**AWS CLOUD COMPUTING**

**CLOUD COMPUTING:**

* It is the on-demand delivery of compute power, database storage, applications and other IT resources.
* The services are provided through a cloud platform with pay as you go pricing.
* One of the best features of this service is it is scalable. You can increase or decrease the service depending on your requirement.
* You can access these services instantly from a single web application.
* Instead of managing your own servers we can simply access these services through AWS. The AWS is responsible for maintaining your servers, database, applications and backup with load balancing.

**Cloud computing:**

The cloud computing is storing data on remote servers, processing data from servers and accessing the data via internet.

Through cloud computing we can access the data from anywhere in the world and can manage the applications for that all we need is a good internet connection.

**Service models**:

1. SaaS:

Cloud provider leases an application or a software for the user. Ex: Gmail. We just have to use it that’s all.

1. PaaS:

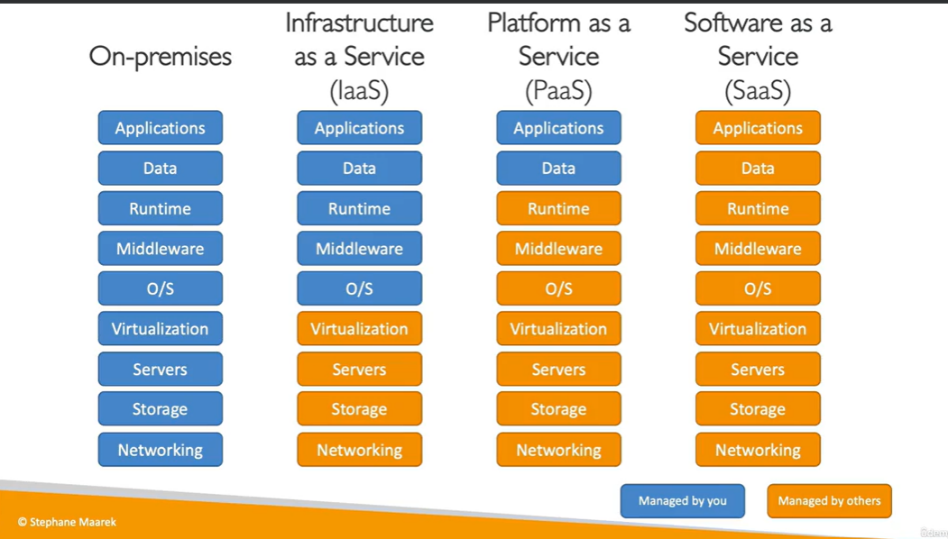
A platform is provided for the users to create our own applications. Ex. Google app engine. Where we can create and let others use these applications.

Ex. Elastic beanstalk

1. IaaS

An infrastructure is provided where we can use any operating system, technology and applications we want to build. Required resources are provided.

Ex.EC2



**Deployment Models:**

1. Public cloud

The service provider make the applications, storage etc available to the general public over the internet. Ex. Google drive.(azure, aws and gcp)

1. Private cloud

Private companies uses these models for few number of people. It has security feature to restrict unauthorized access. EX.rackspace

1. Hybrid cloud

This is a mix of both private and public cloud.

Cloud providers are GCP, AWS, Azure

**AWS:**

It is a secure cloud service platform offering compute power, data storage, content delivery and etc. It is six times more compute power than other cloud services.

Advantage of AWS:

* Flexibility
* Cost effective
* Scalability( easy to increase and decrease the usage)
* Security

**AWS architecture:**

AWS infrastructure is divided into Regions and Availability Zones.

**Regions**: Different locations across the world. Multiple data centers are there in regions.

**Availability zones:** The data centers in the regions are called availability zones.

Each region(country) has multiple availability zones(areas).

**AWS Global services:**

* IAM ( Identity and access management)
* Route 53 ( DNS service)
* Cloud Front (content Delivery network)
* WAF (web application Firewall)

**AWS region specific services:**

* Amazon EC2
* Elastic beanstalk
* Lambda
* **AMI,** S3 buckets

1. **IAM:**

* Users and Groups. Users are people within your organization and can be grouped.
* Users can be members of multiple groups. Groups cannot be grouped.
* For applying privileges and permissions groups are used.
* Users or groups can be assigned JSON documents called IAM policies.
* While creating an aws account root user is created. But we should not use root user for any works. Just like Linux, windows.

****

* This is a json document that defines permissions to users, groups and other group members.
* All the users should not be given with permission to do everything (least privilege principle).
* Create an IAM user. Just like creating a user in windows user with permissions.
* Also along with create a new group with the user in it.

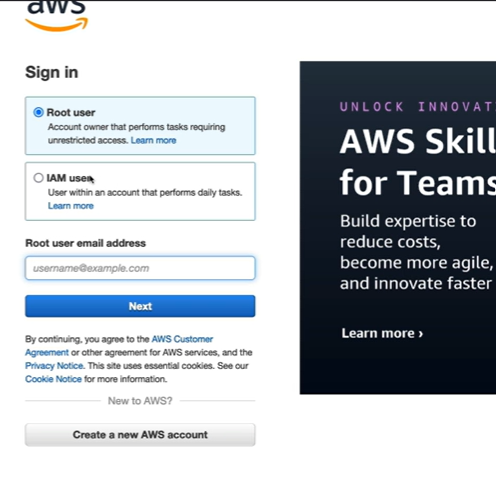
Id: identifier or resource name(optional)

Principle: which account or user or role to which this policy will be attached

Effect: allow or deny

Action: list of actions

Resource: list of resources to which policy is attached

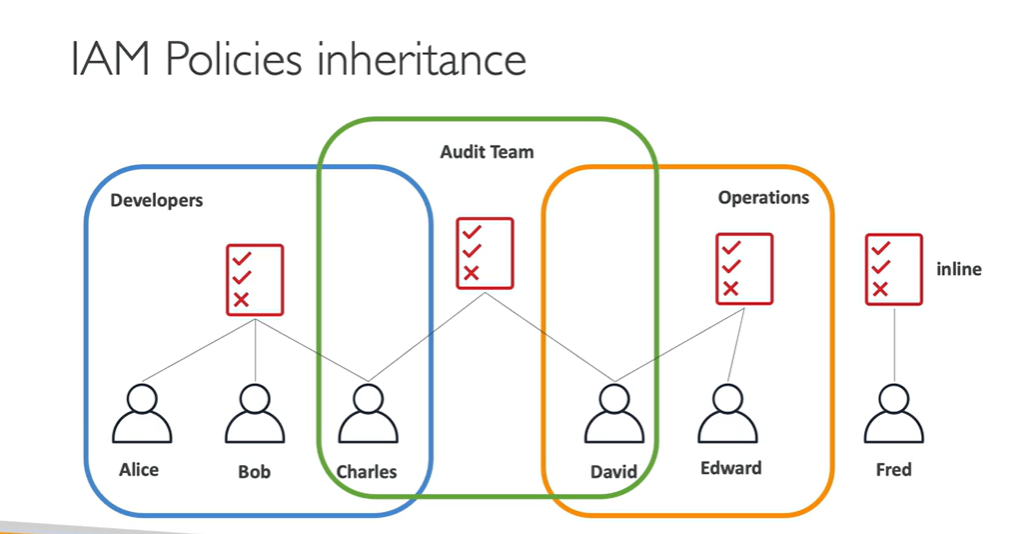


Here the root user can login and the IAM users can also login.

The IAM users are created using the root user.

Similar to windows and linux, multi user login concept.

The account id is same for all users.



* These policies are assigned to groups and the members of the group also gets the same permissions that of the group.
* The root user can restrict the user to do some works. Like creating new users, read only, etc.
* We can create our own Jason iam policies and attach to users or groups.

**Protecting users: (only root users can do this)**

* Passwords: we can set up password policies like size, use of alphanumeric, case sensitive, etc. along with password expiration.
* In account settings there will be password policies where we can customize the password policies. All editing options are available in that settings.
* Multi Factor Authentication(MFA): combination of password we know + device we own.

The advantage is even if password is stolen or hacked, the account is not compromised.

* In security credentials we can add MFA with simple basic steps.

MFA devices are Google Authenticator, Authy(AWS) or else a physical device called Universal second factor key (U2F), Gemalto(used in TechM login)

Users can access their AWS account through

1. AWS management console( Password + MFA) (done)
2. AWS CLI access key
3. AWS SDK (Software development kit)

Access keys are generated by users. It gives Access key ID and Secret access key

**AWS CLI:**

An interface through which we can access the AWS services using commands. It has direct access to AWS API’s. We can develop scripts to manage our services and resources. An open source and free in github.

**AWS SDK:**

Set of libraries or language specific APIs. Enables you to access the resources programmatically. We can use any programming language like Java, Python, Ruby, PHP, .net, etc.

**Create an Access key**: for the user

Open IAM > Users > create access key > CLI > Download it.

To login in AWS CLI using the user credentials

* aws configure

Access key:

Secret access key:

Default: region:

Output:

* aws iam list-users

output will be generated all users

the output is based on the permissions of the user given by the root user. If permission is not given then it won’t generate output.

A computer code with numbers and letters

Description automatically generated CLI

Output

**IAM Roles:**

It is simply like restricting the services or resources to know about the user information. Resources like EC2, Lambda functions.

**Security Groups:**

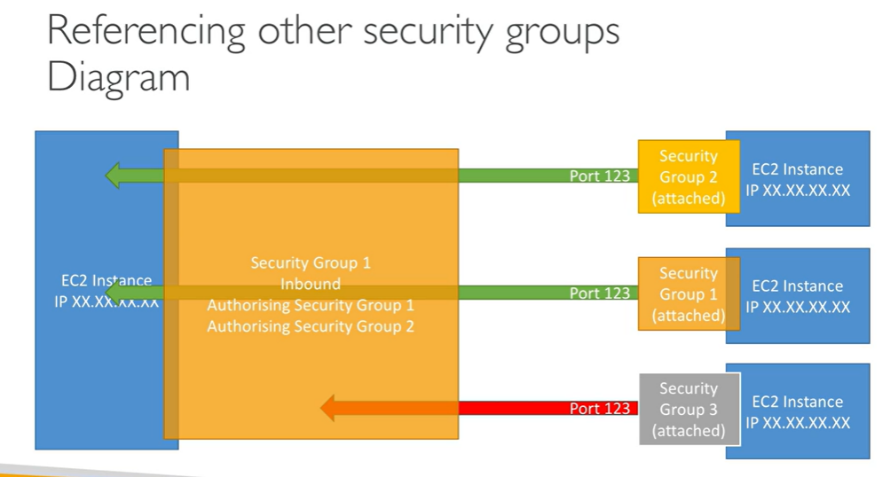
Security groups are the virtual firewall for your instances to control inbound and outbound traffic. We can add the rules like a particular protocol can have access ex: http, ssh, tcp with any IP address.

1. **EC2 (Elastic compute cloud):**

* It comes under IAAS and offers various services like

Virtual machines, virtual drives for storage, load balancers and auto scaling

* EC2 configuration options are OS, RAM, CPU processors, storage space, network and security groups.
* The private IP of the instance will remain the same whereas the public IP changes if the instance is stopped and restarted.
* Instances are different types depending on their usage and user requirements like general purpose, storage optimized, compute optimized, memory optimized etc.
* Security groups of instances acts as firewalls and they regulate access to ports, ip addresses, inbound and outbound networks.
* Based on your IP address and port no. the security group allows you to access the EC2 instance (This is for inbound) and for outbound the Ec2 instance can access any network to fetch the data.
* Security groups are region specific and are outside the EC2 instance.



How to login to instance through linux:

* Download the key pair in pem format and save it in linux in any directory.
* Change the file permissions using chmod 0400 filename.pem.
* Now use the command ssh -i filename.pem ec2-user@public ip of instance
* After login into ec2 user
* Check whether we can ping google or not

We can add the IAM roles to instances so that the authorized people can use the instances.

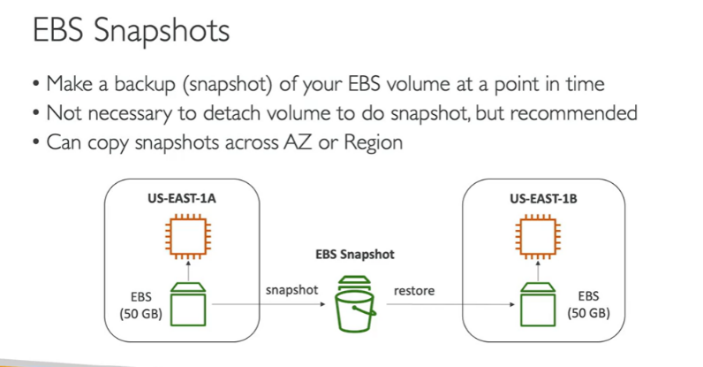
**Key Pair:**

* The key pair is the access key to the instance it a combination of public and private key.
* Through SSH we can login into the instance using the key pair.

1. **EBS (Elastic Block store) volume:**

* These can be attached to our instances to persist data, even after their termination. They can only be mounted to one instance at one time only and are specific to the availability zones.
* It is like a network USB stick or a virtual USB that we can attach to our instance. Even after the instance is terminate the EBS remains still with the data in it.
* They can be easily detached and attached to another Instance easily. We cannot attach the EBS of one availability zone (us-east-1a) to another instance in (us-east-1b).
* **A diagram of a computer network

  Description automatically generated**To do that we need to take a snapshot.
* Multiple EBS can connect to one Instance which are of same AZ.
* When we create an instance the EBS volume is created for the instance and attached to it automatically.
* This root EBS is deleted when the instance is terminated. If you want to change that you have to change the options at the beginning of the instance creation.
* You can check the volumes at the Storage options of the instance.
* The EBS service provides the options to create a new EBS volume.
* The availability zone should be same as that of the instance.
* After creating volume, we can check attach options in Action button. It will provide the instance info to attach them.
* Even If you create a volume with another availability zone and try to connect with this instance it won’t allow you to attach it.
* If you don’t need the EBS you can delete that.

**EBS snapshot:**

* It is a back up procedure. Can copy snapshots across AZ or regions.
* To migrate the data of one EBS of a AZ we need to take snapshots of the EBS and restore that data from the EBS snapshot to another EBS of different AZ.
* It has features like EBS Snapshot archive which takes 24 to 72 hours and EBS recycle bin to restore the EBS which were accidentally deleted.
* The simple way is that we can go to actions at the EBS volume and click take a snapshot.
* The snapshot is created and available at the EBS options snapshot.
* To change the Availability zone we need to click the copy at the snapshot EBS and select another AZ.
* We can recreate the snapshot into the EBS volume and can use it at the desired AZ by clicking the recreate EBS options. We can use the recycle bin to restore the snap shots also.

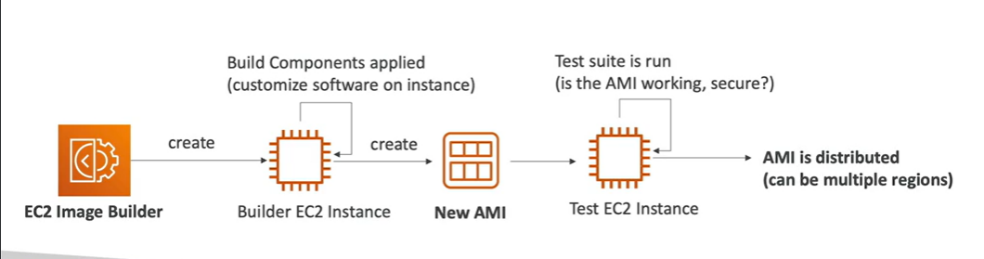
1. **AMI (Amazon Machine Image)**

* AMI are the customization of the EC2 instances like add our own software , configuration, OS, monitoring, etc.
* AMI are built for specific regions and can be copied across the regions.
* **A diagram of a custom ami

  Description automatically generated**We have to make and maintain our own AMIs or else amazon provides Public AMIs which are the instances.

**Create an AMI:**

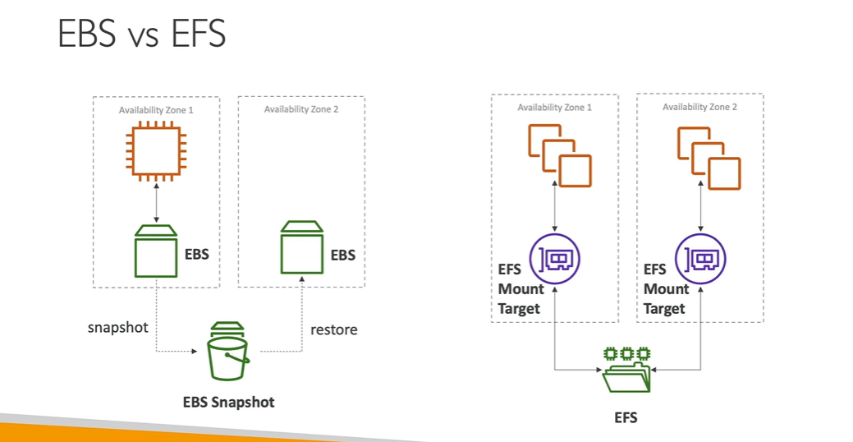
* First create your instance or use the old instance.
* Right click on instance and select image and template option > create image give a name and create it.
* Check the created AMI in the running or not in the AMI resource option.
* There you can have options called launch instance from AMI. You can customize that and launch it.
* It is like replicating the instance and using it for another region.

**5. EC2 Image Builder:**

* Used to automate the creation of virtual machines.
* The EC2 image builder will build an instance and create an AMI, this AMI is tested and is distributed across multiple regions.

EC2 Instance store is one of the ways to store the data just like EBS and it is more powerful in IO through put and has high performance. But it has disadvantage like when instance is terminated then the store gets deleted.

**6. Elastic File System EFS:**

* This is another storage that can be attached to an instance.
* EFS can be attached to 100 EC2 instances at a time and with linux instances it can be used across multiple AZ. Highly available and scalable and expensive.

EBS can be attached to EC2 instance of the same AZ. If we can to attach it to another AZ we need to take a snapshot and restore it another availability zone.

EFS can be attached to multiple EC2 instances of same AZ or different EC2 instance of another AZ using EFS mount target.

**EFS Infrequent Access (EFS IA):**

It is cost optimization technique where we don’t use the EFS frequently then it can be moved to IA which is cost effective and we can access them when ever we require.

**Amazon FSx:**

Another file system specifically for the OS like Windows, Linux. Etc

**Amazon FSx Windows File server**: For Windows server

**Lustre**: It is for the Linux and cluster, it a high performance.

For replicating the EBS volumes we need Snapshots .

For replicating the EC2 instance we need AMI( Amazon Machine Image).

A white text on a white background

Description automatically generated Summary

**Scalability and High Availability:**

* It means the applications or systems can handle greater loads by adapting.
* Types of Scalability are

**Vertical Scalability:**

* It means we can increase the size of the instance from t2.micro to t2.large. It depends on the hardware limit and mostly used for databases.

**Horizontal Scalability:**

* It means increasing the number of instances for your application.

**High Availability:**

* It is similar to horizontal scalability like increasing the number of instances in multiple availability zones. Even if one fails the other still runs.
* Run instances of same application in multiple availability zones.

**7. ELB ( Elastic Load Balancing):**

* Load balancers are the servers that forward the internet traffic to multiple servers. These are the backend EC2 instances and are managed by the AWS.
* Spread the load across multiple servers or instances. Failures can be avoided.
* These can available in multiple AZ so it is highly available.
* ELBs are managed by the AWS like upgrades, maintenance, etc.

**Types of Load Balancers:**

* Application load Balancer( Http/https) - layer 7
* Network Load Balancer (TCP/UDP) - layer 4 (when many people are trying to access the application.)
* Gateway load balancer - layer 3 (security purpose)
* Classic Load Balancer - layer 4&7 (expired in 2023)

A diagram of a network load balancer

Description automatically generated

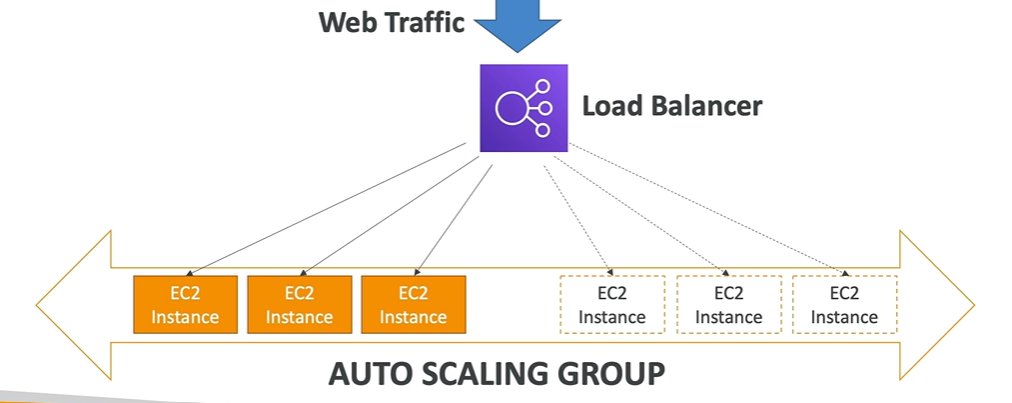
**Target Group:**

A group of instances or lambda or IP to add in for Load balancing. The instances in this target groups are used for routing the traffic into them.

The load balancer provides a DNS name to access the servers. If you refresh the IP address changes between the two instances.

**8. Auto Scaling Groups (ASG):**

* As the load on the servers can increase or decrease so in the cloud we can create and get rid of the severs very quickly.
* The Auto Scaling Groups will do that thing for you.
* Scale out (add Ec2 instances) to match the increased loads.
* Scale in (remove Ec2 instance) to match the decreased loads.
* It will automatically register the new instances to a load balancer and also replaces the unhealthy instances.



Auto Scaling Groups in the Aws works with Load Balancer to control the traffic flow depending on the incoming request.

While launching the ASG we have to create the template and customize the instances that can be created automatically when the load is increased. Scaling are different types

**Manual Scaling:** Update the size manually on ASG.

**Dynamic scaling:** Respond to change on demand.

* Simple or step scaling: Dynamic scaling is based on the Cloud Watch alarm when triggered.
* Target Tracking scaling: We set a limit for the CPU usage and if it exceeds then the scaling will start.
* Scheduled Scaling: Anticipate the scaling based on usage pattern.
* Predictive scaling: Uses ML to predict the future traffic.

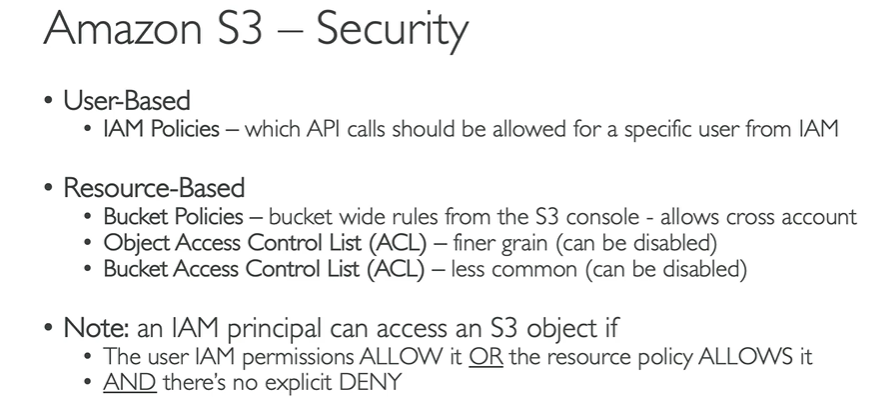
**Overview:**

* IAM create users for console login and groups, security groups for permissions
* EC2 virtual machines
* EBS storage volumes that can be attached to EC2 virtual machines.
* EC2 instance store a volume that is deleted when EC2 instance is terminated
* EBS snapshot and archives For restoring or copy of the EBS volumes
* AMI creating a copy of the EC2 virtual machine with same specifications
* EC2 image builder an automatic EC2 virtual machine builder
* ASG vertical scaling is increasing the size of the EC2 and horizontal scaling is the increasing the number of EC2 instances.
* ELB depending on the load the server spreads the load across multiple servers.

**9. Amazon S3 (simple storage service)- Buckets**

* It has the infinitely scaling storage capacity.
* Many services prefer this as it has a vast storage capability.
* It is used for many purposes like Backup and storage, Disaster recovery, application hosting, media hosting, software delivery, static websites, etc.
* It allows the people to store their object(files) in buckets.
* These buckets should have globally unique name and are region specific.
* The name should not contain upper case letters.
* Objects in the buckets have keys and these keys are the full path.
* The key is composed of prefix + object name.
* s3://my-bucket/my-folder1/another-folder/my\_file.txt. Max size of object is 5TB(5000GB

We can create our own S3 bucket with a unique name which was not used by anyone before.

After creating we can upload the files and create folders and can view the files inside the folders.

The user of the AWS console can only access the files in S3 bucket. The public cannot access them. For this in S3 bucket permissions the Public Access can be allowed by removing the blocking tick mark.

After this using the public url the content can be seen.

**S3 Bucket policy :** The S3 bucket policies are used for allowing or restricting the public access to the objects in the bucket.

There is a policy generator option which can help in creating a policy in Jason format script and can copy paste it.

Without bucket policy the public access is not possible.

**AWS S3 storage classes: per Gb per month**

* S3 standard for 1-15 seconds( costly)
* S3 Glacier & S3 glacier deep archieve (12-48 hrs) cheap
* S3 glacier instant and flexible retrieval ( 1-5 min) cheap

**S3 websites:**

* S3 can host static websites and make them available to public users.
* If we don’t allow the public access then the 403 forbidden error appears
* The static website hosting is available in properties of the bucket.
* Enabling versioning option is best in S3 as the previous version file can be updated with the latest and the older version will be available for usage but it will be hidden.

**S3 Replication:**

* The replication can be done across regions and also in same regions.
* Cross region replication and Same region replication are done only when versioning is enabled.
* The copy is asynchronous simply the data or contents of the one s3 bucket can be copied automatically to the replicated s3 bucket.

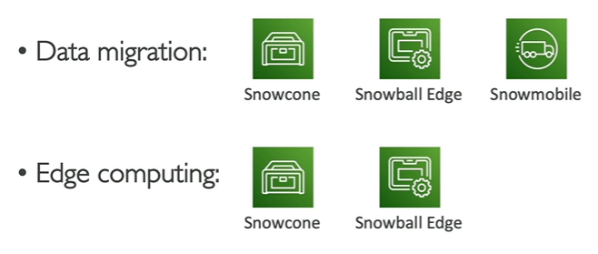
**S3 durability and availability:**

* Durability: Aws s3 has high durability up to 10,000 years and same for all s3 storage classes.
* Availability: The measure of high readily a service is available and it varies depending on the type of storage class.
* Example S3 standard is not available for 53 minutes a year.

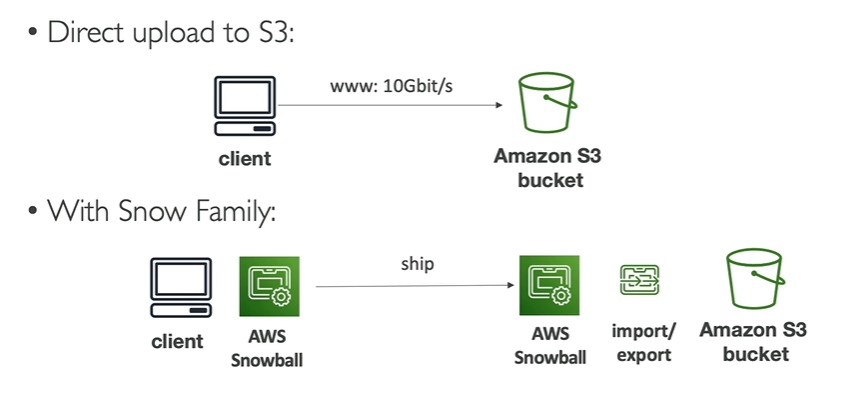
**S3 Encryption:**

* In AWS we have two types of encryptions
* **Client side** encryption where the file is encrypted before uploading it into the S3 bucket.
* **Server side** encryption is where the sever will encrypt the files after uploading into the bucket.

**AWS Snow Family:**

* Portable devices used for collecting and processing the data and migration of data at the edge, and into and out of AWS.

**Data Migration with snow family:**

* The time to transfer the data over the network takes lots of time. By using the Snow family devices we can reduce the time.
* An offline device to perform data migration and for this AWS sends a device through post and you copy the data into that device and send it back to AWS.
* The device that AWS sends is called snow ball and it is used when your data migration takes more than a week.

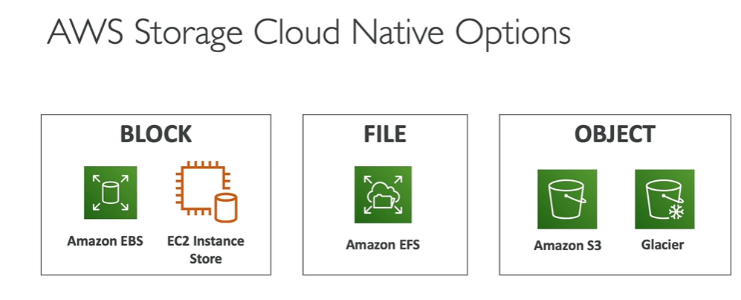
* SnowBall Edge (moves TB or PBs of data)
* Snow Cone (used for 8 Tb to 14 TB of data) smaller then snow ball
* A close-up of a computer

  Description automatically generatedSnowmobile ( a large truck that holds 1EB=1000PB=1000000TB)

**Edge Computing:**

* Process data while it is being created on an edge location and edge location is where there is no internet connectivity or limited connectivity.
* We use Snow cone edge devices with less storage or compute power.

The data transfer to these devices is done by the connecting the snow device through the cables and using only CLI we can communicate with snow device.

For this AWS provided AWS OpsHub which we can install in our laptop and manage the Snow devices. This gives a graphical interface to configure the snow device.

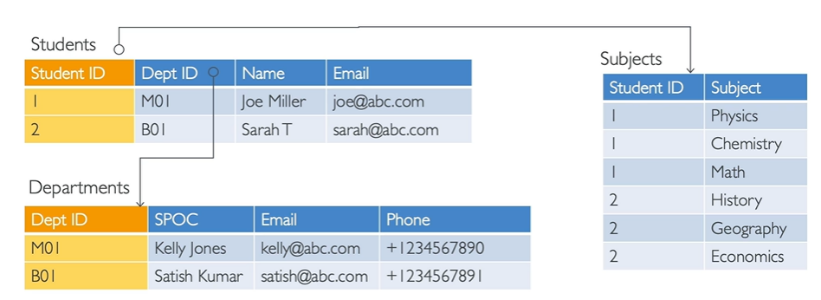
**AWS storage gateway:**

* Bridges the on-premises data and cloud data in s3. The hybrid storage service allows you to seamlessly use the AWS cloud.
* The gateways are File gateway, volume gateway and tape gateway. These gateways bridges the data on-premises and cloud data.

**AWS Database:**

* Storing of the data can be done with any service like EBS, EFS, EC2 instance store and S3.
* But storing a structured data and relational data is done through a database. For this we can build indexes and for retrieval we can query for the data.

1.Relational Database:

* Looks like an excel spread sheet with the link between them.
* We can use SQL to fetch the data in the database.

2. No SQL (Non Relational database):

* These are built for a specific purpose for specific data models.
* This is flexible and scalable unlike RD. A JSON file format is used for the NoSQL.

**AWS Relational Database:**

* It allows us to create the databases and are managed by the AWS. For this we can use Postgre SQL, MariaDB, MySQL, Oracle, Aurora (AWS proprietary).
* Advantages of this is the AWS will monitor our database, back and storage , disaster recovery, maintenance for upgrades, scalability etc.
* We cannot use SSH for the RDS.
* Connecting to the ELB and then to the instances in the ASG which will be connected to the Amazon RDS.
* Read Replicas: helps in creating the multiple copies of Database for high traffic.
* Multi-AZ: The database is made available in multi AZ and regions.

Elastic cache DB: It is fast in read and write from the cache.

A diagram of a computer network

Description automatically generated

**Amazon Aurora:**

* The instances are directly connected to the Aurora. This Aurora supports Postgre SQL and MySQL.
* As it is backed by AWS the performance of Postgre SQL and MySql will increase and need not have to worry about the storage.
* Aurora Serverless is also in serverless where everything like scaling and initiation is automated.

**ServerLess:**

* This doesn’t mean there are servers but it means there I no need for maintaining the servers.
* Like we just deploy the code and need not have to manage the servers.
* In AWS we have server less services like Amazon S3, Lambda, Dynamo DB, Fargate, etc.

RDS deployment:

* Read replicas (multiple data base are replicated from single database for read when more applications try to access the data in it )
* Multi-AZ (The data is read by application from a replica of the data base in another AZ in case of failure.)
* Multi region ( applications in another region can access through a replica of the database of another database.)

Amazon Elastic Cache: Helps to reduce the load on the database for read loads. These are in memory databases. It is used for relational database.

**No SQL( No relational Database):**

**Dynamo DB:**

* It is a serverless database, fully managed and highly available with replication. It can withstand massive workloads. Fast and low latency with milli second request.
* A diagram of a product

  Description automatically generatedDynamo DB is a key/value database.
* For cache in Dynamo DB we have DAX (Dynamo DB Accelerator) the most read data is available in this cache for faster retrieval.

**RedShift:** It is based on Postgre SQL, used for OLAP (online analytical processing) for analytics and data warehousing. Columnar data storage. Red shift serverless for the automated provision and scalability.

**Amazon EMR:** Helps in analyzing and process vast amount of data.

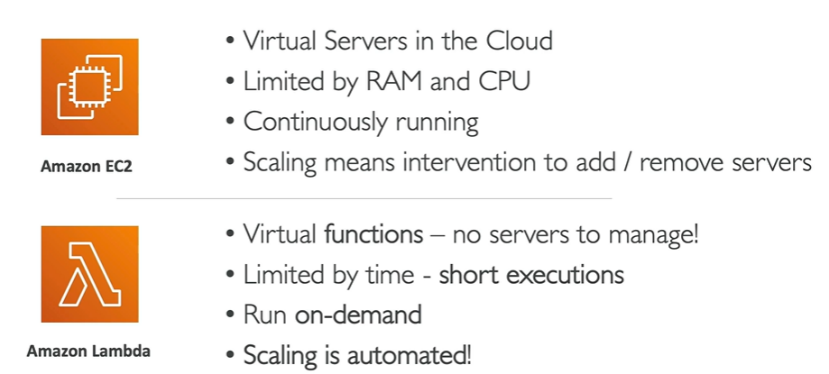
**Athena:** Serverless query service to perform analytics on s3 objects. Uses SQL to query the files.

**Amazon Quicksight:** Allows you to create dashboards on the databases and the insights of the database.

**Document DB:** Uses Mango DB to store query and index JASON data.

**Neptune:** It is a graph database used for social networks.

**Glue**: It does the ETL (extract, transform and load)

**Lambda:**

* These doesn’t require any servers only depending on the functions we can do it. Need not have to create an instance for our functions which save time and money.
* It supports many programming language. It is Function as a Service(FaaS).

**VPC (Virtual Private Cloud):**

* A private cloud server which is not accessible to others, it is very secure and safe to use.
* The AWS will create VPC and takes care of security issues.
* The size of VPC is determined by the size of IP address range for example 172.16.0.0/16 means 65535 ip address are available for the use.

172.16.1.1 to

172.16.1.254

Project 2

172.16.0.1 to

172.16.0.254 Project 1

P

**VPC**

* By subnetting the large IP into multiple subnets we can allocate them to multiple projects where the sub projects use them to the instances inside them.
* Depending on the requirements the instances will be deployed.

**Internet GateWay:**

* To access the VPC the users enters through a gateway. Only through the gateway the request can enter the VPC to access the application.
* Using the public subnet the user enters the internet Gateway and to the load balancer.
* The VPC has both public and private subnets only the public subnets are accessible.

Application/

Instance

Security

Groups

Target group

Route table

Load Balancer

Public Subnet of VPC

User

* The IGW will connect the user to the public subnet IPs of the VPC.

VPC

Internet Gateway

Flow of request from the user to the instance in the VPC.

But when the instance in the VPC want to access anything from internet then it uses NAT Gateway. Because the private IP of the instance should not be disclosed for security reasons.

**Route Table:** Determines the path to the applications from the load balancer. The request follows this path to access the application.

**Public Subnet:** The subnets that are available for the outsiders to access the VPC.

**NAT Gateway:**

* The instance wants to access the internet then the NAT will come into play and MASK the IP address of the instance/application with the public IP of the Load Balancer(SNAT) or Route table (NAT gateway) which will be secured.
* The NAT gateway will mask the IP of the Instance with public IP.

VPC is virtual private cloud in public cloud.

A VPC is simply like Defence quarters. It is protected from the outsiders with a private security. Only through one gate we can enter the VPC that is called Internet Gateway. After entering we have use their vehicles as our public vehicle cannot be used that is the Public subnet of VPC. Depending on the no. of visitors the vehicle is decided which is the Load Balancer. The map is provided for the required quarters which is the Route table. At the entrance of each house the security guards are there those are the Security groups.

We can add security in multiple layers for the VPC in AWS.

**NACL**: A security feature at the subnet level.

**Security Groups:** A security feature at the EC2 instance level.

In AWS the security is Shared Responsibility. It is with the user and AWS.

**Security Groups:**

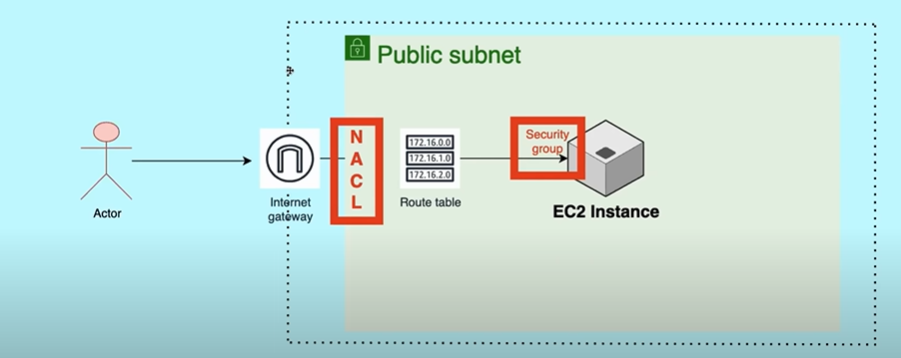
* These are configured at the instance level. Without configuring the security groups we other users cannot access the instance using the public IP.
* The security groups determine the type of request like http from port 80, ssh from port 22, etc.
* Only through this port requests can access the instances.
* It has only configurations to allow a particular request but it cannot deny any particular request from a port. As by default it is denied.

As a part of SG we can manage the traffic into the instance and from instance to outside.

By default the AWS will create default SG where inbound is denied and outbound is allowed.

* **Inbound** : The requests from the users to the application.(like opening the amazon to buy something.
* **Outbound**: The traffic from the application to the internet( like payment option from the amazon the phonepay or other.

**NACL (Network access control list):**

* Security at the subnet level. If the SG allows all traffic this will helps in restricting the traffic at the subnet level.
* This is an additional layer of security. The traffic is initially restricted and allows only specific requests. This Subnet level is applied to all the instances in that particular subnet.
*  We can allow or deny a particular IP request to allow or deny.
* The rules at the NACL are ascending order means the first rule is followed.(lowest number is the first rule)

By default the AWS will provide the IG, NACL, RT, SG while we create a VPC as it a shared responsibility of User and AWS. The user need to configure them based on the requirements.

Both NACL and SG are the securities where the inbound traffic can be blocked.

The inbound request from a particular port is allowed at the Security Groups then that same inbound request can be blocked at the NACL.

The request first enters the NACL, if it allows then it will move to the SG, if not it won’t.

|  |  |  |  |
| --- | --- | --- | --- |
| **Request port or IP** | **NACL(subnet level)** | **SG(instance)** | **Final** |
| 8080 | allow | deny | deny |
| 8080 | allow | allow | Allow |
| 8080 | deny | allow | Deny |
| 8080 | deny | deny | deny |

The inbound traffic rule in the VPC for the port 8080.

We can block the IP address or port no. its upto you. The NACL is the first level of defence.

**Route 53:**

* It is the DNS service of the AWS. DNS means Domain name system.
* Translate the domain name into IP address.
* The applications in the VPC are not accessed by using the IP address. We use the domain name to access that particular domain. (Google 8.8.8.8 which is the public IP address of google)

The applications we host needed to be given a domain name by purchasing from Godaddy and the hosting will take some more money.

But the AWS gave an ease to all of these. The applications running in the AWS will be provided with all using the Route 53 services.

* This Route 53 is placed before the load balancer. When the request enters the internet gateway then it moves to route 53 which resolves the domain name to IP address of the load balancer.

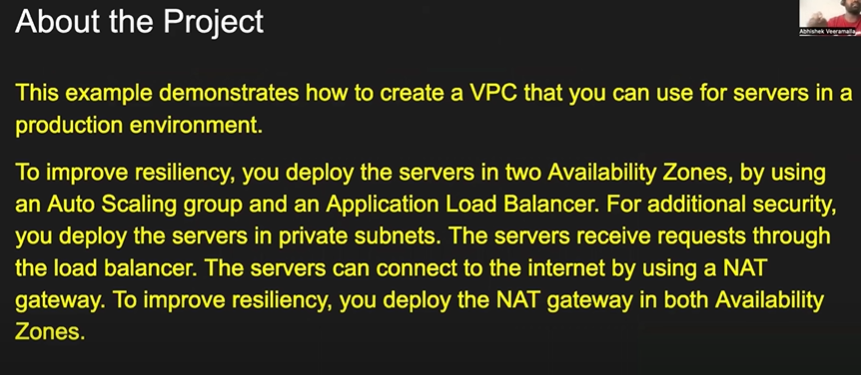
Steps in Route 53:

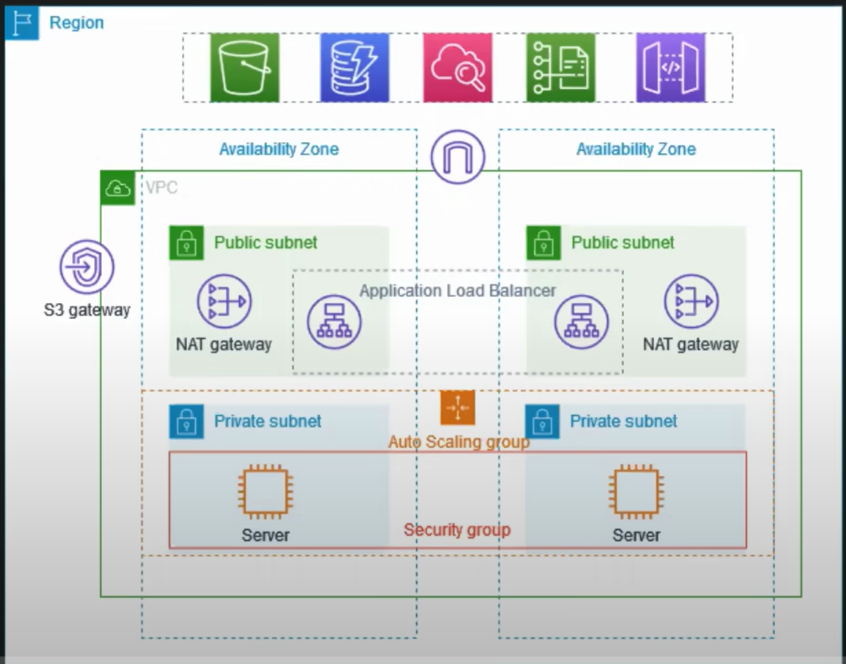
* Domain Registration
* Hosted Zone (create the DNS records with domain names and IP addresses)

Here the Hosted Zone contains the DNS records where the IP and domain names are connected.

Along with these the Health checks are also done by the AWS for the active working of the webservers. This is an added advantage to check the failure of websites in the servers.

**Project 1: VPC with Public Private subnet in production**

Video link : https://youtu.be/FZPTL\_kNvXc?si=UcEupOAmnTs9fBuW



**AWS CLI:**

* The AWS console we were using is a UI( user interface) which is easy to access through a windows OS and at a time we cannot create multiple resources.
* So the automation is the best practice to create vast no. of applications in the cloud.
* Through the UI the automation is difficult but through the CLI interface the automation is easy. For this the AWS created APIs.
* The API (Application Programming interface) acts as a connection between the CLI and the AWS cloud.
* Through API we can automate the creation, deletion and expansion of the resources in the AWS. Using the shell script we request the API to create the resources then the API will communicate with the AWS and do the task without manual intervention.
* AWS CLI (a python utility)
* Terraform
* AWS CDK
* Cloud Formation templates. These are the tools for the automation.

After passing the required things in script or python program form to the API, The API will communicate with the AWS and create the resources. To check the resource the output will be generated in a JSON file format(simple format).

* The format is present in AWS CLI reference documents.

AWS

Cloud

API

AWS

CLI

AWS CLI is for small operations but the CFT and Terraform are for the building the large infrastructure.

To login into AWS CLI with the AWS account we need the Access key and secret access key of the root user or the IAM user. Best practice is to use the IAM user to login into the CLI.

By passing a single command we can list the instances, buckets, etc.

* aws configure is the command used to login into the AWS through CLI.
* aws s3 ls is the command used to list the s3 buckets.

For each and every resource deletion, creation and expansion, modification there will be different commands to pass our arguments. These are available in the AWS CLI documentation.

**IAC (infrastructure as CODE):**

* Write code to create an infrastructure. Generally we write a java code to create a java application but here it is somewhat different.

Principle of IAC tools:

A diagram of a diagram

Description automatically generatedAny tool that has to be IAC tool then that tool should act as a middle man between user and one or multiple cloud provider.

IAC

API cloud

provider

IAC tool should accept the template from the user that may be in YAML, JSON, declarative or versioned.

The tool should convert this template into API calls so that the cloud provider can understand.

**Cloud Formation Template:**

* CFT follows IAC standards which are not possible in AWS CLI. CFT is a tool of IAC.
* This CFT supports an YAML or JSON template which then will be converted into API calls.
* The CFT converts the JSON or YAML template into API calls to the AWS.
* The YAML or JSON templates must be declarative and versioned.

Declarative: The template must be simple and easy to understand by anyone and should be specific to the resources of the cloud provider.

Versioned: Able to get the previous document or file after updating with the new changes. Just like enabling the S3 bucket versioning and github versioning. The previous document will also be available even though a new document is overridden. We can get the changes that are done 5 or 10 days ago.

We use CLI when we want to perform quick actions or short actions like listing instances, s3 buckets, users, etc.

We use CFT when we want to create more instances or buckets etc.

For writing the code for CFT we better use YAML because it has a commenting feature which is not possible in JSON . This commenting feature helps us in easily understand the code.

Along with creating an infrastructure the CFT also supports Drift detection.

**Drift Detection:**

Any changes in the infrastructure will be automatically detected by the drift and informs us when we use the drift. It will be easy for us to identify the changes made by another person easily.

**Stacks:**

The YAML file we have will be uploaded to the stacks in the AWS UI. By creating a new stack and uploading the YAML file the CFT will able to read the YAML file.

**Implementation of CICD on AWS**

**AWS Code Commit:**

* It is like GITHUB or GITLAB which is used for versioning of the codes. It solves the problems of version control.
* It is a managed GIT, scalable and reliable.
* AWS code commit does not work well with root account better use IAM user.
* Code commit is used to store codes in private repositories. The repositories are always private i.e. no access to public.
* After creating the repository we can upload the files into it.
* For the IAM users the permissions should be given like codecommit power user.
* Through the ssh we can connect with the code commit and use git commands.
* This has less features than GIThub, less integration with other services outside AWS and also it is very restricted from outside access.
* Code commit means changes in code or update the changes in code which creates a version of the file.

CD

CI

Kubernete/EC2

AWS Code deploy

AWS Code build

AWS Code Pipeline

AWS Code commit

The CICD implementation in the AWS using AWS services. The Code pipeline will invoke the CI in the Code build and after that it will invoke the CD in the Code Deploy. But outside Jenkins will do this thing.

**AWS Code Build:**

* It supports any of the versioning services like Github or code commit.
* AWS code Pipeline will acts as an orchestrator and invoke the CI in the Code build.
* For all the tasks in the code build we have to write a code in YAML.

Tasks in Codebuild:

* Check out: checks for the code in the repository.
* Build & UT: It checks the code for creating the docker image.
* It creates the docker image and pulls the image.

**AWS Code Pipeline:**

* Integrates the Github and the Code build.
* When there is change in the code of the Github then the code pipeline will invoke the Code pipeline which will do the code build tasks and finally creates a docker image.

**AWS Code Deploy:**

* The code deploy will connect to the EC2 instance where the application can be hosted.
* The Code deploy will be added to the Code Pipeline as a part of Continuous Delivery.
* So when the changes were done in the Github then the Code Pipeline gets triggered and The code pipeline will invoke the Continuous Integration in the Code Build which will create a docker image at the end.
* After successful creation of the docker image the Code pipeline will invoke the Code deploy which will deploy the application into the EC2 instance.

**AWS Cloud Watch:**

* The Cloud watch has the best functions in the AWS where it will monitor, alerting and metrics, logs.
* It is the gatekeeper of the AWS.
* These are most important for the Cost optimization and Scaling.
* The cloud watch will keep a track of each and every activity that was performed in the AWS.
* Based on the metrics, alerts and logs provided by the cloud watch we can do the cost optimization.
* It gives the complete details of the usage of all resources and helps us in identifying the unused resources so that they can be terminated thus saving the money.

**Metrics:**

* The metrics plays important role in the scaling of the resources. It gives the details of the utilization of CPU, memory, etc.
* There are custom metrics where we can customize then for a specific resource also. The metrics are displayed in a bar graph, pie charts, graphs, etc.
* It gives us the information starting from last 5sec to months and years.

**Alerts/ Alarms:**

* Based on the output of the metrics the alerts can be set for example the utilization of the CPU when exceeded above 50% then the alert will be sent to your mail.
* So on this basis the resource management is possible. We can scale up the resources if needed.

**Logs:**

* Logs are the detailed review of the actions and tasks that happened in the AWS.
* Each task has the logs containing even a tiny detail even if the task is failed.
* Even after deleting the application the logs will be stayed.

**AWS Lambda:**

* These are mostly used for cost optimization for the resources.
* It solves the compute and serverless problems. It is similar to EC2 but quite different.
* In EC2 we give the inputs to the AWS about the Type of instance, memory, size , OS, etc. based on the applications we are going to run on the EC2 instance.
* After giving all these the AWS will provide us with a virtual machine.
* But the Lambda functions also gives us the compute to our applications but without thinking about the server.
* We just give the details of the application we are going to run on. Then the AWS will give an instance that has no host or IP address.
* After the function of the application is completed then the AWS will delete the instance.
* Whereas the instance in the EC2 we create will be terminated by the user only.
* Here in the lambda we need not have to think about the server. We just have to use the compute for the application we want to run.
* The scaling up and down of the resources will be taken care by the AWS only depending on the application requirements.

Only when the user request the application then the AWS will create the instance and after the usage the instance will be deleted automatically. The access to lambda functions is also controlled by the AWS itself.

The design team will decide the type of approach we can use for the application it may be either serverless or server.

The lambda functions are event driven. That is by using the cloud watch we can trigger the lambda functions to perform a certain activity at a specific time by a cron job in cloud watch.

In manual practice to perform a task we will create a instance and after the task is completed then we will terminate the instance manually which is not a preferable option for automation.

**Serverless Architecture:**

* It is for the cost optimization, security/ compliance( restricting).
* Serverless means we need not have to maintain a server and give requirements for server creation. We just simply give the application and the requirements will be taken care by the AWS.
* For monitoring the resources we just write a script and use the lambda function and complete the task and it gets deleted after usage.

**Cost Optimization:**

* It is the first priority of the devops engineer. Sometimes the unused resources will be remained the same without intervention they keep on costing the companies.
* To reduce the cloud cost we can send the notifications to the users who created the instances, S3 buckets, snapshots, etc.
* If he has access then the devops engineers can delete the resources.
* Using the SNS to send the notifications to the users.

Here for checking the resources we have to write a code in python (boto 3) using the Lambda functions and that will call the API’s of the AWS to check the resources.

Once it finds the unused resources then it will delete those resources. This lambda functions are event driven which means these can be triggered using the Cloud Watch.

AWS API

Delete/ notify users

Lambda

Function

Python( boto3)

* The Python code in the Lambda function will call the AWS API to list the all the Snapshots.
* When all are available then it will filter out the unused snapshots that are not connected to any instance.
* Now the instances will be either deleted or sent notifications to users who created the snapshots to delete them.
* Using the Python code first we have to identify the instances, volumes and the snapshots of the volumes.
* After that we can filter out the unused snapshots and delete them.

**Cloud Front:**

* It is a solution for the CDN (Content Delivery Network) and this CDN is used in youtube, Instagram, amazon, etc.
* In normal cases without CDN the social media platforms for example Instagram which stores the content of the user in their local servers storage.
* So the users in that particular region will have low latency compared to users trying to request the content from another region.
* This problem can be solved by using the CDN. In normal condition the Images or content is stored in a central location.
* The CDN will create the copies of the content so that they are accessible to the requester with low latency. The copies are stored in Edge locations.
* The DNS will direct us to the CDN to access the content without directy connecting to the Instagram storage center.

When the people from a region try to access the content from S3 or storage centers then the content will be cached in the nearest edge location of the region. So next time any other user wants to look for the content the CND will direct it to the cache of the edge location.

* This reduces the latency of the S3. The Cloud Front has these edge locations.
* Security of the data can also be increased.

**Elastic Container Registery:**

Used to store and manage the containers.

**Containers:** A container is package of your application code and the dependencies that are required to run your applications.

The registries are used to store the docker images. (similar to git hub)

**Elastic Container service:**

Helps in launching the docker containers in AWS. It requires the EC2 instance as infrastructure.

AWS takes care of start and stop of the images.

**Fargate:**

It is similar to ECS but a serverless approach. It does not require an EC2 instance.

The docker images are stored in ECR and are used by the Fargate and ECS whenever required.

**Lightsail:**

It is a simple alternative to all the AWS services. It is of high availability but no auto scaling. For small and simple tasks we can use lightsail. It is a platform as a service.

**Elastic Bean Stalk:**

It is developer centric view of deploying an application on AWS. It uses all the components like EC2, ELB, ASG, etc.

Similar to lightsail but the networks, instance,ASG, ELB etc. selection is done by automatically. We need to select the application that’s all.

**Code deploy:** To deploy applications automatically.

**Code commit:** It stores the code before the code deploying. Automatically versioned. Similar to git hub.

**Code Build:** compiles source codes, run tests, and produce packages that are ready to be deployed by code deploy.

**Code Pipeline:** Orchestrate the different steps to have the code pushed automatically into production. This is for CICD.

**Code Artifact:** Software packages depends on each other to be built. Storing and retrieving these dependencies is called Artifact Management.

**Systems Manager:** Helps you to manage EC2 instance and on premises systems. We can do patching, running commands, storing parameters.

**Session Manager:** Allows you to start a SSH on your EC2 instances. Using the Session manager the user can directly access the EC2 instance when the IAM user is given the IAM policy to access the SSM. We can send session log data to S3 or Cloud watch.

**AWS Global Accelerator:** Improve global application availability and performance using AWS global network.

**AWS Wavelength zones:** For 5G networks

**AWS Local Zones:** Extension of the region. For low latency in your locations.

**Simple Queue Service (SQS):** It is like the request are coming for the application in EC2 instances will be queued and sent in order to the application. The requests will be retained and sent one by one.

**(SNS) Simple Notification Service**: Send one messages to many users. We can send mails, http, lambda, etc. but they should have the subscription to this.

**AWS Cloud Trail:** Provide governance, audits and compliance for your AWS account.Get history of events,api calls, etc.

Anything is deleted then we can use Cloud trail to investigate it.

**AWS Cloud Watch:**

The Cloud watch has the best functions in the AWS where it will monitor, alerting and metrics, logs.

The cloud watch will keep a track of each and every activity that was performed in the AWS.

Based on the metrics, alerts and logs provided by the cloud watch we can do the cost optimization.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EC2** | **S3** | **Database(RDS)** | **ECS** |  |
| EC2 Instance store | Snow Ball | RDS: Aurora | Fargate | **Elastic Bean stalk** |
| EBS | Snow cone | Elastic cache | ECR | **Code commit** |
| EBS snapshot | Snow mobile | NRDS: Dynamo DB | ECS | **Code build** |
| AMI | Storage classes | Redshift, EMR | **Lambda** | **Code pipeline** |
| EFS, Amazon FSx | S3 replication(CRR&SRR) | Athena, Quick sight, Document DB | **Lightsail** | **Code deploy** |
| EC2 image builder |  | , Neptune, Glue, DMS, QLDB | **CFT** | **Code artifact** |
|  |  |  |  |  |
| **Systems Manager** | **Sessions Manager** | **Route 53** | **Cloud Frond(CDN)** | **AWS Global Accelerator** |
| **AWS Wavelength** | **AWS local zones** | **SQS** | **SNS** | **Kinesis, Amazon MQ** |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cloud Monitoring** | **VPC** | **Security** | **Machine** **learning** | **Account** & **Billing** |
| Cloud Watch | Private link, site to site VPN | WAF & Shield | Rekognition, Transcribe | Organization |
| Cloud Trail | Direct connect, VPC endpoints. | Network Firewall, Cloud HSM | Polly, Translate, Lex + connect | AWS, Control Tower, Resource access Manager |
| AWS Health Dashboard | Client VPN, NACL | Secrets Manager,  Artifact, Guard Duty, Inspector, Config, Macie, Amazon Detective | Sagemaker, Forecast, Kendra, Textract | Service Catalogue, Compute Optimizer, Pricing Calculator, AWS Trusted Advisor, |

**Important Notes:**

* A company use Cloud Trail and Cloud Watch to gather information about activity in their AWS account

**AWS Trusted Advisor:**

* provides recommendations on how to optimize performance for AWS services.
* AWS Trusted Advisor is an online tool that provides you real time guidance to help you provision your resources following AWS best practices. Trusted Advisor checks help optimize your AWS infrastructure, improve security and performance, reduce your overall costs, and monitor service limits.

**AWS Health Dashboard:**

* The health dashboard shows issues or upcoming events that may impact your resources. It does not notify of service limit breaches.

**AWS Cost Explorer**:

* Cost Explorer is used for viewing costs and will not assist with service limits.

**AWS CloudTrail:**

* CloudTrail is an auditing service.

**AWS Systems Manager:**

* Systems Manager is used for managing EC2 instances such as installing patches and software.

**AWS Step Functions:**

* It provides a visual console to visualize the steps in the workflow, helping to build and update applications quickly and monitor the status of each step in the process.

**Amazon SNS:**

* it is primarily used for sending notifications.

**Amazon SQS:**

* it can be used to decouple cloud applications.
* This is a message queue service used to decouple the components of a cloud application.

**AWS Web Application Firewall (WAF):**

* used to protect on-premises resources if they are deployed behind an Application Load Balancer (ALB).

**Amazon VPC network ACLs:**

* Network ACLs only filter traffic entering and leaving a VPC subnet.

**AWS CodeBuild:**

* CodeBuild is used for compiling and testing code ahead of deployment.

**AWS CodeDeploy:**

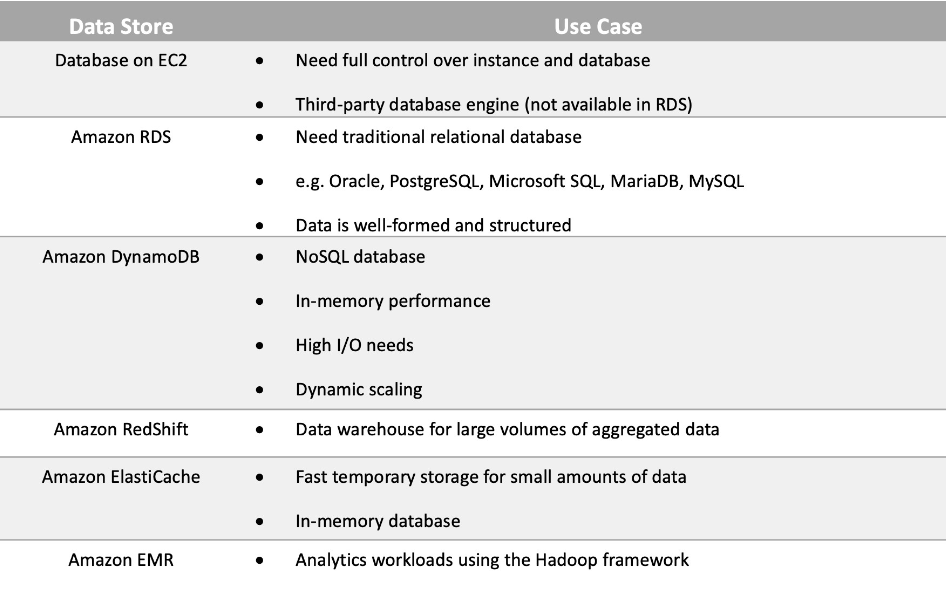
* CodeDeploy is used for deploying code from a code repository and actually installing the application.

**AWS CodePipeline:**

* CodePipeline is used for automating the code release lifecycle.

**AWS Glue:**

* Glue is a fully managed extract, transform, and load (ETL) service that makes it easy to prepare and load data for analytics.

**DATABASE:**

**Amazon Aurora:**

* It is a relational database that is compatible with MySQL and PostgreSQL database engines. Aurora is extremely fast and scales up to 128 TB.

**Amazon DynamoDB:**

* DynamoDB is a NoSQL (non-relational) database and you cannot deploy a MySQL database as it is a relational database type.

**Amazon Athena:**

* It is used for querying data in Amazon S3 using SQL.

**Amazon DocumentDB:**

* DocumentDB is a NoSQL database that supports document data structures.

**Security:**

**Amazon Macie:**

* It is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS. Amazon Macie automates the discovery of sensitive data at scale and lowers the cost of protecting your data.

**Amazon GuardDuty:**

* It is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts, workloads, and data stored in Amazon S3.

**Amazon Detective:**

* It automatically processes terabytes of event data records about IP traffic, AWS management operations, and malicious or unauthorized activity.

**Amazon CloudWatch:**

* CloudWatch monitors performance but does not provide recommendations for optimization.
* to monitor a new Amazon EC2 instances CPU and network utilization.

**Amazon Kinesis:**

* It is used for real-time streaming data and can technically be used to build a real-time analytics solution.

**Amazon CloudFront:**

* It is a content delivery network (CDN) that caches content around the world for lower latency access.

**AWS Global Accelerator:**

* It enables access to your application by leveraging the same Edge Locations as CloudFront and routing connections across the AWS global network.

**AWS Direct Connect:**

* This service provides private connections from data centers to AWS. It is not useful for distributed users as they will not be able to take advantage of it.

**Amazon Connect:**

* It provides a seamless omnichannel experience through a single unified contact center for voice, chat, and task management.

**Loose coupling:**

* It is when you break systems down into smaller components that are loosely coupled together. This reduces interdependencies between systems components.

**AWS Artifact:**

* Artifact provides on-demand access to AWS security and compliance reports.

**AWS Config:**

* Config is a service used for compliance relating the configuration of AWS resources.