Compute Fundamentals

Amazon EC2 - It offers your choice of processor, storage, networking, operating system, etc. to match your applications need. Here, the software gets installed, patch instance, manage scaling and availability, configure storage and then write application code and deploy to your instance.

Instance Types:

- General Purpose - balanced mix of CPU, memory, and storage.
- Compute Optimized - greater focus on compute power and are ideal for applications requiring high performance processors.
- Memory Optimized - They are primarily used for large-scale enterprise class in-memory applications.
- Accelerated Computing - Utilize hardware accelerators or co-processors to perform floating-point calculations faster and more efficiently.
- Storage Optimized - Instances in this family use SSD backed instance storage for low latency and I/O performance.
- PC Optimized - Designed for high performance computing.

Purchasing options for EC2 instances:

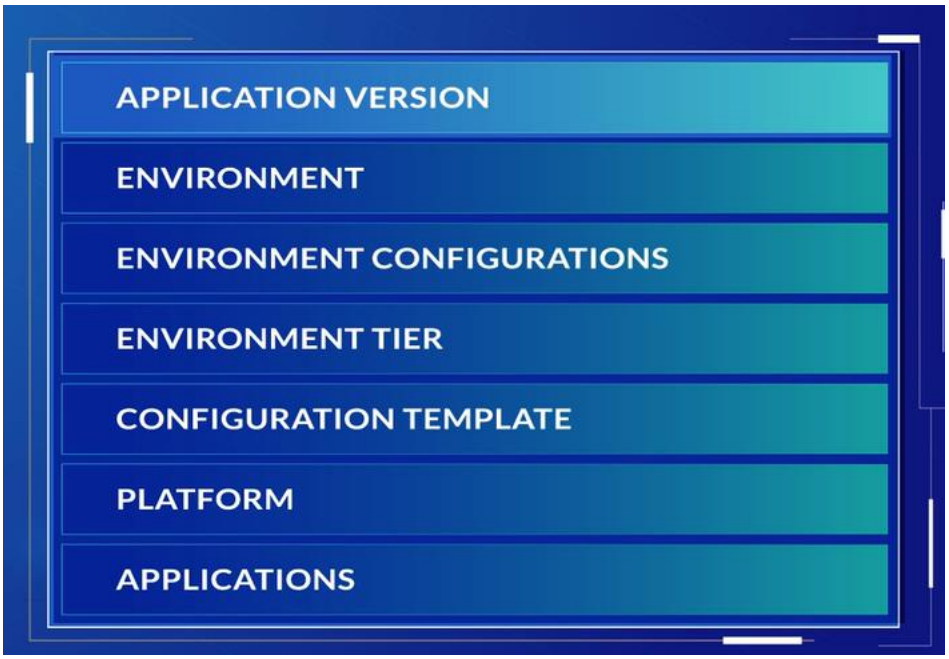
- Scheduled: Scheduled instances run on a recurring schedule, either daily, weekly, or monthly; however, you would still be charged if not running the instance.
- Reserved: Reserved instance type is suitable for long-term predictable workloads.
- Spot: Spot instances are best for unpredictable, periodic workloads.
- On-demand: On-demand instances are used for short-term where workloads can be irregular and where the workload can be interrupted.

EC2 Image Builder – You build an image by making use of dropdowns that have all the resources (for ex: memory, storage, instances etc.) required for an application.

Elastic Beanstalk - Upload your application and elastic beanstalk automatically deploys the AWS resources such as EC2 resources, auto scaling groups, RDS database instances,

elastic load balancers required for the application. AWS Elastic Beanstalk is designed to support multiple versions.

Components:



AWS Lambda - This is a serverless and stateless resource where you don't have to manage the resources like EC2. You can develop your lambda functions and deploy it. With AWS Lambda, you pay for compute time that you consume. On AWS Lambda, the core components are Lambda functions and event sources.

AWS Batch - Runs a series of jobs as a batch (for ex: Training a machine learning model etc.)

Amazon LightSail - Quick and easy way to deploy simple websites, blogs etc.

AWS App Runner - You can build, deploy and run containerized applications, microservices or APIs without provisioning infrastructure. It can integrate with CI/CD pipelines.

AWS Outposts - Hybrid cloud resource which can migrate workloads from on-premises to AWS Outpost. It allows running various AWS services on-premises, including ECS, EKS, S3, RDS, and EMR.

AWS Serverless Application Repository - It includes AWS Lambda, Amazon API Gateway, Dynamo DB and no provisioning of infrastructure is required.

AWS Simspace Weaver - To build large scale spatial simulations such as games, graphics etc.

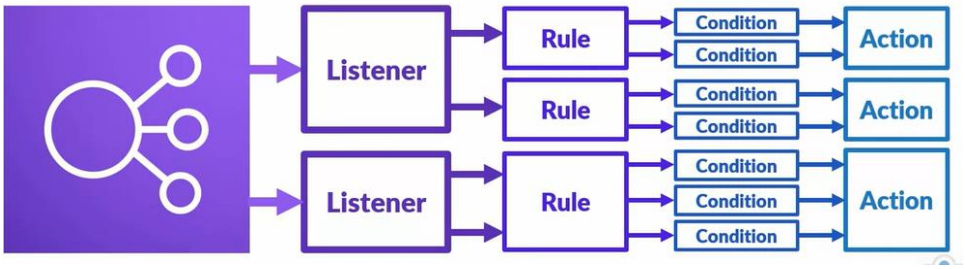
Elastic Load Balancing (ELB) - If you have traffic to your application, then elastic load balancer distributes traffic to multiple EC2 instances.

Load Balancer Types:

- Application Load Balancer (ALB) - This provides a flexible feature set for your web applications running the HTTP or HTTPS protocols. Operates at the request level.
- Network Load Balancer - Ultra-high performance while maintaining very low latencies. Operates at connection level and deals with TCP, TLS etc.
- Classic Load Balancer - Used for applications that were built in the existing EC2 classic environment. Operates at request and connection level.

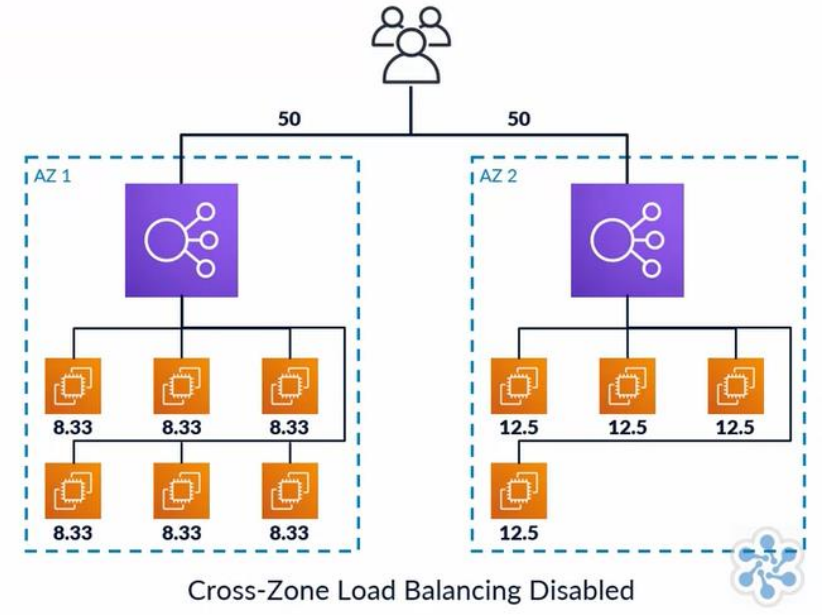
ELB Components:

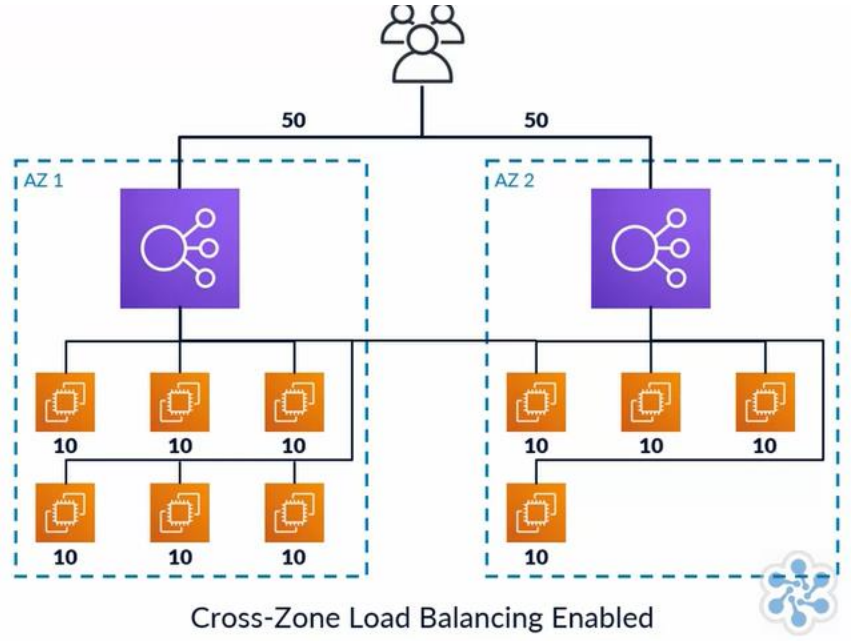
- Listeners - The listener defines how your inbound connections are routed to your target groups based on protocols and ports set as conditions. For every load balancer, you must configure at least one listener.
- Target groups - A target group is simply a group of resources that you want your ELB to route requests to, for example a fleet of EC2 instances.
- Rules - Rules are associated to each listener, and they help to define how an incoming request gets routed to which target group.



- Health checks - The ELB associates a health check that is performed against the resources defined within the target group.
- Internet-facing ELB - Accessible with public IP address.

- Internal ELB - It only has an internal IP address. This means that it can only serve requests that originate from within your VPC itself.
- ELB Nodes - For each Available Zone selected, an ELB node will be placed within that Availability Zone.
- Cross-Zone load balancing - Two types: Enabled and Disabled. In disabled cross-zone load balancing, the traffic is unevenly distributed. In enabled cross-zone load balancing, the traffic is evenly distributed.



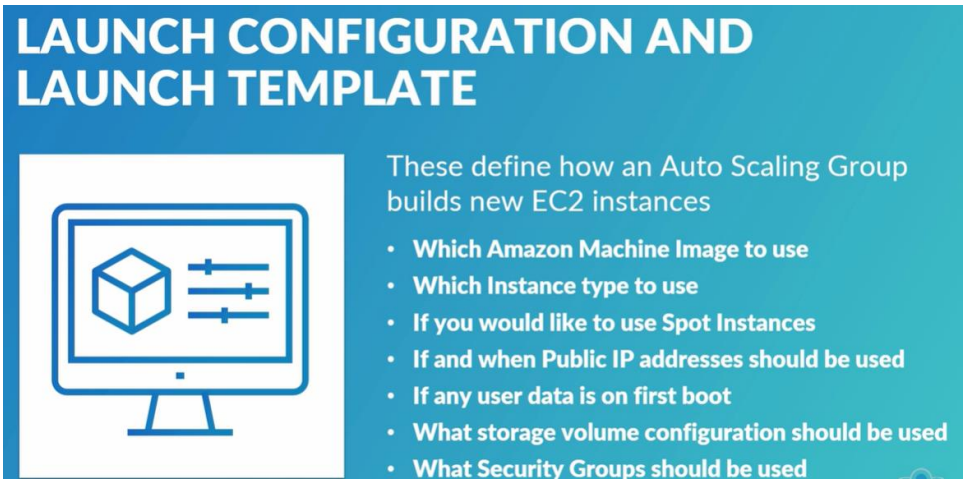


Server Certificates (SSL/TLS) - HTTPS allows encrypted communication. However, to allow HTTPS requests you need a server certificate which is the AWS Certificate Manager (ACM).

EC2 Auto Scaling - It is a mechanism that automatically allows you to increase or decrease your EC2 resources to meet the demand. Advantages are automation, great customer satisfaction, cost reduction.

Components:

- Launch configuration or launch template – It defines how an Auto Scaling Group builds new EC2 instances. Without the launch template, auto scaling would not know what instance it was launching and to which configuration.



- Auto Scaling Group - A group of instances that can automatically increase or decrease and which availability zone it should scale to.

Using ELB and EC2 Auto Scaling Together - When you attach an ELB to an auto scaling group, the ELB will automatically detect the instances and start to distribute all traffic to the resources in the auto scaling group.

One challenge of using an Elastic Load Balancer without EC2 autoscaling is that you must manually add and remove targets based on the demand since you have not enabled autoscaling to do that for you.

Storage Fundamentals

Block storage - Block storage stores data in chunks known as blocks, and these blocks are stored in a volume, and attached to a single instance. It has low latency and is like Directly Attached Storage (DAS). For ex: HDDs, SSDs.

File storage - Data is stored as separate files within a series of directories, forming a data structure hierarchy. The data is then stored on top of a file system. It has shared access between multiple users and is like Network Attached Storage (NAS).

Object Storage - Used for large, unstructured data. For example: key-value pairs.

Amazon S3 - It is a fully managed, object-based storage service. Amazon S3 storage is the cheapest option for archived files.

Storage Classes for Amazon S3:

- S3 Standard
- S3 Intelligent Tiering
- S3 Standard infrequent access
- S3 One Zone Infrequent Access
- S3 Glacier
- S3 Glacier Deep Archive

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days

EC2 Instance Storage or Instance Store Volume - It resides on the EC2 Instance itself. It provides ephemeral(temporary) block-level storage. Not recommended for valuable data. Data is lost when your instance is stopped or terminated. Data stays intact when rebooted. Not all instance types support instance store volumes. The size of your volumes will increase as you increase the EC2 instance size.

Pros: No additional cost, offers a very high I/O speed, ideal for caching, often used within a load balancing group where data is replicated across the fleet.

Cons: It should not be used for data that needs to remain persistent, and data that needs to be accessed by multiple users.

Amazon Elastic Block Store (EBS) - EBS provides persistent and durable block level storage. EBS volumes can be attached to your EC2 instances and are used for Input/Output operations Per Second (IOPS). Only a single EBS volume can be attached to a single EC2 instance. However, multiple EBS volumes can be attached to a single instance. Data remains intact when terminated, stopped or rebooted. Backup of data is called SnapShot and they can be retrieved manually or using amazon CloudWatch. These snapshots are stored in S3. If you lost access to EBS volume you can replicate an EBS volume using SnapShot. It's not suitable for temporary storage.

EBS Volume Types:

- SSD - Suited for work with smaller blocks.
- HD - Suited for work with larger blocks. They are ideal when throughput is essential.

Amazon Elastic File System (EFS) - EFS is a fully managed, highly available and durable service that allows you to create shared file systems that can easily scale petabytes in size with low latency access. EFS has been designed to maintain a high level of throughput. It is regional that means it's spread across multiple availability zones. It can be used with Linux-based EC2 instances and on-premises servers, as well as Lambda functions and other AWS container services such as ECS, EKS, and Fargate. It is best for applications that scale across multiple instances, allowing for parallel access of data.

AWS Storage Gateway - The storage gateway is a software appliance that can be installed within your own data center, which allows integration between your on-premises storage and AWS. It offers file, volume and tape gateway configurations which can be used to help with your data retrieval and data backup solutions. It asynchronously backs up data to AWS.

AWS Snowball - This appliance transfers petabytes of data from the data center to AWS and vice versa.

Networking Fundamentals

Virtual Private Cloud (VPC) - VPC allows you to provision and deploy resources in a safe and secure manner. It's an isolated network segment within the AWS Cloud.

Subnets - Subnets reside inside VPC. The subnets contain EC2 instances or other resources.

Types:

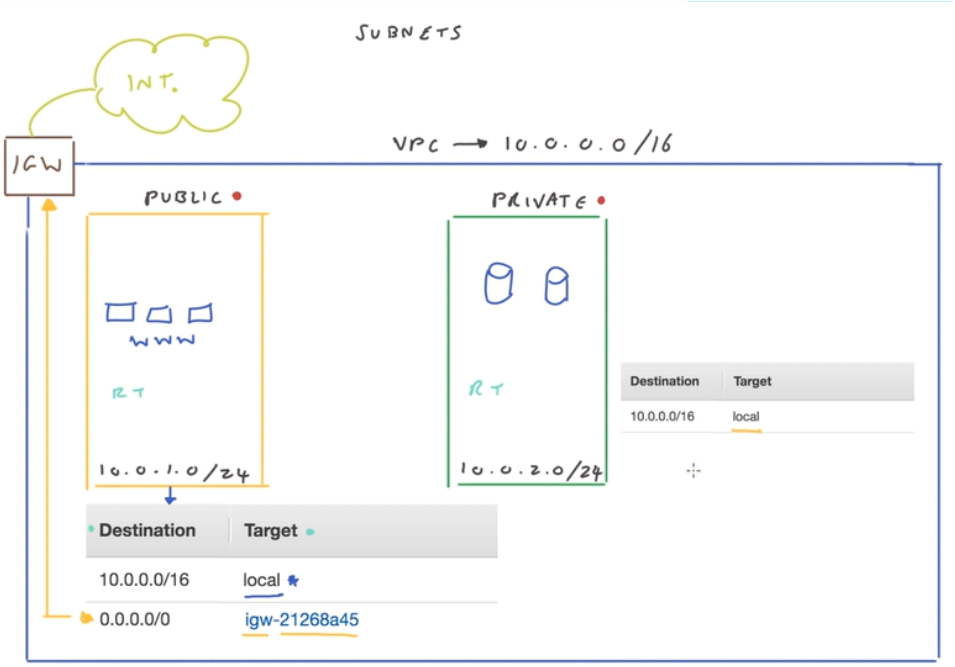
- Private - Private subnets have their own route table and route table will contain a destination field and a target field. It has a local route.
- Public - Public subnets have their own route table and route table will contain a destination field and a target field. It has a local route and additional route to Internet gateway (IGW) which can be accessed by the internet users.

All subnets within a VPC are private when they are created. They become public subnets when the Internet Gateway is configured as a subnet's target in a route table.

Local routes allow subnets within your VPC to communicate with each other.

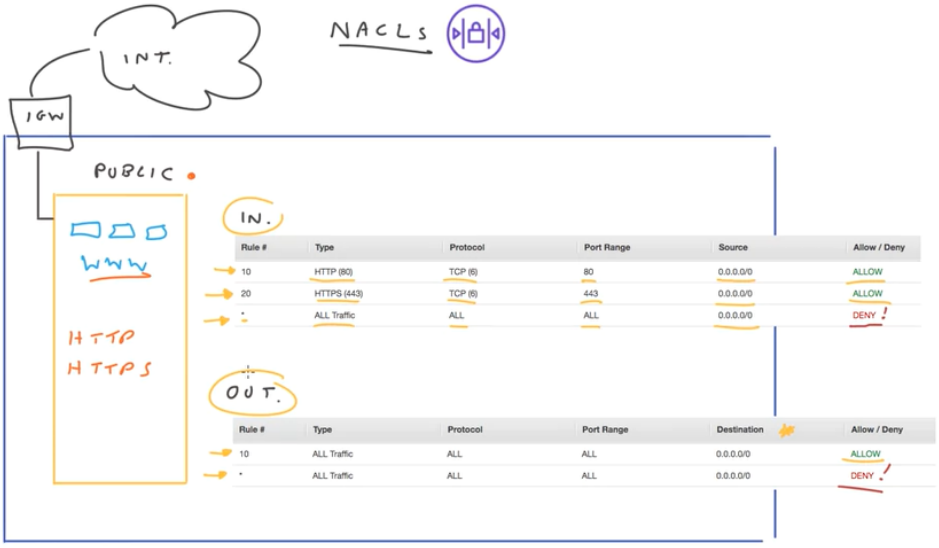
Same route table can be associated to multiple subnets. However, you can't associate more than one route table to a single subnet.

When working with TCP/IP addresses within your subnet, the first four addresses and last address in any subnet are reserved. So, use only 251 IP addresses available to you that you can use to assign to your host resources.

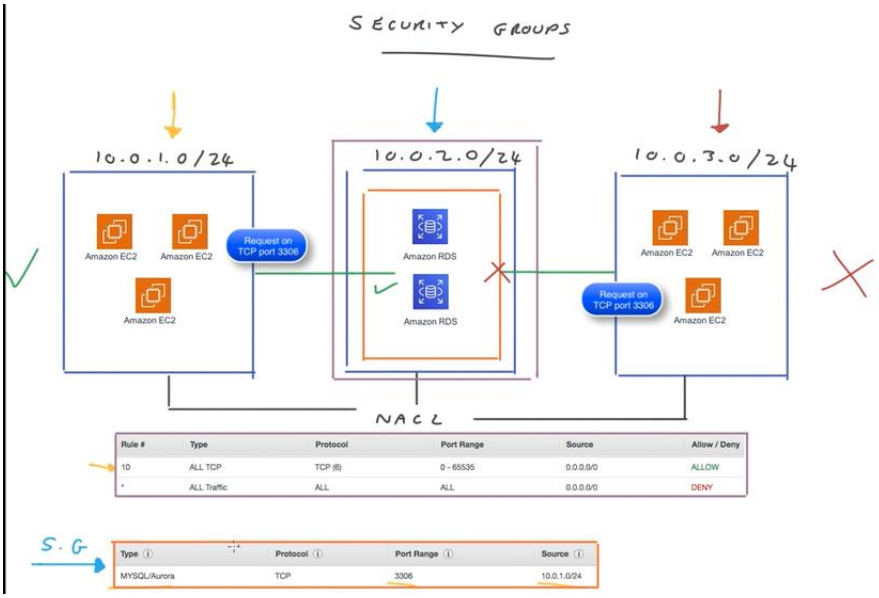


Network Access Control Lists (NACLs) - Its outside the subnet and the only traffic allowed in our public subnet is essentially HTTP and HTTPS. The same NACL can be associated to multiple subnets. However, you can't associate more than one NACL to a single subnet. NACLs are stateless that means any response traffic generated from a

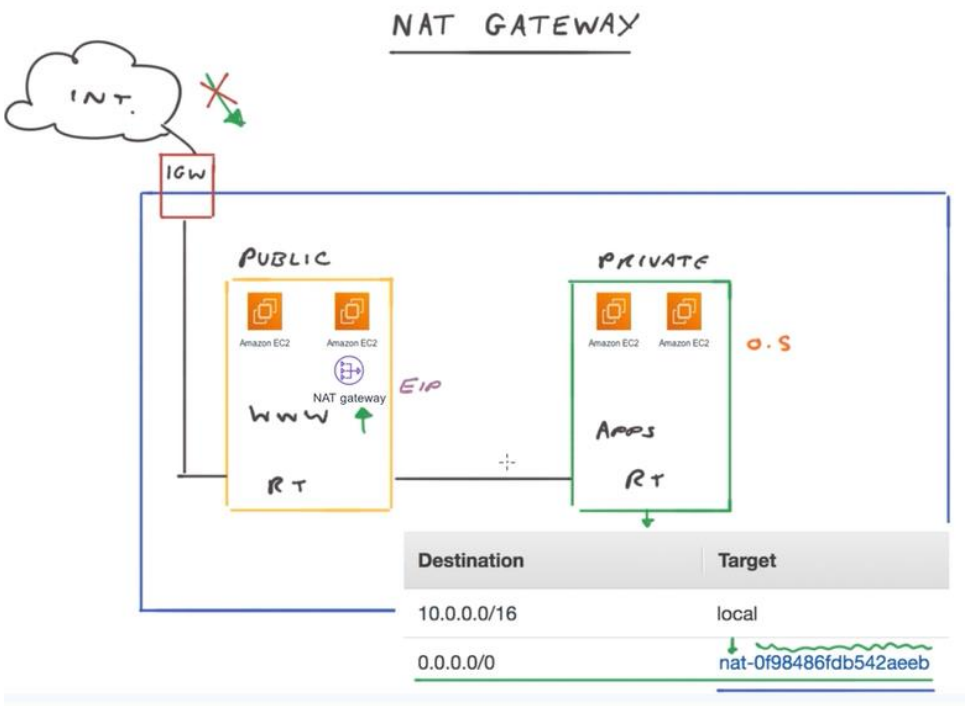
request will have to be explicitly allowed and configured in either the inbound or the outbound ruleset, depending on where the response is coming from.



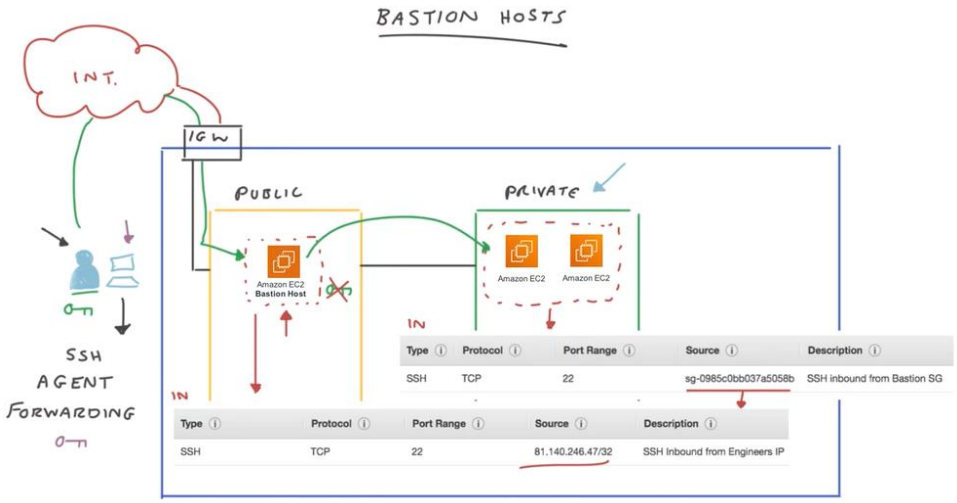
Security Groups - They reside inside Subnet and are stateful. Security Groups are enabled for EC2 instances or other resources.



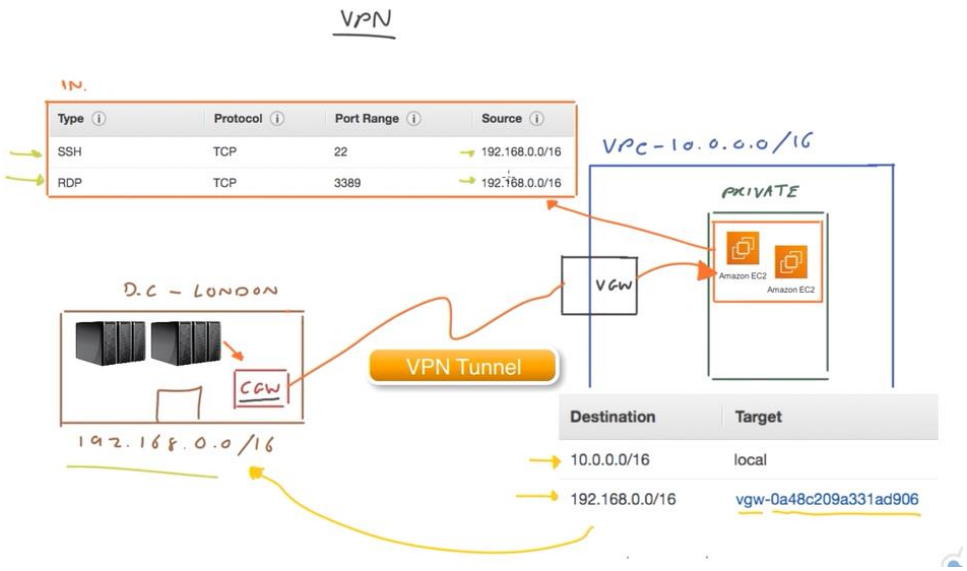
NAT Gateway - The public subnet contains NAT Gateway. If the private subnet wants to access the internet, it goes through the public subnet containing the NAT Gateway. The NAT Gateway takes the request and sends it through the internet gateway. The connections initiated from the internet are blocked.



Bastion Hosts - If an internet user wants to send a request to a private subnet, he can do it via storing the private keys of the EC2 instances which reside in the private subnet at the client side. This is called SSH Agent Forwarding. This way the internet user can SSH by connecting to the internet->internet gateway->public subnet (contains EC2 instance bastion host)->private subnet (contains EC2 instances or other resources).



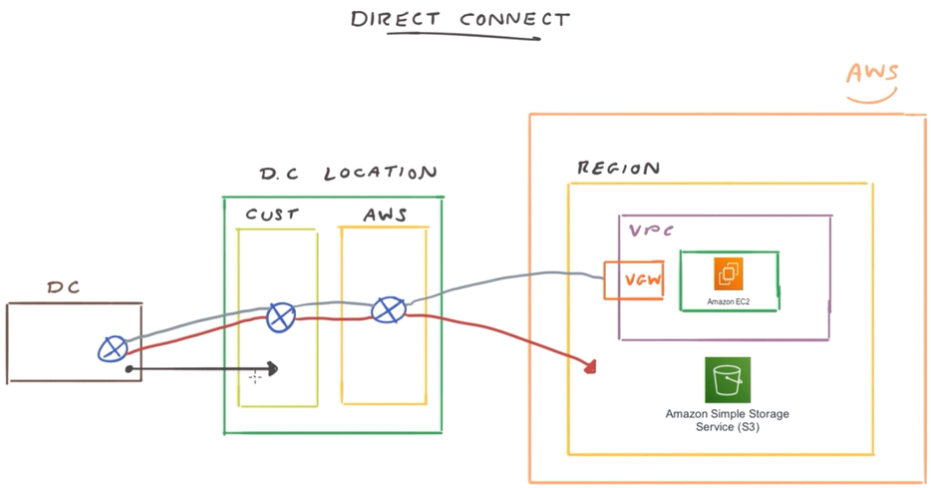
VPN - There is a data center containing a customer gateway and a VPC with a virtual private gateway. To establish a connection between customer gateway and virtual private gateway we need a VPN tunnel. The connection can only be initiated from the customer gateway and not from virtual private gateway. Once the connection is established, virtual private gateway validates whether the request is coming from the source IP by cross checking with its route table. The request then goes to the private subnet->security groups. The security groups protect its instances by checking whether the request is coming from the source IP by cross checking with its route table.



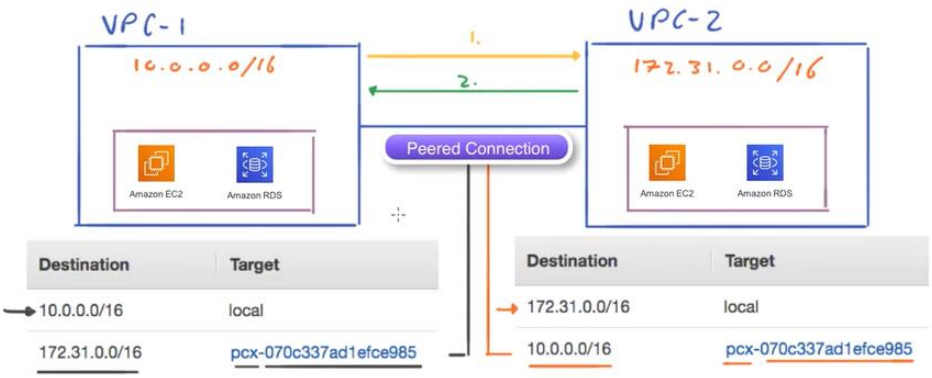
Direct Connect - We have an on-premises data center with a router. The on-premises data center has a private virtual interface and public virtual interface. Direct Connect location sits between the data center and AWS infrastructure. The Direct Connect location contains a customer partner router and an AWS router. The AWS infrastructure contains a region. The region contains a VPC and Amazon S3(for example).

The private virtual interface connects from your on-premises router->customer partner router (Direct Connect location)-> AWS router (Direct Connect location)->virtual gateway. This will allow connectivity to your resources within your VPC.

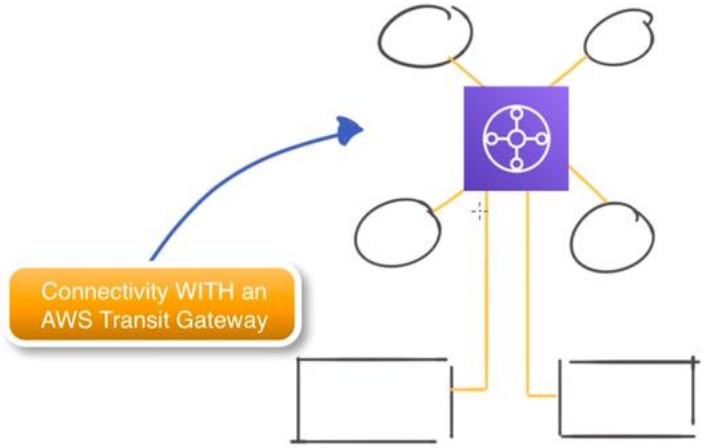
The public virtual interface connects from your on-premises router->customer partner router (Direct Connect location)-> AWS router (Direct Connect location)->Amazon S3. This will allow connectivity to your resources within the AWS region.



VPC Peering - One-to-one connection between VPCs and each VPC has unique IP address. If the IP address is the same the connection does not exist. The one-to-one connection between VPCs can be in the same AWS region or other region as well.



Transit Gateway - It allows all your VPCs to easily communicate with one another and communicate with your data centers as well. All the routing is managed centrally by AWS Transit Gateway.



Database Fundamentals

Amazon Relational Database Service (RDS) - This is a relational database service that provides a simple way to provision, create, and scale a relational database within AWS. Ideal for OLTP use cases. Different database engines are MySQL, MariaDB, PostgreSQL, Amazon Aurora, Oracle, SQL Server. MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server all use Elastic Block Store, for both data and log storage. The database engines that use EBS support general purpose SSD storage, provisioned IOPS SSD storage, and magnetic storage.

Amazon Aurora uses a shared cluster (allows multiple users to simultaneously run workloads such as read/write operations) storage architecture and does not use EBS. The option to configure and select storage options for Amazon Aurora does not exist. Your storage will scale automatically as your database grows.

Amazon RDS backs up your RDS instance to Amazon S3 and you should set a retention period. Manual backups can be done using snapshots and it should be deleted manually and there is no retention period. Relational databases have query analyzer which tells which indexes to use for our query.

Amazon DynamoDB - Amazon DynamoDB is a NoSQL database that means its schemaless. It falls into a category of databases known as key-value pairs. A key value pair is a collection of items or records, and you can look up data using a primary key for each item or through indexes. Amazon DynamoDB is designed to be used for ultra-high performance, with single-digit latency, making this a very powerful database choice used commonly for gaming, web, mobile and IoT applications etc.

When you create a table in the AWS Console, Amazon will configure your table with five read capacity units and five write capacity units. There are two modes: provisioned and on-demand. The provisioned mode allows you to provision set read and writes allowed against your database per second by your application and is measured in capacity units, RCUs for reads and WCUs for writes. Depending on the transaction, each action will use one or more RCUs or WCUs. Provisioned mode is used generally when you have a predicted and forecasted workload of traffic.

On-demand mode does not provision any RCUs or WCUs, instead they are scaled on demand. The downside is that it is not as cost effective as provisioned. This mode is generally used if you do not know how much workload you are expected to experience.

Advantages: Schemaless, fast, serverless, fully managed, highly available

Disadvantages: Eventual consistency (data is replicated however sometimes it returns older data), advanced queries like joins, groups etc. are not possible, workflow limitations in terms of maximum record size of 400 kilobytes and the limit of 20 global indexes and five secondary indexes per table, when provisioned(predefined) databases receive a spike in database requests it will fail.

Amazon ElastiCache - Amazon ElastiCache service improves performance through caching, where web applications allow you to retrieve information from fast managed in-memory data stores (RAMs) instead of relying entirely on slower disk-based solutions. It's generally used to improve read-only performance and is designed to be a temporary data store. Using ElastiCache allows the developer to focus on building and improving apps by simplifying and reducing administrative tasks.

Engines:

- Memcached
- Redis

Both are in-memory key-value engines providing sub-millisecond latency. Redis has more use cases than memcached.

Components:

- Node - A cache node is a fixed sized chunk of secure RAM that is attached to the network.
- Shard - A Redis shard or node group, when working with the API and CLI, is a group of up to 6 ElastiCache nodes.

- Redis Cluster - A Redis cluster contains between one and 90 Redis shards, depending on if cluster mode is enabled or disabled. Data is then partitioned across all the shards in that cluster.

- Memcached clusters - Clusters are a collection of one or more cache nodes. Once you've provisioned a cluster, Amazon ElastiCache automatically detects and replaces failed nodes, which helps reduce the risk of overloaded database, and therefore reduces the website and application load times.

Common Use Cases: Games, social networking sites, real-time analytics etc.

Amazon Neptune - It is a fast, reliable, secure, and fully managed graph database service.

Use Cases: Social Networking, Fraud Detection, Recommendation Engines.

Components:

- Tinkerpop Gremlin - Allows you to query your graph running on your Neptune database, using the Gremlin traversal language.

- Worldwide Web Consortium Sparql - The Sparql query language works with the internet and can be used to run queries against your Neptune database graph.

An Amazon Neptune database cluster is comprised of a single, or multiple database instances across different availability zones. Amazon Neptune has another great feature to help with the durability and reliability of data being stored across your shared cluster, this being Neptune Storage Auto-Repair. Storage Auto-Repair will automatically find and detect any segment failures that are present in the SSDs that make up the shared volume, and then automatically repair that segment using the data from the other volumes in the cluster. This ensures that the data loss is minimized and the need to restore from a failure is drastically reduced. To ensure high availability, each cluster maintains a separate copy of the shared volume in at least three different availability zones.

To connect to Amazon Neptune database, you can use endpoints.

- Cluster endpoint - It points directly to the current primary database instance of that cluster. This endpoint should be used by applications that require both read and write access to the database. If a primary instance fails and you have a read replica available, then Neptune will automatically failover to one of these replicas and act as the primary, providing read and write access. When this happens, the cluster endpoint will then point to the new primary instance without any changes required by your applications accessing the database.

- Reader endpoint - This endpoint is purely used to connect to any read replicas. Only a single reader endpoint exists, even if you have multiple read replicas. Connection served by the read replicas will be performed on a round-robin basis, and it's important to point out that the endpoint does not load balance your traffic in any way across the available replicas in your cluster.

- Instance endpoint - For every instance within your cluster, including your primary and read replica instances, they will each have their own unique instance endpoint that will point to itself. This allows you to direct certain traffic to specific instances within the cluster. You might want to do this for load balancing reasons across your applications reading from your replicas.

Amazon Redshift - It is a fast, fully managed, petabyte-scale data warehouse. It operates as a relational database management system and is compatible with other RDBMS applications. Ideal for OLAP use cases. Redshift is based on PostgreSQL, but it contains several differences from PostgreSQL.

A data warehouse goes through the following processes:

- Extraction is the process of retrieving data from online, brick & mortar, legacy data, Salesforce data and many others. After retrieving the data, ETL is to compute work that loads it into a staging area and prepares it for the next phase.

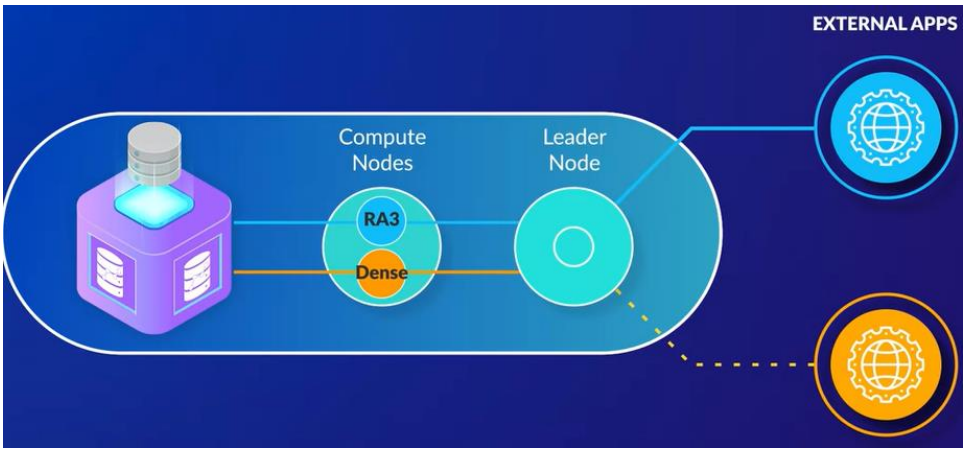
- Transformation is the process of mapping, reformatting, conforming, adding meaning to the data that can be easily consumed.

- Loading involves successfully inserting the transformed data into the target database data store, or a data warehouse.

Components:

Amazon Redshift Cluster - A cluster is the main component of Amazon Redshift service. Every cluster will run its own Redshift engine, which will contain at least one database. A cluster is effectively a grouping of another component, and these being compute nodes.

Each cluster will contain at least one compute node. However, if the cluster is provisioned with more than one compute node, then Amazon Redshift will add another component called a leader node. The leader node of the cluster has the role of coordinating communication between your compute nodes in your cluster and your external applications accessing your Redshift data warehouse.



Amazon Redshift Cluster is three times faster than other cloud data warehouses. Amazon Redshift Cluster has features like: Massively parallel processing, Columnar data storage, Result caching. You can associate 10 different IAM roles to your Amazon Redshift cluster.

Amazon Quantum Ledger Database (Amazon QLDB) - It's a fully managed and serverless database service, which has been designed as a ledger database. Infrastructure administration is managed by AWS. With Amazon QLDB, you can rest assured that nothing has or can be changed through a database journal.

Data for your QLDB database is placed into tables of Amazon ion documents (or JSON documents). These ion documents are open-source, self-describing data serialization format, which is a superset of JSON. The document stores both structured and unstructured data. It's append only and maintains history.

Use Cases: Insurance Claims (Amazon QLDB is an immutable append-only framework, preventing the ability to manipulate previous data entries, which helps to prevent fraudulent activity), payroll processes.

Amazon QLDB uses two different methods of storage:

- Journal storage is used to hold the history of changes made within the ledger database. So, this will hold all immutable changes and history to the ion documents in your table.
- Index storage is used to provision the tables and indexes within the ledger database, and it's optimized for querying.

Amazon QLDB is integrated with Amazon Kinesis through QLDB streams. 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. With Amazon Kinesis, you can ingest real-time data such as application logs, website clickstreams, IoT telemetry data, and more into your databases, your data lakes and data warehouses, or build your own real-time applications using this data.

Amazon DocumentDB(With MongoDB Compatibility) - It runs in a Virtual Private Cloud and is a non-relational (NoSQL) fully managed service, which is highly scalable, very fast, and highly available. With AWS Database Migration Service (DMS), you can transfer data to and from between existing Mongo DB data and Amazon DocumentDB. Its architecture is like Amazon Neptune. DocumentDB performs automatic backups. Automatic backups allow you to restore back to any point in time during your retention period, known as point-in-time-recovery. These backups are automatically stored on Amazon S3 for durability and availability. You can set the retention period by yourself, and it should be a minimum of 1 day.

Amazon Keyspaces(for Apache Cassandra) - Keyspaces is a serverless, fully-managed, highly scalable, highly available service, and compatible with Apache Cassandra(It's a free, open-source, distributed, wide column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure).

When using Amazon Keyspaces, any clients that want to connect to your Cassandra tables will require a TLS connection. Amazon Keyspaces offers low latency, and existing integration with Cassandra. When working with Amazon Keyspaces, you'll need to use

CQL, the Cassandra Query Language which is like SQL. There are several ways to run queries using CQL. Firstly, within the Amazon Keyspaces dashboard within the AWS management console, next using CQLSH client, or you can run them programmatically using an Apache 2 licensed Cassandra client driver.

Keyspaces offers two different throughput capacity modes: On-demand (this can be good for unknown or unpredictable workloads), Provisioned (this can be good for predictable workloads).

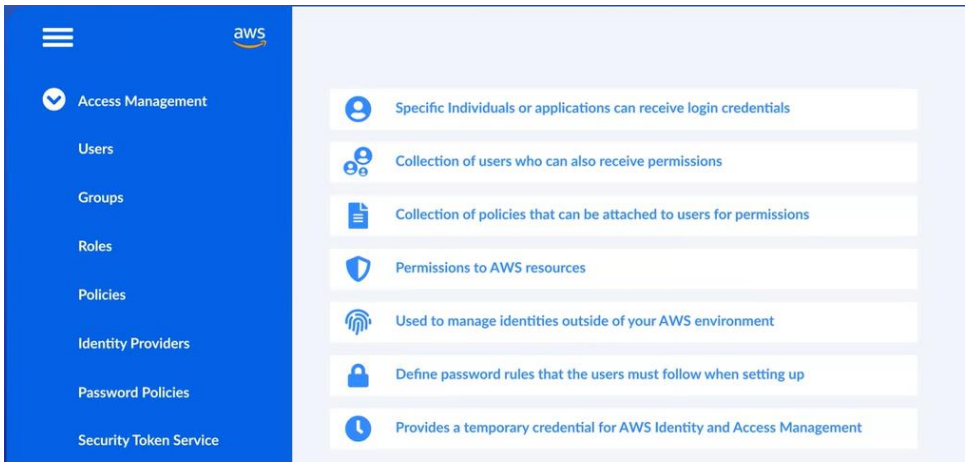
Identity & Access Management

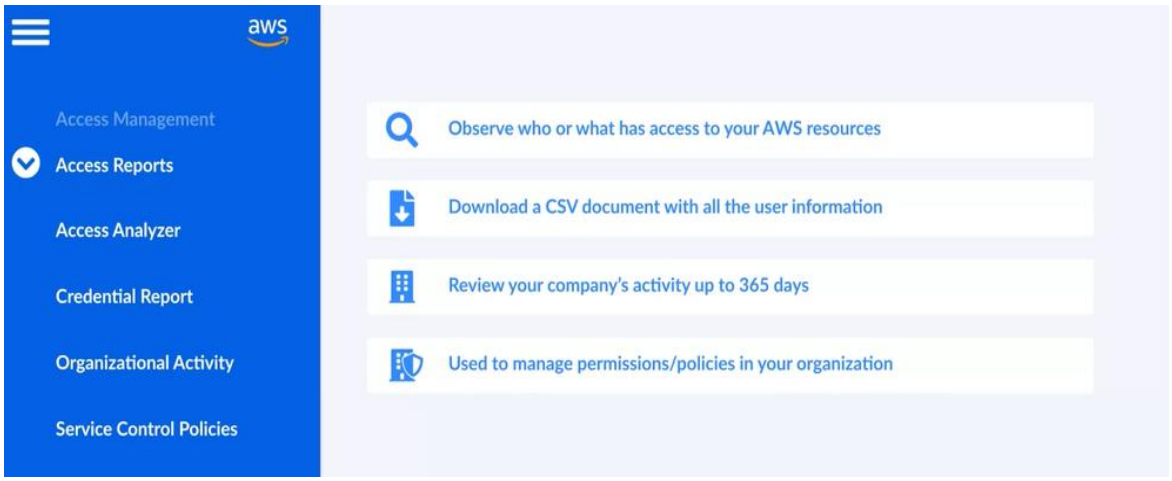
IAM can be defined by its ability to manage, control, and govern authentication, authorization, and access control mechanisms of identities to your resources within your AWS Account.

Identity Management consists of AWS username and password.

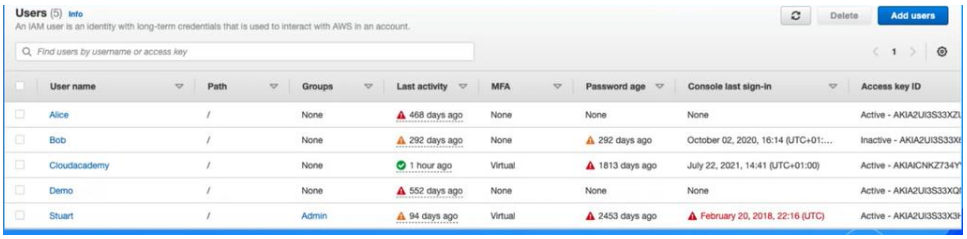
Access Management provides access to secured resources. It consists of Username and Password (Authentication and Verification), Multi-Factor Authentication, or Federated Access (users external to AWS access resources securely without having to supply AWS user credentials from a valid IAM user account. For example, using google account).

Features:





User Dashboard:



A user is an individual, system, or application that interacts with AWS programmatically. Users are objects representing an identity used to authenticate to your AWS account with an associated set of permissions.

Access Key ID: It is made up of 20 random uppercase alphanumeric characters.

Secret Access Key ID: It is made up of 40 random upper and lowercase alphanumeric and non-alphanumeric characters.

On the Users Dashboard, we have a green tick, which symbolizes activity was measured within the last 90 days (about 3 months); an amber exclamation mark showing activity was last measured between 91 and 365 days (about 12 months) ago; and a red exclamation mark, which highlights anything older than 365 days (about 12 months) ago. Users who use MFA will be shown as virtual.

The credential report will only be generated once in 4 hours.

Access Advisor: It shows you which services a user can access based on their current permissions and the last time they were accessed.

Path: If you create your users using the IAM API or the AWS Command Line Interface, then you can specify a path structure for your users. This is useful when you have a large organization, and you want to define a management structure to help you organize your users more effectively.

IAM User Groups and Roles:

It's best practice to apply permissions to groups, instead of users. From a limitation perspective, your AWS account has a default maximum limit of 300 groups. Also, a user can only be associated with 10 groups. Each group can contain 10 different policies attached at once.

A brand new IAM user has no password and no access key (neither an access key ID nor a secret access key), and no credentials of any kind. A brand new IAM user has no permission to do anything. By default, the user is not authorized to perform any AWS actions or to access any AWS resources.

AWS Service roles allow you to apply your own customer managed or AWS Managed policies. AWS Service Roles (For ex. Instance Profile) allows AWS services to assume a role to access other AWS resources within your account on your behalf. This will automatically configure the credentials for you.

AWS service-linked roles come pre-configured with a specific set of read-only AWS-managed policies that can only be used by that service. Examples of AWS service-linked roles are AWS ServiceRoleForAmazonSSM, AWS ServiceRoleForCloudTrail, AWS ServiceRoleForCloudWatchEvents.

In terms of security, you should always associate a Role to an EC2 instance for accessing AWS resources.

Roles for Cross-Account Access offers two options. Providing access between AWS accounts that you own, and providing access between an account that you own, and an AWS account owned by another party.

SAML 2.0 is generally used to authenticate your employees using existing directory services.

IAM AWS Policy Types:

IAM Policies are always aggregated.

- Identity-based policies: These policies can be attached to users, user groups, or roles and they control what permissions each of those entities have. Identity-based policies can either be managed or inline policies.

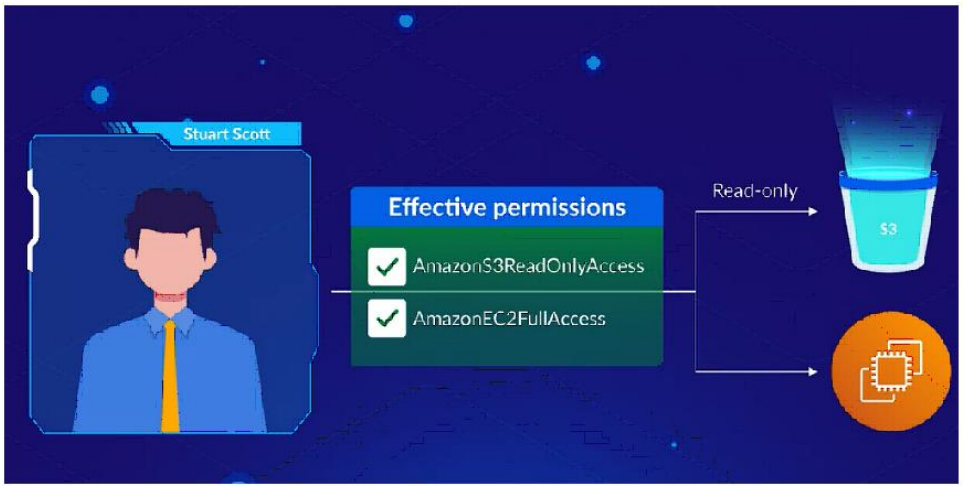
Managed policies are saved within the IAM library of policies and can be attached to any user, user group, or role as and when required, and the same policy can be attached to multiple entities. Managed policies consist of AWS managed policies and customer managed policies.

AWS managed policies are policies that have been pre-configured by AWS. Customer managed policies are those that you have created yourself.

Inline policies are specific to that one user, user group or role, creating a one-to-one relationship.

- Resource-based policies are effectively inline policies that are associated with a resource instead of an identity. If you have been using Amazon S3 for any extended period, you may have come across S3 bucket policies, which is a common resource-based policy where permissions to the bucket are defined at the resource level.

- Permission boundaries: They act as a guide rail to limit the maximum level of permissions that the user or role can be given.



- Service control policies (SCP): For example, let's say a user within an AWS account had full access to S3, RDS and EC2 via an identity-based policy. If the SCP associated with that AWS account denied access to the S3 service, then that user would only be able to access RDS and EC2.

IAM Policy Structure:

IAM Policies are written in JSON.

While creating an IAM policy, it includes many elements. The elements that a policy can contain are as follows: Version, Id, Statement, Sid, Effect, Principal, NotPrincipal, Action, NonAction, Resource, NotResource, Condition, and Supported Data Types.

Statement: It defines the main element of the policy, which will include other sub-elements, including the SID, effect, action, resource, and condition.

SID: It is an optional identifier you provide for a policy statement.

Effect: This element can be set to either Allow or Deny.

Resource: It specifies the object or objects that the statement covers.

Condition: Lets you build expressions in which you use Boolean condition operators (equal, less than, etc.) to match the condition in the policy against values in the request. This is optional.

Action: It describes whether the specific API calls for a specific service will be allowed or denied.

An IAM Policy can be generated in three ways using: Policy Generator, creating a policy from scratch, Copy an existing AWS Managed Policy.

Policy Evaluation Logic:

There is an order in which policies are evaluated: First Organizational Service Control Policies, then Resource-based policies, then IAM permission boundaries, and then finally Identity-based policies.

The rules for reviewing permissions across multiple policies in a single account are quite simple and can be summarized like this: by default, all access to a resource is denied. Access will only be allowed if an allow has been specified within a policy associated with the principle. If a single deny exists within any policy associated with the same principle against the same resource, then that deny will overrule any previous allow that might exist for the same resource and action. So, to reiterate, an explicit deny will always take precedence over an allow.