analytics.

Glue can automatically discover both structured and semi-structured data stored in data lakes on [Amazon S3](https://aws.amazon.com/s3/), data warehouses in [Amazon Redshift](https://aws.amazon.com/redshift/), and various databases running on AWS.

It provides a unified view of data via the Glue Data Catalog that is available for ETL, querying and reporting using services like [Amazon Athena](https://aws.amazon.com/athena/), [Amazon EMR](https://aws.amazon.com/emr/), and [Amazon Redshift Spectrum](https://aws.amazon.com/redshift/).

Glue automatically generates Scala or Python code for ETL jobs that you can further customize using tools you are already familiar with.

AWS Glue is serverless, so there are no compute resources to configure and manage.

Amazon ECS and EKS

The Amazon Elastic Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers.

Amazon ECS allows you to easily run applications on a managed cluster of Amazon EC2 instances.

Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure.

Using API calls you can launch and stop container-enabled applications, query the complete state of clusters, and access features like security groups, Elastic Load Balancing, EBS volumes and IAM roles.

Amazon ECS can be used to schedule the placement of containers across clusters based on resource needs and availability requirements.

There is no additional charge for Amazon ECS. You pay for:

Resources created with the EC2 Launch Type (e.g. EC2 instances and EBS volumes).

The number and configuration of tasks you run for the Fargate Launch Type.

It is possible to use Elastic Beanstalk to handle the provisioning of an Amazon ECS cluster, load balancing, auto-scaling, monitoring, and placing your containers across your cluster.

Alternatively use ECS directly for more fine-grained control for customer application architectures.

It is possible to associate a service on Amazon ECS to an Application Load Balancer (ALB) for the Elastic Load Balancing (ELB) service.

The ALB supports a target group that contains a set of instance ports. You can specify a dynamic port in the ECS task definition which gives the container an unused port when it is scheduled on the EC2 instance.

You can use any AMI that meets the Amazon ECS AMI specification. For example, please follow the link regarding [Amazon ECS-Optimized AMIs](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-optimized_AMI.htm).

Amazon ECS vs Amazon EKS

Amazon also provide the Elastic Container Service for Kubernetes (Amazon EKS) which can be used to deploy, manage, and scale containerized applications using Kubernetes on AWS.

The table below describes some of the differences between these services to help you understand when you might choose one over the other:

|  |  |
| --- | --- |
| Amazon ECS | Amazon EKS |
| Managed, highly available, highly scalable container platform | Managed, highly available, highly scalable container platform |
| AWS-specific platform that supports Docker Containers | Compatible with upstream Kubernetes so it’s easy to lift and shift from other Kubernetes deployments |
| Considered simpler and easier to use | Considered more feature-rich and complex with a steep learning curve |
| Leverages AWS services like Route 53, ALB, and CloudWatch | A hosted Kubernetes platform that handles many things internally |
| “Tasks” are instances of containers that are run on underlying compute but more of less isolated | “Pods” are containers collocated with one another and can have shared access to each other |
| Limited extensibility | Extensible via a wide variety of third-party and community add-ons. |

Launch Types

An Amazon ECS launch type determines the type of infrastructure on which your tasks and services are hosted.

There are two launch types, and the table below describes some of the differences between the two launch types:

|  |  |
| --- | --- |
| Amazon EC2 | Amazon Fargate |
| You can explicitly provision EC2 instances | The control plan asks for resources and Fargate automatically provisions |
| You’re responsible for upgrading, patching, care of EC2 pool | Fargate provisions compute as needed |
| You must handle cluster optimization | Fargate handles customer optimizations |
| More granular control over infrastructure | Limited control, as infrastructure is automated |

Fargate Launch Type

The Fargate launch type allows you to run your containerized applications without the need to provision and manage the backend infrastructure. Just register your task definition and Fargate launches the container for you.

Fargate Launch Type is a serverless infrastructure managed by AWS.

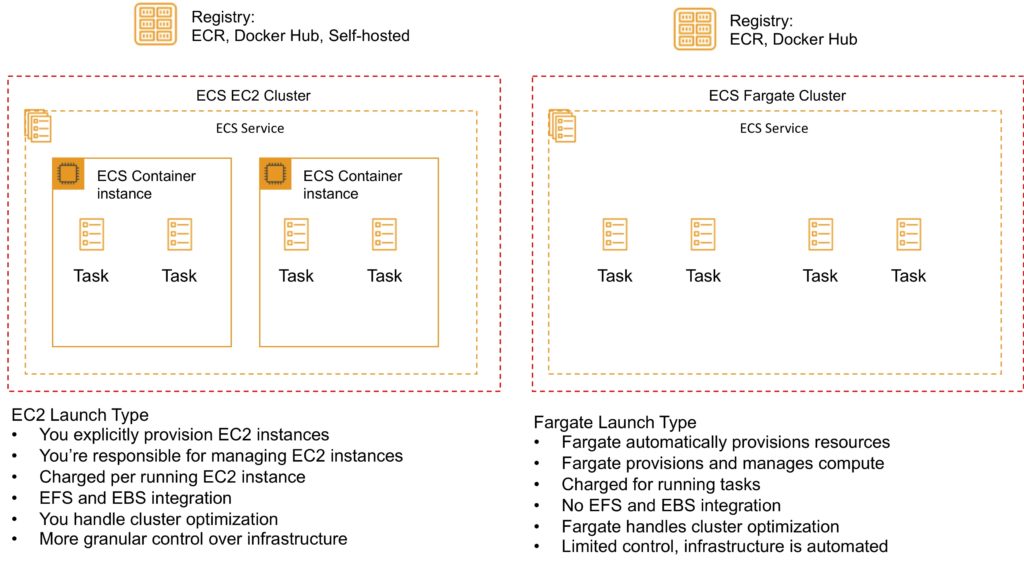
Fargate only supports container images hosted on Elastic Container Registry (ECR) or Docker Hub.

EC2 Launch Type

The EC2 launch type allows you to run your containerized applications on a cluster of Amazon EC2 instances that you manage.

Private repositories are only supported by the EC2 Launch Type.

The following diagram shows the two launch types and summarizes some key differences:



ECS Terminology

The following table provides an overview of some of the terminology used with Amazon ECS:

|  |  |
| --- | --- |
| Amazon ECS Term | Definition |
| Cluster | Logical Grouping of EC2 Instances |
| Container Instance | EC2 instance running the ECS agent |
| Task Definition | Blueprint that describes how a docker container should launch |
| Task | A running container using settings in a Task Definition |
| Service | Defines long running tasks – can control task count with Auto Scaling and attach an ELB |

Images

Containers are created from a read-only template called an image which has the instructions for creating a Docker container.

Images are built from a Dockerfile.

Only Docker containers are currently supported.

An image contains the instructions for creating a Docker container.

Images are stored in a registry such as DockerHub or AWS Elastic Container Registry (ECR).

ECR is a managed AWS Docker registry service that is secure, scalable, and reliable.

ECR supports private Docker repositories with resource-based permissions using AWS IAM to access repositories and images.

Developers can use the Docker CLI to push, pull and manage images.

Tasks and Task Definitions

A task definition is required to run Docker containers in Amazon ECS.

A task definition is a text file in JSON format that describes one or more containers, up to a maximum of 10.

Task definitions use Docker images to launch containers.

You specify the number of tasks to run (i.e. the number of containers).

Some of the parameters you can specify in a task definition include:

Which Docker images to use with the containers in your task.

How much CPU and memory to use with each container.

Whether containers are linked together in a task.

The Docker networking mode to use for the containers in your task.

What (if any) ports from the container are mapped to the host container instances.

Whether the task should continue if the container finished or fails.

The commands the container should run when it is started.

Environment variables that should be passed to the container when it starts.

Data volumes that should be used with the containers in the task.

IAM role the task should use for permissions.

You can use Amazon ECS Run task to run one or more tasks once.

ECS Clusters

ECS Clusters are a logical grouping of container instances that you can place tasks on.

A default cluster is created but you can then create multiple clusters to separate resources.

ECS allows the definition of a specified number (desired count) of tasks to run in the cluster.

Clusters can contain tasks using the Fargate and EC2 launch type.

For clusters with the EC2 launch type clusters can contain different container instance types.

Each container instance may only be part of one cluster at a time.

“Services” provide auto-scaling functions for ECS.

Clusters are region specific.

You can create IAM policies for your clusters to allow or restrict users’ access to specific clusters.

Service Scheduler

You can schedule ECS using Service Scheduler and Custom Scheduler.

Ensures that the specified number of tasks are constantly running and reschedules tasks when a task fails.

Can ensure tasks are registered against an ELB.

Custom Scheduler

You can create your own schedulers to meet business needs.

Leverage third party schedulers such as Blox.

The Amazon ECS schedulers leverage the same cluster state information provided by the Amazon ECS API to make appropriate placement decisions.

ECS Container Agent

The ECS container agent allows container instances to connect to the cluster.

The container agent runs on each infrastructure resource on an ECS cluster.

The ECS container agent is included in the Amazon ECS optimized AMI and can also be installed on any EC2 instance that supports the ECS specification (only supported on EC2 instances).

Linux and Windows based.

For non-AWS Linux instances to be used on AWS you must manually install the ECS container agent.

Auto Scaling

Service Auto Scaling

Amazon ECS service can optionally be configured to use Service Auto Scaling to adjust the desired task count up or down automatically.

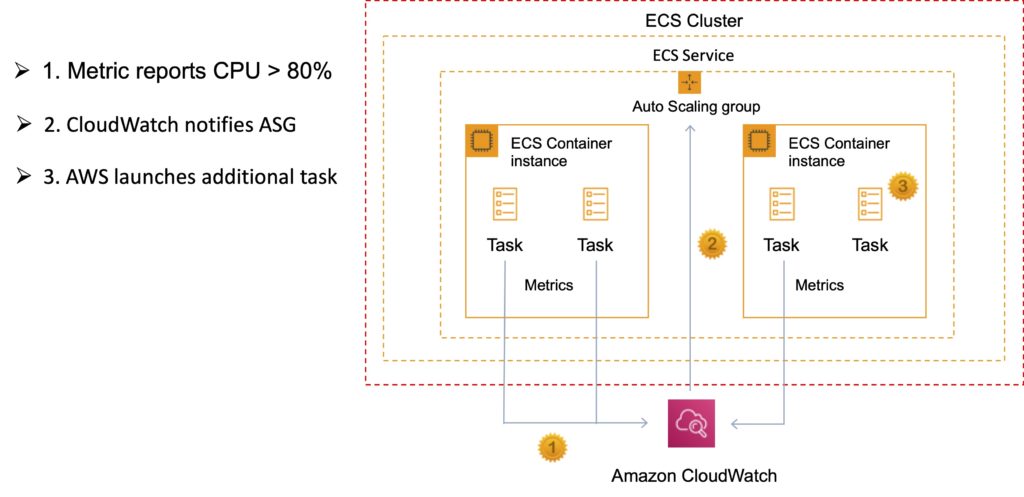
Service Auto Scaling leverages the Application Auto Scaling service to provide this functionality.

Amazon ECS Service Auto Scaling supports the following types of scaling policies:

[Target Tracking Scaling Policies](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-autoscaling-targettracking.html)—Increase or decrease the number of tasks that your service runs based on a target value for a specific CloudWatch metric. This is similar to the way that your thermostat maintains the temperature of your home. You select temperature and the thermostat does the rest.

[Step Scaling Policies](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-autoscaling-stepscaling.html)—Increase or decrease the number of tasks that your service runs in response to CloudWatch alarms. Step scaling is based on a set of scaling adjustments, known as step adjustments, which vary based on the size of the alarm breach.

[Scheduled Scaling](https://docs.aws.amazon.com/autoscaling/application/userguide/application-auto-scaling-scheduled-scaling.html)—Increase or decrease the number of tasks that your service runs based on the date and time.



Cluster Auto Scaling

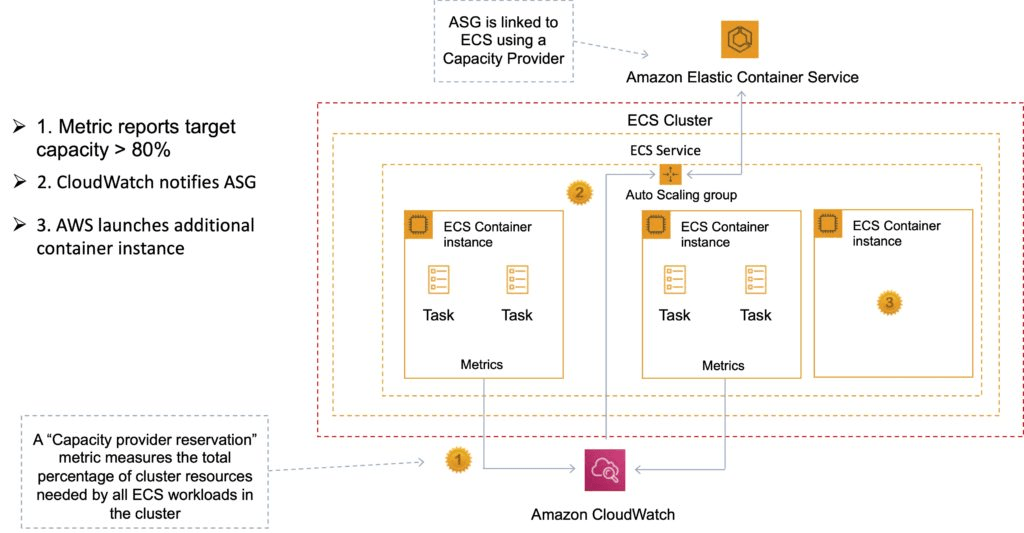
Uses an Amazon ECS resource type called a Capacity Provider.

A Capacity Provider can be associated with an EC2 Auto Scaling Group (ASG).

When you associate an ECS Capacity Provider with an ASG and add the Capacity Provider to an ECS cluster, the cluster can now scale your ASG automatically by using two new features of ECS:

Managed scaling, with an automatically created scaling policy on your ASG, and a new scaling metric (Capacity Provider Reservation) that the scaling policy uses; and

Managed instance termination protection, which enables container-aware termination of instances in the ASG when scale-in happens.



Security/SLA

EC2 instances use an IAM role to access ECS.

IAM can be used to control access at the container level using IAM roles.

The container agent makes calls to the ECS API on your behalf through the applied IAM roles and policies.

You need to apply IAM roles to container instances before they are launched (EC2 launch type).

Assign extra permissions to tasks through separate IAM roles (IAM Roles for Tasks).

ECS tasks use an IAM role to access services and resources.

Security groups attach at the instance or container level.

You have root level access to the operating system of the EC2 instances.

The Compute SLA guarantees a Monthly Uptime Percentage of at least 99.99% for Amazon ECS.

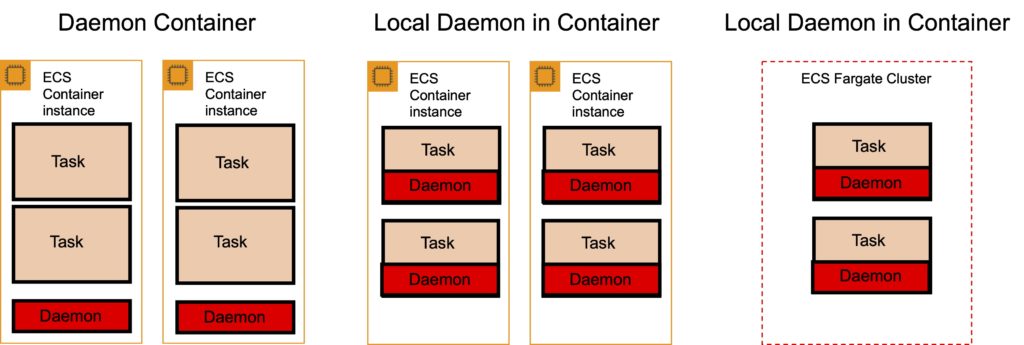
ECS with X-Ray

There are two ways to run X-Ray:

As a daemon: X-Ray agent runs in a daemon container.

As a “sidecar”: X-Ray runs alongside each container.

For Fargate the X-Ray daemon only runs as a sidecar.



Task definition:

X-Ray runs on port 2000 UDP.

Must specify the daemon address.

Must link the containers together.

X-Ray provides a Docker container image that you can deploy alongside your application:

Command: docker pull amazon/aws-xray-daemon

ECS with Elastic Beanstalk

There are two options: Single and Multi- Docker container mode.

Single Container Docker

The single container platform can be used to deploy a Docker image (described in a Dockerfile or Dockerrun.aws.json definition) and source code to EC2 instances running in an Elastic Beanstalk environment.

Use the single container platform when you only need to run one container per instance.

Multicontainer Docker

The other basic platform, Multicontainer Docker, uses the Amazon Elastic Container Service to coordinate the deployment of multiple Docker containers to an Amazon ECS cluster in an Elastic Beanstalk environment.

The instances in the environment each run the same set of containers, which are defined in a Dockerrun.aws.json file.

Use the multicontainer platform when you need to deploy multiple Docker containers to each instance.

ElasticBeanstalk creates the following resources:

ECS cluster.

EC2 container instances.

Load balancers (for high availability mode).

Task definitions and execution.

Requires a config file Dockerrun.aws.json at the root of the source code.

Your Docker images must be pre-built (can be stored in ECR).

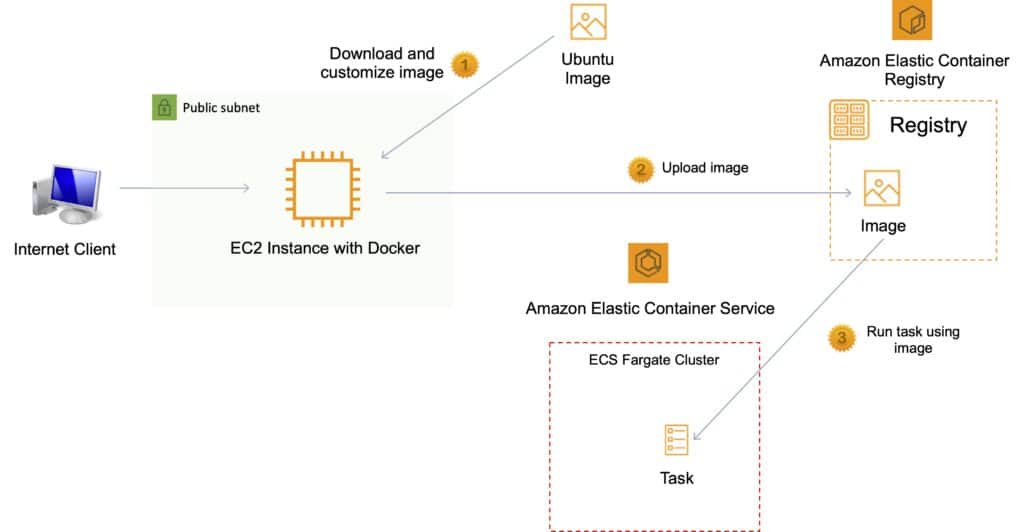
Amazon Elastic Container Registry (ECR)

Amazon Elastic Container Registry (ECR) is a fully managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.

Amazon ECR is integrated with Amazon Elastic Container Service (ECS).

Amazon ECR hosts your images in a highly available and scalable architecture, allowing you to reliably deploy containers for your applications.

Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository.



Pushing and Pulling Images to ECR

You must first authenticate.

To authenticate Docker to an Amazon ECR registry with get-login-password, run the aws ecr get-login-password command:

aws ecr get-login-password –region us-east-1 | docker login –username AWS –password-stdin aws\_account\_id.dkr.ecr.us-east-1.amazonaws.com

Tag your image:

docker tag e9ae3c220b23 aws\_account\_id.dkr.ecr.region.amazonaws.com/my-web-app

Push the image using the docker push command:

docker push aws\_account\_id.dkr.ecr.region.amazonaws.com/my-web-app

Pull the image using the docker pull command. The image name format should be registry/repository[:tag] to pull by tag, or registry/repository[@digest] to pull by digest.

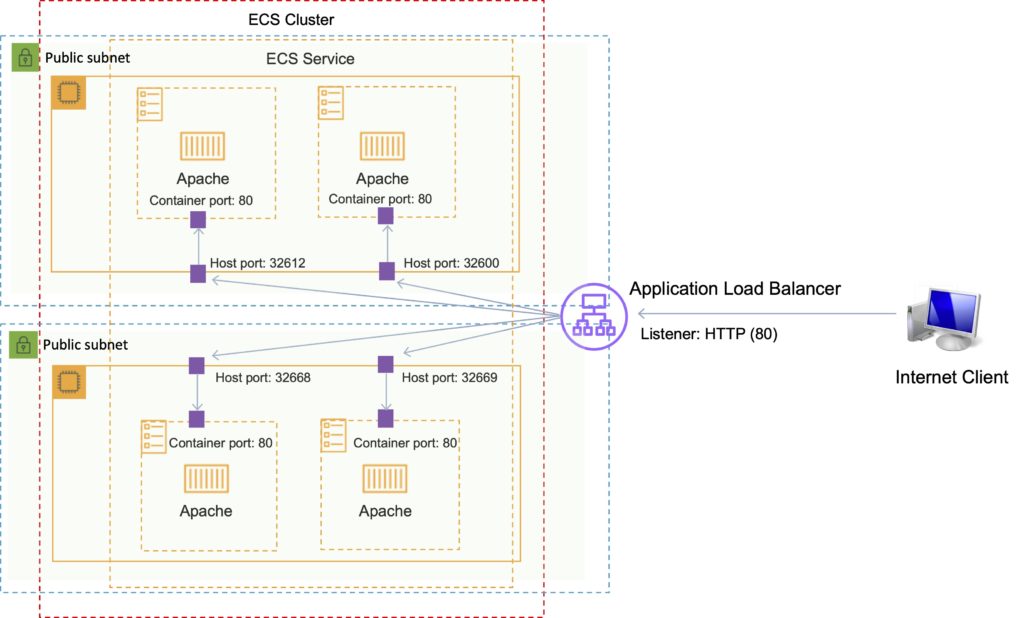
docker pull aws\_account\_id.dkr.ecr.region.amazonaws.com/my-web-app:e9ae3c220b23

Using an Amazon Elastic Load Balancer with ECS

It is possible to associate a service on Amazon ECS to an Amazon Elastic Load Balancer (ELB).

The ALB supports a target group that contains a set of instance ports.

You can specify a dynamic port in the ECS task definition which gives the container an unused port when it is scheduled on the EC2 instance (this is specific to ALB only).



Amazon EKS

The Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service for running Kubernetes on AWS and on-premises.

Amazon EKS can run on Amazon EC2 or AWS Fargate.

Integrates with Application Load Balancers, AWS IAM for RBA and Amazon VPC.

Amazon EKS provides a scalable and highly available Kubernetes control plane running across multiple AWS Availability Zones (AZs).

Amazon EKS automatically manages availability and scalability of Kubernetes API servers and etcd persistence layer.

Amazon EKS runs the Kubernetes control plane across three AZs to ensure high availability, and automatically detects and replaces unhealthy control plane nodes.

Exam tip: The principle use case is when organizations need a consistent control plane for managing containers across hybrid clouds and multicloud environments.

Amazon Cognito

Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily.

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps.

Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, or Google.

Web Identity Federation

AWS Cognito works with external identity providers that support SAML or OpenID Connect, social identity providers (such as Facebook, Twitter, Amazon)

Federation allows users to authenticate with a Web Identity Provider (e.g. Google, Facebook, Amazon).

The user authenticates first with the Web ID provider and receives an authentication token, which is then exchanges for temporary AWS credentials allowing them to assume an IAM role allowing access to the required resources.

Cognito is an Identity Broker which handles interaction between your applications and the Web ID provider (you don’t need to write your own code to do this).

You can use Amazon, Facebook, Twitter, Digits, Google and any other OpenID Connect compatible identity provider.

You can also integrate your own identity provider.

User Pools and Identity Pools

The two main components of AWS Cognito are user pools and identity pools:

User pools are user directories that provide sign-up and sign-in options for your app users.

Identity pools enable you to grant your users access to other AWS services.

You can use identity pools and user pools separately or together.

No need for the application to embed or store AWS credentials locally on the device and it gives users a seamless experience across all mobile devices.

Cognito Identity provides temporary security credentials to access your app’s backend resources in AWS or any service behind Amazon API Gateway.

Cognito exposes server-side APIs.

Users can sign-up and sign-in using email, phone number, or user name.

End users of an application can also sign in with SMS-based MFA.

There is an import tool for migrating users into an Amazon Cognito User Pool.



User Pools

Cognito User Pools are user directories used to manage sign-up and sign-in functionality for mobile and web applications.

With a user pool, users can sign in to your web or mobile app through Amazon Cognito.

Users can also sign in through social identity providers like Facebook or Amazon, and through SAML identity providers.

Whether users sign-in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

Cognito acts as an Identity Broker between the ID provider and AWS.

User pools provide:

Sign-up and sign-in services.

A built-in, customizable web UI to sign in users.

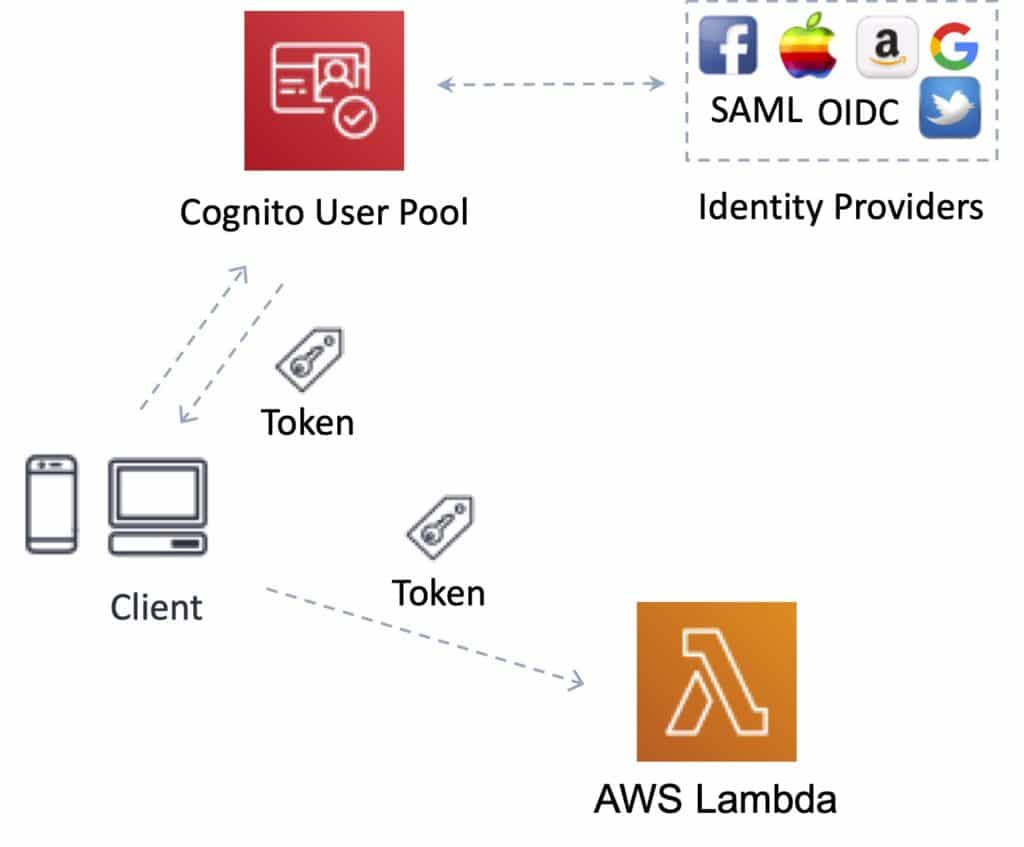
Social sign-in with Facebook, Google, and Login with Amazon, as well as sign-in with SAML identity providers from your user pool.

User directory management and user profiles.

Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.

Customized workflows and user migration through AWS Lambda triggers.

After successfully authenticating a user, Amazon Cognito issues JSON web tokens (JWT) that you can use to secure and authorize access to your own APIs, or exchange for AWS credentials.



Identity Pools

Identity Pools enable you to create unique identities for your users and authenticate them with identity providers.

With an identity, you can obtain temporary, limited-privilege AWS credentials to access other AWS services.

Cognito tracks the association between user identity and the various different devices they sign-in from.

In order to provide a seamless user experience for your application, Cognito uses Push Synchronization to push updates and synchronize user data across multiple devices.

Amazon SNS is used to send a silent push notification to all the devices whenever data stored in the cloud changes.

Amazon Cognito identity pools support the following identity providers:

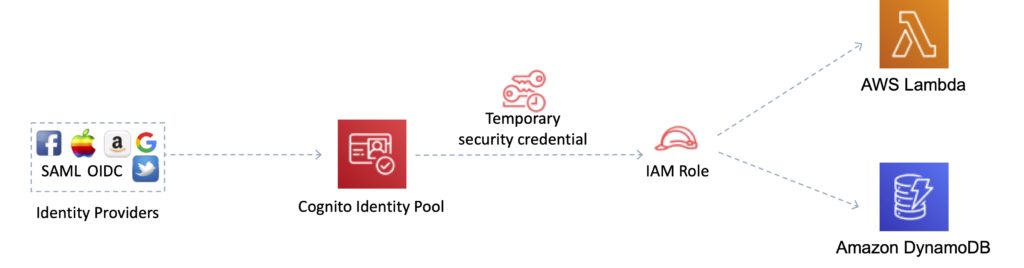
Public providers: Login with Amazon (Identity Pools), Facebook (Identity Pools), Google (Identity Pools).

Amazon Cognito User Pools.

Open ID Connect Providers (Identity Pools).

SAML Identity Providers (Identity Pools).

Developer Authenticated Identities (Identity Pools).



Exam tip: To make it easier to remember the different between User Pools and Identity Pools, think of Users Pools as being like IAM Users or Active Directory and an Identity Pools as being like an IAM Role.

Amazon Cognito Sync

Amazon Cognito Sync is an AWS service and client library that enables cross-device syncing of application-related user data.

You can use it to synchronize user profile data across mobile devices and the web without requiring your own backend.

The client libraries cache data locally so your app can read and write data regardless of device connectivity status.

When the device is online, you can synchronize data, and if you set up push sync, notify other devices immediately that an update is available.

Exam tip: AWS AppSync is a similar service that has additional capabilities. With AppSync you can synchronize mobile app data across devices and users (Cognito Sync cannot synchronize across users, only devices), it has support for additional devices and data types, and is based on GraphQL.

AWS Directory Services

AWS provide several directory types.

The following three types currently feature on the exam and will be covered on this page:

Active Directory Service for Microsoft Active Directory.

Simple AD.

AD Connector.

As an alternative to the AWS Directory service you can build your own Microsoft AD DCs in the AWS cloud (on EC2).

When you build your own you can join an existing on-premises Active Directory domain (replication mode).

You must establish a VPN (on top of Direct Connect if you have it).

Replication mode is less secure than establishing trust relationships.

The table below summarizes the directory services covered on this page as well as a couple of others, and provides some typical use cases:

|  |  |  |
| --- | --- | --- |
| Directory Service Option | Description | Use Case |
| AWS Cloud Directory | Cloud-native directory to share and control access to hierarchical data between applications | Cloud applications that need hierarchical data with complex relationships |
| Amazon Cognito | Sign-up and sign-in functionality that scales to millions of users and federated to public social media services | Develop consumer apps or SaaS |
| AWS Directory Service for Microsoft Active Directory | AWS-managed full Microsoft AD running on Windows Server 2012 R2 | Enterprises that want hosted Microsoft AD or you need LDAP for Linux apps |
| AD Connector | Allows on-premises users to log into AWS services with their existing AD credentials. Also allows EC2 instances to join AD domain | Single sign-on for on-premises employees and for adding EC2 instances to the domain |
| Simple AD | Low scale, low cost, AD implementation based on Samba | Simple user directory, or you need LDAP compatibility |

Active Directory Service for Microsoft Active Directory

Fully managed AWS services on AWS infrastructure.

Best choice if you have more than 5000 users and/or need a trust relationship set up.

Includes software patching, replication, automated backups, replacing failed DCs and monitoring.

Runs on a Windows Server.

Can perform schema extensions.

Works with SharePoint, Microsoft SQL Server and .Net apps.

You can setup trust relationships to extend authentication from on-premises Active Directories into the AWS cloud.

On-premises users and groups can access resources in either domain using SSO.

Requires a VPN or Direct Connect connection.

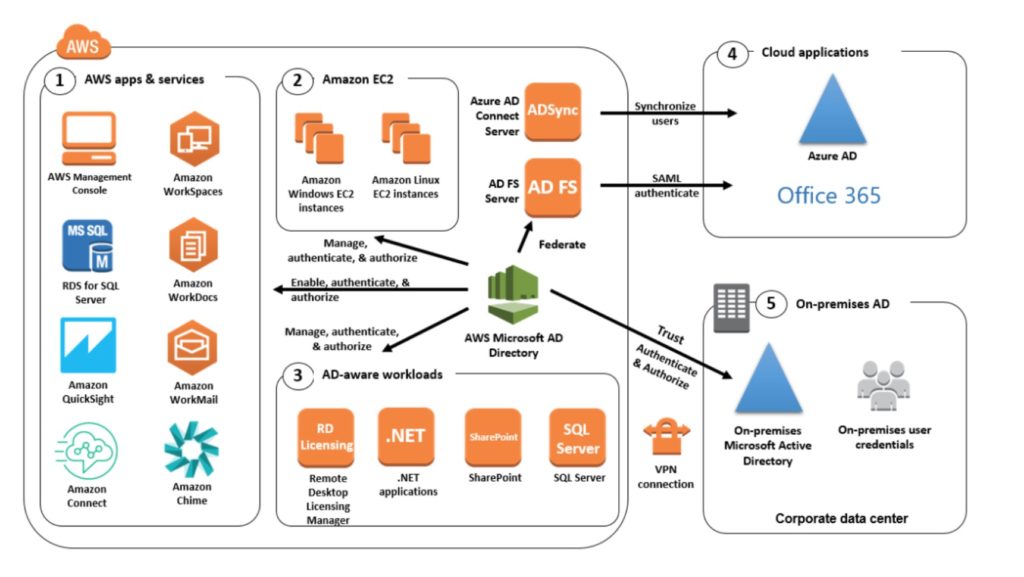
Can be used as a standalone AD in the AWS cloud.

When used standalone users can access 3rd party applications such as Microsoft O365 through federation.

You can also use Active Directory credentials to authenticate to the AWS management console without having to set up SAML authentication.

AWS Microsoft AD supports AWS applications including Workspaces, WorkDocs, QuickSight, Chime, Amazon Connect, and RDS for Microsoft SQL Server.

The following diagram shows some of the use cases for your AWS Microsoft AD directory, including the ability to grant your users access to external cloud applications and allow your on-premises AD users to manage and have access to resources in the AWS Cloud.



Includes security features such as:

Fine-grained password policy management.

LDAP encryption through SSL/TLS.

HIPAA and PCI DSS approved.

Multi-factor authentication through integration with existing RADIUS-based MFA infrastructure.

Monitoring provided through CloudTrail, notifications through SNS, daily automated snapshots.

Scalable service that scales by adding Domain Controllers.

Deployed in a HA configuration across two AZs in the same region.

AWS Microsoft AD does not support replication mode where replication to an on-premises AD takes place.

Two editions:

Standard Edition is optimized to be a primary directory for small and midsize businesses with up to 5,000 employees. It provides you enough storage capacity to support up to 30,000 directory objects, such as users, groups, and computers.

Enterprise Edition is designed to support enterprise organizations with up to 500,000 directory objects.

Directory Sharing:

AWS Directory Service for Microsoft Active Directory allows you to use a directory in one account and share it with multiple accounts and VPCs.

There is an hourly sharing charge for each additional account to which you share a directory.

There is no sharing charge for additional VPCs to which you share a directory, or for the account in which you install the directory.

Simple AD

An inexpensive Active Directory-compatible service with common directory features.

Standalone, fully managed, directory on the AWS cloud.

Simple AD is generally the least expensive option.

Best choice for less than 5000 users and don’t need advanced AD features.

Powered by SAMBA 4 Active Directory compatible server.

Can create users and control access to applications on AWS.

Provides a subset of the features provided by AWS MS AD.

Features include:

Manage user accounts.

Manage groups.

Apply group policies.

Securely connect to EC2 instances.

Kerberos-based SSO.

Supports joining Linux or Windows based EC2 instances.

AWS provides monitoring, daily snapshots, and recovery services.

Manual snapshots possible.

Simple AD is compatible with WorkSpaces, WorkDocs, Workmail and QuickSight.

You can also sign on to the AWS management console with Simple AD user accounts to manage AWS resources.

Available in two editions:

Small – supports up to 500 users (approximately 2000 objects).

Large – supports up to 5000 users (approximately 20,000 objects).

AWS creates two directory servers and DNS servers on two different subnets within an AZ.

Simple AD does not support:

DNS dynamic updates.

Schema extensions.

Multi-factor authentication.

Communication over LDAPS.

PowerShell AD cmdlets.

FSMO role transfer.

Not compatible with RDS SQL server.

Does not support trust relationships with other domains (use AWS MS AD).

AD Connector

AD Connector is a directory gateway for redirecting directory requests to your on-premises Active Directory.

AD Connector eliminates the need for directory synchronization and the cost and complexity of hosting a federation infrastructure.

Connects your existing on-premises AD to AWS.

Best choice when you want to use an existing Active Directory with AWS services.

AD Connector comes in two sizes:

Small – designed for organizations up to 500 users.

Large – designed for organizations up to 5000 users.

The VPC must be connected to your on-premises network via VPN or Direct Connect.

When users log in to AWS applications AD connector forwards sign-in requests to your on-premises AD DCs.

You can also join EC2 instances to your on-premises AD through AD Connector.

You can also login to the AWS Management Console using your on-premises AD DCs for authentication.

Not compatible with RDS SQL.

You can use AD Connector for multi-factor authentication using RADIUS-based MFA infrastructure.

AD Connector vs Simple AD

The table below describes some of the key differences to consider when choosing AD Connector or Simple AD:

|  |  |
| --- | --- |
| AD Connector | Simple AD |
| Must have an existing AD | Standalone AD based on Samba |
| Existing AD users can access AWS assets via IAM roles | Supports user accounts, groups, group policies, and domains |
| Supports MFA via existing RADIUS-based MFA infrastructure | Kerberos-based SSO |
|  | MFA not supported |
|  | Trust relationships not supported |

AWS KMS

AWS Key Management Store (KMS) is a managed service that enables you to easily encrypt your data.

AWS KMS provides a highly available key storage, management, and auditing solution for you to encrypt data within your own applications and control the encryption of stored data across AWS services.

AWS KMS allows you to centrally manage and securely store your keys. These are known as AWS KMS keys (formerly known as customer master keys (CMKs).

AWS KMS Keys

A KMS key consists of:

Alias.

Creation date.

Description.

Key state.

Key material (either customer provided or AWS provided).

KMS keys are the primary resources in AWS KMS.

The KMS key includes metadata, such as the key ID, creation date, description, and key state.

The KMS key also contains the key material used to encrypt and decrypt data.

AWS KMS supports symmetric and asymmetric KMS keys.

KMS keys are created in AWS KMS. Symmetric KMS keys and the private keys of asymmetric KMS keys never leave AWS KMS unencrypted.

By default, AWS KMS creates the key material for a KMS key.

A KMS key can encrypt data up to 4KB in size.

A KMS key can generate, encrypt, and decrypt Data Encryption Keys (DEKs).

A KMS key can never be exported from KMS (CloudHSM allows this).

AWS Managed KMS keys:

KMS keys managed by AWS are used by AWS services that interact with KMS to encrypt data.

They can only be used by the service that created them within a particular region.

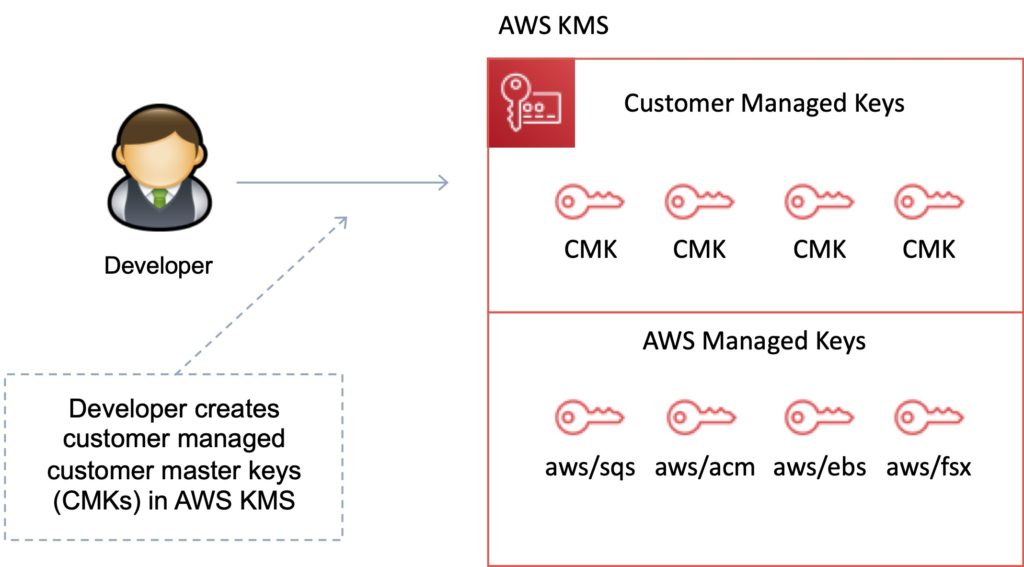
They are created on the first time you implement encryption using that service.

Customer managed KMS keys:

These provide the ability to implement greater flexibility.

You can perform rotation, governing access, and key policy configuration.

You are able to enable and disable the key when it is no longer required.



Customer Managed KMS keys

Customer managed KMS keys are KMS keys in your AWS account that you create, own, and manage.

You have full control over these KMS keys, including establishing and maintaining their key policies, IAM policies, and grants, enabling and disabling them, rotating their cryptographic material, adding tags, creating aliases that refer to the KMS key, and scheduling the KMS keys for deletion.

Customer managed KMS keys incur a monthly fee and a fee for use in excess of the free tier.

AWS Managed KMS keys

AWS managed KMS keys are KMS keys in your account that are created, managed, and used on your behalf by an AWS service that is integrated with AWS KMS.

You cannot manage these KMS keys, rotate them, or change their key policies.

You also cannot use AWS managed KMS keys in cryptographic operations directly; the service that creates them uses them on your behalf.

You do not pay a monthly fee for AWS managed KMS keys. They can be subject to fees for use in excess of the free tier, but some AWS services cover these costs for you.

AWS Owned KMS Keys

AWS owned KMS keys are a collection of KMS keys that an AWS service owns and manages for use in multiple AWS accounts.

Although AWS owned KMS keys are not in your AWS account, an AWS service can use its AWS owned KMS keys to protect the resources in your account.

You do not need to create or manage the AWS owned KMS keys.

However, you cannot view, use, track, or audit them.

You are not charged a monthly fee or usage fee for AWS owned KMS keys and they do not count against the AWS KMS quotas for your account.

Data Encryption Keys

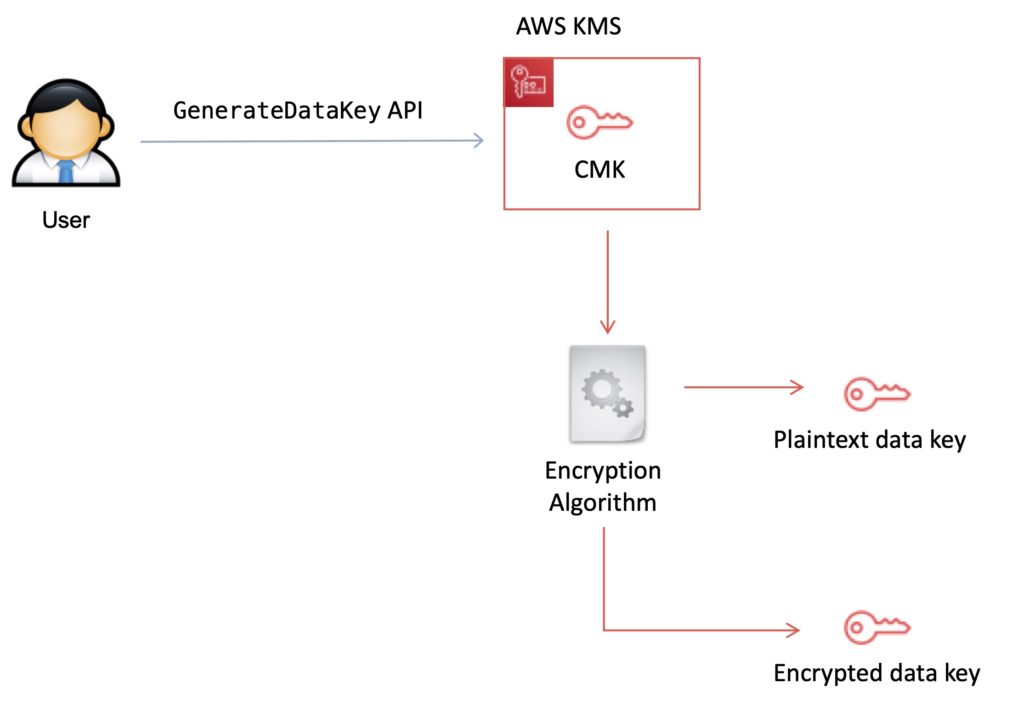
Data keys are encryption keys that you can use to encrypt data, including large amounts of data and other data encryption keys.

You can use AWS KMS keys to generate, encrypt, and decrypt data keys.

AWS KMS does not store, manage, or track your data keys, or perform cryptographic operations with data keys.

You must use and manage data keys outside of AWS KMS.

The GenerateDataKey API can be used to create a data encryption key using a KMS key:



KMS Details

You set usage policies on the keys that determine which users can use them to encrypt and decrypt data and under which conditions.

Key material options:

KMS generated.

Import your own.

You can generate KMS keys in KMS, in an AWS CloudHSM cluster, or import them from your own key management infrastructure.

These master keys are protected by hardware security modules (HSMs) and are only ever used within those modules.

You can submit data directly to KMS to be encrypted or decrypted using these master keys.

KMS now has the option for symmetric and asymmetric keys.

KMS is for encryption at rest only (not in transit, use SSL).

KMS is tightly integrated into many AWS services like Lambda, S3, EBS, EFS, DynamoDB, SQS etc.

Data keys are not retained or managed by KMS.

AWS services encrypt your data and store an encrypted copy of the data key along with the data it protects.

When a service needs to decrypt your data they request KMS to decrypt the data key using your master key.

If the user requesting data from the AWS service is authorized to decrypt under your master key policy, the service will receive the decrypted data key from KMS with which it can decrypt your data and return it in plaintext.

All requests to use your master keys are logged in AWS CloudTrail so you can understand who used which key under which context and when they used it.

You can control who manages and accesses keys via IAM users and roles.

You can audit the use of keys via CloudTrail.

KMS differs from Secrets Manager as its purpose-built for encryption key management.

KMS is validated by many compliance schemes (e.g. PCI DSS Level 1, FIPS 140-2 Level 2).

Exam tip: Encryption keys are regional.

Key Management with KMS

You can perform the following key management functions in AWS KMS:

Create keys with a unique alias and description.

Import your own key material.

Define which IAM users and roles can manage keys.

Define which IAM users and roles can use keys to encrypt and decrypt data.

Choose to have AWS KMS automatically rotate your keys on an annual basis.

Temporarily disable keys so they cannot be used by anyone.

Re-enable disabled keys.

Delete keys that you no longer use.

Audit use of keys by inspecting logs in AWS CloudTrail.

Create custom key stores\*.

Connect and disconnect custom key stores\*.

Delete custom key stores\*.

\* The use of custom key stores requires CloudHSM resources to be available in your account.

Data Encryption Scenarios

Typically, data is encrypted in one of the following three scenarios:

You can use KMS APIs directly to encrypt and decrypt data using your master keys stored in KMS.

You can choose to have AWS services encrypt your data using your master keys stored in KMS. In this case data is encrypted using data keys that are protected by your master keys in KMS.

You can use the AWS Encryption SDK that is integrated with AWS KMS to perform encryption within your own applications, whether they operate in AWS or not.

Custom Key Store

The AWS KMS custom key store feature combines the controls provided by AWS CloudHSM with the integration and ease of use of AWS KMS.

You can configure your own CloudHSM cluster and authorize KMS to use it as a dedicated key store for your keys rather than the default KMS key store.

When you create keys in KMS you can chose to generate the key material in your CloudHSM cluster. Master keys that are generated in your custom key store never leave the HSMs in the CloudHSM cluster in plaintext and all KMS operations that use those keys are only performed in your HSMs.

In all other respects master keys stored in your custom key store are consistent with other KMS keys.

Key Deletion

You can schedule a customer master key and associated metadata that you created in AWS KMS for deletion, with a configurable waiting period from 7 to 30 days.

This waiting period allows you to verify the impact of deleting a key on your applications and users that depend on it.

The default waiting period is 30 days.

You can cancel key deletion during the waiting period.

AWS KMS API’s

The following APIs are useful to know for the exam:

Encrypt (aws kms encrypt):

Encrypts plaintext into ciphertext by using a customer master key (KMS key).

You can encrypt small amounts of arbitrary data, such as  a  personal identifier or database password, or other sensitive information.

You can use the Encrypt operation to move encrypted data from one AWS region to another.

Decrypt (aws kms decrypt):

Decrypts ciphertext that was encrypted by an AWS KMS key using any of the following operations:

Encrypt

GenerateDataKey

GenerateDataKeyPair

GenerateDataKeyWithoutPlaintext

GenerateDataKeyPairWithoutPlaintext

Re-encrypt (aws kms re-encrypt):

Decrypts ciphertext and then re-encrypts it entirely within AWS KMS.

You can use this operation to change the customer master  key  (KMS key)  under which  data  is  encrypted,  such  as when you manually rotate a KMS key or change the KMS key that protects a ciphertext.

You can also use it to re-encrypt  ciphertext  under the same KMS key, such as to change the encryption context of a ciphertext.

Enable-key-rotation:

Enables  automatic  rotation of the key material for the specified symmetric customer master key (KMS key).

You cannot perform this operation  on a KMS key in a different AWS account.

GenerateDataKey (aws kms generate-data-key):

Enables  automatic  rotation of the key material for the specified symmetric customer master key (KMS key).

You cannot perform this operation  on a KMS key in a different AWS account.

GenerateDataKeyWithoutPlaintext (generate-data-key-without-plaintext):

Generates  a  unique  symmetric data key.

This operation returns a data key that is encrypted under a customer master key (KMS key) that you  specify.

To request an asymmetric data key pair, use the  GenerateDataKeyPair or  GenerateDataKeyPairWithoutPlaintext operations.

KMS Envelope Encryption

AWS KMS is integrated with AWS services and client-side toolkits that use a method known as envelope encryption to encrypt your data.

Under this method, KMS generates data keys which are used to encrypt data and are themselves encrypted using your master keys in KMS:

A KMS key is used to encrypt the data key (envelope key).

The envelope key is used to decrypt the data.

Limits

You can create up to 1000 KMS keys per account per region.

As both enabled and disabled KMS keys count towards the limit, AWS recommend deleting disabled keys that you no longer use.

AWS managed master keys created on your behalf for use within supported AWS services do not count against this limit.

There is no limit to the number of data keys that can be derived using a master key and used in your application or by AWS services to encrypt data on your behalf.

AWS CloudHSM

What is AWS CloudHSM?

[AWS CloudHSM](https://aws.amazon.com/cloudhsm/) is a cloud-based hardware security module that provides secure cryptographic key storage and operations within a tamper-resistant hardware device. This service helps you meet corporate, contractual, and regulatory compliance requirements for data security by using FIPS 140-2 Level 3 validated HSMs.

II. AWS CloudHSM Features and Advantages

A. Secure Key Storage

AWS CloudHSM allows you to generate, store, and manage cryptographic keys securely.

B. FIPS 140-2 Level 3 Compliance

The service meets [high compliance standards](https://en.wikipedia.org/wiki/FIPS_140-2), ensuring your data’s security.

C. Scalability

AWS CloudHSM is fully scalable, allowing you to increase capacity as needed.

D. Flexibility

It supports multiple APIs, including [PKCS#11](https://www.ibm.com/docs/en/linux-on-systems?topic=introduction-what-is-pkcs-11), [Java Cryptography Extensions (JCE)](https://www.oracle.com/java/technologies/javase-jce-all-downloads.html), and [Microsoft CryptoNG (CNG)](https://learn.microsoft.com/en-us/windows/win32/seccng/cng-portal) libraries.

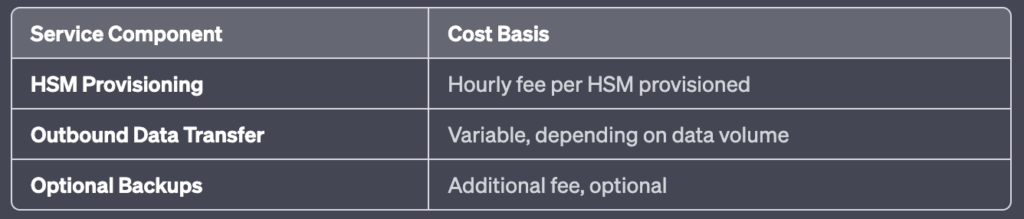
E. Control and Independence

Users have exclusive, single-tenant access to each HSM in their cluster.

III. AWS CloudHSM Monitoring

AWS CloudHSM integrates with [AWS CloudTrail](https://aws.amazon.com/cloudtrail/) and [Amazon CloudWatch](https://aws.amazon.com/cloudwatch/). CloudTrail records API calls for your account, while CloudWatch allows you to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources.

IV. AWS CloudHSM Pricing



Please note, the actual costs can vary based on different regions and specific usage patterns. Always refer to the official [AWS Pricing page](https://aws.amazon.com/pricing/?nc2=h_ql_pr_ln&aws-products-pricing.sort-by=item.additionalFields.productNameLowercase&aws-products-pricing.sort-order=asc&awsf.Free%20Tier%20Type=*all&awsf.tech-category=*all) for the most accurate and up-to-date information.

V. AWS CloudHSM Use Cases

A. Protect Private Keys for an Issuing CA

AWS CloudHSM can securely generate and store the private key of a root or intermediate CA, reducing the risk of unauthorized access.

B. Offload SSL Processing for Web Servers

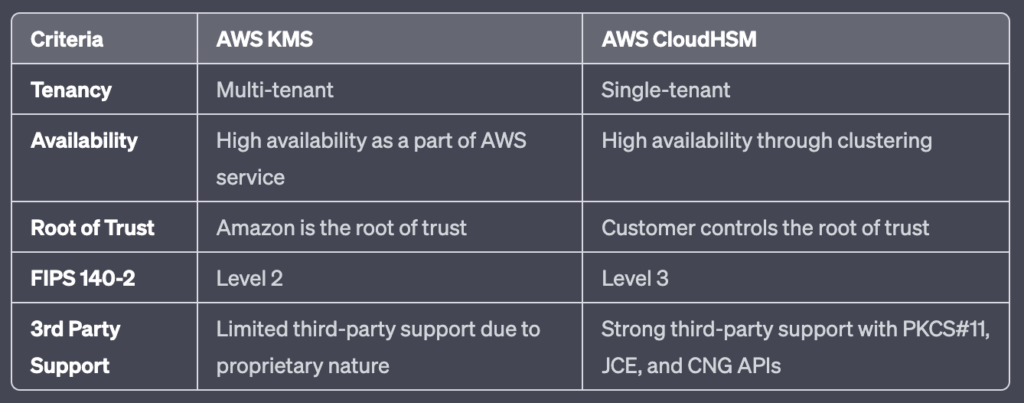
The service can offload SSL/TLS processing from web servers, improving website performance.

C. Encrypt Data at Rest

AWS CloudHSM enables the encryption of data at rest, providing an additional layer of protection for sensitive data.

VI. AWS CloudHSM FAQs

A. What is the difference between AWS KMS and AWS CloudHSM?



B. When should I use AWS CloudHSM instead of AWS KMS?

AWS CloudHSM is recommended when you need to control and manage your own encryption keys in single-tenant hardware, or when you have compliance requirements that cannot be met by AWS KMS.

C. What level of compliance is CloudHSM?

AWS CloudHSM is compliant with FIPS 140-2 Level 3, one of the highest levels of compliance for data security. This signifies that it meets stringent requirements for physical tamper-resistance, role-based access control, and ability to zeroize all plaintext cryptographic keys and critical security parameters within the module. This level of certification provides assurance that CloudHSM is designed to prevent unauthorized physical and logical access to cryptographic keys, making it suitable for handling sensitive data.

VII. Conclusion

AWS CloudHSM offers a robust platform that enables organizations to generate, store, and manage encryption keys securely in the cloud. The service’s flexibility comes from its support for various APIs and its scalability from the ability to increase capacity on-demand. With its high compliance standards, including FIPS 140-2 Level 3, it ensures stringent security for cryptographic operations. As such, businesses seeking a solution for securely managing cryptographic keys, while maintaining control and meeting compliance requirements, will find AWS CloudHSM to be a highly suitable service.

AWS WAF & Shield

AWS WAF and AWS Shield help protect your AWS resources from web exploits and DDoS attacks.

AWS WAF is a web application firewall service that helps protect your web apps from common exploits that could affect app availability, compromise security, or consume excessive resources.

AWS Shield provides expanded DDoS attack protection for your AWS resources. Get 24/7 support from our DDoS response team and detailed visibility into DDoS events.

We’ll now go into more detail on each service.

AWS Web Application Firewall (WAF)

AWS WAF is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources.

AWS WAF helps protect web applications from attacks by allowing you to configure rules that allow, block, or monitor (count) web requests based on conditions that you define.

These conditions include IP addresses, HTTP headers, HTTP body, URI strings, SQL injection and cross-site scripting.

Can allow or block web requests based on strings that appear in the requests using string match conditions.

For example, AWS WAF can match values in the following request parts:

Header – A specified request header, for example, the User-Agent or Referer header.

HTTP method – The HTTP method, which indicates the type of operation that the request is asking the origin to perform. CloudFront supports the following methods: DELETE, GET, HEAD, OPTIONS, PATCH, POST, and PUT.

Query string – The part of a URL that appears after a ? character, if any.

URI – The URI path of the request, which identifies the resource, for example, /images/daily-ad.jpg.

Body – The part of a request that contains any additional data that you want to send to your web server as the HTTP request body, such as data from a form.

Single query parameter (value only) – Any parameter that you have defined as part of the query string.

All query parameters (values only) – As above buy inspects all parameters within the query string.

New rules can be deployed within minutes, letting you respond quickly to changing traffic patterns.

When AWS services receive requests for web sites, the requests are forwarded to AWS WAF for inspection against defined rules.

Once a request meets a condition defined in the rules, AWS WAF instructs the underlying service to either block or allow the request based on the action you define.

With AWS WAF you pay only for what you use.

AWS WAF pricing is based on how many rules you deploy and how many web requests your web application receives.

There are no upfront commitments.

AWS WAF is tightly integrated with Amazon CloudFront and the Application Load Balancer (ALB), services.

When you use AWS WAF on Amazon CloudFront, rules run in all AWS Edge Locations, located around the world close to end users.

This means security doesn’t come at the expense of performance.

Blocked requests are stopped before they reach your web servers.

When you use AWS WAF on an Application Load Balancer, your rules run in region and can be used to protect internet-facing as well as internal load balancers.

Web Traffic Filtering

AWS WAF lets you create rules to filter web traffic based on conditions that include IP addresses, HTTP headers and body, or custom URIs.

This gives you an additional layer of protection from web attacks that attempt to exploit vulnerabilities in custom or third-party web applications.

In addition, AWS WAF makes it easy to create rules that block common web exploits like SQL injection and cross site scripting.

AWS WAF allows you to create a centralized set of rules that you can deploy across multiple websites.

This means that in an environment with many websites and web applications you can create a single set of rules that you can reuse across applications rather than recreating that rule on every application you want to protect.

Full feature API

AWS WAF can be completely administered via APIs.

This provides organizations with the ability to create and maintain rules automatically and incorporate them into the development and design process.

For example, a developer who has detailed knowledge of the web application could create a security rule as part of the deployment process.

This capability to incorporate security into your development process avoids the need for complex handoffs between application and security teams to make sure rules are kept up to date.

AWS WAF can also be deployed and provisioned automatically with AWS CloudFormation sample templates that allow you to describe all security rules you would like to deploy for your web applications delivered by Amazon CloudFront.

AWS WAF is integrated with Amazon CloudFront, which supports custom origins outside of AWS – this means you can protect web sites not hosted in AWS.

Support for IPv6 allows the AWS WAF to inspect HTTP/S requests coming from both IPv6 and IPv4 addresses.

Real-time visibility

AWS WAF provides real-time metrics and captures raw requests that include details about IP addresses, geo locations, URIs, User-Agent and Referers.

AWS WAF is fully integrated with Amazon CloudWatch, making it easy to setup custom alarms when thresholds are exceeded, or attacks occur.

This information provides valuable intelligence that can be used to create new rules to better protect applications.

AWS Shield

AWS Shield is a managed Distributed Denial of Service (DDoS) protection service that safeguards applications running on AWS.

AWS Shield provides always-on detection and automatic inline mitigations that minimize application downtime and latency, so there is no need to engage AWS Support to benefit from DDoS protection.

There are two tiers of AWS Shield – Standard and Advanced.

AWS Shield Standard

All AWS customers benefit from the automatic protections of AWS Shield Standard, at no additional charge.

AWS Shield Standard defends against most common, frequently occurring network and transport layer DDoS attacks that target web sites or applications.

When using AWS Shield Standard with Amazon CloudFront and Amazon Route 53, you receive comprehensive availability protection against all known infrastructure (Layer 3 and 4) attacks.

AWS Shield Advanced

Provides higher levels of protection against attacks targeting applications running on Amazon Elastic Compute Cloud (EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator and Amazon Route 53 resources.

In addition to the network and transport layer protections that come with Standard, AWS Shield Advanced provides additional detection and mitigation against large and sophisticated DDoS attacks, near real-time visibility into attacks, and integration with AWS WAF, a web application firewall.

AWS Shield Advanced also gives you 24×7 access to the AWS DDoS Response Team (DRT) and protection against DDoS related spikes in your Amazon Elastic Compute Cloud (EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator and Amazon Route 53 charges.

AWS Shield Advanced is available globally on all Amazon CloudFront, AWS Global Accelerator, and Amazon Route 53 edge locations.

Origin servers can be Amazon S3, Amazon Elastic Compute Cloud (EC2), Elastic Load Balancing (ELB), or a custom server outside of AWS.

AWS Shield Advanced includes DDoS cost protection, a safeguard from scaling charges because of a DDoS attack that causes usage spikes on protected Amazon EC2, Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator, or Amazon Route 53.

If any of the AWS Shield Advanced protected resources scale up in response to a DDoS attack, you can request credits via the regular AWS Support channel.

AWS Migration Services

AWS Migration Hub provides a single location to track the progress of application migrations across multiple AWS and partner solutions.

AWS Migration Hub allows you to either import information about on-premises servers and applications, or to perform a deeper discovery using our AWS Discovery Agent or AWS Discovery Collector, an agentless approach for VMware environments.

AWS Migration Hub network visualization allows you to accelerate migration planning by quickly identifying servers and their dependencies, identifying the role of a server, and grouping servers into applications.

To use network visualization, first install AWS Discovery agents and start data collection from the Data Collectors page.

AWS Migration Hub provides all application details in a central location.

This allows you to track the status of all the moving parts across all migrations, making it easier to view overall migration progress and reducing the time spent determining current status and next steps.

AWS Migration Hub lets you track the status of your migrations into any AWS region supported by your migration tools.

Regardless of which regions you migrate into, the migration status will appear in Migration Hub when using an integrated tool

Application Discovery Service

AWS Application Discovery Service helps enterprise customers plan migration projects by gathering information about their on-premises data centers.

Planning data center migrations can involve thousands of workloads that are often deeply interdependent.

Server utilization data and dependency mapping are important early first steps in the migration process.

AWS Application Discovery Service collects and presents configuration, usage, and behavior data from your servers to help you better understand your workloads.

The collected data is retained in encrypted format in an AWS Application Discovery Service data store.

You can export this data as a CSV file and use it to estimate the Total Cost of Ownership (TCO) of running on AWS and to plan your migration to AWS.

In addition, this data is also available in AWS Migration Hub, where you can migrate the discovered servers and track their progress as they get migrated to AWS.

Database Migration Service

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.

Supported migration paths include:

On-premises and EC2 databases to Amazon RDS or Amazon Aurora.

Homogeneous migrations such as Oracle to Oracle.

Heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora.

For the full list of supported sources click [here](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.html).

For the full list of supported targets click [here](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Target.html).

With AWS Database Migration Service, you can continuously replicate your data with high availability and consolidate databases into a petabyte-scale data warehouse by streaming data to Amazon Redshift and Amazon S3.

Use along with the Schema Conversion Tool (SCT) to migrate databases to AWS RDS or EC2-based databases.

The AWS Database Migration Service can migrate your data to and from most widely used commercial and open-source databases.

Schema Conversion Tool can copy database schemas for homogenous migrations (same database) and convert schemas for heterogeneous migrations (different database).

DMS is used for smaller, simpler conversions and supports MongoDB and DynamoDB.

SCT is used for larger, more complex datasets like data warehouses.

DMS has replication functions for on-premises to AWS or to Snowball or S3.

Server Migration Service

AWS Server Migration Service (SMS) is an agentless service which makes it easier and faster for you to migrate thousands of on-premises workloads to AWS.

AWS SMS allows you to automate, schedule, and track incremental replications of live server volumes, making it easier for you to coordinate large-scale server migrations.

AWS Transfer Family

The AWS Transfer Family provides fully managed support for file transfers directly into and out of Amazon S3 or Amazon EFS.

With support for Secure File Transfer Protocol (SFTP), File Transfer Protocol over SSL (FTPS), and File Transfer Protocol (FTP), the AWS Transfer Family helps you seamlessly migrate your file transfer workflows to AWS by integrating with existing authentication systems and providing DNS routing with Amazon Route 53, so nothing changes for your customers and partners, or their applications.

With your data in Amazon S3 or Amazon EFS, you can use it with AWS services for processing, analytics, machine learning, archiving, as well as home directories and developer tools.

AWS DataSync

AWS DataSync is an online data transfer service that simplifies, automates, and accelerates moving data between on-premises storage systems and AWS Storage services, as well as between AWS Storage services.

You can use DataSync to migrate active datasets to AWS, archive data to free up on-premises storage capacity, replicate data to AWS for business continuity, or transfer data to the cloud for analysis and processing.

DataSync provides built-in security capabilities such as encryption of data in-transit, and data integrity verification in-transit and at-rest.

It optimizes use of network bandwidth, and automatically recovers from network connectivity failures.

In addition, DataSync provides control and monitoring capabilities such as data transfer scheduling and granular visibility into the transfer process through Amazon CloudWatch metrics, logs, and events.

DataSync can copy data between Network File System (NFS) shares, Server Message Block (SMB) shares, self-managed object storage, AWS Snowcone, Amazon Simple Storage Service (Amazon S3) buckets, Amazon Elastic File System (Amazon EFS) file systems, and Amazon FSx for Windows File Server file systems.

AWS CloudFormation

AWS CloudFormation is a service that allows you to manage, configure and provision your AWS infrastructure as code.

AWS CloudFormation provides a common language for you to describe and provision all the infrastructure resources in your cloud environment.

Resources are defined using a CloudFormation template.

CloudFormation interprets the template and makes the appropriate API calls to create the resources you have defined.

Supports YAML or JSON.

CloudFormation can be used to provision a broad range of AWS resources.

Think of CloudFormation as deploying infrastructure as code.

CloudFormation has some similarities with AWS Elastic Beanstalk though they are also quite different as detailed in the table below:

|  |  |
| --- | --- |
| CloudFormation | Elastic Beanstalk |
| “Template-driven provisioning” | “Web apps made easy” |
| Deploys infrastructure using code | Deploys applications on EC2 (PaaS) |
| Can be used to deploy almost any AWS service | Deploys web applications based on Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker |
| Uses JSON or YAML template files | Uses ZIP or WAR files |
| Similar to Terraform | Similar to Google App Engine |

Key Benefits

Infrastructure is provisioned consistently, with fewer mistakes (human error).

Less time and effort than configuring resources manually.

You can use version control and peer review for your CloudFormation templates.

Free to use (you’re only charged for the resources provisioned).

It can be used to manage updates and dependencies.

It can be used to rollback and delete the entire stack as well.

Key Concepts

The following table describes the key concepts associated with AWS CloudFormation:

|  |  |
| --- | --- |
| Component | Description |
| Templates | The JSON or YAML text file that contains the instructions for building out the AWS environment |
| Stacks | The entire environment described by the template and created, updated, and deleted as a single unit |
| StackSets | AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation |
| Change Sets | A summary of proposed changes to your stack that will allow you to see how those changes might impact your existing resources before implementing them |
| Templates | The JSON or YAML text file that contains the instructions for building out the AWS environment |

Templates

A template is a YAML or JSON template used to describe the end-state of the infrastructure you are either provisioning or changing.

After creating the template, you upload it to CloudFormation directly or using Amazon S3.

CloudFormation reads the template and makes the API calls on your behalf.

The resulting resources are called a “Stack”.

Logical IDs are used to reference resources within the template.

Physical IDs identify resources outside of AWS CloudFormation templates, but only after the resources have been created.

Template elements

Mandatory:

List of resources and associated configuration values.

Not mandatory:

Template parameters (limited to 60).

Output values (limited to 60).

List of data tables.

Template components

Resources – the required Resources section declares the AWS resources that you want to include in the stack, such as an Amazon EC2 instance or an Amazon S3 bucket.

Mandatory.

Represent AWS components that will be created.

Resources are declared and can reference each other.

The following example YAML code declares an EC2 instance as a resource:

Resources:

MyEC2Instance:

Type: "AWS::EC2::Instance"

Properties:

ImageId: "ami-0ff8a91507f77f867"

Parameters

Use the optional Parameters section to customize your templates. Parameters enable you to input custom values to your template each time you create or update a stack.

Provide inputs to your CloudFormation template.

Useful for template reuse.

The following example declares a parameter named InstanceTypeParameter. This parameter lets you specify the Amazon EC2 instance type for the stack to use when you create or update the stack.

Note: the InstanceTypeParameter has a default value of t2.micro. This is the value that AWS CloudFormation uses to provision the stack unless another value is provided.

Parameters:

InstanceTypeParameter:

Type: String

Default: t2.micro

AllowedValues:

- t2.micro

- m1.small

- m1.large

Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.

Pseudo Parameters

Pseudo parameters are parameters that are predefined by AWS CloudFormation. You do not declare them in your template. Use them the same way as you would a parameter, as the argument for the Ref function.

Examples include:

AWS::AccountId – Returns the AWS account ID of the account in which the stack is being created.

AWS::NotificationARNs – Returns the list of notification Amazon Resource Names (ARNs) for the current stack.

AWS::Region – Returns a string representing the AWS Region in which the encompassing resource is being created.

AWS::StackId – Returns the ID of the stack as specified with the aws cloudformation create-stack command.

Mappings

The optional Mappings section matches a key to a corresponding set of named values.

Fixed variables.

Good for differentiating between regions, environments, AMIs etc.

Need to know the values in advance.

For user-specific values use parameters instead.

The following example has region keys that are mapped to two sets of values: one named HVM64 and the other HVMG2.

RegionMap:

    us-east-1:

      HVM64: ami-0ff8a91507f77f867

      HVMG2: ami-0a584ac55a7631c0c

    us-west-1:

      HVM64: ami-0bdb828fd58c52235

      HVMG2: ami-066ee5fd4a9ef77f1

Exam tip: with mappings you can, for example, set values based on a region. You can create a mapping that uses the region name as a key and contains the values you want to specify for each specific region.

Outputs

The optional Outputs section declares output values that you can import into other stacks (to create cross-stack references), return in response (to describe stack calls), or view on the AWS CloudFormation console.

Outputs can be imported into other stacks.

Can view the outputs in the console or using the AWS CLI.

Cannot delete a Stack if its outputs are being referenced by another CloudFormation Stack.

In the following example YAML code, the output named StackVPC returns the ID of a VPC, and then exports the value for cross-stack referencing with the name VPCID appended to the stack’s name

Outputs:

  StackVPC:

    Description: The ID of the VPC

    Value: !Ref MyVPC

    Export:

      Name: !Sub "${AWS::StackName}-VPCID"

Conditions

The optional Conditions section contains statements that define the circumstances under which entities are created or configured.

Control the creation of resources based on a condition.

Applied to resources and outputs.

In the sample YAML code below, resources are created only if the EnvType parameter is equal to prod:

Conditions:

  CreateProdResources: !Equals [ !Ref EnvType, prod ]

Transform

The optional Transform section specifies one or more macros that AWS CloudFormation uses to process your template.

The transform section can be used to reference additional code stored in S3, such as Lambda code or reusable snippets of CloudFormation code.

The AWS::Serverless transform, which is a macro hosted by AWS CloudFormation, takes an entire template written in the AWS Serverless Application Model (AWS SAM) syntax and transforms and expands it into a compliant AWS CloudFormation template.

In the following example, the template uses AWS SAM syntax to simplify the declaration of a Lambda function and its execution role:

Transform: AWS::Serverless-2016-10-31

Resources:

  MyServerlessFunctionLogicalID:

    Type: AWS::Serverless::Function

    Properties:

      Handler: index.handler

      Runtime: nodejs8.10

      CodeUri: 's3://testBucket/mySourceCode.zip'

Intrinsic Functions

AWS CloudFormation provides several built-in functions that help you manage your stacks. Use intrinsic functions in your templates to assign values to properties that are not available until runtime.

EXAM TIP: At a minimum, know the intrinsic functions listed below for the exam. The full list can be found at: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html

Ref

Fn::Ref (or !Ref in YAML),

The intrinsic function Ref returns the value of the specified parameter or resource.

When you specify a parameter’s logical name, it returns the value of the parameter.

When you specify a resource’s logical name, it returns a value that you can typically use to refer to that resource, such as a physical ID.

The following resource declaration for an Elastic IP address needs the instance ID of an EC2 instance and uses the Ref function to specify the instance ID of the MyEC2Instance resource:

MyEIP:

  Type: "AWS::EC2::EIP"

  Properties:

    InstanceId: !Ref MyEC2Instance

Fn::GetAtt

The Fn::GetAtt intrinsic function returns the value of an attribute from a resource in the template.

Full syntax (YAML): Fn::GetAtt: [ logicalNameOfResource, attributeName ]

Short form (YAML): !GetAtt logicalNameOfResource.attributeName

The following example template returns the SourceSecurityGroup.OwnerAlias and SourceSecurityGroup.GroupName of the load balancer with the logical name myELB.

AWSTemplateFormatVersion: 2010-09-09

Resources:

  myELB:

    Type: AWS::ElasticLoadBalancing::LoadBalancer

    Properties:

      AvailabilityZones:

        - eu-west-1a

      Listeners:

        - LoadBalancerPort: '80'

          InstancePort: '80'

          Protocol: HTTP

  myELBIngressGroup:

    Type: AWS::EC2::SecurityGroup

    Properties:

      GroupDescription: ELB ingress group

      SecurityGroupIngress:

        - IpProtocol: tcp

          FromPort: '80'

          ToPort: '80'

          SourceSecurityGroupOwnerId: !GetAtt myELB.SourceSecurityGroup.OwnerAlias

          SourceSecurityGroupName: !GetAtt myELB.SourceSecurityGroup.GroupName

Fn::FindInMap

The intrinsic function Fn::FindInMap returns the value corresponding to keys in a two-level map that is declared in the Mappings section.

Full syntax (YAML): Fn::FindInMap: [ MapName, TopLevelKey, SecondLevelKey ]

Short form (YAML): !FindInMap [ MapName, TopLevelKey, SecondLevelKey ]

The following example shows how to use Fn::FindInMap for a template with a Mappings section that contains a single map, RegionMap, that associates AMIs with AWS regions:

Mappings:

  RegionMap:

    us-east-1:

      HVM64: "ami-0ff8a91507f77f867"

      HVMG2: "ami-0a584ac55a7631c0c"

    us-west-1:

      HVM64: "ami-0bdb828fd58c52235"

      HVMG2: "ami-066ee5fd4a9ef77f1"

Resources:

  myEC2Instance:

    Type: "AWS::EC2::Instance"

    Properties:

      ImageId: !FindInMap

        - RegionMap

        - !Ref 'AWS::Region'

        - HVM64

      InstanceType: m1.small

Fn::ImportValue

The intrinsic function Fn::ImportValue returns the value of an output exported by another stack.

You typically use this function to create cross-stack references.

Fn::ImportValue:

  !Sub "${NetworkStackName}-SecurityGroupID"

Fn::Join

Full syntax (YAML): Fn::Join: [ delimiter, [ comma-delimited list of values ] ]

Short form (YAML): !Join [ delimiter, [ comma-delimited list of values ] ]

The following example uses Fn::Join to construct a string value. It uses the Ref function with the Partition parameter and the AWS::AccountId pseudo parameter.

!Join

  - ''

  - - 'arn:'

    - !Ref Partition

    - ':s3:::elasticbeanstalk-\*-'

    - !Ref 'AWS::AccountId'

Fn::Sub

The intrinsic function Fn::Sub substitutes variables in an input string with values that you specify.

In your templates, you can use this function to construct commands or outputs that include values that aren’t available until you create or update a stack.

The following example uses a mapping to substitute the ${Domain} variable with the resulting value from the Ref function:

Name: !Sub

  - www.${Domain}

  - { Domain: !Ref RootDomainName }

Stacks and Stack Sets

Stacks

Deployed resources based on templates.

Create, update, and delete stacks using templates.

Deployed through the Management Console, CLI or APIs.

Stack creation errors:

Automatic rollback on error is enabled by default.

You will be charged for resources provisioned even if there is an error.

Updating stacks:

AWS CloudFormation provides two methods for updating stacks: direct update or creating and executing change sets.

When you directly update a stack, you submit changes and AWS CloudFormation immediately deploys them.

Use direct updates when you want to quickly deploy your updates.

With change sets, you can preview the changes AWS CloudFormation will make to your stack, and then decide whether to apply those changes.

Stack Sets

AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation.

Using an administrator account, you define and manage an AWS CloudFormation template, and use the template as the basis for provisioning stacks into selected target accounts across specified regions.

An administrator account is the AWS account in which you create stack sets.

A stack set is managed by signing in to the AWS administrator account in which it was created.

A target account is the account into which you create, update, or delete one or more stacks in your stack set.

Before you can use a stack set to create stacks in a target account, you must set up a trust relationship between the administrator and target accounts.

Nested Stacks

Nested stacks allow re-use of CloudFormation code for common use cases.

For example standard configuration for a load balancer, web server, application server etc.

Instead of copying out the code each time, create a standard template for each common use case and reference from within your CloudFormation template.

Best Practices

AWS provides Python “helper scripts” which can help you install software and start services on your EC2 instances.

Use CloudFormation to make changes to your landscape rather than going directly into the resources.

Make use of Change Sets to identify potential trouble spots in your updates.

Use Stack Policies to explicitly protect sensitive portions of your stack.

Use a version control system such as CodeCommit or GitHub to track changes to templates.

Serverless Application Model (SAM)

Use SAM for deploying serverless applications using CloudFormation.

SAM is an extension to CloudFormation used to define serverless applications.

Simplified syntax for defining serverless resources: APIs, Lambda Functions, DynamoDB Tables etc.

Use the SAM CLI to package your deployment code, upload it to S3 and deploy your serverless application.

User data with EC2

User data can be included in CloudFormation.

The script is passed into Fn::Base64

The user data script logs are stored in /var/log/cloud-init-output.log

Binary is available on Amazon EC2 at /opt/aws/bin/cfn-init

CloudFormation Helper Scripts

cfn-init:

The cfn-init helper script reads template metadata from the AWS::CloudFormation::Init key and acts accordingly to:

Fetch and parse metadata from AWS CloudFormation

Install packages

Write files to disk

Enable/disable and start/stop services

cfn-init does not require credentials, so you do not need to use the –access-key, –secret-key, –role, or –credential-file options.

Logs go to /var/log/cfn-init.log

cfn-signal:

The cfn-signal helper script signals AWS CloudFormation to indicate whether Amazon EC2 instances have been successfully created or updated.

If you install and configure software applications on instances, you can signal AWS CloudFormation when those software applications are ready.

You use the cfn-signal script in conjunction with a [CreationPolicy](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-creationpolicy.html) or an Auto Scaling group with a [WaitOnResourceSignals](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-updatepolicy.html) update policy.

When AWS CloudFormation creates or updates resources with those policies, it suspends work on the stack until the resource receives the requisite number of signals or until the timeout period is exceeded.

You can signal a creation policy ([CreationPolicy](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-creationpolicy.html)) or a wait condition handle ([WaitOnResourceSignals](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-updatepolicy.html)).

Troubleshooting errors:

Make sure the AMI has the CloudFormation helper scripts included.

Check that the cfn-init and cfn-signal commands have run successfully.

Verify internet connectivity.

Creation Policies and Wait Conditions

CreationPolicy attribute:

Use the CreationPolicy attribute when you want to wait on resource configuration actions before stack creation proceeds.

You can associate the CreationPolicy attribute with a resource to prevent its status from reaching create complete until AWS CloudFormation receives a specified number of success signals, or the timeout period is exceeded.

To signal a resource, you can use the [cfn-signal](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-signal.html) helper script or [SignalResource](https://docs.aws.amazon.com/AWSCloudFormation/latest/APIReference/API_SignalResource.html) API.

AWS CloudFormation publishes valid signals to the stack events so that you track the number of signals sent.

The following CloudFormation resources support creation policies:

[AWS::AutoScaling::AutoScalingGroup](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html)

[AWS::EC2::Instance](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html)

[AWS::CloudFormation::WaitCondition](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-waitcondition.html)

DeletionPolicy attribute:

With the DeletionPolicy attribute you can preserve or (in some cases) backup a resource when its stack is deleted.

You specify a DeletionPolicy attribute for each resource that you want to control.

If a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource by default.

DependsOn attribute:

With the DependsOn attribute you can specify that the creation of a specific resource follows another.

When you add a DependsOn attribute to a resource, that resource is created only after the creation of the resource specified in the DependsOn attribute.

WaitCondition:

Note: For Amazon EC2 and Auto Scaling resources, AWS recommends that you use a CreationPolicy attribute instead of wait conditions.

You can use a wait condition for situations like the following:

To coordinate stack resource creation with configuration actions that are external to the stack creation.

To track the status of a configuration process.

UpdatePolicy Attribute (WaitOnResourceSignals)

Use the UpdatePolicy attribute to specify how AWS CloudFormation handles updates to the following resources:

[AWS::AutoScaling::AutoScalingGroup](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html),

[AWS::ElastiCache::ReplicationGroup](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-elasticache-replicationgroup)

[AWS::Elasticsearch::Domain](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-elasticsearch-domain.html)

[AWS::Lambda::Alias](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-lambda-alias.html)

UpdateReplacePolicy attribute:

Use the UpdateReplacePolicy attribute to retain or (in some cases) backup the existing physical instance of a resource when it is replaced during a stack update operation.

Rollbacks and Creation Failures

[Stack creation failures](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/troubleshooting.html):

By default everything will be deleted.

You can optionally disable rollback (good for troubleshooting failures).

Stack update failures:

The stack will automatically roll back to the previous known working state.

The logs can assist with understanding what issue occurred.

Monitoring and Reporting

You can monitor the progress of a stack update by viewing the stack’s events. The console’s Events tab displays each major step in the creation and update of the stack sorted by the time of each event with latest events on top.

For resources created by CloudFormation, use AWS monitoring and reporting tools applicable to the service.

Authorization and Access Control

You can use IAM with AWS CloudFormation to control what users can do with AWS CloudFormation, such as whether they can view stack templates, create stacks, or delete stacks.

In addition to AWS CloudFormation actions, you can manage what AWS services and resources are available to each user.

That way, you can control which resources users can access when they use AWS CloudFormation.

For example, you can specify which users can create Amazon EC2 instances, terminate database instances, or update VPCs. Those same permissions are applied anytime they use AWS CloudFormation to do those actions.

Charges

There is no additional charge for AWS CloudFormation.

You pay for AWS resources (such as Amazon EC2 instances, Elastic Load Balancing load balancers, etc.) created using AWS CloudFormation in the same manner as if you created them manually.

You only pay for what you use, as you use it; there are no minimum fees and no required upfront commitments.

AWS Elastic Beanstalk can be used to quickly deploy and manage applications in the AWS Cloud.

Developers upload applications and Elastic Beanstalk handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.

You can use multiple availability zones to improve application reliability and availability.

Considered a Platform as a Service (PaaS) solution.

Supports the following platforms:

[Docker](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.docker)

[Multicontainer Docker](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.mcdocker)

[Preconfigured Docker](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.dockerpreconfig)

[Go](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.go)

[Java SE](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.javase)

[Tomcat](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.java)

[.NET Core on Linux](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.dotnetlinux)

[.NET on Windows Server](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.net)

[Node.js](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.nodejs)

[PHP](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.PHP)

[Python](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.python)

[Ruby](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.ruby)

Developers can focus on writing code and don’t need to worry about deploying infrastructure.

You pay only for the resources provisioned, not for Elastic Beanstalk itself.

Elastic Beanstalk automatically scales your application up and down.

You can select the EC2 instance type that is optimal for your application.

Can retain full administrative control or have Elastic Beanstalk do it for you.

The Managed Platform Updates feature automatically applies updates for your operating system and platform.

Elastic Beanstalk monitors and manages application health and information is viewable via a dashboard.

Integrated with CloudWatch and X-Ray for performance data and metrics.

Integrates with Amazon VPC and AWS IAM.

Can provision most database instances.

Stores your application files and, optionally, server log files in Amazon S3.

Application data can also be stored on S3.

Multiple environments are supported to enable versioning.

Changes from Git repositories are replicated.

Linux and Windows AMI support.

Code is deployed using a WAR file or Git repository.

Can use the AWS toolkit for Visual Studio and the AWS toolkit for Eclipse to deploy Elastic Beanstalk.

Fault tolerance within a single region.

By default applications are publicly accessible.

Can access logs without logging into application servers.

Provides ISO, PCI, SOC 1, SOC 2, and SOC 3 compliance along with the criteria for HIPAA eligibility.

Supports AWS Graviton arm64-based processors.

Elastic Beanstalk Layers

There are several layers that make up Elastic Beanstalk and each layer is described below:

Application:

Within Elastic Beanstalk, an application is a collection of different elements, such as environments, environment configurations, and application versions.

You can have multiple application versions held within an application.

Application version:

An application version is a very specific reference to a section of deployable code.

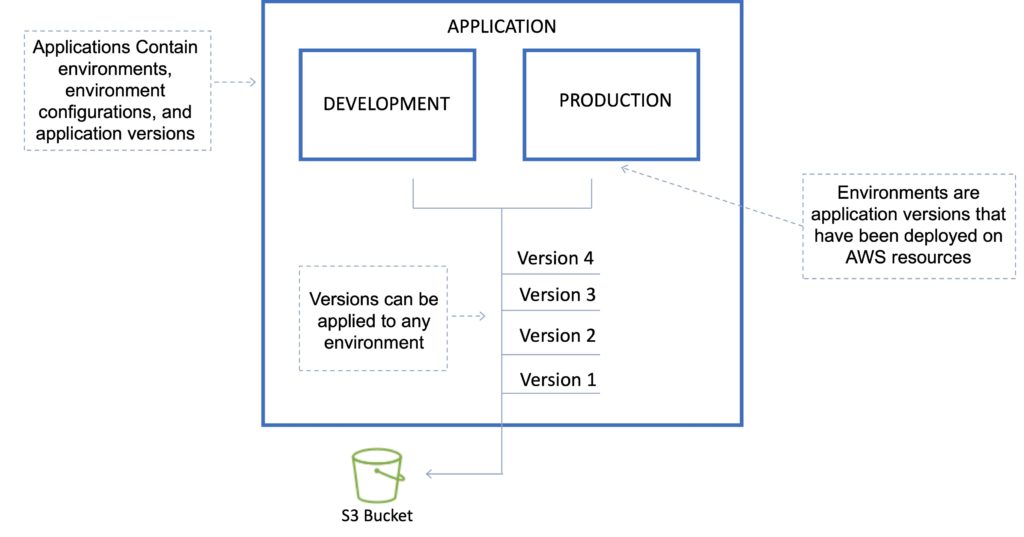
The application version will point typically to an Amazon s3 bucket containing the code.

Environment:

An environment refers to an application version that has been deployed on AWS resources.

The resources are configured and provisioned by AWS Elastic Beanstalk.

The environment is comprised of all the resources created by Elastic Beanstalk and not just an EC2 instance with your uploaded code.



Environment tier:

Determines how Elastic Beanstalk provisions resources based on what the application is designed to do.

Web servers are standard applications that listen for and then process HTTP requests, typically over port 80.

Workers are specialized applications that have a background processing task that listens for messages on an Amazon SQS queue.

Environment configurations:

An environment configuration is a collection of parameters and settings that dictate how an environment will have its resources provisioned by Elastic Beanstalk and how these resources will behave.

Configuration template:

This is a template that provides the baseline for creating a new, unique environment configuration.

Deployment Options

AWS Elastic Beanstalk provides several options for how [deployments](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html) are processed, including deployment policies and options that let you configure batch size and health check behavior during deployments.

Deployment options

Single instance: great for development.

High availability with load balancer: great for production.

Deployment policies

The deployment policies are: All at once, Rolling, Rolling with additional batch, and Immutable.

All at once:

Deploys the new version to all instances simultaneously.

All your instances are out of service while the deployment takes place.

Fastest deployment.

Good for quick iterations in the development environment.

You will experience an outage while the deployment is taking place – not ideal for mission-critical systems.

If the update fails, you need to roll back the changes by re-deploying the original version to all your instances.

No additional cost.

Rolling:

Update a few instances at a time (batch), and then move onto the next batch once the first batch is healthy (downtime for 1 batch at a time).

The application is running both versions simultaneously.

Each batch of instances is taken out of service while the deployment takes place.

Your environment capacity will be reduced by the number of instances in a batch while the deployment takes place.

Not ideal for performance-sensitive systems.

If the update fails, you need to perform an additional rolling update to roll back the changes.

No additional cost.

Long deployment time.

Rolling with additional batch:

Like Rolling but launches new instances in a batch ensuring that there is full availability.

The application is running at capacity.

You can set the bucket size.

The application is running both versions simultaneously.

Small additional cost.

Additional batch is removed at the end of the deployment.

Longer deployment.

Good for production environments.

Immutable:

Launches new instances in a new ASG and deploys the version update to these instances before swapping traffic to these instances once healthy.

Zero downtime.

New code is deployed to new instances using an ASG.

High cost as double the number of instances running during updates.

Longest deployment.

Quick rollback in case of failures.

Great for production environments.

Additionally, Elastic Beanstalk supports blue/green deployment.

Blue / Green deployment:

This is not a feature within Elastic Beanstalk

You create a new “staging” environment and deploy updates there.

The new environment (green) can be validated independently, and you can roll back if there are issues.

Route 53 can be set up using weighted policies to redirect a percentage of traffic to the staging environment.

Using Elastic Beanstalk, you can “swap URLs” when done with the environment test.

Zero downtime.

The following tables summarizes the different deployment policies:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Deployment Policy | Deploy Time | Zero Downtime | Rollback | Extra Cost | Reduction in capacity |
| All at once | Clock | NO | Manual redeploy | NONE | YES (total) |
| Rolling | Clock Clock | YES | Manual redeploy | NONE | YES (batch size) |
| Rolling with additional batch | Clock Clock Clock | YES | Manual redeploy | YES (batch size) | NO |
| Immutable | Clock Clock Clock Clock | YES | Terminate new instances | YES (total) | NO |
| Blue/green | Clock Clock Clock Clock | YES | Swap URL | YES (varies) | NO |

Golden AMIs

When deploying code to Amazon EC2 using Beanstalk, Elastic Beanstalk must resolve application dependencies which can take a long time.

A golden AMI is a method of reducing this time by packaging all dependencies, configuration, and software into the AMI before deploying.

Elastic Beanstalk CLI

There is an additional CLI called “eb cli”.

The EB CLI is a command-line interface for AWS Elastic Beanstalk that provides interactive commands that simplify creating, updating, and monitoring environments from a local repository.

You can use the EB CLI as part of your everyday development and testing cycle as an alternative to the Elastic Beanstalk console.

Lifecycle Policies

Elastic Beanstalk can store at most 1000 application versions.

To phase out old versions use a lifecycle policy:

Time-based – specify max age.

Count based – specify max number to retain.

Versions that are in use will not be deleted.

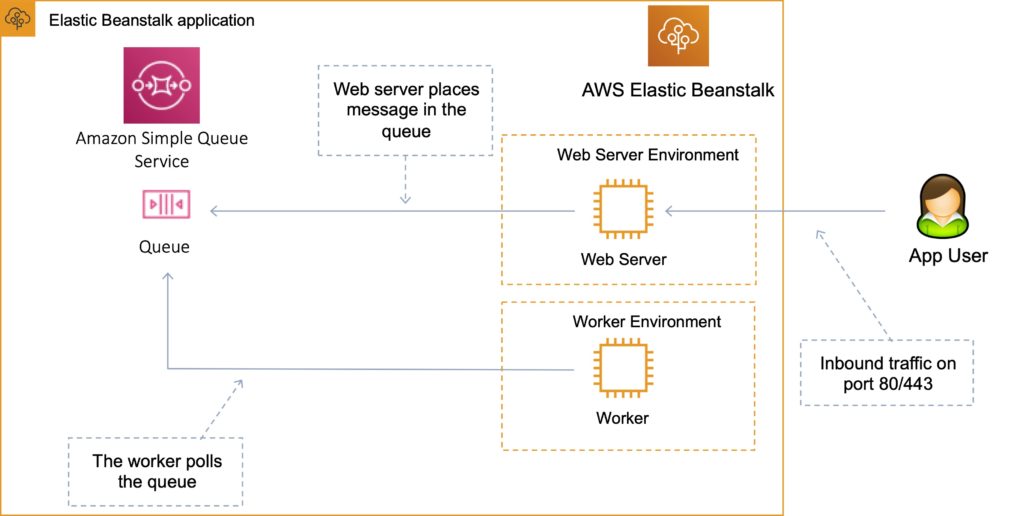
Option to not delete the source bundle in S3 to prevent data loss.

Worker environments

If an application performs tasks that take a long time to complete (long-running tasks), offload to a worker environment.

It allows you to decouple your application tiers.

Can define periodic tasks in the cron.yaml file.



Elastic Beanstalk Extensions

You can add AWS Elastic Beanstalk configuration files (.ebextensions) to your web application’s source code to configure your environment and customize the AWS resources that it contains.

Customization includes defining packages to install, create Linux users and groups, running shell commands, specifying services to enable, configuring a load balancer, etc.

Configuration files are YAML- or JSON-formatted documents with a .config file extension that you place in a folder named .ebextensions and deploy in your application source bundle.

The .ebextensions folder must be included in the top-level directory of your application source code bundle.

All the parameters set in the UI can be configured in the code.

Requirements:

Must be in the .ebextensions/ directory of the source code.

YAML or JSON format.

.config extensions can be included (e.g. logging.config).

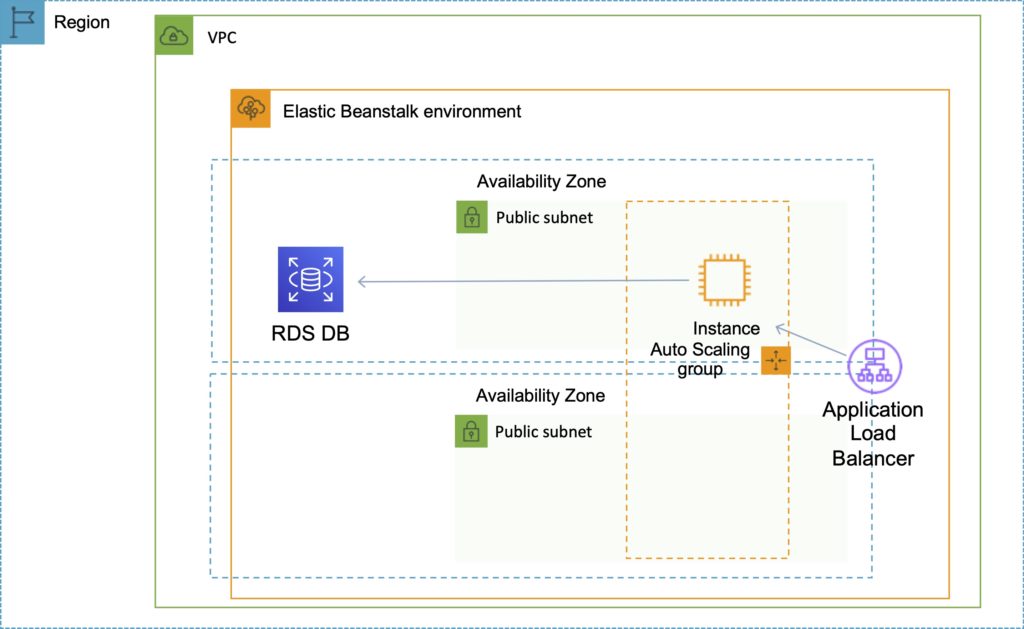
You can modify some default settings using “option\_settings”.

You can add resources such as RDS, ElastiCache, and DynamoDB.

Resources managed by .ebextensions get deleted if the environment is terminated.

Elastic Beanstalk with Amazon Relational Database Service (RDS)

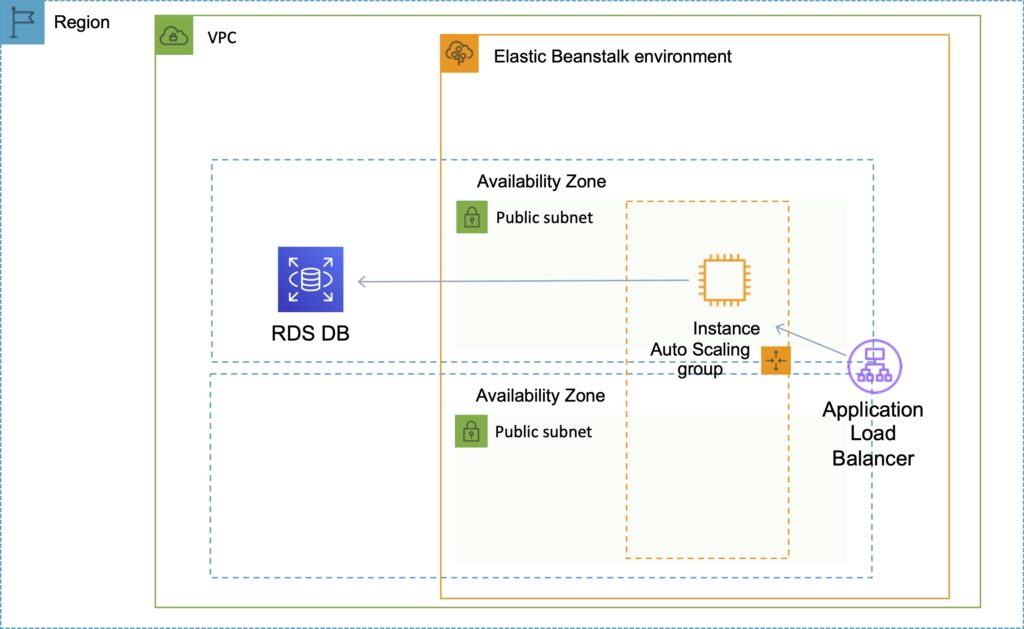
You can deploy Amazon RDS within an Elastic Beanstalk environment as in the diagram below:



However, if you terminate your Elastic Beanstalk environment you also lose the database.

The use case is only for development environments, typically not suitable for production.

For production, it is preferable to create the Amazon RDS database outside of Elastic Beanstalk as in the diagram below:



Steps to migrate from RDS within a Beanstalk environment to standalone RDS:

Take a snapshot of the RDS DB.

Enable deletion protection on the RDS DB.

Create a new environment without an RDS DB and point applications to the existing RDS DB.

Perform a blue/green deployment and swap the new and old environments.

Terminate the old environment (RDS will not be deleted due to termination protection).

Delete the CloudFormation stack (will be in the DELETE\_FAILED state).

Connecting to an Amazon RDS database

When the environment update is complete, the DB instance’s hostname and other connection information are available to your application through the following environment properties:

RDS\_HOSTNAME – The hostname of the DB instance.

RDS\_PORT – The port on which the DB instance accepts connections. The default value varies among DB engines.

RDS\_DB\_NAME – The database name, ebdb.

RDS\_USERNAME – The user name that you configured for your database.

RDS\_PASSWORD – The password that you configured for your database.

Custom Domain Names

If you’re using AWS Elastic Beanstalk to deploy and manage applications in the AWS Cloud, you can use Amazon Route 53 to route DNS traffic for your domain, such as example.com, to a new or an existing Elastic Beanstalk environment.

You create either a CNAME record or an alias record, depending on whether the domain name for the environment includes the Region, such as us-east-2, in which you deployed the environment. New environments include the Region in the domain name; environments that were created before early 2016 do not.

If the domain name does NOT include the Region: create a CNAME record.

If the domain name DOES include the Region: create an Alias record.

Security

Elastic Beanstalk works with HTTPS:

Load the SSL certificate onto the load balancer.

Can be performed from the console or in code (.ebextensions/securelistener-alb.config).

SSL certificate can be provisioned using ACM or CLI.

For redirecting HTTP to HTTPS:

Configure in the application.

Configure the ALB with a rule.

Ensure health checks are not redirected.

Monitoring and Reporting

Elastic Beanstalk automatically uses [Amazon CloudWatch](https://digitalcloud.training/amazon-cloudwatch/) to help you monitor your application and environment status.

You can navigate to the Amazon CloudWatch console to see your dashboard and get an overview of all your resources as well as your alarms.

You can also choose to view more metrics or add custom metrics.

Logging and Auditing

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.

In addition to instance logs, if you enable [enhanced health](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/health-enhanced.html) for your environment, you can configure the environment to stream health information to CloudWatch Logs.

Authorization and Access Control

AWS Elastic Beanstalk supports [identity-based](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.iam.html) policies.

AWS Elastic Beanstalk does not support [resource-based](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html) policies.

AWS Elastic Beanstalk has partial support for [resource-level](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html) permissions.

When you create an environment, AWS Elastic Beanstalk prompts you to provide two AWS Identity and Access Management (IAM) roles: a service role and an instance profile.

The [service role](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts-roles-service.html) is assumed by Elastic Beanstalk to use other AWS services on your behalf.

The [instance profile](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts-roles-instance.html) is applied to the instances in your environment and allows them to retrieve [application versions](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.html#concepts-version) from Amazon Simple Storage Service (Amazon S3), upload logs to Amazon S3, and perform other tasks that vary depending on the environment type and platform.

You can also create [user policies](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts-roles-user.html) and apply them to IAM users and groups in your account to allow users to create and manage Elastic Beanstalk applications and environments. Elastic Beanstalk provides managed policies for full access and read-only access.

AWS Config

AWS Config is a fully managed service that provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance.

With AWS Config you can discover existing AWS resources, export a complete inventory of your AWS resources with all configuration details, and determine how a resource was configured at any point in time.

These capabilities enable compliance auditing, security analysis, resource change tracking, and troubleshooting.

Allow you to assess, audit and evaluate configurations of your AWS resources.

Very useful for Configuration Management as part of an ITIL program.

Creates a baseline of various configuration settings and files and can then track variations against that baseline.

AWS Config vs CloudTrail

[AWS CloudTrail](https://digitalcloud.training/aws-cloudtrail/) records user API activity on your account and allows you to access information about this activity.

AWS Config records point-in-time configuration details for your AWS resources as Configuration Items (CIs).

You can use an AWS Config CI to answer, “What did my AWS resource look like?” at a point in time.

You can use AWS CloudTrail to answer, “Who made an API call to modify this resource?”.

AWS Config Rules

A Config Rule represents desired configurations for a resource and is evaluated against configuration changes on the relevant resources, as recorded by AWS Config.

[AWS Config Rules](https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config.html) can check resources for certain desired conditions and if violations are found the resources are flagged as “noncompliant”.

Examples of Config Rules:

Is backup enabled on [Amazon RDS](https://digitalcloud.training/amazon-rds/)?

Is CloudTrail enabled on the AWS account?

Are [Amazon EBS](https://digitalcloud.training/amazon-ebs/) volumes encrypted.

Configuration Items

A [Configuration Item (CI)](https://docs.aws.amazon.com/config/latest/developerguide/config-item-table.html) is the configuration of a resource at a given point-in-time. A CI consists of 5 sections:

Basic information about the resource that is common across different resource types (e.g., Amazon Resource Names, tags).

Configuration data specific to the resource (e.g., [Amazon EC2](https://digitalcloud.training/amazon-ec2/) instance type).

Map of relationships with other resources (e.g., EC2::Volume vol-3434df43 is “attached to instance” EC2 Instance i-3432ee3a).

AWS CloudTrail event IDs that are related to this state.

Metadata that helps you identify information about the CI, such as the version of this CI, and when this CI was captured.

Charges

With AWS Config, you are charged based on the number configuration items (CIs) recorded for supported resources in your AWS account.

AWS Config creates a configuration item whenever it detects a change to a resource type that it is recording.

AWS Resource Access Manager

AWS Resource Access Manager (RAM) is a service that enables you to share AWS resources easily and securely with any AWS account or within your AWS Organization.

You can share AWS Transit Gateways, Subnets, AWS License Manager configurations, and Amazon Route 53 Resolver rules resources with RAM.

RAM eliminates the need to create duplicate resources in multiple accounts, reducing the operational overhead of managing those resources in every single account you own.

You can create resources centrally in a multi-account environment, and use RAM to share those resources across accounts in three simple steps:

Create a Resource Share.

Specify resources.

Specify accounts.

RAM is available at no additional charge.

Key benefits:

Reduce Operational Overhead – Procure AWS resources centrally and use RAM to share resources such as subnets or License Manager configurations with other accounts. This eliminates the need to provision duplicate resources in every account in a multi-account environment.

Improve Security and Visibility – RAM leverages existing policies and permissions set in AWS Identity and Access Management (IAM) to govern the consumption of shared resources. RAM also provides comprehensive visibility into shared resources to set alarms and visualize logs through integration with Amazon CloudWatch and AWS CloudTrail.

Optimize Costs – Sharing resources such as AWS License Manager configurations across accounts allows you to leverage licenses in multiple parts of your company to increase utilization and optimize costs.

AWS Systems Manager

AWS Systems Manager is an AWS service that provides visibility and control of infrastructure on AWS.

AWS Systems Manager provides a unified interface through which you can view operational data from multiple AWS services.

AWS Systems Manager allows you to automate operational tasks across your AWS resources.

Formally known as SSM, AWS Systems Manager is a central hub to control and view your entire AWS infrastructure.

It can aid with security within your account and helps automate remedial tasks to ensure your environment is as compliant as possible.

AWS Systems manager is a powerful service which allows you to have a holistic view of all of the services you are using to view and control your infrastructure on AWS.

AWS Systems Manager provides a unified interface through which you can view operational data from multiple AWS services.

With AWS Systems Manager, you can group resources, like Amazon EC2 instances, Amazon S3 buckets, or Amazon RDS instances, by application, view operational data for monitoring and troubleshooting, and take action on your groups of resources.

With AWS Systems Manager, you can select a resource group and view its recent API activity, resource configuration changes, related notifications, operational alerts, software inventory, and patch compliance status.

You can also take action on each resource group depending on your operational needs.

AWS Systems Manager simplifies resource and application management, shortens the time to detect and resolve operational problems, and makes it easy to operate and manage your infrastructure securely at scale.

Systems Manager Components

AWS Systems Manager Inventory Manager Automates the process of collecting software inventory from managed instances.

You can simply specify the types of metadata to be collected, the instances from which the metadata should be gathered, and the schedule for collecting metadata.

AWS Systems Manager collects information about your instances and the software installed on them, helping you to understand your system configurations and installed applications.

The gathered data enables you to manage application assets, track licenses, monitor file integrity, discover applications not installed by a traditional installer, and more.

Configuration Compliance

AWS Systems Manager lets you scan your managed instances for patch compliance and configuration inconsistencies.

You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren’t compliant.

By default, AWS Systems Manager displays data about patching and associations. You can also customize the service and create your own compliance types based on your requirements.

AWS Systems Manager Automation

AWS Systems Manager allows you to safely automate common and repetitive IT operations and management tasks across AWS resources.

With Systems Manager, you can create JSON/YAML documents that specify a specific list of tasks or use community published documents.

These documents can be executed directly through the AWS Management Console, CLIs, and SDKs, scheduled in a maintenance window, or triggered based on changes to AWS resources through Amazon CloudWatch Events.

You can track the execution of each step in the documents as well as require approvals for each step.

You can also incrementally roll out changes and automatically halt when errors occur.

AWS Systems Manager Run Command

By using Run Command, you can automate common administrative tasks and make one-time configuration changes in bulk, and at enterprise scale.

This essentially prevents you from having to jump into every one of your instances and run the same command hundreds of times.

It is all actively managed within the console and there is an easy to use console in which to administer your commands, as well as a CLI or through the AWS SDK.

Example tasks include: stopping, restarting, terminating, and resizing instances. Attaching and detaching EBS volumes, creating snapshots and you can install or bootstrap an application, build a deployment pipeline etc.

Commands can be applied to a group of systems based on AWS instance tags or manual selection.

The commands and parameters are defined in an AWS Systems Manager document.

Commands can be issued using the AWS Console, AWS CLI, AWS Tools for Windows PowerShell, the AWS Systems Manager API, or Amazon SDKs.

AWS Systems Manager Session Manager

AWS Systems Manager provides safe; secure and remote management of your instances at scale without logging into your servers, replacing the need for bastion hosts, SSH, or remote PowerShell.

It provides a simple way of automating common administrative tasks across groups of instances such as registry edits, user management, and software and patch installations.

Session Manager provides a command terminal for Linux instances and Windows PowerShell terminal for Windows instances.

Session Manager not only provides a fully auditable terminal environment, but you can administer access to SSH sessions without having to open any ports, strongly enhancing your security posture. You can simply enable the least privileged IAM permissions to the Session Manager console and allow your developers to maximize their efficiency and effectiveness.

All actions taken with AWS Systems Manager are recorded by AWS CloudTrail, allowing you to audit changes throughout your environment.

CloudTrail can intercept StartSession events using Session Manager.

Compared to SSH:

• Do not need to open port 22.

• Do not need bastion hosts for management.

• All commands are logged to S3 / CloudWatch.

• Secure shell access is authenticated using IAM user accounts, not key pairs.

Incident Manager

You can automate solutions in the Incident Manager console to help bring the appropriate internal resources to your attention.

AWS Chatbot links designated chat channels to AWS Incident Manager, and Automation runbooks are executed for AWS Systems Manager using the Incident Manager.

Responders are engaged via SMS and phone calls in predefined response plans.

By suggesting action items based on Amazon’s post-incident analysis template, Incident Manager helps you improve service reliability. For example, you can automate a runbook step or add a new alarm after an incident.

Systems Manager Patch Manager

Patch Manager enables the automated patching your EC2 instances. It includes security patches, as well as other patches for both your applications and your operating systems.

Patch Manager uses what are known as patch baselines, which involve the definition of rules for auto-approving patches, as well as declined patches. By scheduling patching as a maintenance window task in Systems Manager, you can easily install patches on a regular basis.

Understanding the state of your servers that are part of your Systems Manager fleet is paramount to enabling a compliant and secure workload of your applications and your servers.

The process from end-to-end consists of obtaining metadata about your managed Systems Manager nodes (on-premise, EC2 etc.) Regardless of if you are using Systems Manager in the cloud or Systems Manager on-premise you store this metadata in a centralized repository (an S3 Bucket). You can from there query the metadata using native tools to gain insights into trends across your nodes in a number of different categories.

This aggregation of metadata can include multiple regions and multiple accounts, all in one place, and permissions to this sub-service can be granularly assigned using IAM as well as Service Control Policies for AWS Organizations.

AWS Systems Manager Parameter Store

AWS Systems Manager provides a centralized store to manage your configuration data, whether plain-text data such as database strings or secrets such as passwords.

This allows you to separate your secrets and configuration data from your code. Parameters can be tagged and organized into hierarchies, helping you manage parameters more easily.

For example, you can use the same parameter name, “db-string”, with a different hierarchical path, “dev/db-string” or “prod/db-string”, to store different values.

AWS Systems Manager is integrated with AWS Key Management Service (KMS), allowing you to automatically encrypt the data you store.

You are able to store these secrets in a way in which you don’t have to manage any servers, making the entire process much easier.

If your data is also kept separate from your code, you will be in a much better position security wise, as you are enabling separation.

You can also control user and resource access to parameters using AWS Identity and Access Management (IAM).

Parameters can be referenced through other AWS services, such as Amazon Elastic Container Service, AWS Lambda, and AWS CloudFormation.

AWS Systems Manager Distributor

Distributor is an AWS Systems Manager feature that enables you to securely store and distribute software packages in your organization.

You can use Distributor with existing Systems Manager features like Run Command and State Manager to control the lifecycle of the packages running on your instances.

AWS Systems Manager State manager

If you need to manage the state of your AWS EC2 resources, AWS Systems Manager State Manager enables you to maintain your instances in whichever state you like.

There are three steps in using Systems Manager State Manager:

1. Decide which state to apply to your resources

2. You may be able to create the desired state for your AWS resources with a pre-configured SSM document – you need to figure this out.

3. You then create an association between your instance and the state you have defined.

You can query AWS Systems Manager at any time to view the status of your instance configurations, giving you on-demand visibility into your compliance status.

AWS Systems Manager OpsCenter

If you have any operational issues related to your AWS Resources – you can use OpsCenter to provide a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources

AWS Systems Manager Maintenance Windows

AWS Systems Manager lets you schedule windows of time to run administrative and maintenance tasks across your instances.

This ensures that you can select a convenient and safe time to install patches and updates or make other configuration changes, improving the availability and reliability of your services and applications.

AWS Systems Manager Resource Groups

You can use resource groups to organize your AWS resources. Resource groups make it easier to manage, monitor, and automate tasks on large numbers of resources at one time.

AWS Resource Groups provides two general methods for defining a resource group. Both methods involve using a query to identify the members for a group.

The first method relies on tags applied to AWS resources to add resources to a group. Using this method, you apply the same key/value pair tags to resources of various types in your account and then use the AWS Resource Groups service to create a group based on that tag pair.

The second method is based on resources available in an individual AWS CloudFormation stack. Using this method, you choose an AWS CloudFormation stack, and then choose resource types in the stack that you want to be in the group.

AWS Systems Manager Resource Groups allows the creation of logical groups of resources that you can perform actions on (such as patching).

Resource groups are regional in scope.

Systems Manager Document

An AWS Systems Manager document (SSM document) defines the actions that Systems Manager performs on your managed instances.

Systems Manager includes more than a dozen pre-configured documents that you can use by specifying parameters at runtime.

Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify.

Monitoring and Reporting

Insights Dashboard

AWS Systems Manager automatically aggregates and displays operational data for each resource group through a dashboard.

Systems Manager eliminates the need for you to navigate across multiple AWS consoles to view your operational data.

With Systems Manager you can view API call logs from AWS CloudTrail, resource configuration changes from AWS Config, software inventory, and patch compliance status by resource group.

You can also easily integrate your AWS CloudWatch Dashboards, AWS Trusted Advisor notifications, and AWS Personal Health Dashboard performance and availability alerts into your Systems Manager dashboard.

Systems Manager centralizes all relevant operational data, so you can have a clear view of your infrastructure compliance and performance.

Amazon CloudWatch

You can configure and use the Amazon CloudWatch agent to collect metrics and logs from your instances instead of using SSM Agent for these tasks. The CloudWatch agent enables you to gather more metrics on EC2 instances than are available using SSM Agent. In addition, you can gather metrics from on-premises servers using the CloudWatch agent.

Logging and Auditing

Systems Manager is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Systems Manager. CloudTrail captures all API calls for Systems Manager as events, including calls from the Systems Manager console and from code calls to the Systems Manager APIs.

SSM Agent writes information about executions, commands, scheduled actions, errors, and health statuses to log files on each instance. You can view log files by manually connecting to an instance, or you can automatically send logs to Amazon CloudWatch Logs.

Authorization and Access Control

AWS Systems Manager supports identity-based policies.

AWS Systems Manager does not support resource-based policies.

You can attach tags to Systems Manager resources or pass tags in a request to Systems Manager.

To control access based on tags, you provide tag information in the condition element of a policy using the ssm:resourceTag/key-name, aws:ResourceTag/key-name, aws:RequestTag/key-name, or aws:TagKeys condition keys.

AWS Systems Manager Pricing

There is no charge for most components in Systems Manager, and you only pay for the resources that are managed as part of the systems manager service.

There are some exceptions however – OpsCenter and Parameter store cost a small amount of money.

The OpsCenter pricing is as follows:

|  |  |
| --- | --- |
| OpsCenter Details | Pricing |
| Number of OpsItems | $2.97 per 1,000 OpsItems |
| Get, Describe, Update, and GetOpsSummary API requests | $0.039 per 1,000 requests |
|  |  |

The Parameter store pricing is also as follows:

|  |  |
| --- | --- |
| Parameter type | Pricing |
| Standard | No additional charge |
| Advanced | $0.05 per advanced parameter per month (prorated hourly if the parameter is stored less than a month) |

AWS Systems Manager FAQ

• Who should use Systems Manager?

If you are using many different services across many different accounts, and you want to gain keen operational insight into your AWS Resources, you can use Systems Manager. If you are a SysOps administrator you can gain great benefits from using Systems Manager.

• Which operating system can Systems Manager support?

You can use Systems Manager with both Linux and Windows operating systems.

• Which Regions can i use Systems Manager in?

There is a Systems Manager Region table, linked here which you can use to check your region.

• How can i use Systems Manager?

You can use AWS Systems Manager with the CLI, SDK and the AWS Console.

Thanks for reading our breakdown of Systems Manager, and exploring how useful it is as a service.

What is a managed instance?

A managed instance is any on-premises server or any Amazon EC2 instance that can be managed using AWS Systems Manager.

AWS OpsWorks

AWS OpsWorks is a configuration management service that provides managed instances of these two open-source tools (Chef and Puppet).

With OpsWorks AWS manage and automate how your infrastructure configured, deployed, and managed.

The difference between AWS OpsWorks for Chef and Puppet is that Chef works with AWS OpsWorks Chef cookbooks and recipes, while AWS OpsWorks for Puppet Enterprise works with manifests and modules.

Recipes and manifests, as a rule, describe single concepts, while cookbooks and recipes describe more general concepts.

AWS OpsWorks monitoring consists of a direct integration between AWS OpsWorks and CloudWatch.

AWS OpsWorks Logs for use and access of the service are natively and securely shipped to AWS CloudTrail.

AWS OpsWorks Stacks

AWS OpsWorks Stacks is a configuration management service within the wider OpsWorks AWS tooling that enables you to automate any tasks within your environment.

AWS OpsWorks Auto Healing for OpsWorks Stacks also allows your environments to be automatically healed to ensure your environments are proactively healthy.

A good example of these tasks may be things like scaling up and down, setting up of your database clusters, and installing packages and dependencies within your virtual machines.

Linux and Windows workloads are supported by OpsWorks stacks, and it uses Chef Recipes to configure the sorts of tasks which are best suited to being automated.

Every OpsWorks stack is built of multiple layers – each representing a different stack component, such as an individual load balancer, a database, or a set of EC2 instances.

There are number of features of AWS OpsWorks Stacks:

EC2 instances can be deployed using template configurations, including EBS volumes.

The software on your instances can be configured on-demand or automatically based on events that occur over time, from bootstrapping a base OS image into a running server to modifying running services to reflect changes.

Stacks can be automatically healed by the AWS OpsWorks service. OpsWorks Stacks can replace a failed instance in your stack with a new one, ensuring maximum uptime and availability

AWS OpsWorks Stacks integrates perfectly with Amazon CloudTrail and send logs to the CloudTrail console without having to configure this yourself.

AWS OpsWorks for Chef Automate

Chef Automate is an enterprise level-platform that provides actionable insights with enterprise scale and performance across your cloud architecture.

AWS OpsWorks for Chef Automate is a managed way of launching a Chef Automate server in OpsWorks AWS.

Within AWS, you can run a Chef Automate server without having to worry about provisioning any hardware.

AWS OpsWorks Auto Healing for Chef Automate also allows your environments to be automatically healed to ensure your environments are proactively healthy.

Chef Automate handles its own operations, backups, restorations, and software upgrades also.

You can connect any on-premises computer or any EC2 instance that is running a supported operating system ([list here](http://man.hubwiz.com/docset/Chef.docset/Contents/Resources/Documents/docs.chef.io/platforms.html)) which has network access to an AWS OpsWorks for Chef Automate server.

You can configure automatic backups natively, and the backups are stored in S3.

Adding new nodes to your Chef server is as simple as inserting the user-data code provided by OpsWorks for Chef Automate into your Auto Scaling groups.

All Chef communication is encrypted using SSL to ensure that the server will communicate safely and securely with other aspects of your architecture.

You receive the full Chef Automate platform which includes premium features that you can use with Chef server, like Chef Workflow, Chef Visibility, and Chef Compliance, all within the same service.

OpsWorks for Chef Automate can also install updates automatically during a weekly maintenance window, one which you define.

You can launch a Chef Automate server through the console, using CloudFormation, or using the CLI or the SDK.

AWS OpsWorks for Puppet Enterprise

OpsWorks for Puppet Enterprise is another subset of OpsWorks which lets you launch a Puppet Enterprise master in minutes, compared to hours.

You can configure automatic backups natively, and the backups are stored in S3.

Adding new nodes to your Puppet Enterprise server is as simple as inserting the user-data code provided by AWS OpsWorks for Puppet Enterprise into your Auto Scaling groups.

Puppet Enterprise communication is encrypted using SSL to ensure that the server will communicate safely and securely with other aspects of your architecture.

Deleting a Puppet Enterprise server will also delete everything associated with it, including backups and all other supporting resources.

AWS OpsWorks for Puppet Enterprise can also install updates automatically during a weekly maintenance window, one which you define.

You can launch a Puppet Enterprise server through the console, using CloudFormation, or using the CLI or the SDK.

AWS OpsWorks Pricing

For OpsWorks Stacks, there is no additional charge. You pay for any resources you launch as part of the service, but nothing for OpsWorks Stacks.

On-premises deployments of OpsWorks stacks cost $0.02 per hour and with both deployments there are no minimum fees and no upfront commitments.

AWS OpsWorks for Chef Automate and for Puppet Enterprise are charged separately.

You are charged based on the number of nodes connected to your Chef Automate instance the times they are running for and the EC2 instance which is running.

The final price considers the number of nodes per month, and the number of node hours.

For both OpsWorks Puppet Enterprise and Chef Automate you benefit from 7,500 node hours free per month with the AWS Free Tier.

OpsWorks AWS Endpoints and Quotas

OpsWorks for Chef Automate and AWS OpsWorks for Puppet Enterprise isn’t currently available in all Regions – and only exists in nine regions currently ([list here](https://docs.aws.amazon.com/general/latest/gr/opsworks-service.html).)

OpsWorks Stacks exists in fifteen regions instead, and any resources cannot be managed from other regions.

Please see the table below for information on the service quotas for AWS OpsWorks stacks.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Default Limit (Per Region) | Hard / Soft Limit |
| Stacks | Number of Stacks | 40 | Soft |
| Layers per Stack | Multiple different stack components, such as an individual load balancer, a database, or a set of EC2 instances. | 40 | Soft |
| Instances per Stack | The maximum number of instances that you can have in an OpsWorks stack. | 40 | Soft |
| Apps per Stack | The maximum number of apps that you can have in an OpsWorks stack. | 40 | Soft |

Please see the table below for information on the service quotas for OpsWorks for Chef Automate and AWS OpsWorks for Puppet Enterprise.

|  |  |  |
| --- | --- | --- |
| Name | Default Limit (Per Region) | Hard / Soft Limit |
| Manual backups per server | 10 | Soft |
| Number of scheduled (automated) backups per server | 10 | Soft |
| Number of servers | 5 | Soft |

Amazon Route 53

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) service.

Amazon Route 53 offers the following functions:

Domain name registry.

DNS resolution.

Health checking of resources.

Route 53 can perform any combination of these functions.

Route 53 provides a worldwide distributed DNS service.

Route 53 is located alongside all edge locations.

Health checks verify Internet connected resources are reachable, available, and functional.

Route 53 can be used to route Internet traffic for domains registered with another domain registrar (any domain).

When you register a domain with Route 53 it becomes the authoritative DNS server for that domain and creates a public hosted zone.

To make Route 53 the authoritative DNS for an existing domain without transferring the domain create a Route 53 public hosted zone and change the DNS Name Servers on the existing provider to the Route 53 Name Servers.

Changes to Name Servers may not take effect for up to 48 hours due to the DNS record Time To Live (TTL) values.

You can transfer domains to Route 53 only if the Top-Level Domain (TLD) is supported.

You can transfer a domain from Route 53 to another registrar by contacting AWS support.

You can transfer a domain to another account in AWS however it does not migrate the hosted zone by default (optional).

It is possible to have the domain registered in one AWS account and the hosted zone in another AWS account.

Primarily uses UDP port 53 (can use TCP).

AWS offer a 100% uptime SLA for Route 53.

You can control management access to your Amazon Route 53 hosted zone by using IAM.

There is a default limit of 50 domain names, but this can be increased by contacting support.

Private DNS is a Route 53 feature that lets you have authoritative DNS within your VPCs without exposing your DNS records (including the name of the resource and its IP address(es) to the Internet.

You can use the AWS Management Console or API to register new domain names with Route 53.

[](https://www.adplugg.com/track/click/A48221584/60486/click?toi=21584&hn=digitalcloud.training&bu=%2Famazon-route-53%2F&rf=&zn=16420&pm=8397&ct=&next=)

Hosted Zones

A hosted zone is a collection of records for a specified domain.

A hosted zone is analogous to a traditional DNS zone file; it represents a collection of records that can be managed together.

There are two types of zones:

Public host zone – determines how traffic is routed on the Internet.

Private hosted zone for VPC – determines how traffic is routed within VPC (resources are not accessible outside the VPC).

Amazon Route 53 automatically creates the Name Server (NS) and Start of Authority (SOA) records for the hosted zones.

Amazon Route 53 creates a set of 4 unique name servers (a delegation set) within each hosted zone.

You can create multiple hosted zones with the same name and different records.

NS servers are specified by Fully Qualified Domain Name (FQDN), but you can get the IP addresses from the command line (e.g. dig or nslookup).

For private hosted zones you can see a list of VPCs in each region and must select one.

For private hosted zones you must set the following VPC settings to “true”:

enableDnsHostname.

enableDnsSupport.

You also need to create a DHCP options set.

You can extend an on-premises DNS to VPC.

You cannot extend Route 53 to on-premises instances.

You cannot automatically register EC2 instances with private hosted zones (would need to be scripted).

Health checks check the instance health by connecting to it.

Health checks can be pointed at:

Endpoints.

Status of other health checks.

Status of a CloudWatch alarm.

Endpoints can be IP addresses or domain names.

You can associate the Route 53 private hosted zone in one account with a VPC in another account.

To associate a Route 53 private hosted zone in one AWS account (Account A) with a virtual private cloud that belongs to another AWS account (Account B), follow these steps using the AWS CLI:

From an instance in Account A, authorize the association between the private hosted zone in Account A and the virtual private cloud in Account B.

From an instance in Account B, create the association between the private hosted zone in Account A and the virtual private cloud in Account B.

Delete the association authorization after the association is created.

Health Checks

Health checks check the instance health by connecting to it.

Health checks can be pointed at:

Endpoints.

Status of other health checks.

Status of a CloudWatch alarm.

Endpoints can be IP addresses or domain names.

You can create the following types of health checks:

HTTP: Route 53 tries to establish a TCP connection. If successful, Route 53 submits an HTTP request and waits for an HTTP status code of 200 or greater and less than 400.

HTTPS: Route 53 tries to establish a TCP connection. If successful, Route 53 submits an HTTPS request and waits for an HTTP status code of 200 or greater and less than 400.

HTTP\_STR\_MATCH: Route 53 tries to establish a TCP connection. If successful, Route 53 submits an HTTP request and searches the first 5,120 bytes of the response body for the string that you specify in SearchString.

HTTPS\_STR\_MATCH: Route 53 tries to establish a TCP connection. If successful, Route 53 submits an HTTPS request and searches the first 5,120 bytes of the response body for the string that you specify in SearchString.

TCP: Route 53 tries to establish a TCP connection.

CLOUDWATCH\_METRIC: The health check is associated with a CloudWatch alarm. If the state of the alarm is OK, the health check is considered healthy. If the state is ALARM, the health check is considered unhealthy. If CloudWatch doesn’t have sufficient data to determine whether the state is OK or ALARM, the health check status depends on the setting for InsufficientDataHealthStatus: Healthy, Unhealthy, or LastKnownStatus.

CALCULATED: For health checks that monitor the status of other health checks, Route 53 adds up the number of health checks that Route 53 health checkers consider to be healthy and compares that number with the value of HealthThreshold.

Records

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).

Alias (an Amazon Route 53-specific virtual record).

The Alias record is a Route 53 specific record type.

Alias records are used to map resource record sets in your hosted zone to Amazon Elastic Load Balancing load balancers, Amazon CloudFront distributions, AWS Elastic Beanstalk environments, or Amazon S3 buckets that are configured as websites.

You can use Alias records to map custom domain names (such as api.example.com) both to API Gateway custom regional APIs and edge-optimized APIs and to Amazon VPC interface endpoints.

The Alias is pointed to the DNS name of the service.

You cannot set the TTL for Alias records for ELB, S3, or Elastic Beanstalk environment (uses the service’s default).

Alias records work like a CNAME record in that you can map one DNS name (e.g. example.com) to another ‘target’ DNS name (e.g. elb1234.elb.amazonaws.com).

An Alias record can be used for resolving apex / naked domain names (e.g. example.com rather than sub.example.com).

A CNAME record can’t be used for resolving apex / naked domain names.

Generally use an Alias record where possible.

Route 53 supports wildcard entries for all record types, except NS records.

The following table details the differences between Alias and CNAME records:

|  |  |
| --- | --- |
| CNAME Records | Alias Records |
| Route 53 charges for CNAME queries | Route 53 doesn’t charge for alias queries to AWS resources |
| You can’t create a CNAME record at the top node of a DNS namespace (zone apex) | You can create an alias record at the zone apex (however you can’t route to a CNAME at the zone apex) |
| A CNAME record redirects queries for a domain name regardless of record type | Route 53 follows the pointer in an alias record only when the record type also matches |
| A CNAME can point to any DNS record that is hosted anywhere | An alias record can only point to a CloudFront distribution, Elastic Beanstalk environment, ELB, S3 bucket as a static website, or to another record in the same hosted zone that you’re creating the alias record in |
| A CNAME record is visible in the answer section of a reply from a Route 53 DNS server | An alias record is only visible in the Route 53 console or the Route 53 API |
| A CNAME record is followed by a recursive resolver | An alias record is only followed inside Route 53. This means that both the alias record and its target must exist in Route 53 |

Routing Policies

Routing policies determine how Route 53 responds to queries.

The following table highlights the key function of each type of routing policy:

|  |  |
| --- | --- |
| Policy | What it Does |
| Simple | Simple DNS response providing the IP address associated with a name |
| Failover | If primary is down (based on health checks), routes to secondary destination |
| Geolocation | Uses geographic location you’re in (e.g. Europe) to route you to the closest region |
| Geoproximity | Routes you to the closest region within a geographic area |
| Latency | Directs you based on the lowest latency route to resources |
| Multivalue answer | Returns several IP addresses and functions as a basic load balancer |
| Weighted | Uses the relative weights assigned to resources to determine which to route to |

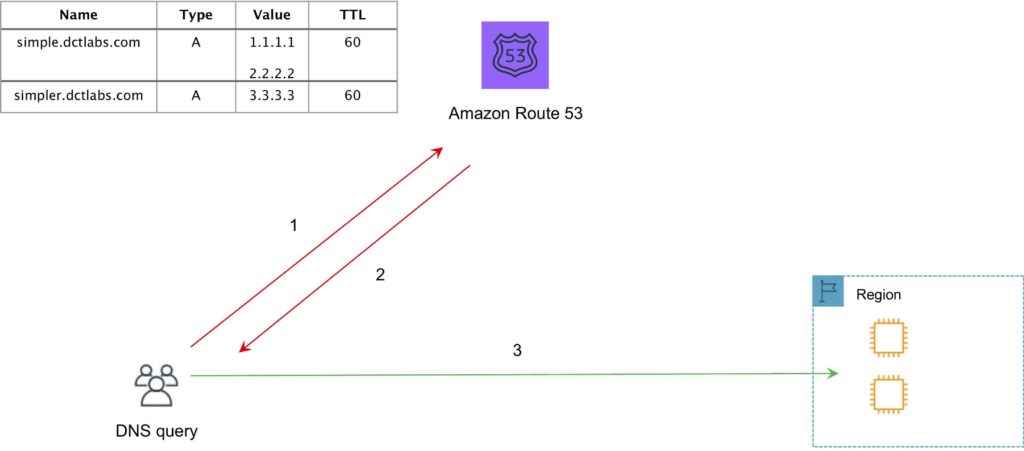
Simple Routing Policy

An A record is associated with one or more IP addresses.

Uses round robin.

Does not support health checks.

The following diagram depicts an Amazon Route 53 Simple routing policy configuration:



Failover:

Failover to a secondary IP address.

Associated with a health check.

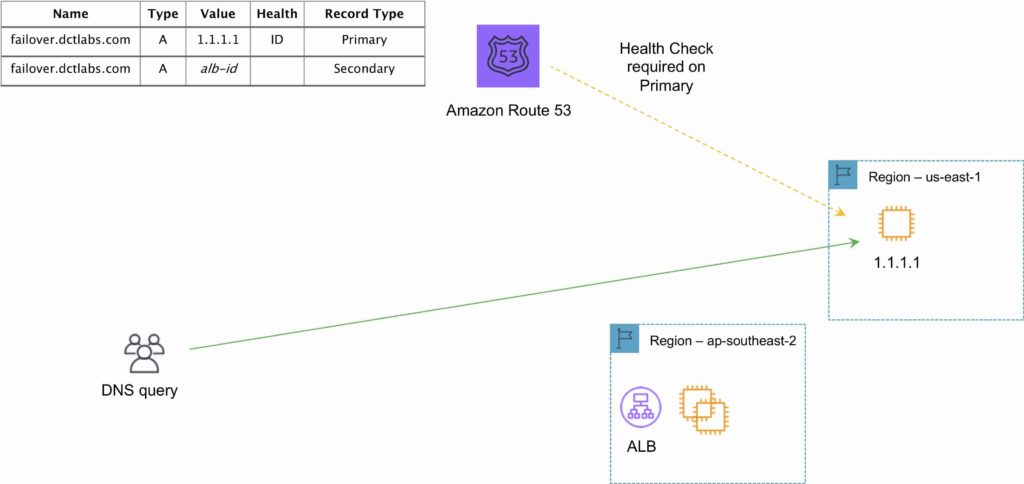
Used for active-passive.

Routes only when the resource is healthy.

Can be used with ELB.

When used with Alias records set Evaluate Target Health to “Yes” and do not use health checks.

The following diagram depicts an Amazon Route 53 Failover routing policy configuration:



Geo-location Routing Policy

Caters to different users in different countries and different languages.

Contains users within a particular geography and offers them a customized version of the workload based on their specific needs.

Geolocation can be used for localizing content and presenting some or all your website in the language of your users.

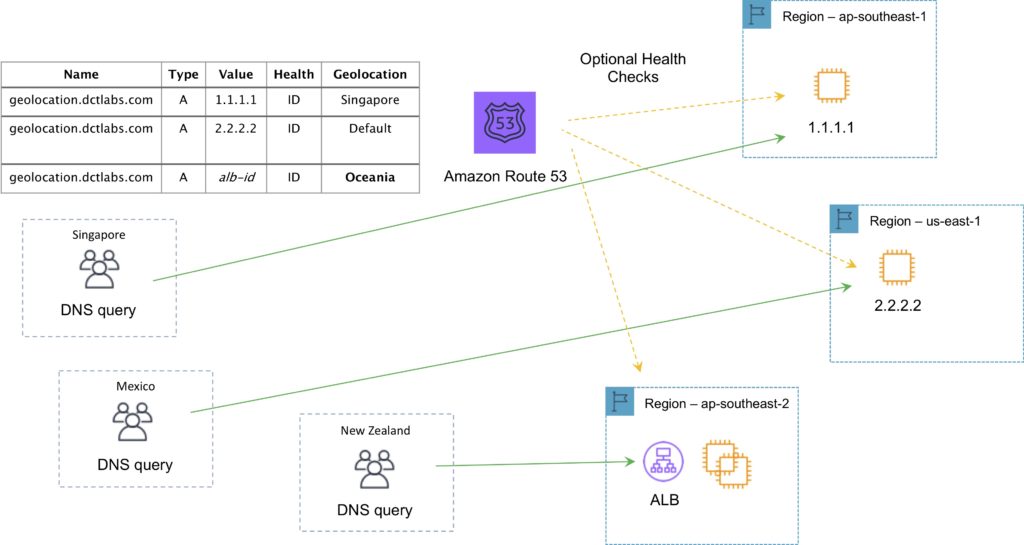
Can also protect distribution rights.

Can be used for spreading load evenly between regions.

If you have multiple records for overlapping regions, Route 53 will route to the smallest geographic region.

You can create a default record for IP addresses that do not map to a geographic location.

The following diagram depicts an Amazon Route 53 Geolocation routing policy configuration:



Geo-proximity routing policy (requires Route Flow):

Use for routing traffic based on the location of resources and, optionally, shift traffic from resources in one location to resources in another.

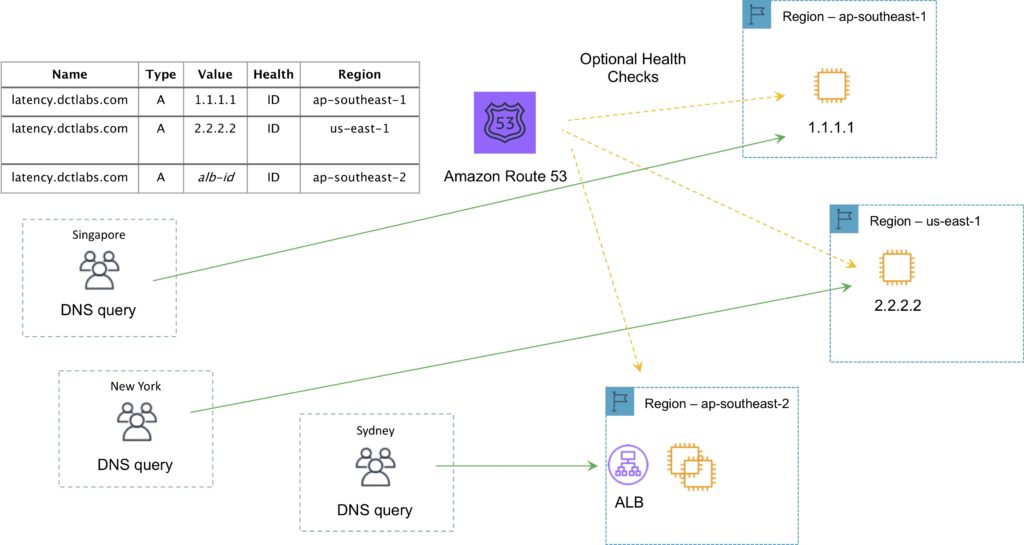
Latency Routing Policy

AWS maintains a database of latency from different parts of the world.

Focused on improving performance by routing to the region with the lowest latency.

You create latency records for your resources in multiple EC2 locations.

The following diagram depicts an Amazon Route 53 Latency based routing policy configuration:



Multi-value Answer Routing Policy

Use for responding to DNS queries with up to eight healthy records selected at random.

The following diagram depicts an Amazon Route 53 Multivalue routing policy configuration:

Weighted Routing Policy

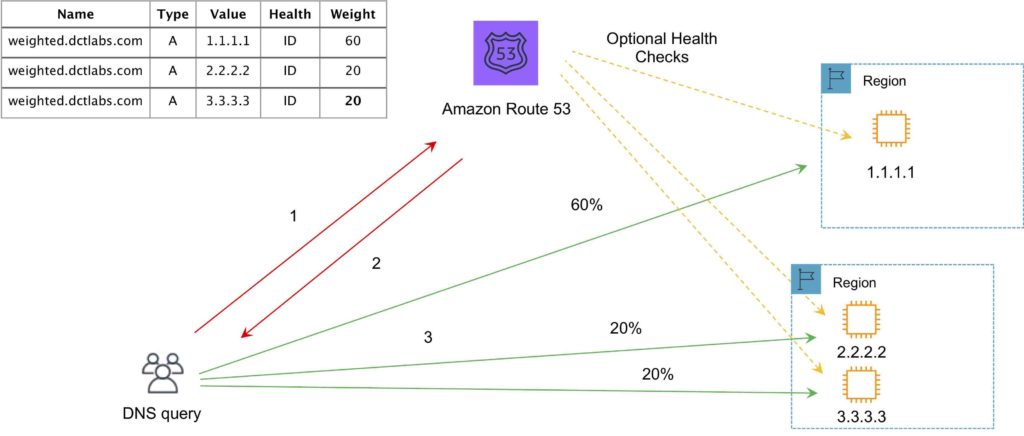
Like simple but you can specify a weight per IP address.

You create records that have the same name and type and assign each record a relative weight.

Numerical value that favors one IP over another.

To stop sending traffic to a resource you can change the weight of the record to 0.

The following diagram depicts an Amazon Route 53 Weighted routing policy configuration:



Traffic Flow

Route 53 Traffic Flow provides Global Traffic Management (GTM) services.

Traffic flow policies allow you to create routing configurations for resources using routing types such as failover and geolocation.

Create policies that route traffic based on specific constraints, including latency, endpoint health, load, geo-proximity, and geography.

Scenarios include:

Adding a simple backup page in Amazon S3 for a website.

Building sophisticated routing policies that consider an end user’s geographic location, proximity to an AWS region, and the health of each of your endpoints.

Amazon Route 53 Traffic Flow also includes a versioning feature that allows you to maintain a history of changes to your routing policies, and easily roll back to a previous policy version using the console or API.

Route 53 Resolver

Route 53 Resolver is a set of features that enable bi-directional querying between on-premises and AWS over private connections.

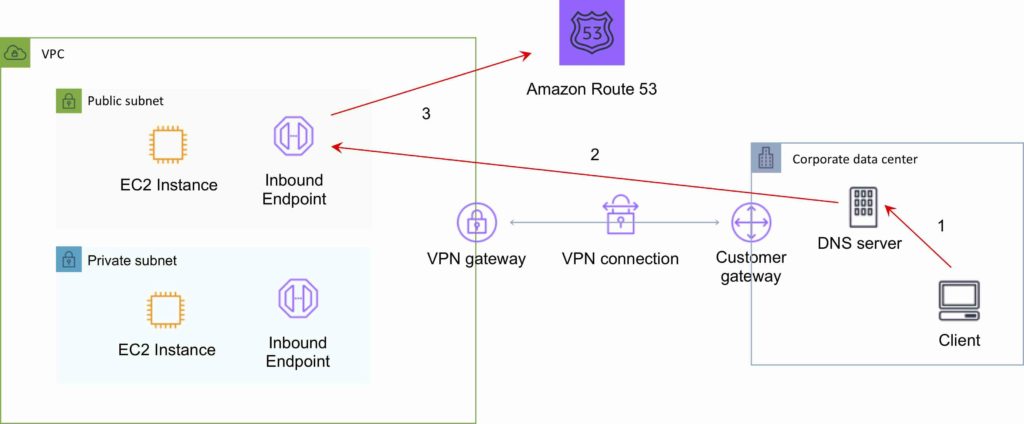
Used for enabling DNS resolution for hybrid clouds.

Route 53 Resolver Endpoints.

Inbound query capability is provided by Route 53 Resolver Endpoints, allowing DNS queries that originate on-premises to resolve AWS hosted domains.

Connectivity needs to be established between your on-premises DNS infrastructure and AWS through a Direct Connect (DX) or a Virtual Private Network (VPN).

Endpoints are configured through IP address assignment in each subnet for which you would like to provide a resolver.



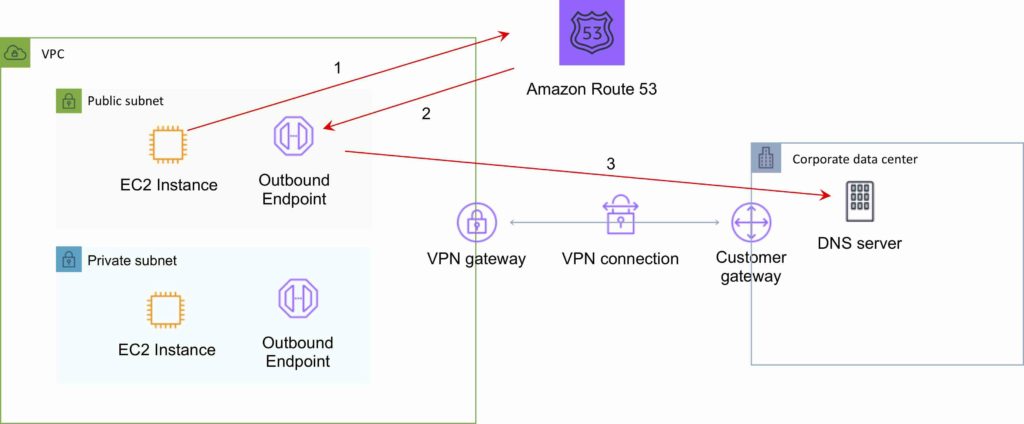
Conditional forwarding rules:

Outbound DNS queries are enabled using Conditional Forwarding Rules. .

Domains hosted within your on-premises DNS infrastructure can be configured as forwarding rules in Route 53 Resolver.

Rules will trigger when a query is made to one of those domains and will attempt to forward DNS requests to your DNS servers that were configured along with the rules.

Like the inbound queries, this requires a private connection over DX or VPN.



There are a couple of ways to provide resolution of Microsoft Active Directory Domain Controller DNS zones and AWS records:

Define an outbound Amazon Route 53 Resolver. Set a conditional forwarding rule for the Active Directory domain to the Active Directory servers. Configure the DNS settings in the VPC DHCP options set to use the AmazonProvidedDNS servers.

Configure the DHCP options set associated with the VPC to assign the IP addresses of the Domain Controllers as DNS servers. Update the DNS service on the Active Directory servers to forward all non-authoritative queries to the VPC Resolver.

Charges

You pay per hosted zone per month (no partial months).

A hosted zone deleted within 12 hours of creation is not charged (queries are charged).

Additional charges for:

Queries.

Traffic Flow.

Health Checks.

Route 53 Resolver ENIs + queries.

Domain names.

Alias records are free of charge when the records are mapped to one of the following:

Elastic Load Balancers.

Amazon CloudFront distributions.

AWS Elastic Beanstalk environments.

Amazon S3 buckets that are configured as website endpoints.

Health checks are charged with different prices for AWS vs non-AWS endpoints.

You do not pay for the records that you add to your hosted zones.

Latency-based routing queries are more expensive.

Geo DNS and geo-proximity also have higher prices.

Amazon CloudFront

CloudFront is a web service that gives businesses and web application developers an easy and cost-effective way to distribute content with low latency and high data transfer speeds.

CloudFront is a good choice for distribution of frequently accessed static content that benefits from edge delivery—like popular website images, videos, media files or software downloads.

Used for dynamic, static, streaming, and interactive content.

CloudFront is a global service:

Ingress to upload objects.

Egress to distribute content.

Amazon CloudFront provides a simple API that lets you:

Distribute content with low latency and high data transfer rates by serving requests using a network of edge locations around the world.

Get started without negotiating contracts and minimum commitments.

You can use a zone apex name on CloudFront.

CloudFront supports wildcard CNAME.

Supports wildcard SSL certificates, Dedicated IP, Custom SSL and SNI Custom SSL (cheaper).

Supports Perfect Forward Secrecy which creates a new private key for each SSL session.

Edge Locations and Regional Edge Caches

An edge location is the location where content is cached (separate to AWS regions/AZs).

Requests are automatically routed to the nearest edge location.

Edge locations are not tied to Availability Zones or regions.

Regional Edge Caches are located between origin web servers and global edge locations and have a larger cache.

Regional Edge Caches have larger cache-width than any individual edge location, so your objects remain in cache longer at these locations.

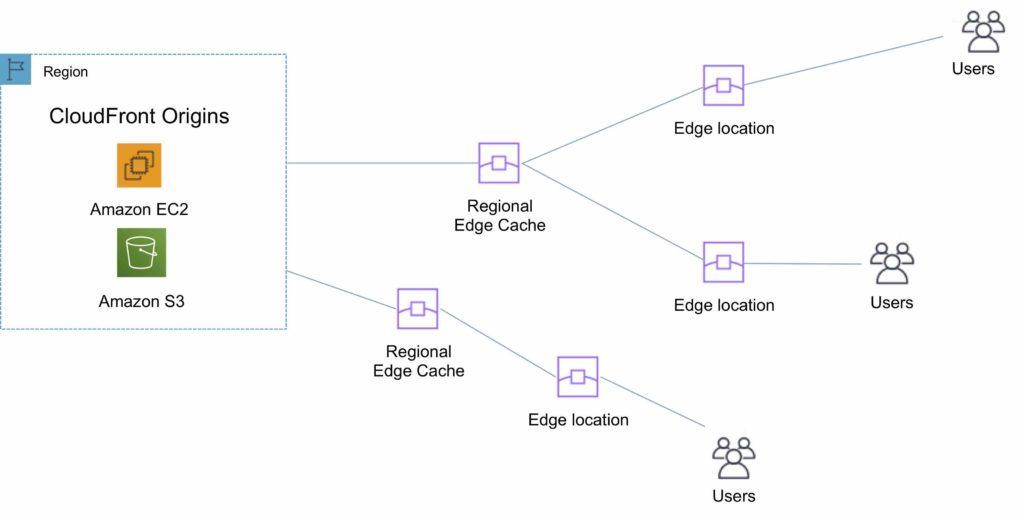
Regional Edge caches aim to get content closer to users.

Proxy methods PUT/POST/PATCH/OPTIONS/DELETE go directly to the origin from the edge locations and do not proxy through Regional Edge caches.

Dynamic content goes straight to the origin and does not flow through Regional Edge caches.

Edge locations are not just read only, you can write to them too.

The diagram below shows where Regional Edge Caches and Edge Locations are placed in relation to end users:



Origins

An origin is the origin of the files that the CDN will distribute.

Origins can be either an S3 bucket, an EC2 instance, an Elastic Load Balancer, or Route 53 – can also be external (non-AWS).

When using Amazon S3 as an origin you place all your objects within the bucket.

You can use an existing bucket and the bucket is not modified in any way.

By default all newly created buckets are private.

You can setup access control to your buckets using:

Bucket policies.

Access Control Lists.

You can make objects publicly available or use CloudFront signed URLs.

A custom origin server is a HTTP server which can be an EC2 instance or an on-premises/non-AWS based web server.

When using an on-premises or non-AWS based web server you must specify the DNS name, ports, and protocols that you want CloudFront to use when fetching objects from your origin.

Most CloudFront features are supported for custom origins except RTMP distributions (must be an S3 bucket).

When using EC2 for custom origins Amazon recommend:

Use an AMI that automatically installs the software for a web server.

Use ELB to handle traffic across multiple EC2 instances.

Specify the URL of your load balancer as the domain name of the origin server.

S3 static website:

Enter the S3 static website hosting endpoint for your bucket in the configuration.

Example: http://<bucketname>.s3-website-<region>.amazonaws.com.

Objects are cached for 24 hours by default.

The expiration time is controlled through the TTL.

The minimum expiration time is 0.

Static websites on Amazon S3 are considered custom origins.

AWS origins are Amazon S3 buckets (not a static website).

CloudFront keeps persistent connections open with origin servers.

Files can also be uploaded to CloudFront.

High availability with Origin Failover:

Can set up CloudFront with origin failover for scenarios that require high availability.

Uses an origin group in which you designate a primary origin for CloudFront plus a second origin that CloudFront automatically switches to when the primary origin returns specific HTTP status code failure responses.

For more info, check this [article](https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html).

Also works with Lambda@Edge functions.

Distributions

To distribute content with CloudFront you need to create a distribution.

The distribution includes the configuration of the CDN including:

Content origins.

Access (public or restricted).

Security (HTTP or HTTPS).

Cookie or query-string forwarding.

Geo-restrictions.

Access logs (record viewer activity).

There are two types of distribution.

Web Distribution:

Static and dynamic content including .html, .css, .php, and graphics files.

Distributes files over HTTP and HTTPS.

Add, update, or delete objects, and submit data from web forms.

Use live streaming to stream an event in real time.

RTMP:

Distribute streaming media files using Adobe Flash Media Server’s RTMP protocol.

Allows an end user to begin playing a media file before the file has finished downloading from a CloudFront edge location.

Files must be stored in an S3 bucket.

To use CloudFront live streaming, create a web distribution.

For serving both the media player and media files you need two types of distributions:

A web distribution for the media player.

An RTMP distribution for the media files.

S3 buckets can be configured to create access logs and cookie logs which log all requests made to the S3 bucket.

Amazon Athena can be used to analyze access logs.

CloudFront is integrated with CloudTrail.

CloudTrail saves logs to the S3 bucket you specify.

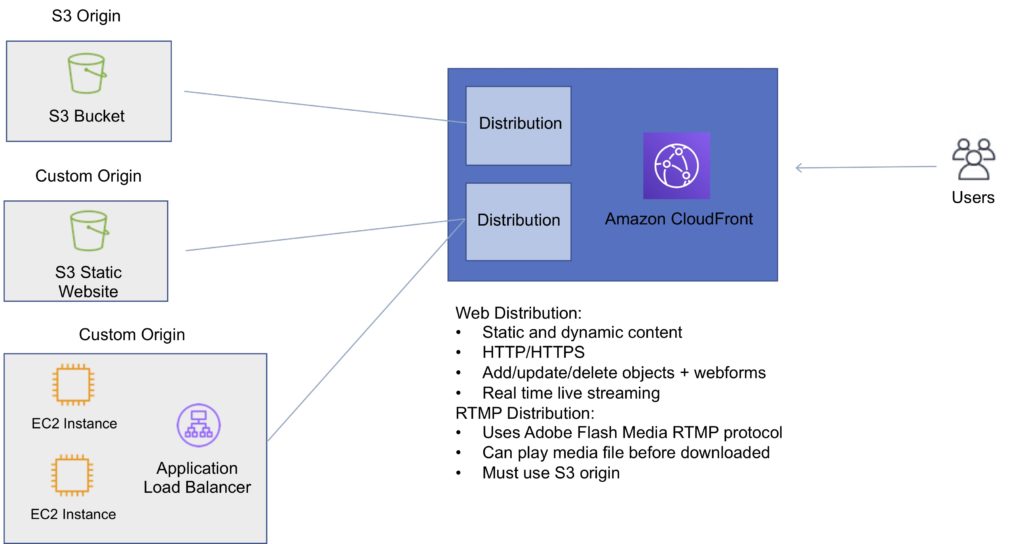
CloudTrail captures information about all requests whether they were made using the CloudFront console, the CloudFront API, the AWS SDKs, the CloudFront CLI, or another service.

CloudTrail can be used to determine which requests were made, the source IP address, who made the request etc.

To view CloudFront requests in CloudTrail logs you must update an existing trail to include global services.

To delete a distribution it must first be disabled (can take up to 15 minutes).

The diagram below depicts Amazon CloudFront Distributions and Origins:



Cache Behavior

Allows you to configure a variety of CloudFront functionality for a given URL path pattern.

For each cache behavior you can configure the following functionality:

The path pattern (e.g. /images/\*.jpg, /images\*.php).

The origin to forward requests to (if there are multiple origins).

Whether to forward query strings.

Whether to require signed URLs.

Allowed HTTP methods.

Minimum amount of time to retain the files in the CloudFront cache (regardless of the values of any cache-control headers).

The default cache behavior only allows a path pattern of /\*.

Additional cache behaviors need to be defined to change the path pattern following creation of the distribution.

You can restrict access to content using the following methods:

Restrict access to content using signed cookies or signed URLs.

Restrict access to objects in your S3 bucket.

A special type of user called an Origin Access Identity (OAI) can be used to restrict access to content in an Amazon S3 bucket.

By using an OAI you can restrict users so they cannot access the content directly using the S3 URL, they must connect via CloudFront.

You can define the viewer protocol policy:

HTTP and HTTPS.

Redirect HTTP to HTTPS.

HTTPS only.

You can define the Allowed HTTP Methods:

GET, HEAD.

GET, HEAD, OPTIONS.

GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE.

For web distributions you can configure CloudFront to require that viewers use HTTPS.

Field-Level Encryption:

Field-level encryption adds an additional layer of security on top of HTTPS that lets you protect specific data so that it is only visible to specific applications.

Field-level encryption allows you to securely upload user-submitted sensitive information to your web servers.

The sensitive information is encrypted at the edge closer to the user and remains encrypted throughout application processing.

Origin policy:

HTTPS only.

Match viewer – CloudFront matches the protocol with your custom origin.

Use match viewer only if you specify Redirect HTTP to HTTPS or HTTPS only for the viewer protocol policy.

CloudFront caches the object once even if viewers make requests using HTTP and HTTPS.

Object invalidation:

You can remove an object from the cache by invalidating the object.

You cannot cancel an invalidation after submission.

You cannot invalidate media files in the Microsoft Smooth Streaming format when you have enabled Smooth Streaming for the corresponding cache behavior.

Objects are cached for the TTL (always recorded in seconds, default is 24 hours, default max is 1 year).

Only caches for GET requests (not PUT, POST, PATCH, DELETE).

Dynamic content is cached.

Consider how often your files change when setting the TTL.

Invalidation can be used to immediately revoke cached objects – chargeable.

Deletions propagate.

Cache Hit Ratio

A good cache hit ratio means more requests are served from the cache.

Methods of improving the cache hit ratio include:

Use the Cache-Control max-age directive to increase the time objects remain in the cache

Use Origin Shield.

Forward only the query string parameters for which your origin will return unique objects.

Configure CloudFront to forward only specified cookies instead of forwarding all cookies.

Configure CloudFront to forward and cache based on only specified headers instead of forwarding and caching based on all headers.

Restrictions

Blacklists and whitelists can be used for geography – you can only use one at a time.

There are two options available for geo-restriction (geo-blocking):

Use the CloudFront geo-restriction feature (use for restricting access to all files in a distribution and at the country level).

Use a 3rd party geo-location service (use for restricting access to a subset of the files in a distribution and for finer granularity at the country level).

Lambda@Edge

Can be used to run Lambda at Edge Locations.

Lets you run Node.js and Python Lambda functions to customize content that CloudFront delivers.

Executes the functions in AWS locations closer to the viewer.

You can use Lambda functions to change CloudFront requests and responses at the following points:

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).

Lambda@Edge can do the following:

Inspect cookies and rewrite URLs to perform A/B testing.

Send specific objects to your users based on the User-Agent header.

Implement access control by looking for specific headers before passing requests to the origin.

Add, drop, or modify headers to direct users to different cached objects.

Generate new HTTP responses.

Cleanly support legacy URLs.

Modify or condense headers or URLs to improve cache utilization.

Make HTTP requests to other Internet resources and use the results to customize responses.

Exam tip: Lambda@Edge can be used to load different resources based on the User-Agent HTTP header.

Signed URLs and Signed Cookies

A signed URL includes additional information, for example, an expiration date and time, that gives you more control over access to your content. This additional information appears in a policy statement, which is based on either a canned policy or a custom policy.

CloudFront signed cookies allow you to control who can access your content when you don’t want to change your current URLs or when you want to provide access to multiple restricted files, for example, all the files in the subscribers’ area of a website.

Application must authenticate user and then send three Set-Cookie headers to the viewer; the viewer stores the name-value pair and adds them to the request in a Cookie header when requesting access to content.

Use signed URLs in the following cases:

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 in the following cases:

You want to provide access to multiple restricted files, for example, all the files for a video in HLS format or all the files in the subscribers’ area of website.

You don’t want to change your current URLs.

Origin Access Identity

Used in combination with signed URLs and signed cookies to restrict direct access to an S3 bucket (prevents bypassing the CloudFront controls).

An origin access identity (OAI) is a special CloudFront user that is associated with the distribution.

Permissions must then be changed on the Amazon S3 bucket to restrict access to the OAI.

If users request files directly by using Amazon S3 URLs, they’re denied access.

The origin access identity has permission to access files in your Amazon S3 bucket, but users don’t.

AWS WAF

AWS WAF is a web application firewall that lets you monitor HTTP and HTTPS requests that are forwarded to CloudFront and lets you control access to your content.

With AWS WAF you can shield access to content based on conditions in a web access control list (web ACL) such as:

Origin IP address.

Values in query strings.

CloudFront responds to requests with the requested content or an HTTP 403 status code (forbidden).

CloudFront can also be configured to deliver a custom error page.

Need to associate the relevant distribution with the web ACL.

Security

PCI DSS compliant but recommended not to cache credit card information at edge locations.

HIPAA compliant as a HIPAA eligible service.

Distributed Denial of Service (DDoS) protection:

CloudFront distributes traffic across multiple edge locations and filters requests to ensure that only valid HTTP(S) requests will be forwarded to backend hosts. CloudFront also supports geo-blocking, which you can use to prevent requests from geographic locations from being served.

Domain Names

CloudFront typically creates a domain name such as a232323.cloudfront.net.

Alternate domain names can be added using an alias record (Route 53).

For other service providers use a CNAME (cannot use the zone apex with CNAME).

Moving domain names between distributions:

You can move subdomains yourself.

For the root domain you need to use AWS support.

High Availability

CloudFront caches content at Edge Locations around the world. The more objects served by the cache, the fewer the requests to the origin.  This reduces the load on your origin server and reduces latency.

You can set up CloudFront with [origin failover](https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html) for scenarios that require high availability.

To set up origin failover, you must have a distribution with at least two origins. Next, you create an origin group for your distribution that includes two origins, setting one as the primary. Finally, you create or update a cache behavior to use the origin group.

Monitoring and Reporting

You can view [operational metrics](https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/viewing-cloudfront-metrics.html) about your CloudFront distributions and Lambda@Edge functions in the CloudFront console.

The following default metrics are included for all CloudFront distributions, at no additional cost:

Requests

The total number of viewer requests received by CloudFront, for all HTTP methods and for both HTTP and HTTPS requests.

Bytes downloaded

The total number of bytes downloaded by viewers for GET, HEAD, and OPTIONS requests.

Bytes uploaded

The total number of bytes that viewers uploaded to your origin with CloudFront, using POST and PUT requests.

4xx error rate

The percentage of all viewer requests for which the response’s HTTP status code is 4xx.

5xx error rate

The percentage of all viewer requests for which the response’s HTTP status code is 5xx.

Total error rate

The percentage of all viewer requests for which the response’s HTTP status code is 4xx or 5xx.

In addition to the default metrics, you can enable additional metrics for an additional cost.

These additional metrics must be enabled for each distribution separately:

Cache hit rate

The percentage of all cacheable requests for which CloudFront served the content from its cache. HTTP POST and PUT requests, and errors, are not considered cacheable requests.

Origin latency

The total time spent from when CloudFront receives a request to when it starts providing a response to the network (not the viewer), for requests that are served from the origin, not the CloudFront cache. This is also known as first byte latency, or time-to-first-byte.

Error rate by status code

The percentage of all viewer requests for which the response’s HTTP status code is a particular code in the 4xx or 5xx range. This metric is available for all the following error codes: 401, 403, 404, 502, 503, and 504.

Logging and Auditing

S3 buckets can be configured to create access logs and cookie logs which log all requests made to the S3 bucket.

Amazon Athena can be used to analyze access logs.

CloudFront is integrated with CloudTrail.

CloudTrail saves logs to the S3 bucket you specify.

CloudTrail captures information about all requests whether they were made using the CloudFront console, the CloudFront API, the AWS SDKs, the CloudFront CLI, or another service.

CloudTrail can be used to determine which requests were made, the source IP address, who made the request etc.

To view CloudFront requests in CloudTrail logs you must update an existing trail to include global services.

Charges

There is an option for reserved capacity over 12 months or longer (starts at 10TB of data transfer in a single region).

You pay for:

Data Transfer Out to Internet.

Data Transfer Out to Origin.

Number of HTTP/HTTPS Requests.

Invalidation Requests.

Dedicated IP Custom SSL.

Field level encryption requests.

You do not pay for:

Data transfer between AWS regions and CloudFront.

Regional edge cache.

AWS ACM SSL/TLS certificates.

Shared CloudFront certificates.

AWS Global Accelerator

AWS Global Accelerator is a service that improves the availability and performance of applications with local or global users.

It provides static IP addresses that act as a fixed entry point to application endpoints in a single or multiple AWS Regions, such as Application Load Balancers, Network Load Balancers or EC2 instances.

Uses the AWS global network to optimize the path from users to applications, improving the performance of TCP and UDP traffic.

AWS Global Accelerator continually monitors the health of application endpoints and will detect an unhealthy endpoint and redirect traffic to healthy endpoints in less than 1 minute.

Details and Benefits

Uses redundant (two) static anycast IP addresses in different network zones (A and B).

The redundant pair are globally advertised.

Uses AWS Edge Locations – addresses are announced from multiple edge locations at the same time.

Addresses are associated to regional AWS resources or endpoints.

AWS Global Accelerator’s IP addresses serve as the frontend interface of applications.

Intelligent traffic distribution: Routes connections to the closest point of presence for applications.

Targets can be Amazon EC2 instances or Elastic Load Balancers (ALB and NLB).

By using the static IP addresses, you don’t need to make any client-facing changes or update DNS records as you modify or replace endpoints.

The addresses are assigned to your accelerator for as long as it exists, even if you disable the accelerator and it no longer accepts or routes traffic.

Does health checks for TCP only – not UDP.

Can assign target weight within a region to control routing and “dial” up or down traffic to a region.

Fault tolerance:

Has a fault-isolating design that increases the availability of your applications.

AWS Global Accelerator allocates two IPv4 static addresses that are serviced by independent network zones.

Like Availability Zones, these network zones are isolated units with their own set of physical infrastructure and service IP addresses from a unique IP subnet.

If one IP address from a network zone becomes unavailable, due to network disruptions or IP address blocking by certain client networks, client applications can retry using the healthy static IP address from the other isolated network zone.

Global performance-based routing:

AWS Global Accelerator uses the vast, congestion-free AWS global network to route TCP and UDP traffic to a healthy application endpoint in the closest AWS Region to the user.

If there’s an application failure, AWS Global Accelerator provides instant failover to the next best endpoint.

Fine-grained traffic control:

AWS Global Accelerator gives you the option to dial up or dial down traffic to a specific AWS Region by using traffic dials.

The traffic dial lets you easily do performance testing or blue/green deployment testing for new releases across different AWS Regions, for example.

If an endpoint fails, AWS Global Accelerator assigns user traffic to the other endpoints, to maintain high availability.

By default, traffic dials are set to 100% across all endpoint groups so that AWS Global Accelerator can select the best endpoint for applications.

Continuous availability monitoring:

AWS Global Accelerator continuously monitors the health of application endpoints by using TCP, HTTP, and HTTPS health checks.

It instantly reacts to changes in the health or configuration of application endpoints, and redirects user traffic to healthy endpoints that deliver the best performance and availability to end users.

Client affinity:

AWS Global Accelerator enables you to build applications that require maintaining state.

For stateful applications where you need to consistently route users to the same endpoint, you can choose to direct all requests from a user to the same endpoint, regardless of the port and protocol.

Distributed denial of service (DDoS) resiliency at the edge:

By default, AWS Global Accelerator is protected by AWS Shield Standard, which minimizes application downtime and latency from denial-of-service attacks by using always-on network flow monitoring and automated in-line mitigation.

You can also enable AWS Shield Advanced for automated resource-specific enhanced detection and mitigation, as well as 24×7 access to the AWS DDoS Response Team (DRT) for manual mitigations of sophisticated DDoS attacks.