**\* \* \* AWS Session \* \* \***

**01-27-Aws-27-DEC-24**

------------------------------------------------------------------------------------------------------------------

1) What is IT infrastructure ?

2) Challenges with On-Prem infrastructure

3) What is Cloud Computing

4) Cloud Service Models

5) Cloud Providers

6) AWS Introduction

7) AWS Services Overview

**=============================**

**What is IT Infrastructure ?**

**=============================**

=> To run a software company then we need to purchase below resources

a) machines

b) network

c) power

d) storage

e) backup

f) security

=> The above resources are called as IT infrastructure.

=> We can maintain infrastructure in 2 ways

a) On-Prem infrastructure

b) Cloud infrastructure

**==================================**

**What is On-Prem Infrastructure ?**

**==================================**

=> On-Prem means we need to purchase and we need to maintain our resources to run our business.

=> We have several challenges with On-Prem infrastructure

1) Lot of money investment

2) Lot of man power

3) Scalability (increase / decrease)

4) Availability

5) Network issues

6) Security issues

=> To overcome the problems of on-prem infrastructure companies are preferring Cloud Infrastructure.

**===========================**

**What is Cloud Computing ?**

**===========================**

=> The process of delivering IT resources over the internet on demand basis is called as Cloud Computing.

=> We have below advantages with cloud computing.

- Pay as you go

- Less cost

- Scalability

- Availability

- Security

- Backup

Note: Cloud computing works based on pay as you go model.

**================**

**Cloud Providers**

**================**

=> The companies which are providing IT infrastructure based on "pay as you go" model are called as Cloud Providers.

1) Amazon (AWS)

2) Microsoft (Azure)

3) Google (GCP)

4) Salesforce

5) Ali Baba

6) Digital Ocean

**=======================**

**Cloud Service Models**

**=======================**

=> We have 3 types cloud service models

**1) IAAS**

**2) PAAS**

**3) SAAS**

**=============**

**What is IaaS**

**=============**

=> IAAS stands for infrastructure as a service

=> Provider will give infrastructure for us

Ex: Machines, Network, Storage

=> As a customer we need to prepare platform to run our application.

Ex: install required softwares + setup web servers + deploy application.

**=============**

**What is PaaS**

**=============**

=> PaaS stands for Platform as a service

=> Provider will give ready made platform to run our application directley.

=> As a customer we need to take care of only our application deployment.

**=================**

**What is SaaS**

**=================**

=> SaaS stands for software as a service

-> Cloud Provider will give their application to run our business.

Ex: zoom, google drive, dropbox, microsoft teams, jira ...

**===========**

**AWS Cloud**

**===========**

=> AWS stands for Amazon webservices.

=> AWS providing cloud services from 2006 onwards

=> AWS works based on Pay as you go model

=> 190+ countries using AWS cloud services to run their businesses

=> AWS having global infrastructure

34 - Regions

108 - Availability Zones

**===============**

**AWS Services**

**===============**

=> We have 200+ Services.....

1) EC2 : To Create Virtual Machines (Hourly Billing)

2) S3 : Unlimited storage

3) RDS : Relational Database service

4) EFS : Elastic File System (shared file system)

5) IAM : Identity & Access Management

6) VPC : Virtual Private Cloud

7) Elastic Beanstack : End to end web-application mgmt (PaaS)

8) Lambdas : Serverless computing

9) Route 53 : Domain Mapping (DNS)

10) ECS : Elastic Container Service

11) EKS : Elastic Kubernetes Service

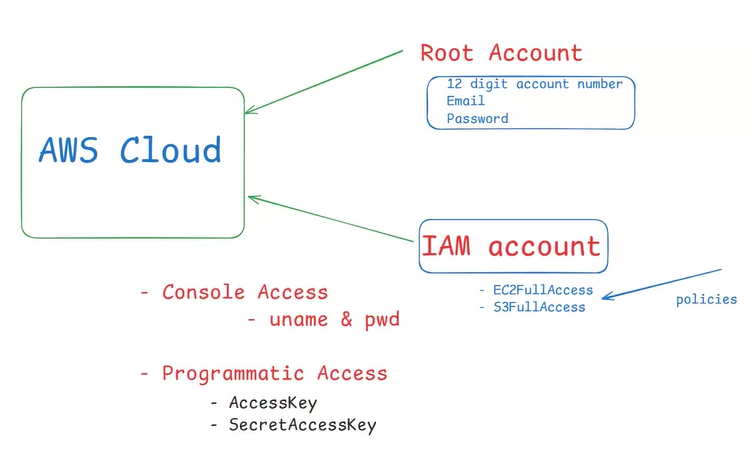
12) Cloud Watch : Monitoring

13) SNS : Simple Notification Service .....

**=====**

**IAM 02-27-Aws-30-DEC-24**

**=====**

****

**\* Identity and Access management**

=> It is used to manage users, groups, policies and roles

=> IAM is a free service

=> In AWS cloud platform we will have 2 types of accounts

1) Root Account

2) IAM Account

Note: When we signup in aws website then by default it will consider that as root account.

=> Root account is very powerfull account with no restrictions.

=> If we login with Root user credentials, we can access everything in AWS cloud.

Note-1 : We shouldn't use root account for day to day activities.

Note-2 : We shouldn't share root account credentials with anyone.

Note-3: Company will not provide root account credentials for team members.

Note-4: It is recommended to enable MFA for root account.

MFA : Multi Factor Authentication

**===================================**

**Multi Factor Authentication (MFA)**

**===================================**

-> It is used to provide additional security for root account.

-> Enable MFA for root account using Google Authenticator app.

-> After enabling MFA, logout and login into root account and check behaviour.

**============**

**IAM Account**

**============**

=> For team members IAM accounts will be created with limited access

=> For daily activities in aws cloud we should use IAM account only

=> For IAM user we can provide below types of accesses

1) Console Access (web login)

- uname & pwd

2) Programmatic Access

- AccessKey and SecretAccessKey

**Note: To communicate with AWS cloud using terraform then IAM user should have programmatic access**.

1) Create IAM account and attach policies (RDSFullAcces, S3FullAccess)

2) Login into IAM account and check EC2 service (can't access because no permission)

**=================**

**IAM User Group**

**=================**

**Note - Policies are only applicable for users and group level only.**

=> When we want to provide some permissions for multiple users then we can create IAM user group and we can add users to that group and we can attach policies to group.

**1) Create User Group**

**2) Attach Policies to group**

**3) Add Users to group**

**=========**

**IAM Role**

**=========**

**=> IAM role nothing but set of permissions**

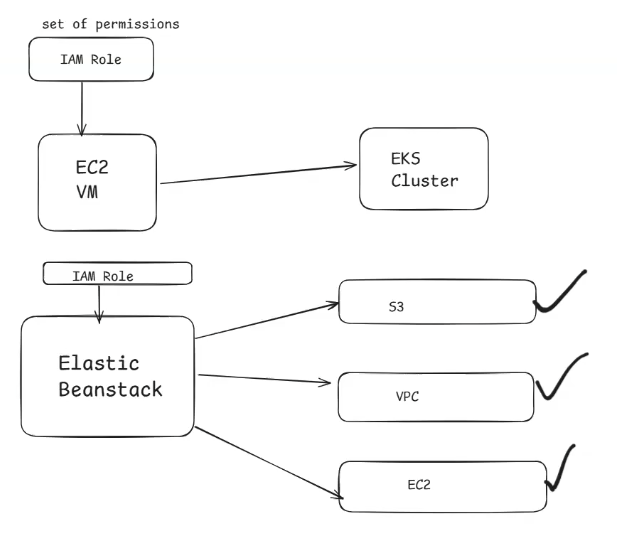
Ex-1: EC2 VM wants to create EKS cluster, Then EC2 VM should have IAM Role with EKS permissions.

1. create IAM role with all EKS policies

2) Attach IAM role to ec2 vm

**Note – AWS policies will be saved in the JSON format.**

**If one service want to communicate with the another service then IAM role comes into the picture.**

****

**============**

**IAM Summary**

**============**

1) What is IAM

2) What is Root Account

3) How to enable MFA

4) What is IAM account

5) Console Access Vs Programmatic Access

6) Users Creation

7) User Groups

8) Policies / Permissions

9) Roles

10) Working with Custom Policies

**=========**

**AWS RDS 03-27-Aws-02-JAN-25**

**=========**

1) What is Database

2) Why we need database

3) On-Prem Database & Challenges

4) What is RDS & Why

5) RDS Setup

6) Connecting with RDS DB Server using MySQL Workbench

**===================**

**What is Database ?**

**===================**

Database : It is a software which is used to store the data permanently.

Whatsapp Application -------------> Database

- Text msgs

- Audio files

- Video files

- Documents

=> Every s/w application will use database to store the data permanently.

=> Software applications will use SQL to communicate with databases.

SQL

Application --------------> Database

=> Using SQL we can perform CRUD operations in the database.

**C -> Create**

**R -> Retrieve**

**U -> Update**

**D -> Delete**

=> We have several databases in the market

- Oracle

- MySQL

- SQLServer

- PostGres ....

=> The above databases are called as Relational Databases.

=> Relational databases will store the data using tables.

=> Table represents data using Rows and columns.

**=================**

**Database Setup**

**=================**

=> We can setup database in 2 ways

1) On-Prem Database

2) Cloud Database

**==================================**

**Challenges with On-Prem Database**

**==================================**

You Download and Install database on your own.

1) Purchase DB server license

2) Install DB Server s/w

3) Security

4) Network

5) Availability

6) Scalability

7) Backup & Restore

8) Adminstration (DBA)

* To overcome above challenges it is highly recommended to use Cloud Databases.
* If we use cloud database then cloud provider will manage database server for us.
* **RDS is a service which is used to manage relational databases.**
* AWS RDS service providing cloud databases.
* **RDS stands for relational database service in AWS cloud.**
* RDS is used to create & manage relational databases.
* **RDS is a fully managed service in AWS cloud.**
* RDS works based on "pay as you go" model.

**=============**

**RDS Lab Task**

**=============**

**Step-1 :** Create MySQL DB Server using RDS

**Step-2 :** Enable "**MySQL : 3306**" port number in Security Group Inbound Rules

**Step-3 :** Test MySQL DB Connection using "**MySQL Workbench software (client s/w)**"

**Step-4 :** Execute some SQL queries for practice (optional)

**Step-5 :** Delete RDS instance to avoid billing.

**============================**

**MySQL DB Creation Steps**

**============================**

Creation method : Standard Create

Engine Type : MySQL

Templates: Free Tier

DB instance identifier : ashokit-db-instance

public access : Yes

Credentials : Self Managed

Additional Configuarations : Initial DB name : ashokitdb

**====================**

**Database Details**

**====================**

DB Endpoint - database-1.cnuuke2i0lav.eu-north-1.rds.amazonaws.com

DB username - admin

DB password - Masterpassword

DB port - 3306

Note : Using above details we can check database connectivity.

=> Once connectivity is successful then we will share database details with developmen team.

In realtime public access will be NO

**============================**

**SQL Queries For Practice**

**============================**

=> Execute below sql queries using workbench

show databases;

use ashokitdb;

show tables;

## table creation query

create table emp(

eid int(10),

ename varchar(100),

esal int(10)

);

## retrieve records query

select \* from emp;

## insert query

insert into emp values(1, 'john', 1000);

insert into emp values(2, 'smith', 2000);

## retrieve records query

select \* from emp;

===========

Assigment

===========

1. Connect with RDS DB Server using EC2 Linux VM and execute above sql queries.

* Create one RDS database and ec2 instance
* Connect RDs instance to the ec2 using command
* Command - mysql -h <rds-endpoint> -u <username> -p
* Then enter password and your database is connected.

**=================================**

**AWS S3 (Simple Storage Service) 04-27-Aws-03-JAN-25**

**=================================**

-> S3 is a storage service in AWS cloud.

-> S3 supports unlimited storage.

-> Using S3 we can store any amount of data from anywhere at anypoint of time.

-> S3 supports object based storage (files).

One Object = One file

**Note: We can store any type of file in s3**

**Ex:** .txt, .mp4, .png, .jpeg, .doc, .pdf, .xls ...

-> In S3, we need to create buckets to store objects (files)

**Note: In one bucket we can store group of objects.**

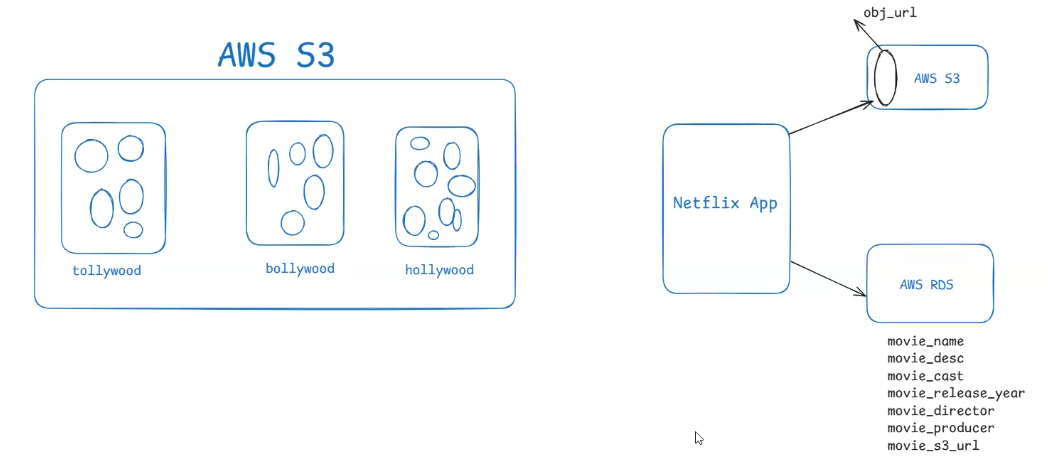
**Note: Every bucket should have unique name.**

-> When we create a bucket, end-point url will be generated to access bucket.

-> When we upload object into bucket, every object will get its own end-point url.

Note: By default, buckets and objects are private (we can them as public).

**In RDS we can store only txt format data. Ex. Urls**

****

**Where we will use s3 storage ?**

**Ans –** If our project is dealing with the some files , binary files (audio , video ,images) that is not recommended to store in the database , so that’s why file based storage is required for that s3 service comes into the picture

**Note - Our job is only to create a database and the buckets**

**=================================**

**Static Website Hosting using S3 05-27-Aws-06-JAN-25**

**=================================**

-> Website nothing but collection of web pages.

Ex: login page, register page, services, about us page, contact us page...

-> Websites are divided into 2 types

**1) static website**

**2) dynamic website**

-> The website which gives same response/content for all users is called as **static website.**

-> The website which gives response based on user is called as **Dynamic website.**

**=> Static websites we can deploy in 2 ways**

1) Create EC2 Linux VM and Install HTTPD Webserver and Deploy

2) Deploy using S3 bucket

Step-1: Create s3 bucket with unique name

Step-2: Upload website files & folders into bucket with public read-access

Step-3: Enable Static website hosting (in bucket properties)

index-document : index.html

error-document : error.html

Note: After enabling static website hosting it generates end-point URL for our website.

Step-4: Access our website using website endpoint url.

**===================**

**S3 Storage Classes**

**===================**

=> Storage classes are used to specify how frequently we want to access our objects from S3.

=> At the time of uploading object into S3 bucket we can select storage class for that object.

=> We have several storage classes in s3 like below..

1) Standard (default) : To access object more than once in a month

2) Intelligent - Tiering : Unknown access patterns

3) Standard-IA : Infrequent Accesed Data (Only once in month)

4) One Zone-IA : Stored in single availability zone (once in a month)

5) Glacier Instant Retrieval : Long Live Archieve Data

(once in quarter -> Milli Seconds)

6) Glacier Flexible Retrieval : Once in a year (Minutes to Hours)

7) Glacier Deep Archieve : Less Than once in a year (Hours to download)

8) Reduced Redundency : Not Cost Effective (Not recommended)

**============**

**Versioning**

**============**

=> It is used to maintain multiple variants of same file.

=> By default versioning will be disabled for S3 bucket.

=> As versioning is disabled, when we upload file again it will override old file.

=> If we don't want to replace old objects from bucket then we can enable Versioning.

=> Versioning we will enable at bucket level and it is applicable at object level

**================**

**Object Locking**

**================**

-> It is used to enable the feature WORM (Write once read many times) model.

-> We can enable object lock on **versioning enabled** buckets.

-> Object Lock will be enabled at bucket level and it is applicable at object level.

- created object cant be deleted from other users. They only can read .

Note: If we enable object locking then versioning will be enabled by default.

**=================================**

**What is Transfer Accelaration**

**=================================**

=> It is used to speed up data transfer process in s3 bucket.

=> When we enable Transfer Accelaration it provides endpoint url to upload the data to s3 bucket quickly.

Note: If we enable Transfer Accelaration bill will be generated.

Janabhoomi express (6 hrs, 300 Rs)

Hyd -----------------------------------------------------------> Vijaywada

Vande Bharat Express (3 hrs, 1000 Rs)

Hyd -----------------------------------------------------------> Vijaywada

**===============**

**AWS S3 Limits**

**===============**

**\* Interview Question - how many buckets we can create in an s3 ?**

**ANS -**

=> By default we can create upto 100 s3 buckets per aws account. However we can request AWS support team to increase the limit.

**\* Interview Question -** **what is the size of the object we can upload ?**

**Ans-**

=> Individual object size can be upto 5TB. For uploading large object we can use multipart upload.

For uploading large object we can use multipart upload

Note: In one bucket we can upload unlimited objects. ( Files )

**=========================**

**Realtime Usecases of S3**

**=========================**

1) Application files (images, audios, videos, docs...)

2) Database backup files ( Snapshots )

3) EBS volumes snapshots

4) Server log files

**===========**

**Summary**

**===========**

1) What is AWS S3 & Why ?

2) S3 Buckets & Objects

3) Static website Hosting using S3

4) S3 Storage Classes

5) Versioning

6) Bucket Locking

7) Transfer Accelaration

8) S3 Limits

9) Real-Time usecases

**========**

**AWS EC2 06-27-Aws-07-JAN-25**

**========**

=> Elastic Compute Cloud

=> It is Most demanded service in AWS

=> It is Used to create Virtual Machines in AWS cloud

=> EC2 VM is called as EC2 instance

EC2 Instance = Computer / Server / VM / Virtual Machine / V Box

=> EC2 instance is re-sizable (we can change configuration based on demand)

=> EC2 is a paid service (hourly billing)

=> EC2 VM Minimum billing period is 1 hour

9:00 AM to 9:15 AM => 15 mins => 1 hour billing

9:30 AM to 9:45 AM => 15 mins => 1 hour billing

Note: To encourage beginners, AWS provided t2.micro/t3.micro for 1 year free

(monthly 750 hours)

EC2 has storage with EBS service .

* **EBS stands for Elastic Block Store**

**Note: To encourage beginners, AWS provided t2.micro/t3.micro for 1 year free (monthly 750 hours)**

=> EC2 VM will have storage with EBS service

=> EBS stands for elastic block store.

**Ec2 VM with Windows OS : 30 GB (default)**

**EC2 VM with Linux OS : 8 GB (default)**

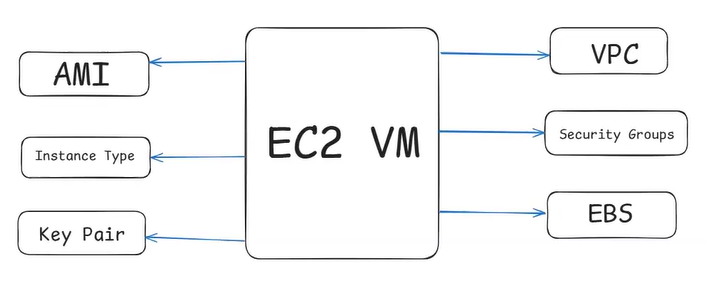
**EBS max capacity : 16 TB (16000 GB)**

**===================**

**Creating EC2 VM**

**===================**

=> To create EC2 instance we will use below components

****

**1.AMI**

* Ami stands for **Amazon Machine Image.**
* AMI represents operating system like **Windows, Linux, MAC etc...**

**2.Instance Type**

* Instance Type represents machine configuration ( Memory,RAM)
* Ex : t2.micro, t3.micro, t2.medium, t2.large etc....

**3. Keypair**

* Keypair is used to connect with EC2 VM securely
* **Ex: .pem file**

**4.VPC**

* VPC provides network required for the EC2 vm

**5.Security Group**

Outgoing

Incoming

* Security Group represents inbound and outbound rules to handle traffic.(Request)

**6. EBS (Elastic Block Store ) Storage / SSD**

* EBS Volume represents storage required for EC2 instance.

Note-1 : One Key Pair we can use for multiple EC2 instances

Note-2 : One Security Group (SG) we can use for multiple EC2 instances

**===============================**

**Types of IP's in AWS Cloud**

**===============================**

=> We have 3 types of IP's in AWS cloud

**1) private ip**

* Private IP is a fixed IP in AWS. It is used for internal communication (With in VPC).
* Even if we restart the EC2 instance, private IP will not change.

**2) public ip**

* public ip is a dynamic IP in aws. It is used to connect with EC2 VM from outside.
* When we restart our EC2 instance then public ip gets changed.
* If we want fixed public ip then we need to use Elastic IP.

**3) Elastic IP**

* Elastic IPs are commercial (bill will be generated).

**================================================================**

**Assignment : Create EC2 VM with Windows AMI and connect with that using RDP client.**

**================================================================**

Create 1 instance and select windows ami and create.

Click on connect 🡪 RDS Client 🡪 Get password🡪upload pem file🡪decrypt password.

Search in windows🡪Remote desktop connection🡪show option🡪enter url🡪username=Adminnistrator 🡪save and connect.

We can connect physical machine to the windows virtual machine using RDP protocall.

Remote Desktop Protocall. It is running on port 3389.

**=================================**

**What is user-data in EC2 VM ? 07-27-Aws-08-JAN-25**

**=================================**

- It is used to execute script while launching ec2 machine.

- User data will execute only once

- Create EC2 VM (amazon linux) with below user data

**Que – Why we use user data in the EC2 VM ?**

**Ans-** We will use user data at the creation of the machine for installing some application .

It is time saving .

#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Life Insurance Server - 1</h1></html>" > index.html

service httpd start

**========================**

**What is Load Balancer ?**

**========================**

* When we run our application in single server then we have to face below challenges

1) One server should handle all request

2) Burden will increase on server

3) Response will be delayed for clients

4) Server can crash

5) Single Point Of Failure

* To avoid above problems, we will run our application in multiple servers.
* We will use Load Balancer to distribute load to all servers in the round robbin fashion.
* We have below advantages with Load Balancer

1) App will run in multiple severs

2) Load will be distributed

3) Burden will be reduces on servers

4) Fast Performance

5) High Availability

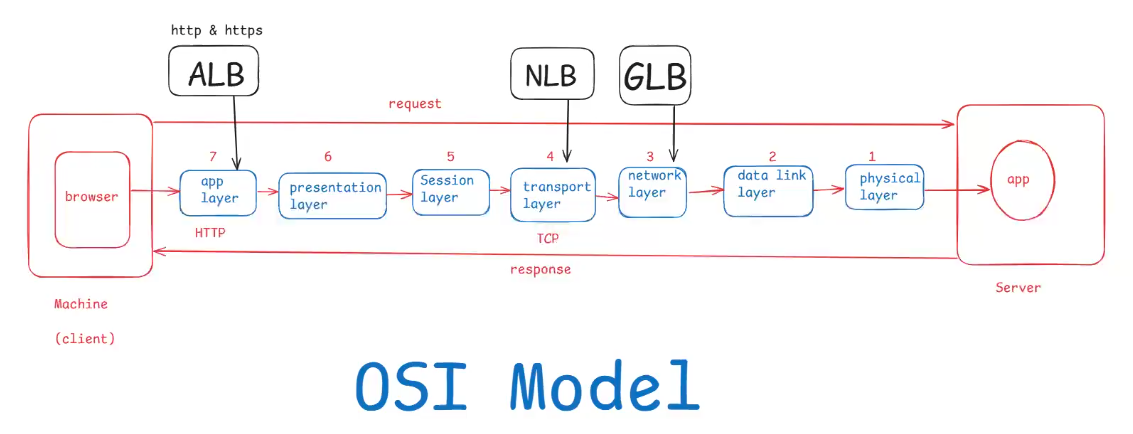
* In AWS we have 4 types of Load Balancers

1) Application Load Balancer (ALB)

2) Network Load Balancer (NLB)

3) Gateway Load Balancer (GLB)

4) Classic Load Balancer (previous generation) Not used



---------------------------------------------------------------------------------------------------------------

**Application Load Balancer (ALB)**

•ALB work at the Layer 7 (**Application Layer**) of the OSI model.

• Designed to route HTTP and HTTPS traffic based on content.

• Supports advanced routing features such as host-based routing, path-based routing, and routing based on HTTP headers.

• Ideal for modern web applications, microservices, and container-based applications.

--------------------------------------------------------------------------------------------------------------

**Network Load Balancer(NLB)**

• Works at Layer 4 (Transport Layer) of the OSI model.

Designed to handle high volumes of TCP, TLS, and UDP traffic.

• Provides ultra-high performance, low-latency load balancing. Suitable for applications requiring high scalability and low-latency communication, such as gaming applications, loT applications, and services requiring high-throughput TCP/UDP traffic.

--------------------------------------------------------------------------------------------------------------

**Gateway Load Balancer**

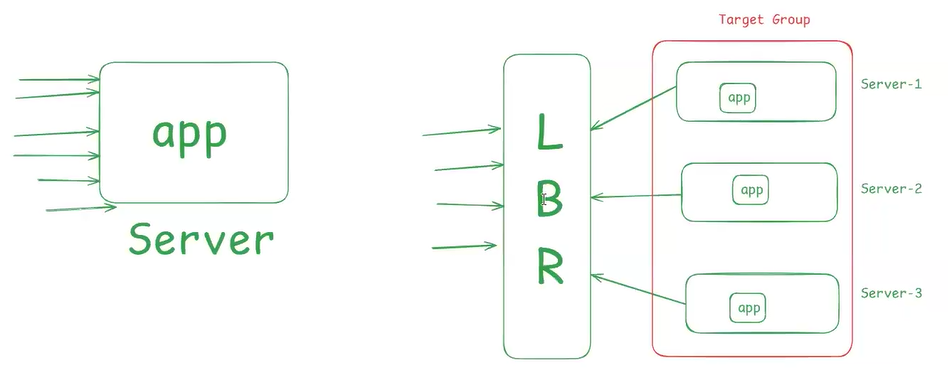
Gateway Load Balancer will work at the network layer.

* It will work at third party applications.

**=========================**

**Load Balancer Lab Task**

**=========================**

****

**Target Group = Set of servers.**

Step-1) Create EC2 VM-1 with below user data

#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Life Insurance Server - 1</h1></html>" > index.html

service httpd start

Step-2) Create EC2 VM-2 with below user data

#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Life Insurance Server - 2</h1></html>" > index.html

service httpd start

Step-3) Add these 2 instances to one "Target Group"

* Click on create target group 🡪 instances🡪Give name for target group🡪
* Select instances🡪include as pending below🡪create target group🡪register targets

Step-4) Create Load Balancer with Target Group (ALB)

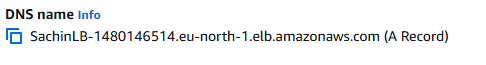
* Create load balancer🡪application load balancer🡪give load balancer name
* Select availability zones 🡪create load balancer.

**Listeners and Routing**

* **We request comes to the load balancer it should forward request to the target groups.**

Step-5) Access Load Balancer DNS in browser

* Before hitting url check security inbound rule and enable HTTP port 80.



**After Use Delete All target group ,load balancer , instances.**

**============**

**Assignment**

**============**

##### Microservices Load Balancing with multiple Target Groups :

###https://www.youtube.com/watch?v=QvEJ8--zneU ####

**===============**

**Auto Scaling**

**===============**

* **Increase or Decrease our resources**

=> It is used to adjust the capacity required to handle the load of our application.

Ex: For big billon day sale we can't guess traffic

=> If requests are increasing then servers should be increased and if requests are decreasing then servers should be reduced.

=> If we use Auto scaling then it will increase or decrease the no.of servers based on incoming traffic.

=> We have below advantages with Auto Scaling

**1) Cost Management**

**2) High Availability**

* If one server is damaged then it will replace with the another server.
* 24/7 server are available.

**3) Fault Tolerance**

* If any server is down / there is any problem with the server then that will replace with that server.

**Note – EKS cluster internally working with the auto scaling.**

**====================**

**EC2 Instance Types 08-27-Aws-10-JAN-25**

**====================**

=> Amazon EC2 (Elastic Compute Cloud) offers a variety of instance types to suit different use cases.

1) General Purpose

- T series : t2, t3, t4...( t= generation)

- M series : m6g, m5, m5a, m5n...

2) Compute Optimized

- C series : c7gm c6i, c6g, c5, c5a

3) Memory Optimized

- R series : r6g, r5, r5a, r5n

- X series : x2gd, x1e

4) Storage Optimized

- I series : i4i, i3, i3en

- D series : d2

5) Accelarated Computing

- P series : p4, p3, p2

- G series : g5, g4ad, g4dn

- inf series : inf1

6) High Performance Computing

- H series : hpc6id

**==============**

**What is EBS 09-27-Aws-11-JAN-25**

**==============**

* EBS stands for Elastic Block Store
* It is block level storage device (Hard Disc / SSD)
* When we create ec2 instance then "EBS Volume" gets created automatically.

**Note: If we remove EBS volume from EC2 instance then we can't start/use that EC2 instance.**

------------------------------------------------------------------------------------------------------------------

* In EBS we have 2 types of volumes (storages)

**1) Root Volume**

**2) Additional Volume**

**Note: When we launch EC2 instance by default we will get one Root volume.**

* Root volume is mandatory to launch EC2 instance.
* Additional EBS volumes are optional devices (we can add/remove)
* For windows VM, we will get 30 GB as default volume size.
* For Linux VM, we will get 8 GB as default volume size.

**Note: EBS volume can have upto 16 TB (16000 GB)**

* One EC2 VM can have multiple EBS volumes.
* One EBS volume can be attached to one EC2 VM at a time.
* EBS volumes are Availability Zone Specific

Mumbai : ap-south-1

ap-south-1a

ap-south-1b

ap-south-1c

Note: In which availability zone our EC2 VM got created, in same availability zone we have to create EBS volumes for attaching.

**=================**

**EBS Volume Types**

**=================**

=> We have 5 types of EBS volumes

1) General Purpose Volume (Min: 1 GiB, Max: 16384 GiB)

2) Provisioned IOPS (io1) (Min: 4 GiB, Max: 16384 GiB)

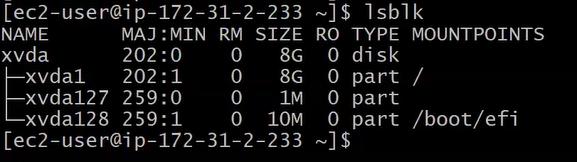
3) Cold HDD (Min: 125 GiB, Max: 16384 GiB)

4) Throuput Optimized (Min: 125 GiB, Max: 16384 GiB)

5) Magnetic (standard) (Min: 1 GiB, Max: 1024 GiB)

**Que – How to check root volumes for linux VM ?**

**Ans – Command – lsblk**

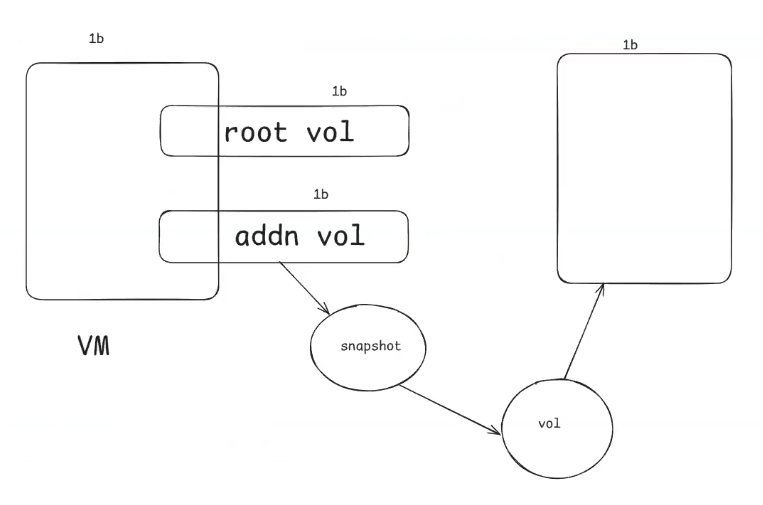
****

* **Xvda** is the root volume name

**==========**

**Snapshots**

**==========**



=> Snapshots are used for volumes backup

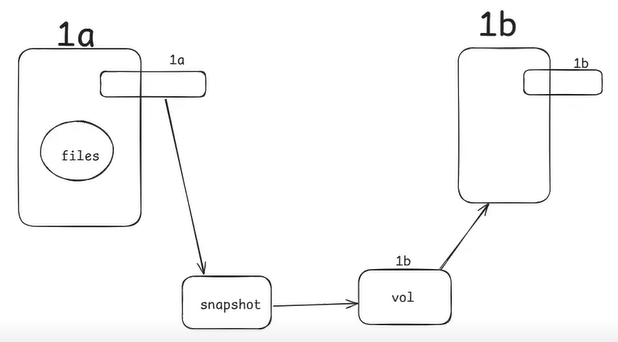
=> snapshots are region specific and volumes are zones specific.

=> From Volume we can create snapshot and from snapshot we can create volume

volume =====> snapshot ========> volume

=> Snapshot can't be attached to EC2 instance directley

(volumes can be attached to ec2 instances directley).



**Interview Question**

**How to copy data from 1a availability zone to 1b availability zone ?**

**Ans –**

Since **snapshots are region-specific** and **volumes are zone-specific**, we cannot directly move an EBS volume from **us-east-1a to us-east-1b**. However, we can achieve this by:

1. **Creating a Snapshot** of the EBS volume in **AZ 1a**.
2. **Creating a new Volume** from the snapshot in **AZ 1b**.
3. **Attaching the new Volume** to an EC2 instance in **AZ 1b**."

----------------------------------------------------------------------------------------------------------------

**Que - How to crate a snapshot ?**

**Ans-** select volume 🡪 Action 🡪 create snapshot 🡪 Add Dicription.

**Que - How to Open and Use an AWS EBS Snapshot?**

**Ans -** An AWS EBS snapshot is a backup of an EBS volume, and it cannot be opened directly. However, you can use it by **creating a new EBS volume** and **attaching** it to an **EC2 instance**.

1) What is EC2 & Why

2) What is AMI

3) What is Key Pair

4) What is Security Group

5) Launch Windows VM and connect with RDP client

6) Launch Linux VM and connect with SSH Client

7) What is User Data

8) Types of IPs

9) Launching Static website using HTTPD webserver

10) EBS Volumes & Snaphosts

11) Load Balancers

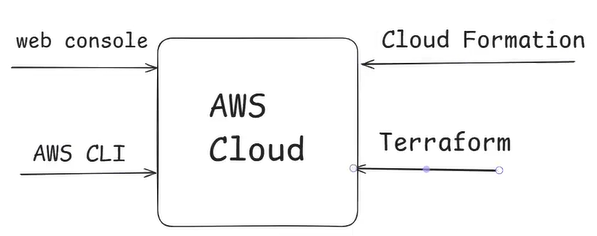
12) OSI Model

13) Auto Scaling Group

**================================================================**

**AWS CLI 10-27-Aws-16-JAN-25**

**================================================================**



=> AWS provides multiple ways to configure and manage infrastructure

=> We can use below 4 ways to manage AWS infrastructure

**1) AWS Management Web Console (browser)**

**2) AWS CLI (Command Line Interface)**

**3) Cloud Formation**

**4) Terraform**

**===========================**

**Using the AWS web console**

**===========================**

=> It is a graphical user interface to connect with various AWS resources, their configuration, modification, etc.

=> It is very simple to use and does not require knowledge of scripting.

Create a user 🡪select user 🡪 security credentials 🡪 create access key 🡪select cli.

**============================**

**AWS Command Line Interface**

**============================**

=> Using AWS CLI we can manage infrastructure in AWS cloud using commands.

=> Nowadays it is not required just for the knowledge we are using.

Ex :

$ aws s3 ls

$ aws ec2 describe-instances

**============================**

**Configuring AWS CLI**

**============================**

1) Login into AWS account & generate access keys

Access Key : **AKIAVFIWIS4WAOQV4LY2**

Secret Access key : 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

Note: Not recommended to use root user access keys.

2) Download and Install AWS CLI s/w

For windows : https://awscli.amazonaws.com/AWSCLIV2.msi

3) Configure AWS CLI

$ aws --version

$ aws configure

Note: AWS configure command will ask for access key, secret access key, region and output format.

######### CLI Documentation : https://docs.aws.amazon.com/cli/latest/reference/ ##########

**=======================================**

**Working with AWS S3 Service using AWS CLI**

**=======================================**

=> AWS S3 is an object storage service.

# list down all buckets available

* $ aws s3 ls

# list down all objects of bucket

* $ aws s3 ls s3://mybucket --recursive

# create bucket in s3

* $ aws s3 mb s3://ashokit097978

# copy file into bucket

* $ aws s3 cp test.txt s3://ashokit097978/test.txt

# remove file from bucket

* $ aws s3 rm s3://ashokit097978/test.txt

# remove bucket

* $ aws s3 rb s3://ashokit097978

**==========================**

**Working with EC2 AWS CLI**

**==========================**

# list down all ec2 instances available

$ aws ec2 describe-instances

# create key-pair in ec2

$ aws ec2 create-key-pair --key-name test --output text > test.pem

# Launch EC2 instance

$ aws ec2 run-instances --image-id ami-07b69f62c1d38b012 --instance-type t2.micro --key-name test

Instance ID : i-03cec5835807f5d8a

# stop ec2 instance

$ aws ec2 stop-instances --instance-ids i-03cec5835807f5d8a

# start ec2 instance

$ aws ec2 start-instances --instance-ids i-03cec5835807f5d8a

# terminate ec2 instance

$ aws ec2 terminate-instances --instance-ids i-03cec5835807f5d8a

**================================**

**Working with AWS RDS using CLI**

**================================**

# creating RDS instance

$ aws rds create-db-instance --db-instance-identifier test-mysql-instance --db-instance-class db.t3.micro --engine mysql --master-username admin --master-user-password secret99 --allocated-storage 20

# delete RDS instance

$ aws rds delete-db-instance --db-instance-identifier test-mysql-instance --skip-final-snapshot --no-delete-automated-backups

**=======================================**

**Installing AWS CLI in Ubunut Linux VM**

**=======================================**

$ sudo apt-get install -y python-dev python-pip

$ sudo pip install awscli

$ aws --version

$ aws configure

**=======================================**

**Installing AWS CLI in Amazon Linux VM**

**=======================================**

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

**============**

**Cloud Watch**

**============**

=> It is used for monitoring aws resources.

=> CloudWatch enables real-time monitoring of AWS resources such as Ec2 , EBS volumes, Elastic Load Balancing and RDS instances.

**Note : -** we cant monitor resources 24/7 Instead of watching cloud resources we can monitor resources by using cloud watch.

**==========**

**Usecase**

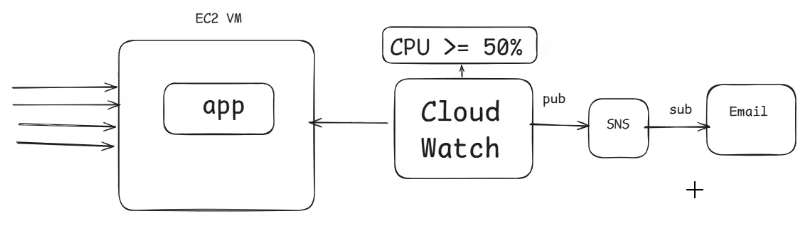
**==========**

=> Create EC2 instance in aws cloud

=> Monitor that EC2 Instance 24/7

=> If Ec2 instance CPU utilization is above 50 % then trigger Cloud watch alarm.

=> When alarm got triggered send email notification to management.

****

**======================================**

**How to send Email notification in AWS**

**======================================**

=> In AWS we have **SNS** to send notifications

**SNS : Simple Notifcation Service**

=> SNS works based on Pub and Sub Model

**Pub** => Publisher (will send msg to sns topic)

**Sub =**> Subscriber (will recieve msg from topic)

**==============================**

**Cloud Watch & SNS - Lab Task**

**==============================**

1) Create SNS Topic with Standard Create option

Topic Name : ashokit

2) Create Email Subscription in SNS Topic (Confirm Subscription recieved in email)

3) Create EC2 Instance and Configure Cloud Watch Alaram To Monitor

-> Select EC2 Instance -> Action -> Monitor and trouble Shoot -> Manager Cloud Watch Alarms -> Create cloud watch Alarm

Alaram Notification : Select SNS Topic which we have created

Alaram Thresh Hold : Avg CPU Utiliation >= 3%

4) Connect to EC2 VM using ssh client and increase load

$ sudo yum install stress -y

$ sudo stress --cpu 8 -v --timeout 180s

Note: Execute stress command 3 to 4 times...

5) Observe the behaviour of Cloud Watch / Alaram / SNS

(We should get Email Notification)

**Note: When Alarm got triggered, its status will be changed to 'In Alarm'**

=> We can Monitor our Alarm history (how many times it got triggered)

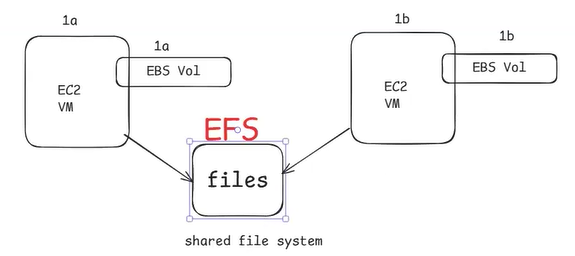
(Goto Cloud Watch -> Select Alarms -> Click Alarm -> Click on History).

**==========================**

**Elastic File System (EFS) 11-27-Aws-17-JAN-25**

**==========================**

* AWS EFS lets you create scalable file storage to be used in Ec2 Machines.
* EFS we can share with multiple EC2 instances at a time.
* If we have some configuration files of application and we want to access them in mulitple ec2 instances at a time then we can use EFS concept.



**When to Use Which?**

**EFS**

* **Use EFS** if you need shared file storage across multiple EC2 instances.

------------------------------------------------------------------------------------------------------------------

**EBS**

* Only single machine at a time
* **Use EBS** if you need high-performance, low-latency storage attached to a single EC2 instance.

------------------------------------------------------------------------------------------------------------------

**S3**

* We can upload data and can download.
* **Use S3** if you need to store large-scale, unstructured data like backups, images, or logs.
* **Not related to the any machine002E**

**=============================**

**Step-1 : Create EFS in AWS**

**=============================**

=> Login into the AWS console

=> Go to Services and select EFS under storage

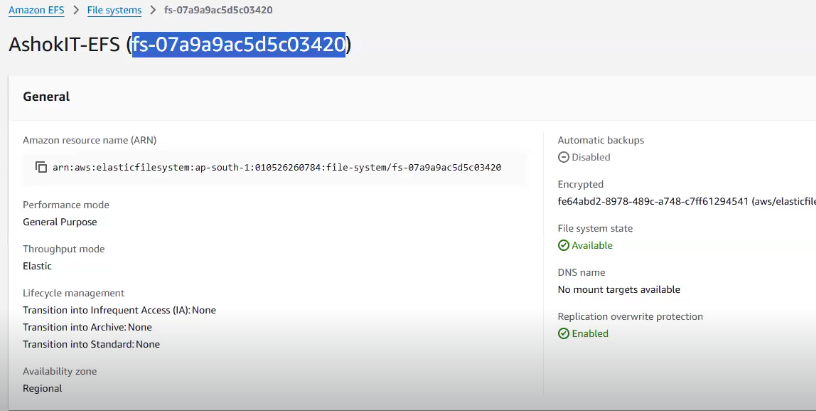
=> Click on Create file system

Whenever we create a file system - id will be generated.

@@@@@@@@@ File System ID : fs-07a9a9ac5d5c03420

=> Create 2 EC2 instances (Amazon Linux AMI)

**Enable security Group inbound rule with "NFS" protocol with 2049 port number.**



**===============================================**

**Step-2 : Mount EFS to First EC2 VM**

**===============================================**

# Login to EC2 instance and install the NFS client

$ sudo yum install -y amazon-efs-utils

# Let's create a folder where you want to mount the EFS

$ sudo mkdir efsdir

# Mount EFS Filesystem (Make sure you changed FileSystem ID)

$ sudo mount -t efs -o tls fs-0eb0cb7d871512c09:/ efsdir

# Change the directory to the mount point that is created above using the command:

$ cd efsdir

# Create a sample text file:

$ sudo touch f1.txt f2.txt

# Run ls command to list the contents of directory.

**=======================================**

**Step-2 : Mount EFS to Second EC2 VM**

**=======================================**

# Login to EC2 instance and install the NFS client

$ sudo yum install -y amazon-efs-utils

# Let's create a folder where you want to mount the EFS.

$ sudo mkdir efsdir

# Mount EFS Filesystem (Make sure you changed FileSystem ID)

$ sudo mount -t efs -o tls fs-0eb0cb7d871512c09:/ efsdir

# Change the directory to the mount point that is created above using the command:

$ cd efsdir

# check the files available

$ ls

**Note : The files we have created in First EC2 instance, should display in second ec2 instance.**

|  |  |
| --- | --- |
| **Part** | **Explanation** |
| sudo | Runs the command with superuser privileges (required for mounting). |
| mount | The Linux command to attach a filesystem to a directory. |
| -t efs | Specifies the type of filesystem as **EFS (Elastic File System)**. |
| -o tls | Enables **TLS encryption** for secure data transfer between EC2 and EFS. |
| fs-0eb0cb7d871512c09:/ | The **EFS File System ID** (unique identifier for your EFS instance in AWS). |
| efsdir | The **mount point** (a local directory where EFS will be accessible). |

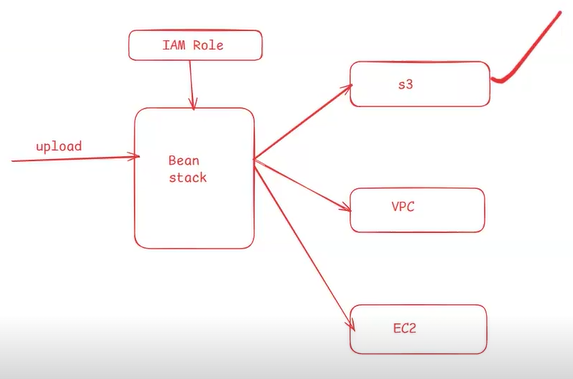
**==================**

**Elastic Beanstack 11-27-Aws-17-JAN-25**

**==================**

=> End to end web application management service.

=> It provides platform as a service (PaaS).

****

* First it will talk to the S3 for upload the code.
* One service want to talk to another service then policy ( Roles ) required.
* Application code will stored in s3 bucket that why beanstack talk to the s3
* Talk VPC for the network
* Talk to the EC2 for creating virtual machine.

**=======================================**

**Dynamic Web App - Deployment process**

**=======================================**

1) Create Network (VPC)

2) Create Security Groups

3) Enable Inbound Rules

4) Create EC2 Instances

5) Install Required Softwares to run the code

Ex : Java, Tomcat, IIS

6) Create Load Balancer

7) Setup Auto Scaling Group for High Availability

8) Deploy our application code

**Note: When we use AWS Elastic Bean stack services then first 7 steps will be taken by Elastic Bean stack. We are responsible for only application deployment.**

**================================**

**Elastic Beanstack Pricing Model**

**================================**

=> There is no additional charge for Elastic Beanstalk.

=> We need to pay the amount for the resources which are created by bean stack.

Ex: EC2 instances, S3 buckets, LBR, ASG etc...

**================================**

**Lab Task on Elastic Beanstack**

**================================**

Step-1) Create IAM Role with below 3 policies

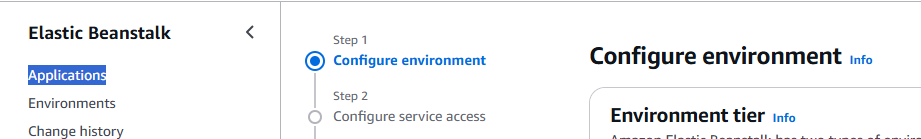
- AWSElasticBeanstalkMulticontainerDocker

- AWSElasticBeanstalkWebTier

- AWSElasticBeanstalkWorkerTier

**Ex : Role Name : Sachin-IAM**

Step-2) Create Application using Beanstack

Click on applications🡪 Create application 🡪first-app

Fisrtapp🡪 Platform 🡪java 🡪application code🡪sample🡪next 🡪

Select Service Access 🡪next 🡪tick public ip address 🡪select all availability zones 🡪next

Security group 🡪default 🡪 next 🡪 submit

Step-3) Create Environment for the application by choosing required Runtime.

Ex: Java or Python or Dot Net

Note: Once environment is created it will generate DNS to access our application.

**====================================**

**Uploading Java SpringBoot Web App 12-27-Aws-20-JAN-25**

**====================================**

=> Take jar file of java springboot web app

=> Go to Elastic Beanstack environment and upload your jar and give version number for your application

Ex: v1.0

=> Go To enviornment Properties and set SERVER\_PORT as 5000

Ex : SERVER\_PORT = 5000

- Select Environment

- Go to Configuration

- Edit "Updates, monitoring, and logging" option

- Set Environment Property and apply

=> After environment got re-started, we can access our application by using DNS url.

**=============**

**AWS Lambdas**

**=============**

-> AWS lambdas are used to achieve serverless computing.

-> Serverless computing means run the application without thinking about servers.

-> AWS Lambda will take care of servers required to run our application.

**==============================**

**Elastic Beanstack vs Lambdas**

**==============================**

=> When we deploy our app using Elastic Beanstack we need to pay bill amount for the resources which got created by beanstack based on **pay as you go model**. Even if customers are not accessing our application bill be generated for the resources created bean stack like s3, ec2, lbr etc...

* Always server will be available to handle the request

------------------------------------------------------------------------------------------------------------------

**Lambdas**

=> When we deploy our app using AWS lambdas then it will charge based on Pay as you use model. If your application code is executed then only bill be generated. If nobody accessing your application then no bill.

**code executed for only 5 mins : bill will generate only for 5 mins**

* Whenever request will come container will prepare to run our application.

**==================================**

**Running Java Code with AWS Lambda**

**==================================**

1) Create Lambda Function with 'java 21' runtime

- Enable Functional URL

- Auth Type None (Public Access)

Note: Once lambda function got created we can see URL to access that function.

2) Access Lambda function using its URL.

2) Upload jar file in 'Code Source'

3) Configure Handler in Runtime

Class Name : in.ashokit.LambdaHandler

Method Name : handleRequest

Handler Syntax : className :: methodName

Ex: in.ashokit.LambdaHandler::handleRequest

4) Access Lambda function using its URL (we can latest msg coming from our java app)

**===========================**

**Static website deployment**

**===========================**

Approach-1) S3 static website hosting

Approach-2) Take EC2 VM + Install HTTPD + Run Static website

**============================**

**Dynamic Website Deployment**

**============================**

Approach-1) Take EC2 VM Then install required softwares and run your application.

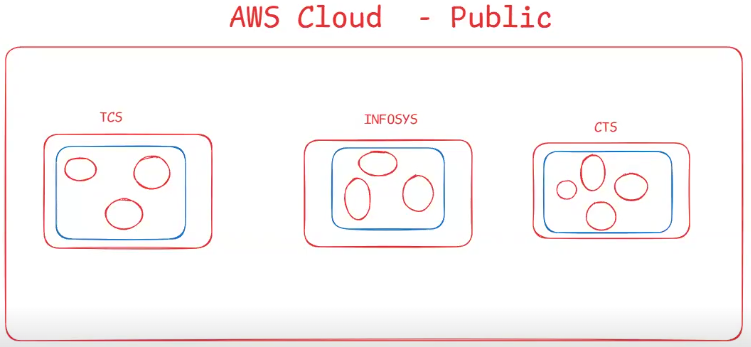
Approach-2) Elastic Beanstack (PaaS)

Approach-3) Lambdas (Serverless Computing)

**\* \* \* \* VPC : Virtual Private Cloud \* \* \* \* \***

**--------------------------------------------------------**

**14-27-Aws-24-JAN-25**

****

Isolated Infrastructure

Isolated Infrastructure

Isolated Infrastructure

=> VPC stands for Virtual Private Cloud.

=> VPC provides Virtual Network Environment.

=> A VPC allows users to create and manage their own isolated virtual networks within the cloud.

=> VPC provides flexible and secured network environment to manage our resources in AWS cloud.

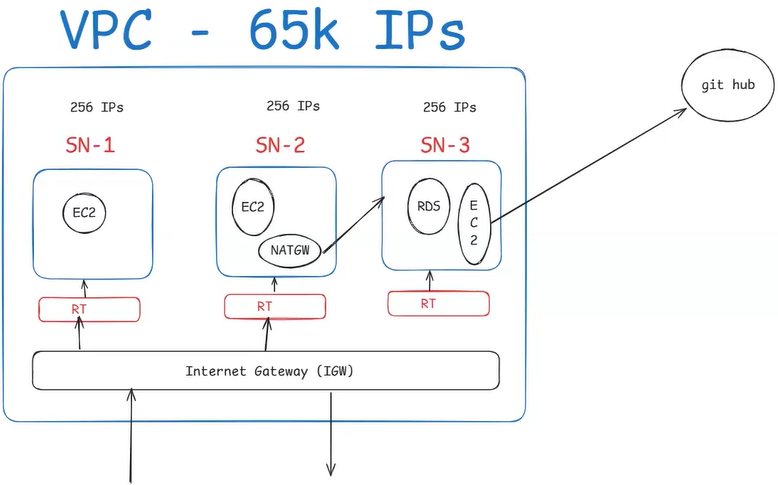
When everybody creating infrastructure in AWS cloud they created isolated, so other people should not access our infrastructure, that’s where VPC comes into the picture.

**================**

**VPC terminology**

**================**

**Note – Maximum for 1 VPC we can get 65,000 ip’s.**

****

Public Subnet

**1) Isolated Network**

* In AWS, a VPC (Virtual Private Cloud) allows you to create an isolated network environment within AWS

**2) CIDR Blocks (Classless Inter-Domain Routing)**

- Used to allocate ip address within the network.

* Define ip address ranges.

**3) Subnets (public & Private)**

**The process of diving VPC into the small network is called as subnets.**

* There are two types of subnets:
* **Public Subnet →** Has direct access to the internet (via an Internet Gateway). Used for resources like web servers.
* **Private Subnet →** Has no direct internet access. Used for databases, backend servers, and internal applications.

**4) Route Tables**

* A Route Table contains a set of rules that determine how network traffic is directed.
* Each subnet is associated with a route table to define whether traffic goes to the internet, stays within the VPC, or goes to another network.

**5) Internet Gateway (IGW)**

* provides incoming and outgoing traffic for the VPC.
* If we provide internet to the subnet then it will become public subnet , so outside people can access it.

**6) NAT Gateway (NGW)**

* I don’t want incoming access , I want outgoing access then we can go for NAT-GW.
* A NAT (Network Address Translation) Gateway enables instances in a private subnet to access the internet while keeping them inaccessible from the internet.
* It is used for outbound traffic like updates, patches, and API calls.

**7) VPC Peering**

* VPC Peering is used to connect two VPCs privately using AWS’s network.
* It enables secure communication between VPCs across different AWS accounts or regions.
* Peered VPCs can share databases, applications, and other resources without using public internet routes.

**8) Security Groups (Resource level – Only one machine )**

* It controls inbound and outbound traffic based on rules that specify allowed IP addresses, ports, and protocols.
* Security group will work at resource level.

**9) NACL (subnet level)**

* NACL will work at subnet level.

**Que- Why we need VPC in an organization ?**

**Ans –** Isolated environment for the organization. What environment we are going to create so other people should not able to access it.

Crating VPC is nothing but the allocating ip addresses.

**Que – how to make subnet public and private ?**

**Ans –** Through the internet gateway.

**Interview Question – What is NAT and Internet Gateway ?**

**===============**

**Types of IP's 15-27-Aws-27-JAN-25**

**===============**

**1) IPV4**

**2) IPV6**

**=========**

**IPV4**

**=========**

=> IPv4 addresses are 32-bit numeric addresses written in four sets of numbers separated by period (e.g : 192.168.0.1)

=> It is the most widely used IP version and supports approximately 4.3 billion unique addresses

=> However, due to the increasing number of devices connected to the internet, IPv4 addresses are running out, leading to the adoption of IPv6.

**===========**

**IPV6**

**===========**

IPv6 addresses are 128-bit alphanumeric addresses written in eight sets of four hexadecimal digits separated by colons

e.g : 2001:0db8:85a3:0000:0000:8a2e:0370:7334

=> IPv6 provides a significantly larger address space than IPv4, with approximately 340 undecillion unique addresses.

=> It was introduced to overcome the IPv4 address exhaustion issue and support the growing number of internet-connected devices.

**============**

**VPC Sizing**

**============**

=> The process of deciding no.of IPs required for VPC and no.of IPs required for Subnets is called as VPC Sizing.

-> VPC Sizing will be calculated in 2 power

10.0.0.0/16 => 2 power (32-16) => 2 power 16 => 65, 536

10.0.0.0/32 => 2 power(32-32) => 2 power 0 => 1

10.0.0.0/31 => 2 power (32-31) => 2 power 1 => 2

10.0.0.0/30 => 2 power (32-30) => 2 power 2 => 4

10.0.0.0/29 => 2 power (32-29) => 2 power 3 => 8

10.0.0.0/24 => 2 power (32-24) => 2 power 8 => 256

Note: For VPC we will use /16 and for subnet we will use /24 as CIDR range.

**=======================**

**VPC Lab Task For Today**

**=======================**

1) Create VPC

Use VPC CIDR as : 10.0.0.0/16

Note : It will create one Route Table by default. Rename it as "ashokit-private-rt"

2) Create 2 Subnets (Public and Private Subnets)

public Subnet CIDR : 10.0.0.0/24 (256 ips)

private Subnet CIDR : 10.0.1.0/24 (256 ips)

3) Create Internet Gateway and Attach to our VPC

4) Create one new Route Table (Name it as public Route Table)

5) Peform Subnet Association with Route Tables

=> public-rt => public-sn

=> private-rt => private-sn

6) Attach IGW to Public Route Table in Routes, so that subnet will become public-sn (internet will be available).

7) Create One EC2 VM in public subnet and another EC2 vm in private subnet.

8) Test connectivity of both vms using SSH client.

**=====================**

**Step-1 : Create VPC**

**=====================**

-> Create a VPC

-> CIDR Block : 10.0.0.0/16

-> Select No IPV6 CIDR blok

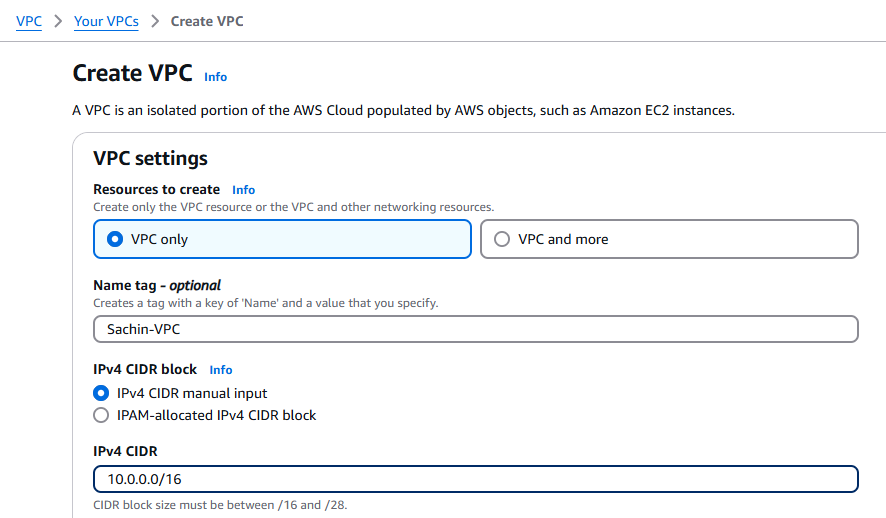
-> Select Default Tenancy

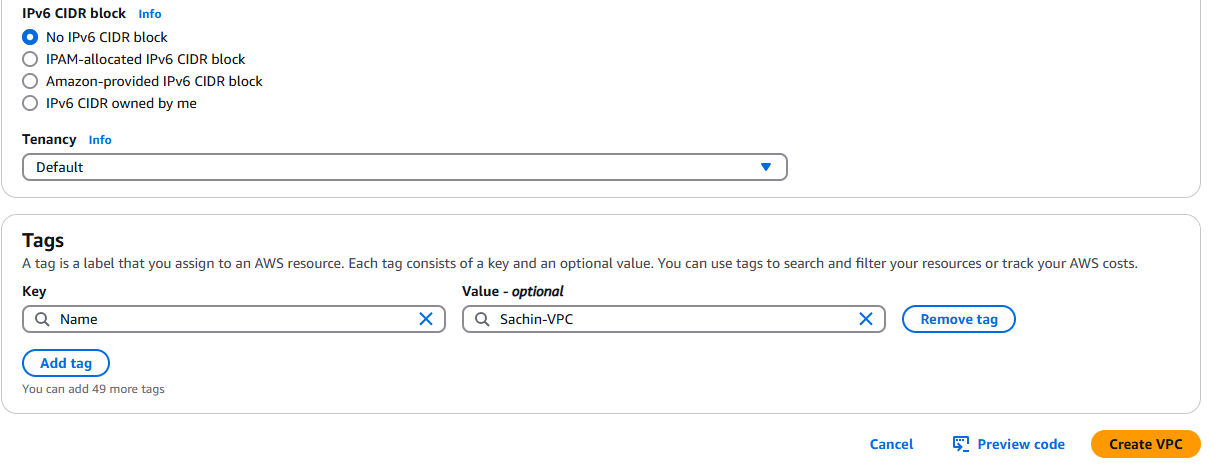
-> Create VPC

Note: After creating VPC verify its details

(DNS hostnames -> Disabled)

Note: One Route Table will be created for VPC by default. Rename it as "Ashokit-Private-Route-Table"



****

**===========================**

**Step-2 : Create 2 Subnets**

**===========================**

**------------------------------------------------------------------------------------------------**

**Create Subnet-1**

**-----------------------------------------------------------------------------------------------**

* Create Subnet
* Select VPC
* Name : Sachin-Public-SN1
* select availability zone as 1a
* CIDR Block : 10.0.0.0/24 (It will take 256 ips)

**----------------------------------------------------------------------------------------------------**

**Create Subnet-2**

**---------------------------------------------------------------------------------------------------**

* Create Subnet
* Name : private-subnet-az-1b
* select availability zone as 1b
* CIDR Block : 10.0.1.0/24 (It will take 256 ips)
* AWS will reserve 5 ips in every subnet (we will get only 251)

**Note: Every subnet will have Route Table and NACL**

**=================================**

**Stpe-3 : Create Internet gateway**

**=================================**

Note: By default one IGW will be available and it will be attached to default VPC

* Create custom Internet Gateway (ashokit-vpc-igw)
* Attach this IGW to VPC (we can attach IGW to only one VPC)

**=================================**

**Step-4 : Create Route Table**

**=================================**

**Note: When we create VPC, we will get only route table by default. It is called as Main route table.**

Note : Change existing route table name ‘Sachin-VPC-Private-SN'

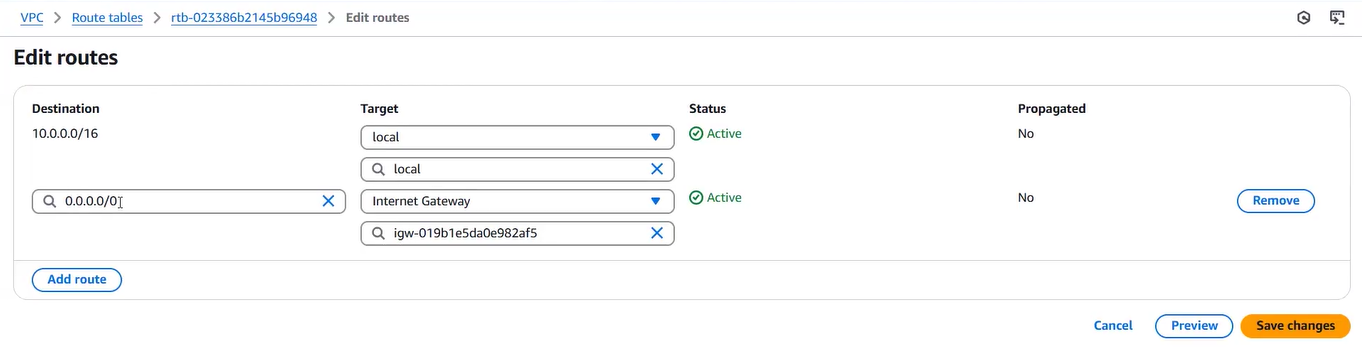
* Create one new Route Table ( Sachin-VPC-Public-SN )
* Choose vpc and create it
* Now We have 2 route tables
* Goto route table and attach route tables to subnets (Subnets association for Route Tables)
* Private Route Table should have Private Subnet
* Public Route Table should have Public Subnet

**==========================================**

**Step-5 : Making Subnet as public**

**==========================================**

* Goto public Route Table -> Edit Routes
* Add Destination as 0.0.0.0/0 and Target as IGW -> Save
* Subnet Associations -> Edit SNET -> Select Public Subnet



**======================================**

**Step - 6 : Create EC2 (Public EC2)**

**======================================**

* Choose AMI
* Select Public Subnet
* Change default VPC to desired.
* Assign Public IP as Enable
* Add SSH and Http Protocols
* Download KeyPair
* Launch Instance
* Check incoming and outgoing traffic is allowed or not
* Command to check internet is connected or not
* Ping www.google.com

**Note: Goto VPC and Enable DNS Host Enable**

**========================================**

**Step - 7 : Create EC2 (Private EC2)**

**========================================**

* Choose AMI
* Select VPC
* Select Private Subnet
* Assign Public IP as Enable
* Add existing security group and change from default
* Add SSH (source : custom, Range : 10.0.0.0/16)
* Download KeyPair
* Launch Instance



Connection is refused because we have not attach the Internet gateway.

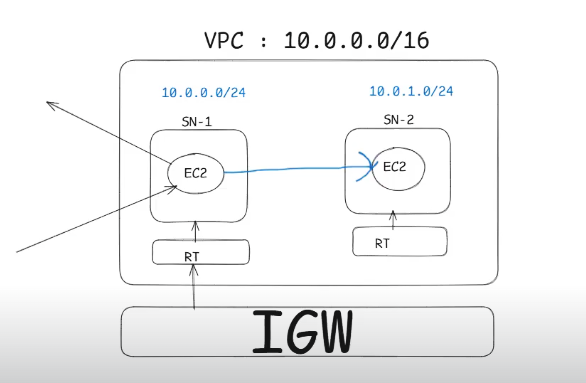
**=================================**

**Step - 8 : Test EC2 Connections**

**=================================**

-> Connect to Public EC2 using MobaXterm (It should allow to connect)

-> Connect to Private EC2 using MobaXterm (It shouldn't allow to connect)

****

**\* Interview Question**

**I have a private EC2 machine which don’t have internet gateway how to use that machine ?**

**Ans –** Directly we cannot connect the private subnet machine for that we need to connect from the public subnet machine.

* First we enter public subnet machine the we can enter to the private subnet machine using private IP

**================================================================**

**Step - 9 : Connect with 'private-ec2' from 'public-ec2' using 'ssh' connection**

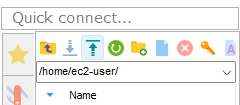
**================================================================**

**Note: As both Ec2 instances are available under same VPC, we should be able to access one machine from another machine.**

**----------------------------**

**Procedure to access**

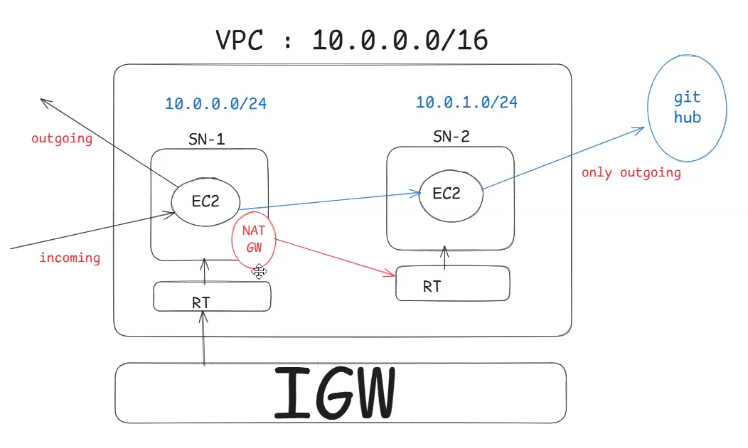
**----------------------------**

* Upload pem file into public-ec2 machine (in mobaxterm we have upload option)
* 
* Execute below command to provide permission to access pem file
* $ chmod 400 <pem-file-name>
* Execute below command to make ssh connection from public-ec2 to private-ec2
* $ ssh -i "pem-file-name" ec2-user@<Private VM IP>
* Ex: ssh -i "ashokitnewkey.pem" ec2-user@65.2.73.111
* **Note: It should establish connection (this is internal connection)**
* Try to ping google from private ec2 (it should not allow because IGW is not available)

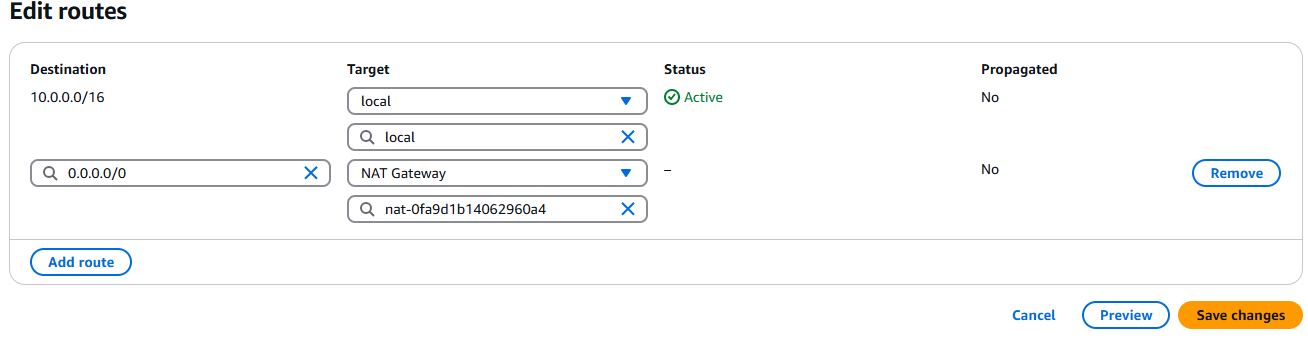
**Que - I want to add outgoing internet to the private ec2-VM ?**

**Ans –** Through the NAT gateway we can achieve outgoing internet access.

* We need to crete NAT gateway in the public subnet and attach to the private subnet route table.
* NAT gateway is paid service that internally uses elastic IP.

****

* After creating NAT gateway we are going to attach NAT GW to the private subnet.
* Only outgoing internet not the outgoing.
* Go to route tables 🡪 select private subnet 🡪Routes 🡪 edit Routes 🡪add route.



**=============================**

**VPC with NAT Gateway Lab Task**

**=============================**

1) Create NAT gateway in public subnet

2) Add NAT gateway in 'private-subnet-route-table'

3) After NAT Gateway, we should be able to ping google from 'private-ec2' also

Note: Delete Elastic IP and NAT Gateway after practice

**====================**

**What is VPC Peering 16-27-Aws-28-JAN-25**

**====================**

* It is used to establish communication between 2 VPCs.
* Using VPC peering technique, we can access one VPC resources in another VPC
* We can establish connection between different account VPC also.
* Default VPC should allow custom VPC traffic and Custom VPC should allow default VPC traffic.
* To establish the connection, lets create VPC peering
* On the left navigation panel under VPC -> Peering Connections:
* **VPC (Requester) = ashokit\_aws\_custom\_vpc**
* **VPC (Accepter) = default\_vpc**
* Now you would see the status Pending Acceptance which means, Requestor has sent a request to the peer now target VPC needs to accept the request.
* Go to VPC Peering -> Click on Actions -> Accept Request
* Now we need to make entries in Route Tables
* Now navigate to Route Tables, in Default VPC RT(Route Table) -> Edit routes

**\* \* \* \* \* \* \* \* \* Default VPC Route Table should have 3 routes \* \* \* \* \* \* \* \* \***

(Local + all traffic with IGW + ashokit\_aws\_custom\_vpc IP Range)

* 172.31.0.0/16 - local
* 0.0.0.0/0 - Internet-gateway
* 10.0.0.0/16 - vpc peering (We need to add this)

------------------------------------------------------------------------------------------------------------------

**########### Custom VPC Route Table should have 3 routes #########**

* (Local + All traffic with IGW + Default VPC IP Range)
* 10.0.0.0/16 - local
* 0.0.0.0/0 - Internet-gateway
* 172.31.0.0/16 - vpc (We need to add this)

------------------------------------------------------------------------------------------------------------------

**########### Allow Traffic in VPC Security Groups ###########**

Edit Security Group of Default and Custom VPC to allow traffic from each other

* Default VPC Security Group looks like
* SSH - 22 - all
* All Traffic

----------------------------------------------------------------------------------------

* Custom VPC Security Group would look like
* SSH - 22 - all
* All Traffic

-----------------------------------------------------------------------------------------

* First create 1 EC2 VM in custom VPC
* Name 🡪custom VPC VM 🡪Network setting 🡪 custom v
* Create 1 EC2 VM in default VPC and create 1 EC2 VM in custom vpc

--------------------------------------------------------------------------------------------------------

* # Ping default-vpc EC2-VM private IP from ashokit-custom-vpc vm
* $ ping <private-ip>
* From the custom VPC we need to connect to the default VPC
* Public ip anybody can ping we need to ping private ip.
* If ping is not working check default VM security group and add All Traffic 🡪IPV4 Save.

**Note: If ping is successfully then VPC peering is working.**

**===============================================================**

**Q ) What is the difference between NACL and Security Groups ?**

**===============================================================**

* Security group will act at the **resource level** .
* NACL will act at the **Subnet level.**

**================**

**Security Group**

**================**

* Security Group acts as a Firewall to secure our resources
* Security Group contains Inbound Rules & Outbound Rules

1. **inbound rules ---> incoming traffic**
2. **outbound rules ---> outgoing traffic**

* In One security group we can add 50 Rules
* Security Group supports only Allow rules (by default all rules are denied)
* We can't configure deny rule in security group
* Ex : 172.32.31.90 ----> don't accept request from this IP (we can't do this in SG)
* Security Groups are applicable at the resource level (manually we have to attach SG to resource)
* Multiple Security Groups can be attached to single instance & one instance can have 5 security groups

**======**

**NACL**

**======**

* **NACL stands for Network Access Control List**
* NACL acts as a firewall for our Subnets in VPC
* NACL applicable at the subnet level
* NACL rules are applicable for all the resources which are part of that Subnet
* In NACL we can configure both Allow & Deny rules

Ex: We can block particual IP address (192.168.2.4) to connect with EC2 instance

* **One subnet can have only one NACL**

**Note: One NACL can be added to multiple subnets**