# AWS

Before cloud computing every organisation has, they own assets (visual machine etc) which manages by visualization team by working with DCOPS team

# Cloud computing:

They are two types:

1. Private – it is for the organizations (big company)
2. Public - it is for the public

* Aws
* Azure
* Google cloud

# **What is cloud computing?**

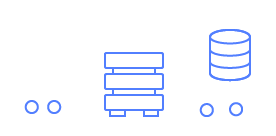
Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

Benefits of cloud computing:

1. Agility 🡪 quick start of cloud computing.
2. Elasticity
3. Cost saving
4. Deploying globally in minutes

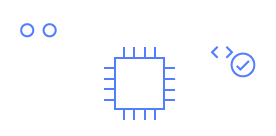
# **Types of cloud computing**

The three main types of cloud computing include Infrastructure as a Service, Platform as a Service, and Software as a Service. Each type of cloud computing provides different levels of control, flexibility, and management so that you can select the right set of services for your needs.



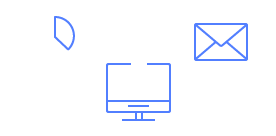
## **Infrastructure as a Service (IaaS)**

IaaS contains the basic building blocks for cloud IT. It typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS gives you the highest level of flexibility and management control over your IT resources. It is most similar to the existing IT resources with which many IT departments and developers are familiar.



## **Platform as a Service (PaaS)**

PaaS removes the need for you to manage underlying infrastructure (usually hardware and operating systems), and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don’t need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.



## **Software as a Service (SaaS)**

SaaS provides you with a complete product that is run and managed by the service provider. In most cases, people referring to SaaS are referring to end-user applications (such as web-based email). With a SaaS offering, you don’t have to think about how the service is maintained or how the underlying infrastructure is managed. You only need to think about how you will use that particular software.

# AWS Infrastructure

### Region

Availability zones—inside many data centers.

<https://aws.amazon.com/about-aws/global-infrastructure/?p=ngi&loc=0>

at now 32 regions 102 availability zone

region inside( 2 or more availability zones inside (many data centers)))

AWS has over 200 fully featured services for a wide range of technologies, industries, and use cases.

In devops:

Compute , storage, database, transfer, network, developer tools, manage and governances, secops services

# EC2

## EC2 Features

EC2 provides web services API for provisioning, managing, and deprovisioning virtual servers inside amazon cloud.

➢ Ease In Scaling Up/Down

➢ Pay only for what you use

➢ Can be integrated into several other services

## Ec2 Pricing

* On Demand Pay per hour or seconds.
* Spot Bid your price for unused ec2 capacity.
* Reserved Reserve Capacity(1 or 3 yrs) for discounts.
* Dedicated Hosts Physical Server dedicated for you.

AMI: Amazon Machine Image (AMI) provides the information required to launch an instance, which is a virtual server in the cloud.

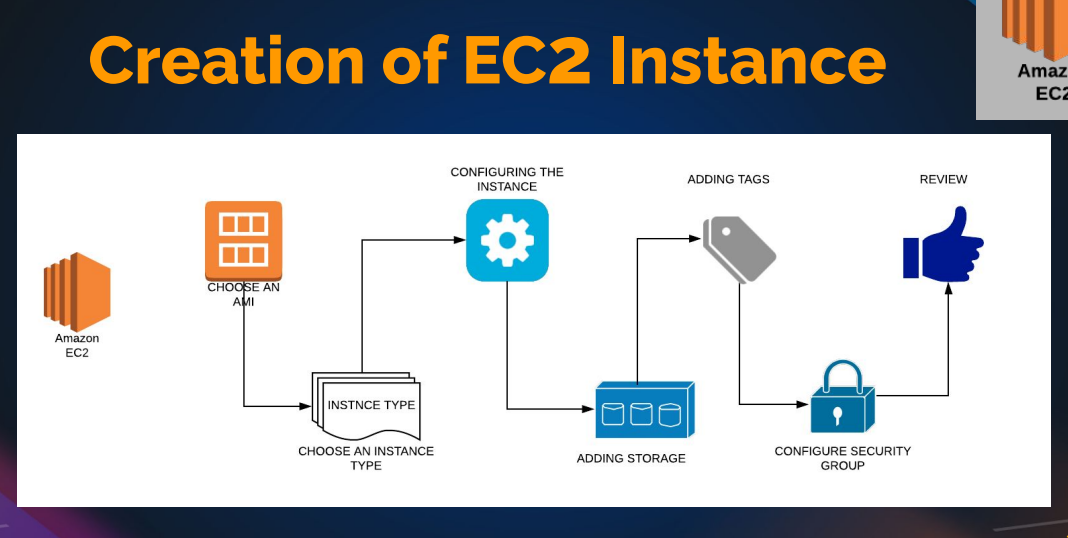
INSTACES TPYES: When you launch an instance, the instance type that you specify determines the hardware of the host computer used for your instance

EBS: amazon elastic bock store(EBS) Amazon EC2 provides you with flexible, cost effective, and easy-to-use data storage options for your instances.

Tag :Tag is a simple label consisting of a customer-defined key and an optional value that can make it easier to manage, search for, and filter resources.

security group: A security group acts as a virtual firewall that controls the traffic for one or more instances

public–key: Amazon EC2 uses public–key cryptography to encrypt and decrypt login information.



* Hand-on: go to ec2 instances and create a instances in free tiar and create key-pair(ssh, http) and create security group .

Do the proper Tags

Change/give the instances name and description

Add the security group

Check the network interface

* Do the provision step startbootstap and launch the website
* Launch website using both ssh and public Ip
* Create a elastic Ip and use it (it will get some cost)
* Action 🡪 change instances type ---> to increase /decrease the site of the instances
* Check the system log ----> in monitor & networking

To check the open ports : ss -tunlp | grep 80

8080 is Apache port number to assign it

# EC2 INSTANCES CRATION

Gathering information

1. Key pairs

2. Security Group

3. Instance Launch

sudo apt update

sudo apt install apache2 wget unzip -y

wget https://www.tooplate.com/zip-templates/2128\_tween\_agency.zip

unzip 2128\_tween\_agency.zip

sudo cp -r 2128\_tween\_agency/\* /var/www/html/

sudo systemctl restart apache2

# Gathering information

Toolplate website

2. OS a. Centos

3. Size => Ram, CPU, Network etc a. Min

4. Storage size a.

5 gigs for web server images

5. Project

6. Services/Apps Running a. SSH, Http, Mysql etc

7. Environment (Dev, QA, Staging, Prod) 8. Login User/ Owner

## Link : https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html

### Best Practices for Launching an EC2 Instance

#### 1. **Requirement Gathering**

* **Operating System**: Determine the OS you will use (e.g., Ubuntu, CentOS).
* **Instance Size**: Identify required CPU, RAM, and network speed.
* **Storage**: Specify the required storage space for the OS and applications.
* **Services**: List the services and applications that will run on the instance (e.g., SSH, HTTP, MySQL).
* **Environment**: Define whether it’s for development, staging, or production.
* **Login User and Ownership**: Identify the users who will log in and the owner for tagging purposes.

#### 2. **Creating a Key Pair**

* Create a key pair before launching the instance to ensure secure access.
* Use a naming convention that reflects the project and environment, e.g., **twin-dev-region**.

#### 3. **Setting Up Security Groups**

* Security groups act as virtual firewalls to control inbound and outbound traffic.
* Avoid opening all ports to all IPs; instead, specify necessary ports and IP ranges.
* For SSH access, limit it to specific IP addresses.
* Add rules for HTTP (port 80) or HTTPS (port 443) as needed.

#### 4. **Launching the Instance**

* **Tags**: Use tags to organize and manage instances (e.g., name, project, environment, owner).
* **AMI Selection**: Choose the appropriate Amazon Machine Image (AMI) based on the OS requirement.
* **Instance Type**: Select the instance type according to your requirement (e.g., t2.micro for free tier).
* **Key Pair**: Select the previously created key pair for SSH access.
* **Security Group**: Apply the created security group with the appropriate rules.
* **Storage Configuration**: Configure storage as per requirements.
* **Advanced Details**: Skip additional provisioning during initial setup if not needed.

#### 5. **Connecting to the Instance**

* Use SSH to connect to the instance using the key pair.
* Verify the default username (e.g., **ubuntu** for Ubuntu instances).

#### 6. **Setting Up a Web Server**

* **Install Apache**: Set up Apache or your preferred web server.
* **Deploy Website**: Place website files in the web server's root directory (e.g., **/var/www/html**).
* **Verify Apache Service**: Ensure the Apache service is running.

#### 7. **Testing Website Access**

* Modify security group to allow HTTP traffic on port 80.
* Access the website using the instance's public IP address.

### Example Workflow

1. **Delete Existing Key Pair**:

aws ec2 delete-key-pair --key-name "previous-key"

Create a new key pair:

aws ec2 create-key-pair --key-name "twin-dev-region" --query "KeyMaterial" --output text > twin-dev-region.pem

**Create Security Group**:

aws ec2 create-security-group --group-name "twin-web-dev-sg" --description "Security group for twin web dev"

Add inbound rules:

aws ec2 authorize-security-group-ingress --group-name "twin-web-dev-sg" --protocol tcp --port 22 --cidr "your-ip-address/32"

aws ec2 authorize-security-group-ingress --group-name "twin-web-dev-sg" --protocol tcp --port 80 --cidr "0.0.0.0/0"

**Launch EC2 Instance**:

aws ec2 run-instances --image-id "ami-xxxxxxxx" --count 1 --instance-type "t2.micro" --key-name "twin-dev-region" --security-groups "twin-web-dev-sg" --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=web01},{Key=Project,Value=twin},{Key=Environment,Value=dev},{Key=Owner,Value=lead-devops}]'

**Connect to Instance**:

ssh -i "twin-dev-region.pem" ubuntu@<instance-public-ip>

**Set Up Apache and Deploy Website**:

sudo apt update

sudo apt install apache2

sudo systemctl start apache2

sudo systemctl enable apache2

1. **Verify Website**:
   * Access the website using the public IP: **http://<instance-public-ip>**

By following these steps, you ensure a secure, organized, and efficient process for launching and managing EC2 instances.

### Managing EC2 Instances and Elastic IPs

#### **Powering Off and Restarting an Instance**

1. **Check IP Addresses**:
   * Note the public and private IP addresses of the instance.
2. **Stopping the Instance**:
   * Stop the instance. The public IP will be released, but the private IP will remain.
3. **Restarting the Instance**:
   * When you restart the instance, it will get a new public IP. The private IP remains unchanged.

#### **Using Elastic IPs**

1. **Allocating an Elastic IP**:
   * Allocate an Elastic IP from AWS. This IP is reserved for your use only.
2. **Associating Elastic IP**:
   * Associate the Elastic IP with your instance. This IP remains the same even after stopping and starting the instance.

#### **Benefits of Elastic IPs**

* **Static IP**: Remains unchanged across instance stops and starts.
* **Reusability**: Can be reassigned to different instances as needed.

#### **Network Interfaces**

* Security groups, public IPs, and private IPs are associated with network interfaces.
* Multiple network interfaces can be attached to an instance for more complex networking setups.

#### **Managing Instances**

1. **Changing Instance Type**:
   * **Stopping the Instance**: Required before changing the instance type.
   * **Change Type**: Select a new instance type and apply changes.
   * **Starting the Instance**: The instance will start with the new instance type.
2. **Networking Options**:
   * **Attach/Detach Network Interfaces**: Manage additional network interfaces.
   * **Disassociate Elastic IP**: Disassociate the Elastic IP if no longer needed.
   * **Change Security Groups**: Modify or assign additional security groups.
3. **Instance Settings**:
   * **Modify IAM Role**: Assign or modify IAM roles for the instance.
   * **Create Image**: Create an AMI from your instance for future use.
   * **Launch More Like This**: Create similar instances with the same settings.
4. **Monitoring and Troubleshooting**:
   * **System Logs**: Access boot and system logs for troubleshooting.

#### **Cleaning Up Resources**

1. **Terminate the Instance**:
   * **Terminate**: Completely remove the instance and release resources.
   * **Release Elastic IP**: Release the Elastic IP back to the AWS pool to avoid charges.
2. **Check Dashboard**:
   * Ensure there are no running instances, volumes, or elastic IPs remaining to avoid unnecessary charges.

#### **Key Practices**

* **Frequent Cleanup**: Regularly terminate and release resources not in use.
* **Resource Tags**: Use tags to manage and identify resources efficiently.
* **Stay Within Free Tier**: Use T2 or T3 micro instances for experiments to avoid charges.
* **Security**: Keep security groups limited to necessary ports and IPs for better security.
* **Elastic IPs**: Use only when static IP is needed; otherwise, release to avoid charges.

By following these steps and best practices, you can effectively manage EC2 instances and associated resources, ensuring efficient and secure use of AWS infrastructure.

# Aws cli

IAM 🡪 user 🡪 add user 🡪 in security credentials 🡪Access key(cli)

## Some commands

Aws configure

Aws –version

Aws sts get-caller-identity

Aws ec2 describe-instances

Etc

<https://aws.amazon.com/cli/>

to check commands

and follow the command line interface pdf in devops learning

## link : <https://awscli.amazonaws.com/v2/documentation/api/latest/index.html>

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

# EBS -ELASTIC BLOCK STORAGE

* Block based storage

★ Runs ec2 OS, store data from db, file data, etc

★ Placed in specific AZ. Automatically replicated within the AZ to protect from failure.

★ Snapshot is backup of a volume

## EBS Types

● General Purpose (SSD) ○ Most Work Loads

● Provisioned IOPS ○ Large Databases

● Throughput Optimized HD ○ Big Data & Data Warehouses

● Cold HDD ○ File Servers

● Magnetic ○ Backups & Archives

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html>

or ebs volume types

## volumes

instances 🡪 storage 🡪 volume 🡪 click and rename the volume (both instances and volume should be in the same AZ (Availability zone)

exercise:

Size => Ram, CPU, Network etc

Min

4. Storage size

5 gigs for web server images

In the 30 gb you are in free trie

* Create a volume(make sure instances and volume should be same AZ) 🡪 attach the volume to the exiting once in the (actions 🡪 attach)

Partition:

Partition web images only :

Cd /var/www/html/images/

fdisk -l

fdisk /dev/xvdf

help -n

-a for partition

-1

+3g

Click enter

mkfs (double tab)

mkfs -ext4 /dev/xvdf **# formating**

mount & umount

take a back-up of images : mkdir /tmp/backup-imgs

mv /var/www/html/images/\* /tmp/backup-imgs/

mount /dev/xvdf /var/www/html/images/

**# mount filesystem path**

df -h (to check mounted or not

umount /var/www/html/images/

**# umount path**

**this is tempory mount**

**to do permanently**

* Vi /etc/fstab

Add the line

/dev/xvdf /var/www/html/images/ ext4 defaults 0 0

# **( 0 – no dump, 0 – file system sfc code)**

Mount -a

mv /tmp/backup-imgs/\* /var/www/html/images/

if still website is not working then use

vi /etc/selinux/cofig

* disable selinux

reboot the machine

short note

check

df -h (list to filesystem to mounted)

fdisk -l (list of disks)

mkfs -🡪 format for files(file type) 🡪 create a file system

“he mkfs command is used to create a filesystem on a device or partition in Linux. The command you provided seems to have a small typo. The correct command should be:

sh

Copy code

mkfs.ext4 /dev/xvdf

Here's a breakdown of the command:

* mkfs.ext4: This invokes the mkfs command with the ext4 filesystem type.
* /dev/xvdf: This specifies the device on which you want to create the filesystem.

### Steps to Format a Device with ext4 Filesystem

1. **Check the Device**: Ensure that /dev/xvdf is the correct device you want to format. You can list all block devices using:

lsblk

1. **Unmount the Device**: If the device is mounted, you need to unmount it before formatting:

sudo umount /dev/xvdf

1. **Format the Device**: Create an ext4 filesystem on the device:

sudo mkfs.ext4 /dev/xvdf

1. **Verify the Filesystem**: After formatting, you can verify the filesystem using lsblk or blkid:

sudo blkid /dev/xvdf

### Example Usage

sudo mkfs.ext4 /dev/xvdf

This command will create an ext4 filesystem on /dev/xvdf. Be very careful when running this command, as it will erase all data on the specified device.

”

add the volume🡪 make new partition(fdisk xvdf) 🡪 mkfs (files format 🡪 do mount 🡪 check the mount status

🡪it temporary

* to make permanent 🡪 vi /etc/fstab 🡪 add the line dev/xvdf1 /var/www/html ext4 defaults 00

# **filesystem** path to mount **file-format defaults 0 0**

# Snapshot Backup & Restore

* Unmount partition

• Detach volume

• Create new volume from snapshot

• Attach the volume created from snapshot

• Mount it back

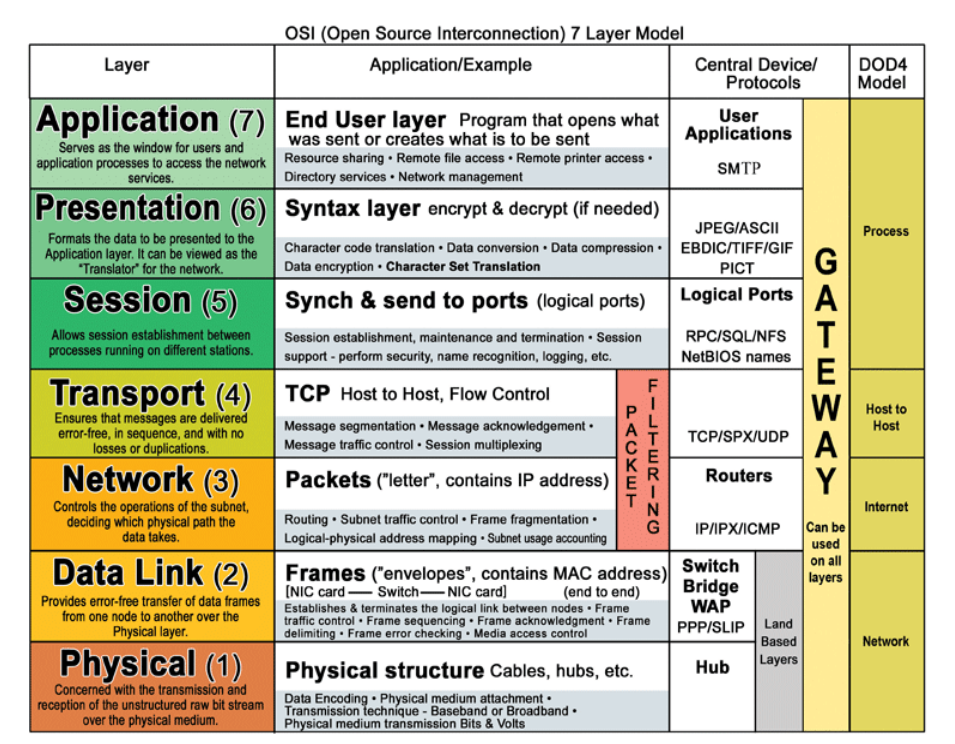
Snapshot :

Create a snapshot from exiting volume(like for backup&restore), if you remove the file or corrupt the file then we can replace the new volume by using snapshot

* Umount the exiting volume🡪 detach the volume🡪 create a new volume from snapshot(action)🡪attach the new volume 🡪 mount it again (mount -a)
* Snapshot can be used for exception , increase the volume , change the type of the volume , image , copy from one region to other region etc

If you need to delate the volume

* Detach the volume 🡪delate the volume



**Application(Layer 7)** This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services.  
  
**Presentation(Layer 6)** This layer provides independence from differences in data representation (e.g., encryption) by translating from application to network format, and vice versa. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the syntax layer.  
  
**Session(Layer 5)** This layer establishes, manages and terminates connections between applications. The session layer sets up, coordinates, and terminates conversations, exchanges, and dialogues between the applications at each end. It deals with session and connection coordination.  
  
**Transport(Layer 4)**This layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.  
  
**Network(Layer 3)** This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing.  
  
**Data Link(Layer 2)** At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronization. The data link layer is divided into two sublayers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sublayer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronization, flow control and error checking.  
  
**Physical(Layer 1)** This layer conveys the bit stream - electrical impulse, light or radio signal -- through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier, including defining cables, cards and physical aspects.

**OSI Layer Model for concentrators**

**Hubs/Repeaters** are found in the **Physical Layer**  
  
**Switches /Bridges/Wireless Access Point**are found in the **Data Link Layer**

**Multilayer Switch** are found in both the **Data Link Layer and Network Layer**  
  
**Routers** are found in the**Network Layer**  
  
**Gateway** are found in **All 7 of the OSI Layers**

**Brouter** are found in both the **Data Link** and **Network Layer**

|  |
| --- |
| [**OSI**](http://en.wikipedia.org/wiki/OSI_model)**OSI 7 Layer Model** |
| **7. Application Layer**- DHCP, DNS, FTP, HTTP, IMAP4, NNTP, POP3, SMTP, SNMP, SSH, TELNET and NTP[more)](http://en.wikipedia.org/wiki/Category:Application_layer_protocols) |
| **6.** **Presentation layer** – SSL, WEP, WPA, Kerberos, |
| **5. Session layer** – Logical Ports 21, 22, 23, 80 etc… |
| **4. Transport -**TCP, SPX and UDP[more)](http://en.wikipedia.org/wiki/Category:Transport_layer_protocols) |
| **3. Network -** IPv4, IPV6, IPX, OSPF, ICMP, IGMP and ARP[MP](http://en.wikipedia.org/wiki/Internet_Group_Management_Protocol) |
| **2. Data Link-**802.11 (WLAN), Wi-Fi, WiMAX, ATM, Ethernet, Token Ring, Frame Relay, PPTP, L2TP and ISDN[-](http://en.wikipedia.org/wiki/Wi-Fi)[ore)](http://en.wikipedia.org/wiki/Category:Link_protocols) |
| **1. Physical-**Hubs, Repeaters, Cables, Optical Fiber, SONET/SDN,Coaxial Cable, Twisted Pair Cable and Connectors [(more)](http://en.wikipedia.org/wiki/Category:Physical_layer_protocols) |

# Clusters

● Cluster of servers needs Endpoints

● Endpoints are usually of a Load Balancer

● Load Balancer balances incoming traffic to backend servers

## LOAD BALANCER Ports

* Frontend Port: Listens from the User Requests on this port AKA Listeners.
* e:g 80, 443, 25 etc
* Backend Ports: Services running on OS listening on this port
* e:g 80, 443, 8080 etc

ELASTIC LOAD BALANCER

● Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, in multiple Availability Zones.

● Elastic Load Balancing supports three types of load balancers:

* Application Load Balancer
* Network Load Balancer
* Classic Load Balancer
* Gateway Load Balance

# CLASSIC LOAD BALANCER

● The Classic Load Balancer that routes traffic based on either application or network level information

● The Classic Load Balancer is ideal for simple load balancing of traffic across multiple EC2 instance

# APPLICATION LOAD BALANCER

● Application Load Balancer that routes traffic based on advanced application level information that includes the content of the request

## NETWORK LOAD BALANCER

● A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model.

● It can handle millions of requests per second.

● Static IP

# Refer aws sides in devops learning

# ELB HANDS-ON

Create a instances with amazon – linux with provision tooplate template website 🡪 create a AMI(from the instances)[ instances🡪action🡪image and templates🡪 create image]

**“snapshot we can create a volume”**

**“AMI we can create an instance” AMI equlent shapshot + matadata**

* We can launch the instances from the AMI

OR

We can create a launch template from template, we can create a same type of instances very quickly

ELB: importance’s

Now we have two instances with blog website, they cannot be access separately by user,

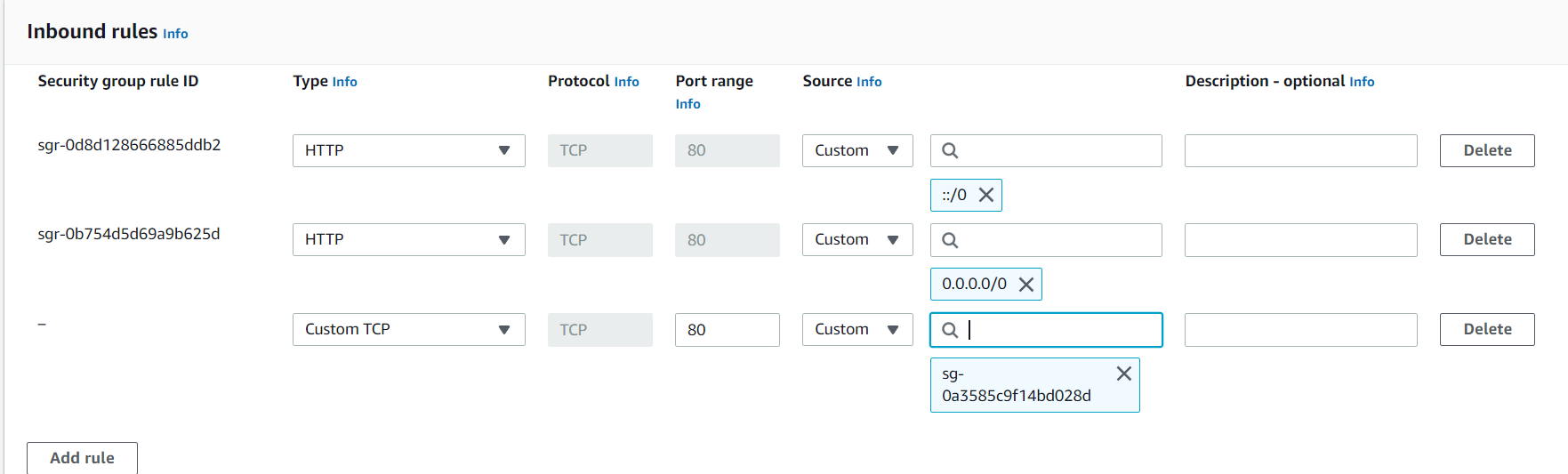
They can be accessing the website from the single endpoint that can be done by the load balances

Load balances: goto target group

[ **target group is basically a group of instances, with health . checks** (it checks the website health by loading again and again upto the value we given, if it all loaded good then it is health]

* Create a load balances (application load balances) with instances and create a security group and add the target group and check the option again launch it)

🡪Once load balance in active copy the DNS name and check the website is working or not. If not, check the security group (if you using different sg’s for the instance the pass the elb-sg in sg (sg- security group)



In target group(tg) we can register or deregister the tg 🡪(deregister is help to stop the traffic on maintance work)

# Cloud watch

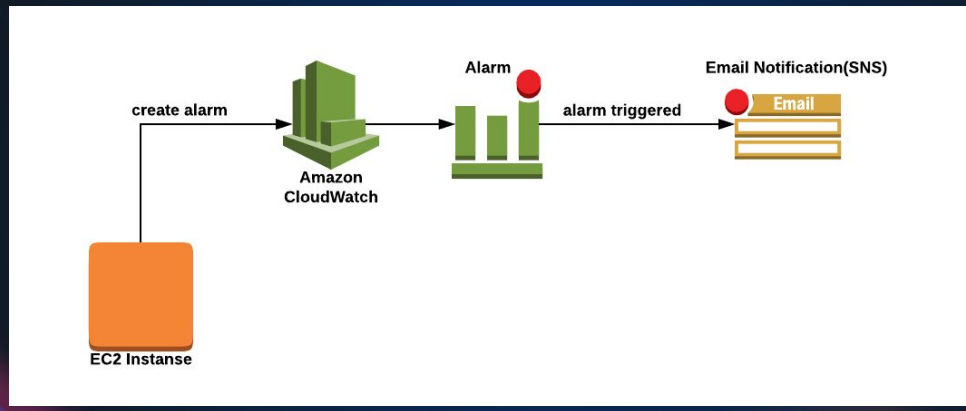
Cloud Watch - Monitor performance of AWS environment - standard infrastructure metrics.

● Metrics: AWS cloud watch allows. you to record metrics for services such as EBS, EC2, ELB, Route53 Health checks, RDS, Amazon S3, cloudfront etc etc…

● Events: AWS Events delivers a near real-time stream of system events that describe changes in Amazon Web Services (AWS) resources

● Logs: You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS CloudTrail, Route 53, and other sources

* Alarm monitors CloudWatch metrics for Instances.
* Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.



In instance🡪monitor(can see the metrics logs)

Normally/default cloud watch will monitor the metrics every five minutes (but can customize the time but some cost will applicable)

Instances🡪monitor🡪managing &detail monitor🡪can change the time here

* Launch the instance🡪 install the stress🡪 check the monitor graft

Yum install stress -y

Stress

Stress -c 4 (to stress the cpu 4 proess)

nohup stress -c 4 -t 300 &

Top (to check the all utilization)

Script

Script.sh

Sleep 60 & stress -c 4 -t 60 & Sleep 30 & stress -c 4 -t 60 &

Sleep 60 & stress -c 4 -t 500 & Sleep 60 & stress -c 4 -t 30 &

Nohup ./script.sh &

Cloudwatch🡪alarm🡪create alarm🡪select metrics🡪browse🡪ec2🡪 cpu-unitization🡪create topic🡪set alarm (name : warning | high cpu untilztion )

Some other alarm tools : promethus, nagios, icinga, zenos..etc

# EFS – ELASTIC FILE SYSTEM

* + 1. 🡪 create a security group 🡪 NFS (NETWORK FILE SYSTEM) -type
    2. 🡪 CREATE EFS
    3. 🡪 access point
    4. 🡪 install efs file
* Amazon linux

sudo yum install -y amazon-efs-utils

* Utuntu

$ sudo apt-get update

$ sudo apt-get -y install git binutils rustc cargo pkg-config libssl-dev

$ git clone https://github.com/aws/efs-utils

$ cd efs-utils

$ ./build-deb.sh

$ sudo apt-get -y install ./build/amazon-efs-utils\*deb

https://aws.amazon.com/efs

EFS – is a shared storage on aws (shared file system over the network)

* Create a instance 🡪 create a security group (efs-blog-sg) 🡪 create a EFS file system 🡪 create a access point
* To mount the EFS file system using this link :
* <https://docs.aws.amazon.com/efs/latest/ug/wt1-test.html>

to install amzon-efs-ulity is easily in amazon linux

sudo yum install -y amazon-efs-utils

**#please check the doc for more details**

# Using the EFS mount helper to automatically re-mount EFS file systems

file-system-id:/ efs-mount-point efs \_netdev,noresvport,tls,iam,accesspoint=access-point-id 0 0

fs-47a77ccb /var/www/html/img efs \_netdev,noresvport,tls,iam,accesspoint=fsan-03f6334520365d3 0 0

* Take a backup of images

Mkdir /tmp/backup-img

Mv /var/www/html/img/\* /tmp/backup-img

Vi /etc/fstab

fs-47a77ccb /var/www/html/img efs \_netdev,noresvport,tls,iam,accesspoint=fsan-03f6334520365d3 0 0

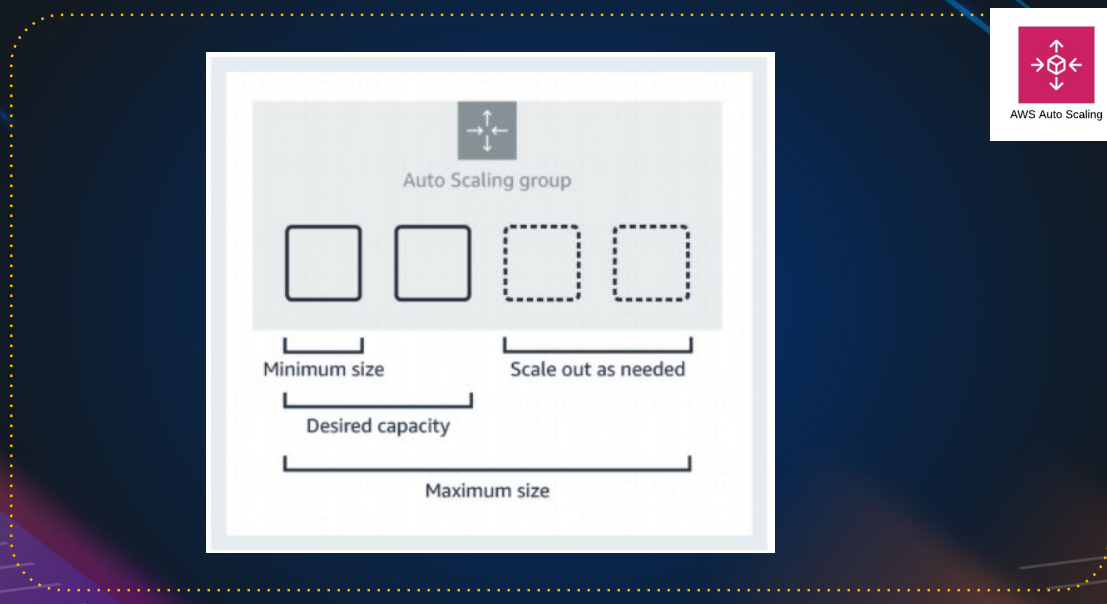
mount -fav

Mv /tmp/backup-img/\* /var/www/html/img/

Df -h

# AUTO SCALING

* Auto Scaling is a service that automatically monitors and adjusts compute resources to maintain performance for applications hosted in the AWS.
* Alarm monitors CloudWatch metrics for Instances
* A launch configuration/Template is an instance configuration template that an Auto Scaling group uses to launch EC2 instances
* Scaling policy is used to increase and decrease the number of running instances in the group dynamically to meet changing conditions.

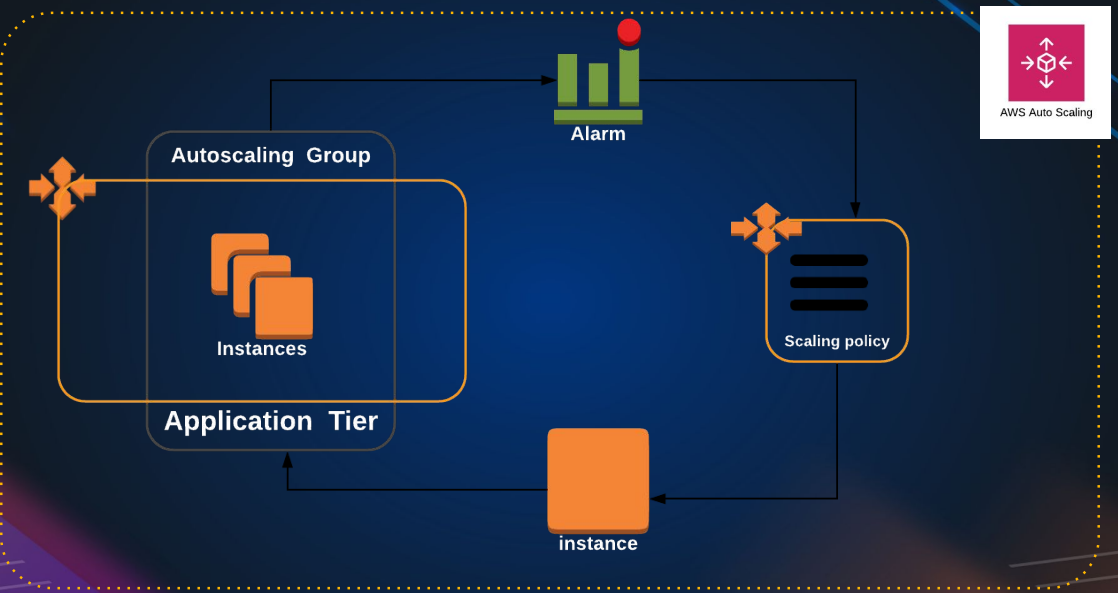


Example:

Minimum size: 1 instance #we can’t remove it

Desired capacity: 2 instances #it will create 2 instances on auto scaling

Maximum size: 4 instances #ASG can create max of 4 instances



Hands-on: create a lunch template🡪 set the target group(if it empty also no-issues ASG will assign automatically)🡪create a ASG🡪

Launch template: we can edit

Launch configurations: we can’t edit it

it

In Health check : ec2 health check by default it will do hardware health check and vm health check

ELB health check “ target group checks every 30 seconds, if instance is unhealth it will declare as unhealth, means basically it checking the port number or process whether it’s up or not.

If instance is health, target group will declare as health. If tg declare is not health then ASG remove the unhealth instance and create a new instance automatically.

Note:

If you update manually in instance, then after sometime instance get deleted and it will launch instance with OLD AMI, so always create a new launch template and edit the launch template in ASG and update they

ASG🡪 select the ASG🡪 details🡪edit the launch template and update the new launch template over there 🡪 start instances refresh in **instances refresh**

Important

ASG is dynamic instances, make sure they do not store any information.

The storage should be out of this easy instance, like on EFS OR NFS, somewhere out

### Ingredients Needed:

1. **Launch Template**: Ensure you have the launch template from the load balancer section. If not, recreate it.
2. **Load Balancer**: Necessary for distributing traffic across instances.

### Steps to Set Up Auto Scaling Group:

1. **Create a Target Group**:
   * Go to Target Groups and create a new one named **health-TG**.
   * Set it up as you did in the load balancer section.
   * Leave it empty; the Auto Scaling Group will add instances to it later.
2. **Create a Load Balancer**:
   * Create a new Application Load Balancer named **health-elb**.
   * Use the same security groups as before, except for the default security group.
   * Link it to the target group **health-TG**.
3. **Set Up Auto Scaling Group**:
   * Name it **healthy-ASG**.
   * Select the previously created launch template.
   * Choose the availability zones for the instances.
   * Attach the Auto Scaling Group to the existing load balancer and target group.
4. **Configure Health Checks**:
   * Enable ELB (Elastic Load Balancer) health checks for more accurate health status of instances.
   * EC2 health checks are basic and may not detect issues with the application running on the instances.
5. **Define Capacity Settings**:
   * Desired Capacity: 2 instances.
   * Minimum Capacity: 1 instance.
   * Maximum Capacity: 8 instances.
   * This ensures you always have at least one instance running and no more than eight.
6. **Set Scaling Policies**:
   * Use target tracking scaling policy based on CPU utilization (e.g., add instances if CPU > 50%, remove if CPU < 50%).
   * Adjust settings according to your needs.
7. **Review and Create**:
   * Add any necessary notifications.
   * Tag the instances (e.g., **Name: web-server**).
   * Review settings and create the Auto Scaling Group.

### Post-Creation:

* **Monitor Instances**: Check the instances created by the Auto Scaling Group in the Instances section.
* **Target Group Update**: Ensure the new instances are added to the target group and are healthy.

### Important Considerations:

* **Dynamic Instances**: Avoid manual changes to instances as they can be terminated and recreated based on the launch template.
* **Launch Template Changes**: For any updates, modify the launch template or create new versions, then refresh instances in the Auto Scaling Group.
* **Logging and Storage**: Ensure logs and dynamic data are stored outside the instances (e.g., using EFS or NFS).

### Cleanup:

* **Delete Auto Scaling Group**: This will automatically terminate the associated instances.
* **Delete Load Balancer**: Clean up resources to avoid unnecessary charges.

### Summary:

This session covered setting up an Auto Scaling Group with a load balancer and target group, defining health checks, and configuring scaling policies. It emphasized the importance of using the launch template for changes and storing dynamic data outside the instances. Finally, it included steps for cleaning up the resources after the setup.

# S3- Simple Storage Service

(it is like a google drive & drop box but much more feature)

Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

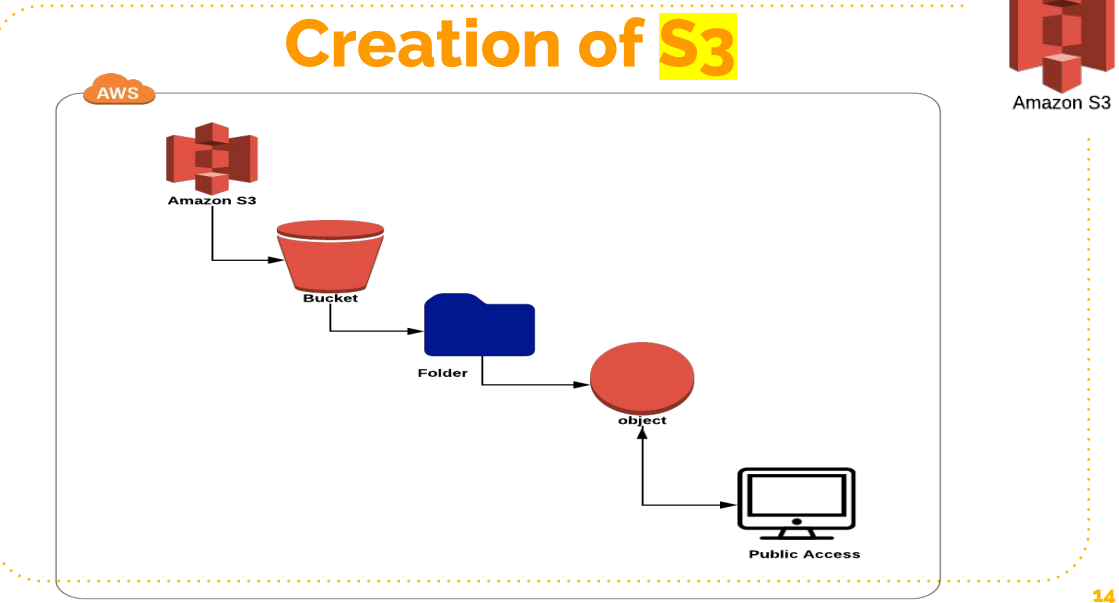
## S3 Basics

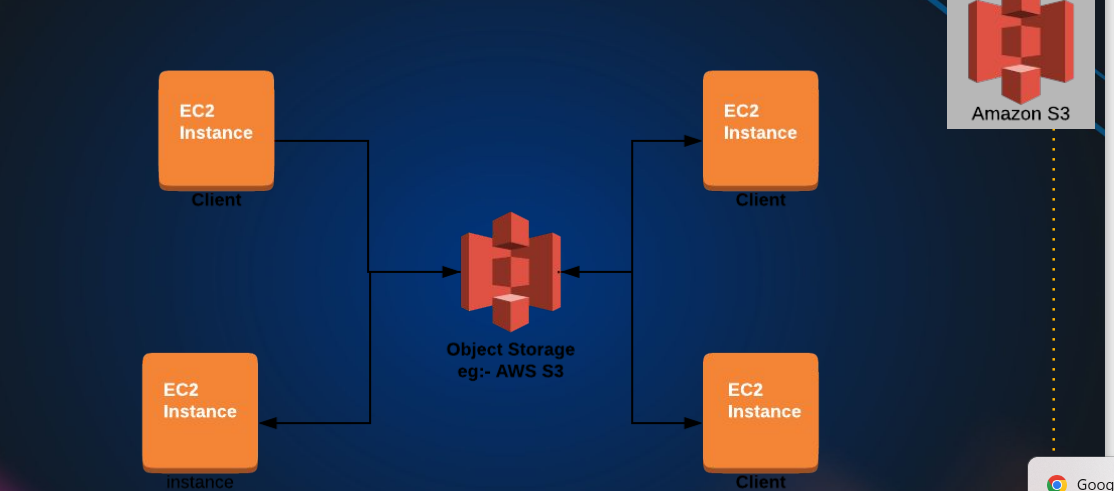
* It is Object-Based Storage
* Data is replicated across multiple facilities
* Unlimited Storage
* Amazon S3 stores data as objects within buckets
* Bucket name has to be unique

#Because you will be getting end point to access it and then end point will have the name of the bucket, so it has to be unique in the internet

Bucket: A bucket is a logical unit of storage in Amazon Web Services (AWS).

Object: Object Storage is a computer data storage architecture that manages data as Objects.





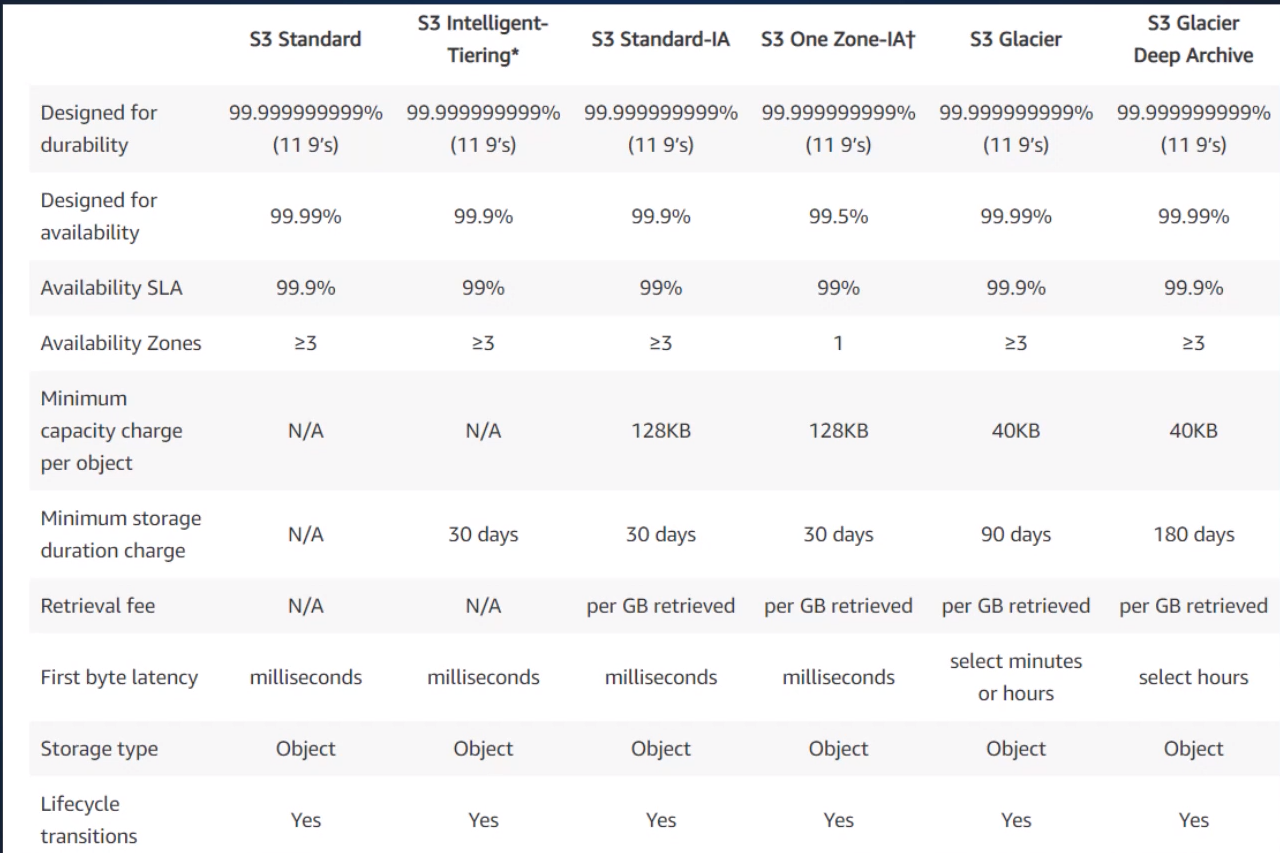
It seems like EFS through this architectural design, but in EFS we mount the filesystem at the operating system level.so we get a folder at the operating system layer, level, where we store the data

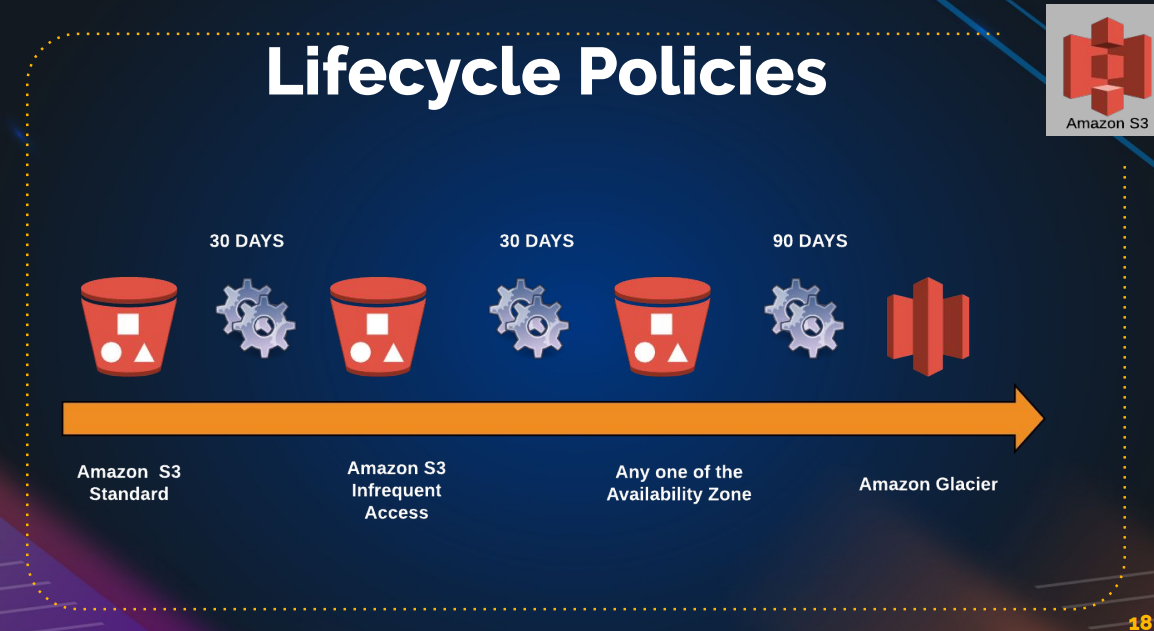
But S3 we are going programmatically access through our application.

We can also mount s3 bucket to a folder but that comes though a different driver S3FS

# S3 Storage Classes

1. S3 Standard: general-purpose storage of frequently accessed data. Fast access & object replication in multi-AZ.
2. S3 IA– Infrequent Access: Long-lived, but less frequently accessed data. Slow access, object replication in multi-AZ.
3. S3 One Zone-IA is for data that is accessed less frequently, but requires rapid access when needed. Slow access, no object replication.
4. S3 Intelligent Tiering Automatically moves data to most cost effective tier.
5. S3 Glacier: Low Cost Storage class for data Archiving.
6. S3 Glacier Deep Archive Lowest cost storage, retrieval time of 12 Hrs.





**This to save the cost** ,

we are applying the lifecycle policy, usually on log’s archive and you can also set expiry, also you can set after sometime(1year or two years) delete this data.

## Amazon S3 Session Overview

### Introduction to Amazon S3

* **What is Amazon S3?**
  + Amazon Simple Storage Service (S3) is an internet-accessible storage service.
  + Similar to Google Drive or Dropbox but with more features.
  + Allows storing and accessing any amount of data from anywhere.
  + Popular and one of the oldest AWS services.

### Basic Concepts of S3

* **Object Storage:**
  + S3 is an object storage service.
  + Data is stored as objects within buckets (similar to folders).
  + Objects can be documents, pictures, videos, etc.
  + Buckets must have unique names globally.
* **Data Replication and Durability:**
  + Data in S3 buckets is replicated across multiple facilities for durability.
  + S3 offers virtually unlimited storage capacity.

### Bucket and Object Management

* **Buckets:**
  + Logical containers for storing objects.
  + Can set permissions and properties at the bucket level.
* **Objects:**
  + Individual data items stored in buckets.
  + Can set permissions and properties at the object level.

### Access and Permissions

* **Access Control:**
  + Access can be managed through IAM policies, bucket policies, and ACLs.
  + By default, all objects are private.
  + Public access must be explicitly enabled.

### Common Use Cases

* **Integration with EC2:**
  + EC2 instances can store file-based data in S3 rather than locally.
  + Useful for applications requiring storage of web service data, etc.

### Storage Classes

1. **S3 Standard:**
   * High availability and durability.
   * Fast data access and replication across multiple availability zones.
2. **S3 Infrequent Access (IA):**
   * Lower cost for less frequently accessed data.
   * Slower access compared to Standard but still replicated across multiple availability zones.
3. **S3 One Zone IA:**
   * Lower cost for infrequently accessed data that doesn't require high durability.
   * Data stored in a single availability zone.
4. **S3 Intelligent Tiering:**
   * Automatically moves data to the most cost-effective access tier.
5. **S3 Glacier:**
   * Low-cost storage for data archiving.
   * Suitable for data accessed infrequently (e.g., once a year).
6. **S3 Glacier Deep Archive:**
   * Lowest-cost storage for long-term data archiving.
   * Data retrieval can take up to 12 hours.

### Lifecycle Policies

* **Lifecycle Policies:**
  + Automate transitioning of data between different storage classes.
  + Example: Move data to IA after 30 days, Glacier after 90 days, and delete after a year.

### S3 Charges

1. **Storage Charges:**
   * Based on the amount of data stored.
2. **Request Charges:**
   * Based on the number of requests made to S3.
3. **Data Transfer Charges:**
   * Charges for data transferred out of S3.
4. **Replication Charges:**
   * Additional charges for cross-region replication.

### Creating and Managing Buckets and Objects

#### Steps to Create an S3 Bucket and Upload Files:

1. **Create a Bucket:**
   * Open the AWS console and navigate to S3.
   * Click "Create bucket" and provide a unique name.
2. **Configure Bucket Settings:**
   * Enable or disable Object Ownership and ACLs.
   * Block public access by default for security.
   * Enable versioning and encryption if needed.
3. **Upload Objects:**
   * Select the bucket and click "Upload".
   * Choose files and configure storage class, permissions, and encryption.
4. **Manage Object Access:**
   * Objects are private by default.
   * To make objects public, adjust bucket permissions and enable public access through ACLs.

#### Making Objects Public:

1. **Enable ACLs:**
   * Go to bucket permissions and enable ACLs.
2. **Unblock Public Access:**
   * Edit the block public access settings and uncheck the blocking options.
3. **Set Object Permissions:**
   * Select objects and use actions to make them public.

### Practical Exercise

* **Create an S3 Bucket:**
  + Follow steps to create and configure the bucket.
* **Upload and Manage Objects:**
  + Upload files, check permissions, and try accessing them via URL.
  + Practice making objects public and understanding access controls.

### Next Steps

* **Hosting a Website on S3:**
  + In the next session, learn how to host a static website using S3.

By following this structure, you can efficiently navigate through the core aspects of Amazon S3, ensuring a clear understanding of its functionalities and practical applications.

# S3 Charges

* Storage
* Requests
* Tiers
* Data Transfer
* Region Replication

Some of the feature’s like **bucket versioning**: if you enable this data is overwrite or deleted , then is present there only then we can revive deleted once. But if we delete more then more storage is required

S3 object 🡪 delete the file(index.html)🡪 it will not delete(because versioning is enble) 🡪if you delete permanent the click the enable-icon on the left and select deleted files do permanent delete to remove the files

Use case: upload the date and access to from anywhere by making public. We can host to public

Create a S3 bucket 🡪 upload the data(code, files and artifacts etc) 🡪unblock the public serves 🡪 goto object🡪 make it public (to files/object) 🡪 use the object URL to access the object/name from anywhere

Exercise:

Launch the website from tooplate.com using s3 bucket

Move/upload all the files to S3 bucket 🡪 select all files and make as public 🡪 check the block permission if in block state 🡪 properties under bottom enable the static website hosting 🡪 get the end point URL and check the website is working or not.

# Lifecycle roles

[Amazon S3](https://s3.console.aws.amazon.com/s3/get-started?region=us-east-1) 🡪[Buckets](https://s3.console.aws.amazon.com/s3/buckets?region=us-east-1) 🡪[blog-storage-s3](https://s3.console.aws.amazon.com/s3/buckets/blog-storage-s3) 🡪[Lifecycle configuration](https://s3.console.aws.amazon.com/s3/management/blog-storage-s3/lifecycle?region=us-east-1) 🡪 **Create lifecycle rule**

**Set the roles for storage type to cost “refer the lifecycle policies**

## Replication

[Amazon S3](https://s3.console.aws.amazon.com/s3/get-started?region=us-east-1) 🡪 [Buckets](https://s3.console.aws.amazon.com/s3/buckets?region=us-east-1) 🡪[blog-storage-s3](https://s3.console.aws.amazon.com/s3/buckets/blog-storage-s3) 🡪[Replication rules](https://s3.console.aws.amazon.com/s3/management/blog-storage-s3/replication?region=us-east-1) 🡪 C**reate replication rule**

**“” UPTO 5GB S3 under free tair “”**

### Step-by-Step Guide to Hosting a Static Website on AWS S3

1. **Download and Extract Website Template**
   * Visit [Tooplate](https://www.tooplate.com/) or any other source for free website templates.
   * Download a template and extract the contents of the downloaded zip file to a folder.
2. **Create S3 Buckets**
   * Log in to the AWS Management Console.
   * Go to the S3 service.
   * **Create the main bucket for your website:**
     + Click on "Create bucket".
     + Name the bucket uniquely (e.g., **barista123**).
     + Enable versioning (can be enabled later if not done now).
     + Complete the creation process.
   * **Create a bucket for access logs:**
     + Click on "Create bucket".
     + Name it uniquely (e.g., **barista123-accesslogs**).
     + Ensure the name is in all lowercase.
     + Complete the creation process.
3. **Upload Website Files to S3 Bucket**
   * Open the main bucket (e.g., **barista123**).
   * Click on "Upload", then "Add files".
   * Drag and drop all the files from the extracted folder or use the file picker to upload.
   * Click "Upload".
4. **Make the Bucket Public**
   * Go to the bucket's permissions.
   * Edit the "Block public access" settings and uncheck the options to allow public access.
   * Save the changes and confirm.
   * Enable ACLs (Access Control Lists) if not already enabled.
5. **Make All Objects Public**
   * Select all objects in the bucket.
   * Choose "Actions" and then "Make public using ACL".
   * Confirm the warning and proceed.
6. **Enable Static Website Hosting**
   * Go to the bucket's properties.
   * Scroll down to "Static website hosting".
   * Edit the settings and enable static website hosting.
   * Set the index document (e.g., **index.html**) and error document (e.g., **error.html**).
   * Save the changes.
   * Note the website endpoint URL provided.
7. **Set Up Access Logs**
   * In the main bucket's properties, locate "Server access logging".
   * Edit the settings and enable access logs.
   * Specify the access log bucket (e.g., **barista123-accesslogs**).
   * Save the changes.
8. **Verify Website Access**
   * Open a web browser.
   * Enter the website endpoint URL provided in the static website hosting section.
   * Verify that the website loads correctly.
9. **Enable Versioning and Manage Versions**
   * In the main bucket’s properties, enable versioning if not already done.
   * To view versions, use the "Show versions" toggle in the S3 console.
   * Demonstrate adding a new version of a file by uploading a new **index.html**.
   * Make the new object public if necessary.
   * Use the versions view to delete and restore previous versions.
10. **Access Logs Verification**
    * The logs for access will appear in the access logs bucket.
    * These logs will provide details about who accessed the website, the browser used, regions, etc.
    * Check after some time as it might take a while for the logs to appear.

### Summary

Hosting a static website on AWS S3 is straightforward and eliminates the need for traditional web servers. By using S3's static website hosting capabilities, you can easily deploy and scale your website. Additionally, versioning helps manage changes, and access logs provide valuable insights into website traffic.

### Additional Notes

* **CNAME Configuration**: Later, you can configure a custom domain and set up CNAME records to point to the S3 website endpoint.
* **Cost Considerations**: Be mindful of the storage costs, especially with versioning enabled, as each version of a file is stored separately.

This guide captures the essence of the lecture and provides a clear, practical approach to hosting a static website on AWS S3.

### Advanced S3 Bucket Features: Lifecycle Rules and Disaster Recovery

In this lecture, we will explore advanced features of Amazon S3 buckets that help manage costs, comply with disaster recovery protocols, and optimize storage efficiency. These features are essential for DevOps engineers and architects to understand and implement effectively.

#### Lifecycle Rules for Cost Management

\*\*Lifecycle Rules\*\* allow you to define the transition of objects between different storage classes based on their age, helping to optimize storage costs.

1. \*\*Access the Management Tab\*\*

- Navigate to the S3 bucket.

- Click on the "Management" tab.

2. \*\*Create a Lifecycle Rule\*\*

- Click "Create lifecycle rule".

- Name the rule (e.g., `CostEffectiveTransitions`).

3. \*\*Define the Scope\*\*

- Apply the rule to all objects or limit it by specifying a prefix (e.g., files that start with "image").

4. \*\*Set Transition Actions\*\*

- \*\*Current Version Objects:\*\*

- Transition from `Standard` to `Standard-IA` after 30 days.

- Transition from `Standard-IA` to `One Zone-IA` after 60 days.

- Transition to `Glacier Flexible Retrieval` after 90 days.

- Transition to `Glacier Deep Archive` after 180 days.

- \*\*Non-Current Version Objects:\*\*

- Transition from `Standard` to `Standard-IA` after 35 days.

- Transition from `Standard-IA` to `One Zone-IA` after 65 days.

- Transition to `Glacier Flexible Retrieval` after 95 days.

- Transition to `Glacier Deep Archive` after 185 days.

5. \*\*Expire Objects\*\*

- Set expiration for objects (e.g., after 450 days).

6. \*\*Delete Non-Current Versions\*\*

- Set a rule to permanently delete non-current versions (e.g., after 455 days).

7. \*\*Handle Incomplete Uploads\*\*

- Delete incomplete uploads after a specified period (e.g., 15 days).

8. \*\*Create the Rule\*\*

- Click "Create rule" to apply it.

#### Disaster Recovery with Cross-Region Replication

To ensure data durability and compliance with disaster recovery protocols, configure cross-region replication.

1. \*\*Create a Destination Bucket\*\*

- In a different region (e.g., Oregon), create a new bucket (e.g., `barista908dr`).

2. \*\*Set Up Replication\*\*

- Go back to the source bucket.

- Click on "Management" and then "Create replication rule".

- Name the rule (e.g., `DisasterRecoveryBarista908`).

3. \*\*Define the Replication Scope\*\*

- Apply the rule to all objects or filter based on specific criteria.

4. \*\*Choose Destination Bucket\*\*

- Select the destination bucket created in the other region.

5. \*\*Enable Versioning\*\*

- Ensure versioning is enabled in the destination bucket.

6. \*\*Configure IAM Role\*\*

- Create a new IAM role to grant necessary permissions for replication.

7. \*\*Set Destination Storage Class\*\*

- Choose an appropriate storage class for the destination bucket (e.g., `Standard-IA` or `One Zone-IA`).

8. \*\*Additional Options\*\*

- Configure replication metrics, replication time control, and delete markers as needed.

9. \*\*Save the Rule\*\*

- Submit the rule to start replication for new objects.

#### Clean Up

1. \*\*Delete Lifecycle and Replication Rules\*\*

- Remove lifecycle and replication rules to avoid unintended costs.

2. \*\*Delete Buckets\*\*

- Empty and delete the created buckets to clean up resources.

### Summary

This lecture covered essential S3 features for cost management and disaster recovery. By using lifecycle rules, you can transition objects to cheaper storage classes based on their age. Cross-region replication ensures your data is resilient to regional failures, adhering to disaster recovery best practices. These advanced configurations are crucial for efficient and reliable data storage management in any AWS-based infrastructure.

### Next Steps

In the next lecture, we will delve into more advanced S3 features and AWS services, continuing to build your expertise as a DevOps engineer or architect. Be sure to clean up the resources used in this lecture to avoid unnecessary charges.

# Relational Database (RDB)

## DB Administration

* Installs
* Patching
* Monitoring
* Performance Tuning
* Backups
* Scaling
* Security
* Hardware upgrades
* Storage Management

## RDS

❖ Amazon Relational Database Service is a distributed relational database service.

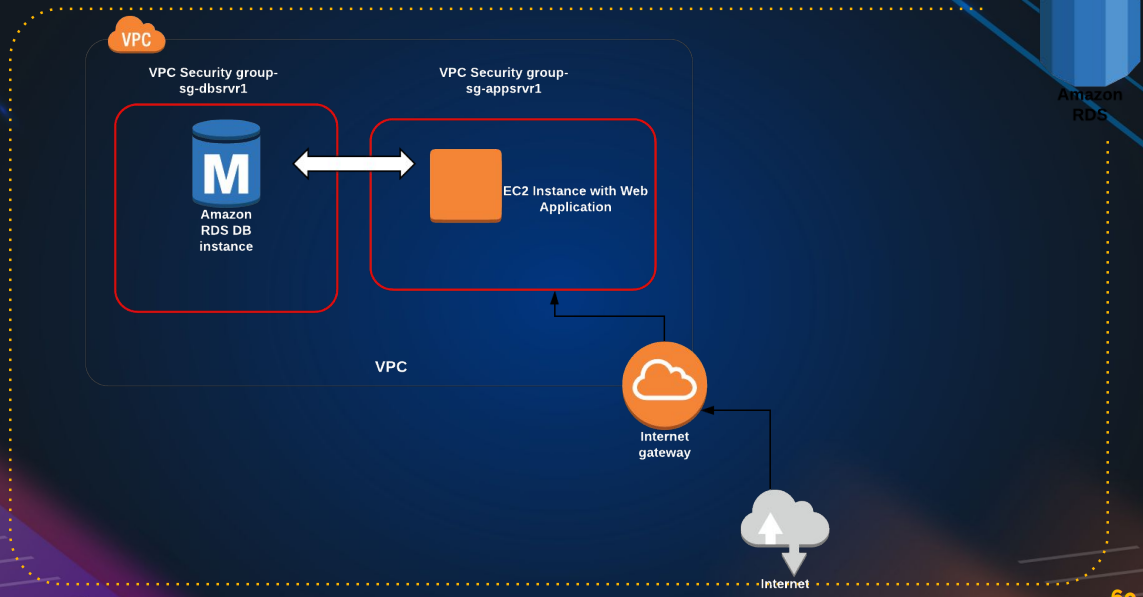
❖ High Availability Multi-AZ Deployments.

❖ Effortless Scaling.

❖ Read Replicas for performance

“RDS support various database engines like MySQL, PostgreSQL, Oracle etc. if you using MySQL or PostgreSQL then amazon suggest to go with “amazon Aurora” because

* amazon Aurora is 5 time faster than MySQL.
* amazon Aurora is 3 time faster than PostgreSQL.
* Cheater than compare to MySQL, PostgreSQL.
* It is serverless and support 64 TB.



RDS 🡪create a db (eg: mysql)🡪create instance EC2🡪login to instances 🡪make sure SG have proper roles🡪 connect the db

Sudo yum install mariadb-server -y

If it not found the search for available server

sudo yum/dnf search mariadb

and install it

to connect the server: mysql -h YourDatabaseEndpoint -u YourUsername -p current password

“””need to check for password reset”””

RDS cost’s more accounting to usage. Use properly

# Welcome to the RDS Session

Welcome to the RDS session, where we'll explore Amazon's Relational Database Service (RDS) and how it can significantly simplify database administration.

## The Challenges of Database Administration

Managing a database involves several complex tasks:

1. **Setting up the database:** Initial configuration and deployment.
2. **Regular patching:** Keeping the database software up-to-date.
3. **Continuous monitoring:** Ensuring performance and health.
4. **Performance tuning:** Optimizing database operations.
5. **Regular backups:** Ensuring data safety and availability for rollback in case of failures.
6. **Scaling:** Adjusting resources as business grows, which is challenging for SQL databases.
7. **Security:** Protecting data from unauthorized access.
8. **Hardware upgrades:** Managing physical resources.
9. **Storage management:** Scaling storage as needed.

## Introduction to AWS RDS

AWS RDS automates many of these tasks:

* **High Availability:** With Multi-AZ deployments, you can have a primary database and a standby replica in different availability zones. If the primary fails, the standby takes over.
* **Read Replicas:** For performance improvement, read replicas can handle read requests, reducing the load on the primary database.
* **Scalability:** Easy to scale databases and storage.
* **Security:** Integrated security features to protect your data.

## Setting Up an RDS Instance

### Step-by-Step Guide:

1. **Navigate to RDS:**
   * Go to the RDS section in your AWS Management Console.
2. **Choose Database Engine:**
   * Supports MySQL, PostgreSQL, Oracle, MariaDB, SQL Server, and Amazon Aurora.
   * For this demo, we'll use MySQL.
3. **Select the Database Configuration:**
   * Choose between Production, Dev/Test, and Free Tier templates.
   * For this demo, we’ll use the Dev/Test option.
4. **Configure Instance Details:**
   * Instance name: **vprofile-mysql-rds**
   * Login credentials: **admin** (password auto-generated for this demo).
   * Instance size: **t3.micro** for free tier eligibility.
   * Storage: General Purpose SSD, 20GB with auto-scaling enabled.
5. **Set Network and Security:**
   * VPC: Default
   * Subnet: Default
   * Public accessibility: No (private access within the VPC)
   * Security group: Create a new one named **vprofile-rds-sg**
6. **Database Options:**
   * Database name: **accounts**
   * Backup retention: 7 days
   * Enable encryption and enhanced monitoring (default settings)
7. **Launch the Database:**
   * Review settings and click **Create Database**.

### Accessing the RDS Instance

1. **Create an EC2 Instance in the Same VPC:**
   * Launch an Ubuntu 18.04 instance named **mysql-client**.
   * Ensure it’s in the same VPC as your RDS instance.
2. **Install MySQL Client:**
   * SSH into your EC2 instance.
   * Run **sudo apt update && sudo apt install mysql-client -y**.
3. **Connect to RDS:**
   * Use the RDS endpoint, username, and password to connect.
   * Command: **mysql -h <RDS\_ENDPOINT> -u admin -p**

### Troubleshooting Connection Issues

* **Security Group Settings:**
  + Ensure the RDS security group allows inbound connections on port 3306 from your EC2 instance's security group.
* **Network Issues:**
  + Verify network connectivity using tools like **telnet**.

## RDS Features and Management

* **Automatic Backups and Snapshots:** RDS automatically backs up your database and creates snapshots, which can be restored if needed.
* **Read Replicas:** Create replicas for load balancing read traffic.
* **Parameter Groups:** Customize database parameters to optimize performance.

## Conclusion

AWS RDS automates many aspects of database management, reducing the need for a dedicated database administrator. It offers high availability, easy scalability, enhanced security, and regular backups, making it an ideal choice for businesses of all sizes.

**Remember:** Always clean up your resources after use to avoid unnecessary charges. Delete the RDS instance and any associated EC2 instances once you are done.

Thank you for joining this session. See you in the next one!

# Aws lift & shift project

🡪Refer the pdf in the devops learning

Vproject vagrant project shift to aws server

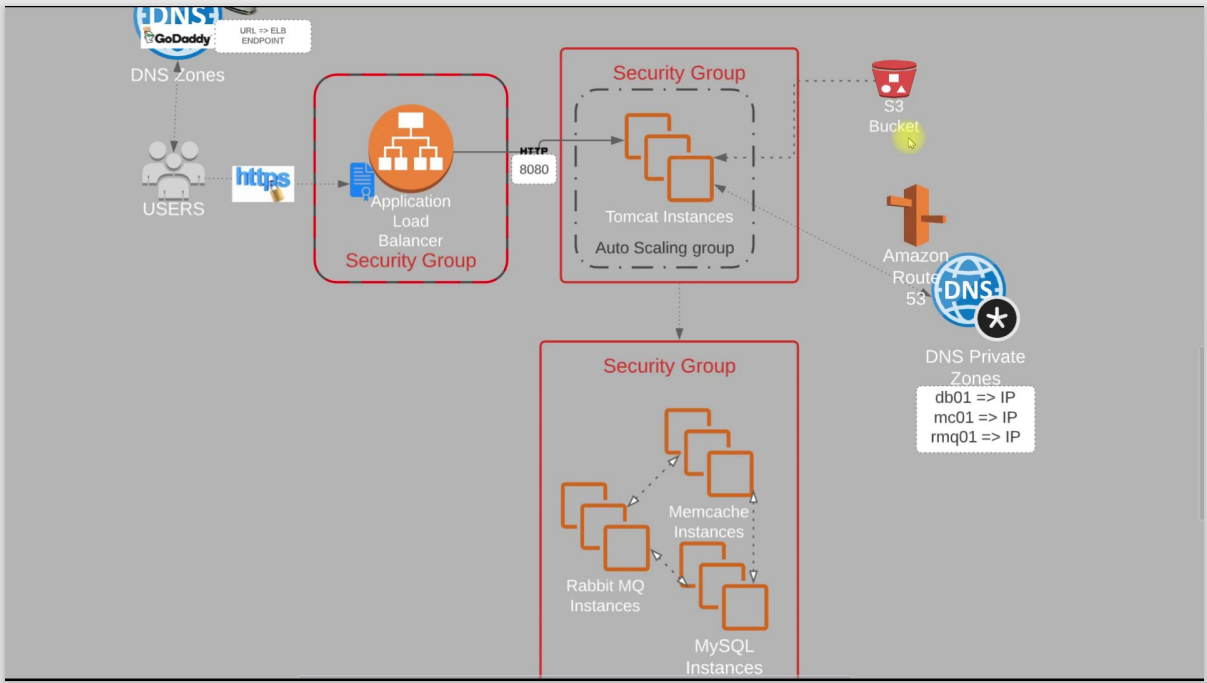
Vagrant:

User 🡪Nginx(load balancer🡪single end point) 🡪 Apache tomcat server(httpd servers)🡪 rabbit mq(message service)🡪Memcached(temporary che service) 🡪 database (mysql)

Vs

Aws

User 🡪 load balancer 🡪 instance (tomcat, RabbitMQ, Memcached,mysql instances)🡪key pairs 🡪 security group🡪s3/EFS🡪Rout 53



Key pair : bproject-KP.pem

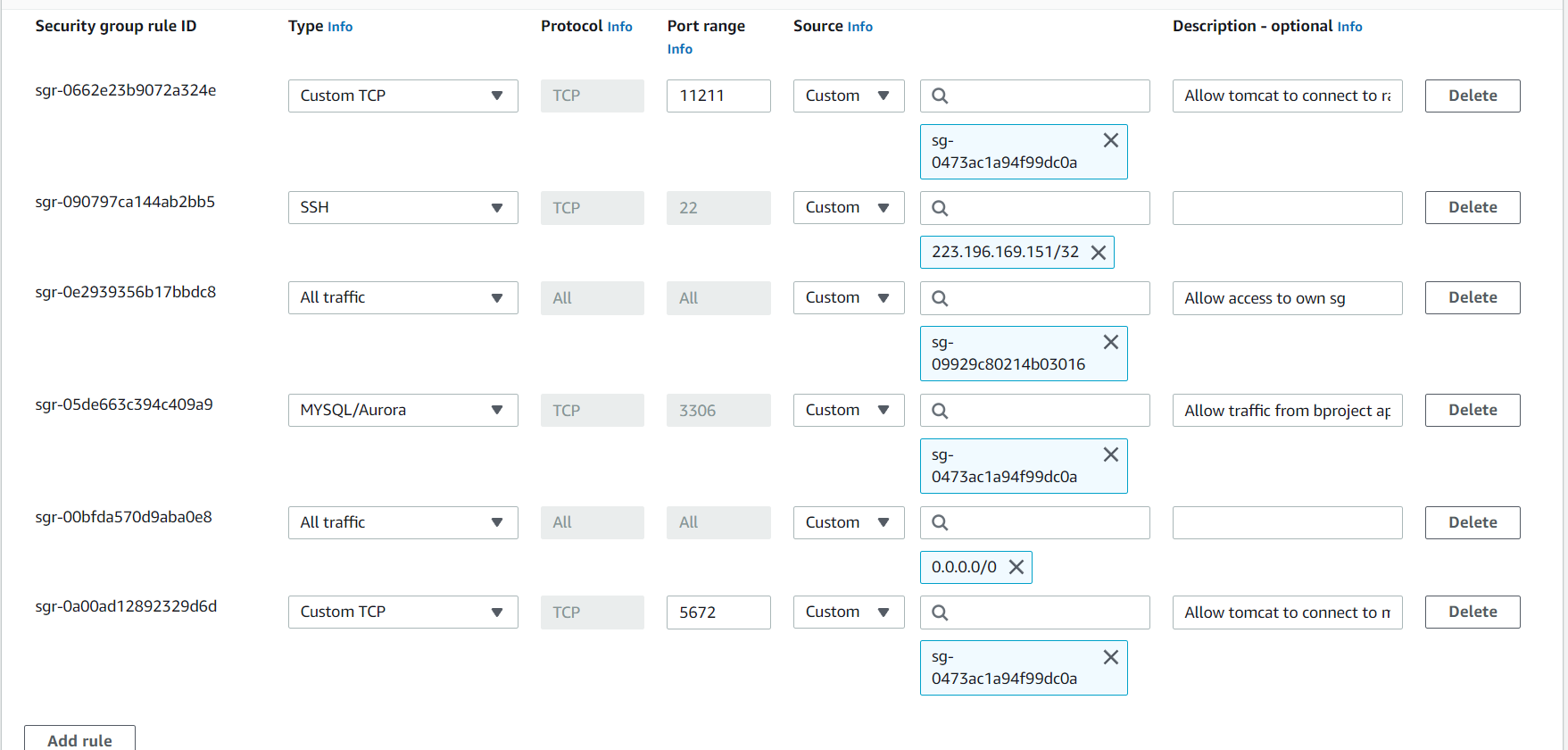
Security group :

ELB: inbound roles http & https to 8080 port for 0.0.0.0/0 & ::/0

APP: inbound roles 8080 port / allow the elb-sg to app & port 22

BACKEND: here we are using three services : mysql, Memcached and rabbitmq

Allow every port app-sg 3306 for sql , 5672 for rabbitmq and memecached 11211 and allow same sg for internal data transfer.



Then create ec2 for all 4 services mysql,rabbitmq,memecacahed & tomcat7(ubuntu)

By provision bash script in git repo aws lift and shift

“”default tomcat application can be they in this provision is /var/lib/tomcat9” we find webfiles here and inside ROOT(default tomcat page) will be there

<https://github.com/hkhcoder/vprofile-project/tree/main>

* Check the service active are not
* We can check the retrieve data for ec2 by curl <http://169.254.169.254/latest/user-data>
* We can check status by systemctl or ss -tunlp | grep “given the service port number”

Normally we use /etc/hosts to ip addr in vagrant but in ROUTE 53 service in aws

https://github.com/devopshydclub

# ROUTE 53

Domain name service (DNS)

# VPC (virtual private cloud)

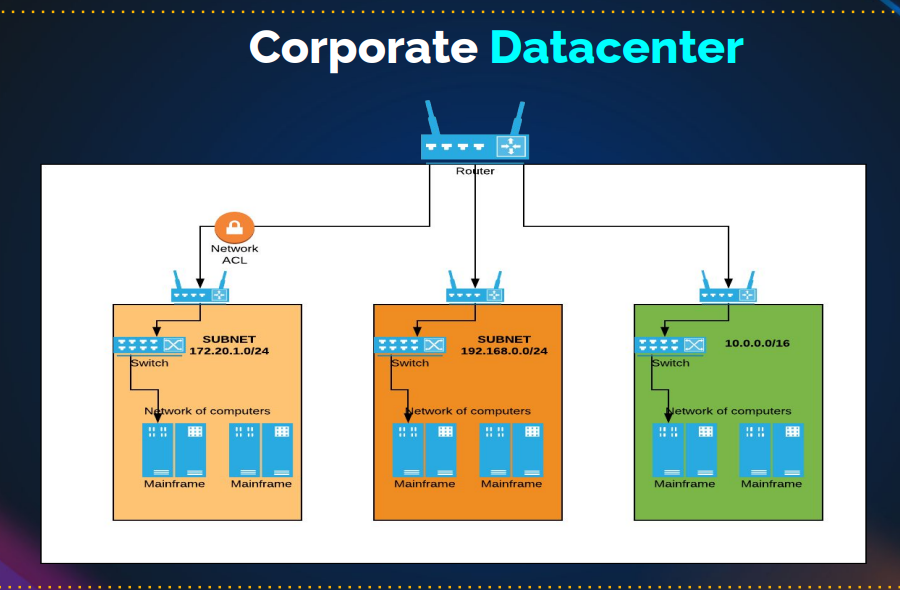
At the early stage of AWS(cloud computing started), aws come up with SQS,S3, classic EC2 instance.

"We initially had data centers, and we still maintain them. In these data center networks, there are a multitude of components, including switches, routers, firewalls, and numerous interconnected networks. Within these networks, smaller subnets play a crucial role. A subnet serves as a dedicated portion of the network, often allocated for specific projects or front-end and back-end services associated with those projects. Security levels vary, with some subnets being highly restricted while others remain open to the internet.

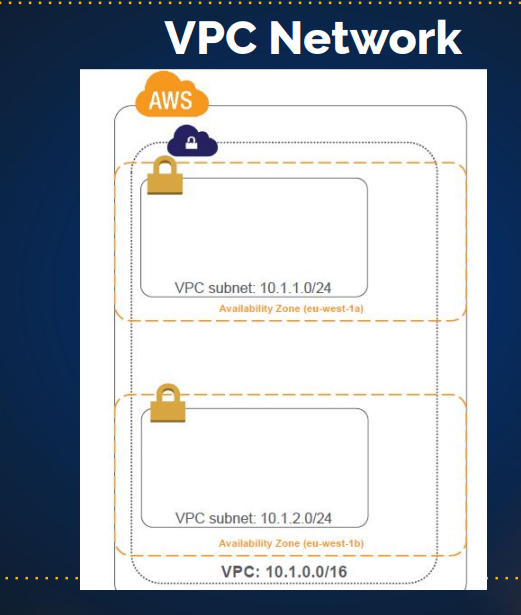
Network access control lists (ACLs) come into play to determine what traffic can enter or exit, while IP addresses are meticulously planned as part of the network scheme. All of this falls under the jurisdiction of the networking team, responsible for designing and managing the networks in corporate data centers, whether they are vast or more compact in scale.

Then, a significant development occurred with the introduction of AWS's EC2 service, which revolutionized our capabilities. It allowed us to launch virtual machines within AWS or Amazon Data Centers for computing purposes. However, the demand for greater networking control grew. People wanted to customize their networking schemes and regulate inbound and outbound traffic according to their specific requirements. Unfortunately, these features were not initially available in the early versions of EC2.

In response to these demands, AWS introduced the VPC (Virtual Private Cloud), which provided the desired networking flexibility and control."



VPC:



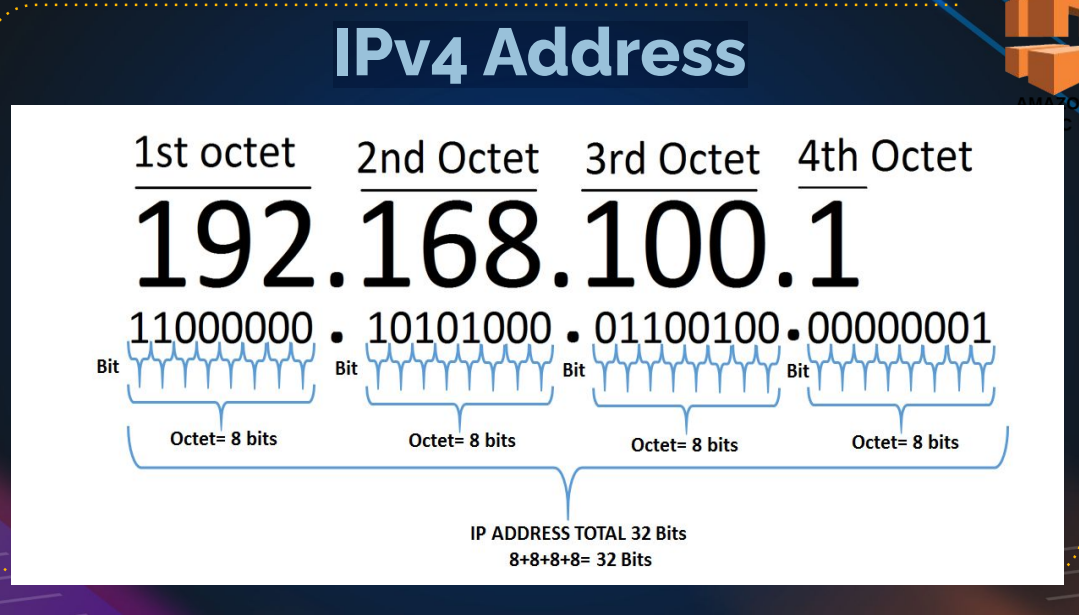
## Virtual Private Cloud (VPC)

1. VPC is a logical data center within an AWS Region.

Normally you have the default vpc but when created the VPC you have more control in it. You create instance, RDS etc inside VPC.

1. virtual private cloud is an on-demand configurable pool of shared computing resources allocated within a public cloud environment.
2. Control over network environment, select IP address range, subnets and configure route tables and gateways.

## IPv4 Address



### IPv4 Range

1. 0.0.0.0 – 255.255.255.255

🡪 0000000.00000000.00000000.00000000 (0.0.0.0)

🡪11111111.11111111.11111111.11111111 (255.255.255.255)

### public and Private IP Division

* Public IP => Internet

• E:g 54.86.23.90

• Private IP => For local network design

• E:g 192.168.1.10

### Private IP Ranges

1. Class A 10.0.0.0 - 10.255.255.255
2. Class B 172.16.0.0 - 172.31.255.255
3. Class C 192.168.0.0 - 192.168.255.255

### Subnet Masks

● 255.0.0.0

● 255.255.0.0

● 255.255.255.0

A subnet mask is a 32-bit address that segments an IP address into network and host portions. It is used in networking to determine which part of an IP address identifies the network and which part identifies the specific host within that network. Subnet masks consist of a series of contiguous 1s followed by a series of contiguous 0s, with the 1s indicating the network portion and the 0s indicating the host portion.

For example, in the common IPv4 notation, a subnet mask might look like this:

* 255.255.255.0

In binary, this subnet mask would be represented as:

* 11111111.11111111.11111111.00000000

In this example, the first 24 bits (the 1s) are used to represent the network, while the last 8 bits (the 0s) are used to represent individual hosts within that network. This allows for the segmentation of IP addresses into different subnets, each with its own network and hosts.

Subnet masks are crucial in IP addressing and routing because they help routers and devices determine whether an IP address is on the same local network or if it needs to be forwarded to another network. By applying the subnet mask to an IP address, you can determine the network address and use that information for routing traffic within or between networks.

Subnetmask will decide on the network that an IP address

range starts at what IP, ends at what IP.

How many IP address do you get from the range?

What will be the network address?

What will be the broadcast address?

All these things are decided based on the subnet mask.

Top of Form

DSCP server will give the subnet mark and the network range

## Example

Your Ip address: 192.168.0.174

Subnet mark: 255.255.255.0

So

192.168.0.0 🡪 Network IP

192.168.0.1 🡪First usable IP

192.168.0.2

.

.

.

192.168.0.254 🡪 Last usable IP

192.168.0.255 🡪 broadcast

TOTAL IP : 256

TOTAL USABLE IP : 254

Your Ip address: 172.16.12.16

Subnet mark: 255.255.0.0

So

172.16.0.0 🡪 Network IP

172.16.0.1 🡪 Last Usable IP

172.16.0.2

.

.

.

172.16.0.255

172.16.1.0

172.16.1.1

.

.

.

172.16.255.254 🡪 Last usable IP

172.16.255.255 🡪 Broadcast IP

TOTAL IP : 256\*256

TOTAL USABLE IP : 256\*256 -2

Your Ip address: 10.23.12.56

Subnet mark: 255.0.0.0

SO

10.0.0.0/8 🡪 /24 is notation in cloud completed representation of subnet mask

10.1.0.0/24

10.2.0.0/24

.

.

10.255.0.0/24

10.255.1.0/24

10.255.2.0/24

.

.

.

10.255.255.254/24

10.255.255.255/24

TOTAL IP : 256\*256\*256

TOTAL USABLE IP : 256\*256 \*256-2

## **CIDR** - Classless Inter-Domain Routing

Subnet mask

255.255.255.0

11111111. 11111111. 11111111.00000000

CIDR notation 🡪 /24

255.255.0.0

11111111. 11111111. 00000000. 00000000

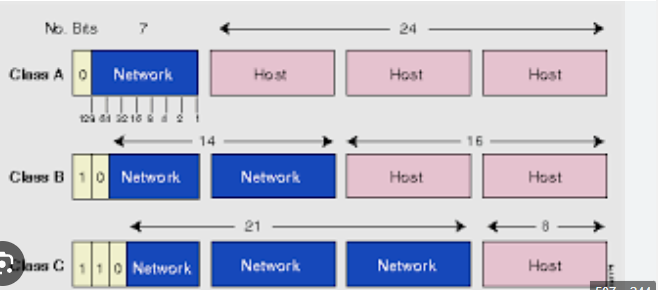
CIDR notation 🡪 /16

255.255.0.0

11111111.00000000.00000000.00000000

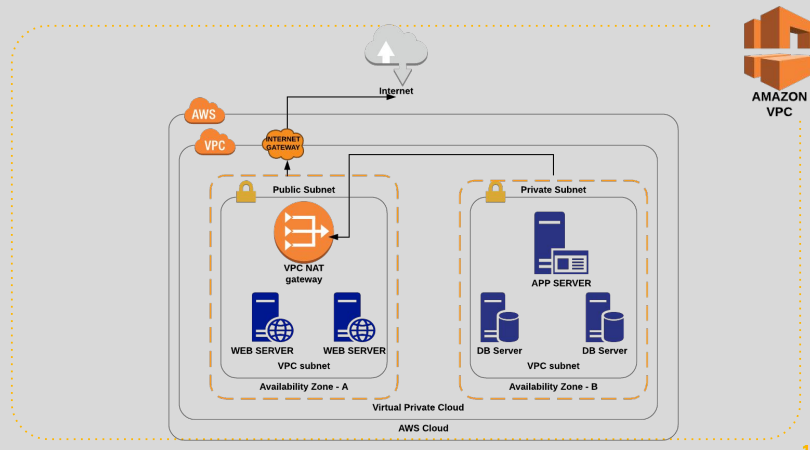
CIDR notation 🡪 /8

Note **: also we have online subnet calculators**

****

**Network address & host address**

# VPC design and components



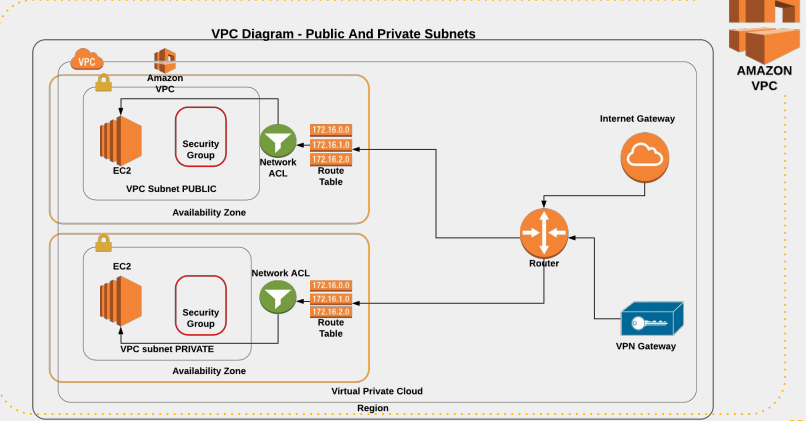
This is basic VPC, they two kinds of subnet in VPC ( public & private ).

* PUBLIC SUBNET can access the internet directly thought internet gateway (it will manage interner though and flow).
* private subnet can’t access directly, first it will route to public subnet though NAT gateway(NAT gateway act as wi-fi router and connected to the internet , consider private subnet like laptop).

\*\* private subnet can access the internet thought NAT gateway but internet can not access the private subnet\*\*

Network Address Translation (NAT) gateway to enable instances in a private subnet to connect to the internet or other AWS services.

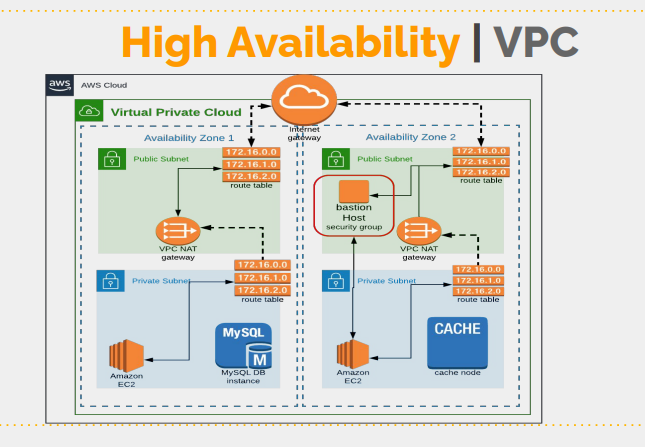
An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the internet.



In the above design , route table decide the network connect need to go to Nat gateway or internet gateway

Private network 🡪 route table decide network traffic to nat gateway

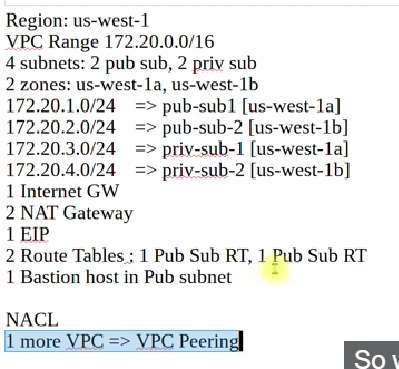
Public network 🡪 route table decide 🡪 network traffic goes to internet gateway



In this we have deploy the instances in two AZ’S for the high availability and set the 2 NAT gateway(only nat gateway is chargeable in above diagram)

See the above design and understand it

# VPC SETUP DETIALS



### Creating a VPC and its Components on AWS

In this tutorial, we will set up a Virtual Private Cloud (VPC) with subnets, an internet gateway, a NAT gateway, route tables, and other components. Here's a step-by-step guide to accomplish this:

#### Step 1: Define the VPC and Subnet Structure

* **VPC CIDR Block:** 172.20.0.0/16
  + This provides over 65,000 IP addresses.
* **Subnets:**
  + Public Subnet 1: 172.20.1.0/24 (Availability Zone: us-west-1a)
  + Public Subnet 2: 172.20.2.0/24 (Availability Zone: us-west-1b)
  + Private Subnet 1: 172.20.3.0/24 (Availability Zone: us-west-1a)
  + Private Subnet 2: 172.20.4.0/24 (Availability Zone: us-west-1b)

#### Step 2: Create the VPC

1. **Navigate to the VPC Dashboard:**
   * Open the AWS Management Console.
   * Switch to the North California region (us-west-1).
   * Search for and open the **VPC** service.
2. **Create the VPC:**
   * Click on **Create VPC**.
   * Set the **Name Tag** to vprofile-vpc.
   * Enter 172.20.0.0/16 as the **IPv4 CIDR Block**.
   * Leave the **Tenancy** as default.
   * Click **Create VPC**.

#### Step 3: Create Subnets

1. **Public Subnet 1:**
   * Click on **Subnets** > **Create subnet**.
   * Select vprofile-vpc for the VPC.
   * Set the **Name Tag** to vpro-pubsub-1.
   * Select **Availability Zone** us-west-1a.
   * Set the **IPv4 CIDR Block** to 172.20.1.0/24.
   * Click **Create subnet**.
2. **Public Subnet 2:**
   * Repeat the above steps with:
     + **Name Tag:** vpro-pubsub-2
     + **Availability Zone:** us-west-1b
     + **IPv4 CIDR Block:** 172.20.2.0/24
3. **Private Subnet 1:**
   * Repeat the steps with:
     + **Name Tag:** vpro-privsub-1
     + **Availability Zone:** us-west-1a
     + **IPv4 CIDR Block:** 172.20.3.0/24
4. **Private Subnet 2:**
   * Repeat the steps with:
     + **Name Tag:** vpro-privsub-2
     + **Availability Zone:** us-west-1b
     + **IPv4 CIDR Block:** 172.20.4.0/24

#### Step 4: Create and Attach the Internet Gateway

1. **Create the Internet Gateway:**
   * Click on **Internet Gateways** > **Create internet gateway**.
   * Set the **Name Tag** to vprofile-igw.
   * Click **Create internet gateway**.
2. **Attach the Internet Gateway to the VPC:**
   * Select vprofile-igw.
   * Click **Actions** > **Attach to VPC**.
   * Select vprofile-vpc.
   * Click **Attach internet gateway**.

#### Step 5: Create the Route Tables

1. **Public Route Table:**
   * Click on **Route Tables** > **Create route table**.
   * Set the **Name Tag** to vpro-public-RT.
   * Select vprofile-vpc for the VPC.
   * Click **Create route table**.
   * Select the newly created route table.
   * Click on the **Routes** tab, then **Edit routes**.
   * Add a route:
     + **Destination:** 0.0.0.0/0
     + **Target:** vprofile-igw (internet gateway ID)
   * Click **Save routes**.
   * Go to the **Subnet Associations** tab.
   * Click **Edit subnet associations**.
   * Select vpro-pubsub-1 and vpro-pubsub-2.
   * Click **Save**.
2. **Private Route Table:**
   * Repeat the steps with:
     + **Name Tag:** vpro-private-RT
   * For now, skip the route configuration and subnet association until we create the NAT gateway.

#### Step 6: Create the NAT Gateway

1. **Elastic IP:**
   * Click on **Elastic IPs** > **Allocate Elastic IP address**.
   * Click **Allocate**.
   * Note the allocated Elastic IP address.
2. **NAT Gateway:**
   * Click on **NAT Gateways** > **Create NAT gateway**.
   * Select the **Subnet** as vpro-pubsub-1.
   * Select the allocated **Elastic IP**.

#### Step 6: Create the NAT Gateway (continued)

1. **NAT Gateway:**
   * Click on **NAT Gateways** > **Create NAT gateway**.
   * Set the **Name Tag** to vprofile-nat-gateway.
   * Select the **Subnet** as vpro-pubsub-1.
   * Select the allocated **Elastic IP**.
   * Click **Create NAT gateway**.
2. **Configure Private Route Table:**
   * Go to **Route Tables** and select vpro-private-RT.
   * Click on the **Routes** tab, then **Edit routes**.
   * Add a route:
     + **Destination:** 0.0.0.0/0
     + **Target:** NAT Gateway ID (vprofile-nat-gateway)
   * Click **Save routes**.
   * Go to the **Subnet Associations** tab.
   * Click **Edit subnet associations**.
   * Select vpro-privsub-1 and vpro-privsub-2.
   * Click **Save**.

#### Step 7: Create a Bastion Host (Jump Server)

1. **Launch EC2 Instance:**
   * Navigate to the **EC2 Dashboard**.
   * Click on **Launch Instance**.
   * Choose an Amazon Machine Image (AMI) (e.g., Amazon Linux 2).
   * Select an instance type (e.g., t2.micro).
   * Configure instance details:
     + **Network:** Select vprofile-vpc.
     + **Subnet:** Select vpro-pubsub-1.
   * Add storage (use default settings).
   * Add tags (e.g., **Name**: Bastion-Host).
   * Configure security group:
     + Create a new security group (e.g., BastionSG).
     + Allow **SSH** from your IP.
   * Review and launch the instance.

#### Step 8: Configure Network Access Control Lists (NACLs)

1. **Create a NACL for the Public Subnets:**
   * Navigate to **Network ACLs**.
   * Click on **Create Network ACL**.
   * Set the **Name Tag** to Public-NACL.
   * Select the **VPC** as vprofile-vpc.
   * Click **Create**.
   * Select the created NACL, go to the **Subnet Associations** tab, and associate it with vpro-pubsub-1 and vpro-pubsub-2.
2. **Inbound Rules:**
   * Click on the **Inbound Rules** tab and **Edit inbound rules**.
   * Add rules to allow HTTP, HTTPS, and SSH traffic from 0.0.0.0/0.
   * Add rule to allow all traffic from 172.20.0.0/16 for internal communication.
3. **Outbound Rules:**
   * Click on the **Outbound Rules** tab and **Edit outbound rules**.
   * Add rules to allow all traffic to 0.0.0.0/0.
4. **Create a NACL for the Private Subnets:**
   * Repeat the above steps for the private subnets with appropriate rules.
   * Set the **Name Tag** to Private-NACL.
   * Associate it with vpro-privsub-1 and vpro-privsub-2.

#### Step 9: Create and Peer with Another VPC

1. **Create a Second VPC:**
   * Go to the **VPC Dashboard** and click on **Create VPC**.
   * Set the **Name Tag** to secondary-vpc.
   * Enter 172.21.0.0/16 as the **IPv4 CIDR Block**.
   * Click **Create VPC**.
2. **Create a VPC Peering Connection:**
   * Go to **VPC Peering Connections** > **Create Peering Connection**.
   * Set the **Name Tag** to vpc-peering-connection.
   * Select the **Requester VPC** as vprofile-vpc.
   * Select the **Accepter VPC** as secondary-vpc.
   * Click **Create Peering Connection**.
   * Accept the peering request in the secondary-vpc.
3. **Update Route Tables for Peering:**
   * Go to **Route Tables** and update the routes for both VPCs to allow communication between them:
     + In vprofile-vpc route table, add a route to 172.21.0.0/16 pointing to the peering connection.
     + In secondary-vpc route table, add a route to 172.20.0.0/16 pointing to the peering connection.

### Final Configuration Verification

1. **Verify Subnets and Route Tables:**
   * Ensure all subnets are associated with the correct route tables.
   * Check the route entries in the public and private route tables to ensure they point to the correct gateways.
2. **Test Connectivity:**
   * Launch instances in the public and private subnets.
   * Use the Bastion Host to connect to instances in the private subnets.
   * Verify internet access for instances in public subnets and internet connectivity via NAT gateway for instances in private subnets.

This completes the setup of a VPC with public and private subnets, internet and NAT gateways, route tables, a bastion host, and NACLs, along with a peering connection to another VPC.

4o

# Default VPC

Normally, a default VPC is create in AWS. and we can’t create a default VPC. (We can create VPC but can’t make it default)

Similarly, in subnets also they are two default public subnet will be they

If you need find subnet are public or private

Select any subnet 🡪 route table (check port connection whether it is connecting to internet gateway or NAT gateway)

## Note

Don’t delete default VPC, internet gateways or route table. (if you delete you need to reach AWS helpdesk)

### AWS VPC Setup and Understanding Default VPC

#### Step 1: Switch to the North California Region

* Open the AWS Management Console.
* Switch to the **North California (us-west-1)** region.

#### Step 2: Explore the Default VPC

* **Navigate to the VPC Dashboard:**
  1. In the AWS Management Console, search for **VPC** and open it.
  2. You will see the VPC dashboard with various sections like VPC, Subnets, Route Tables, etc.
* **Examine the Default VPC:**
  1. Click on **Your VPCs**.
  2. Identify the default VPC by looking for the one marked with **Default VPC: Yes**. It typically has the CIDR block 172.31.0.0/16.
  3. Rename it to **DEFAULT VPC** for easier identification.

#### Step 3: Understand Default Subnets

* **Check Subnets:**
  1. Click on **Subnets** in the VPC Dashboard.
  2. Identify the subnets associated with the default VPC. They usually have the CIDR blocks within the range 172.31.0.0/16.
  3. Rename them for clarity, e.g., **Default-pubsub1** and **Default-pubsub2**.
  4. Note that the subnets are typically /20 giving around 4091 usable IP addresses each (AWS reserves 5 IPs per subnet).

#### Step 4: Identify Public and Private Subnets

* **Public Subnets:**
  1. Click on **Route Tables**.
  2. Select the route table associated with your default VPC.
  3. Check the **Routes** tab. If you see a route like 0.0.0.0/0 pointing to an Internet Gateway (IGW), it’s a public subnet.
  4. Subnets routing traffic to the IGW are public.
* **Private Subnets:**
  1. Subnets routing traffic to a NAT Gateway instead of an IGW are private.
  2. Typically, private subnets will route through a NAT Gateway to access the internet securely.

#### Summary of Key Concepts

* **Default VPC:** Automatically created by AWS in each region with a specific CIDR block (172.31.0.0/16).
* **Default Subnets:** Associated with the default VPC, usually in /20 ranges.
* **Public Subnets:** Have route table entries that send non-local traffic (e.g., 0.0.0.0/0) to an Internet Gateway.
* **Private Subnets:** Have route table entries that send non-local traffic to a NAT Gateway.
* **Internet Gateway (IGW):** Allows instances in the VPC to connect to the internet.
* **NAT Gateway:** Allows instances in private subnets to connect to the internet while preventing inbound traffic initiated from the internet.

### Practical Steps for VPC Setup

1. **Create a VPC:**
   * Go to the **VPC Dashboard**.
   * Click on **Create VPC**.
   * Name it vprofile-vpc.
   * Set the **IPv4 CIDR Block** to 172.20.0.0/16.
   * Leave **Tenancy** as default.
   * Click **Create VPC**.
2. **Create Subnets:**
   * Click on **Subnets** in the VPC Dashboard.
   * Click on **Create subnet**.
   * **VPC:** Select vprofile-vpc.
   * Add subnets:
     + **Name Tag:** vpro-pubsub-1, **Availability Zone:** us-west-1a, **CIDR Block:** 172.20.1.0/24.
     + **Name Tag:** vpro-pubsub-2, **Availability Zone:** us-west-1b, **CIDR Block:** 172.20.2.0/24.
     + **Name Tag:** vpro-privsub-1, **Availability Zone:** us-west-1a, **CIDR Block:** 172.20.3.0/24.
     + **Name Tag:** vpro-privsub-2, **Availability Zone:** us-west-1b, **CIDR Block:** 172.20.4.0/24.
3. **Create Internet Gateway:**
   * Click on **Internet Gateways**.
   * Click **Create internet gateway**.
   * Name it vprofile-igw.
   * Click **Create**.
   * Select the newly created IGW, click on **Actions**, and select **Attach to VPC**.
   * Choose vprofile-vpc and click **Attach internet gateway**.
4. **Configure Route Tables:**
   * Go to **Route Tables**.
   * Create a new route table for public subnets:
     + **Name:** vpro-public-RT, **VPC:** vprofile-vpc.
     + Add a route 0.0.0.0/0 pointing to the IGW.
     + Associate it with vpro-pubsub-1 and vpro-pubsub-2.
   * Create a new route table for private subnets:
     + **Name:** vpro-private-RT, **VPC:** vprofile-vpc.
     + Add a route 0.0.0.0/0 pointing to the NAT Gateway (once created).
     + Associate it with vpro-privsub-1 and vpro-privsub-2.

### Final Steps

* **Review Configuration:**
  + Verify all settings, CIDR blocks, and subnet associations.
* **Test Setup:**
  + Launch instances in public subnets and check internet connectivity.
  + Configure NAT Gateway and test instances in private subnets for internet access via NAT.

This setup provides a solid foundation for understanding and implementing VPCs in AWS, ensuring both public and private subnets are correctly configured for internet access and security.

# Create VPC

Nat gateway is expensive

### Step-by-Step Guide to Creating a VPC in AWS

#### 1. **Create the VPC**

* **Navigate to VPC Dashboard:**
  1. Go to the **VPC Dashboard** in the AWS Management Console.
  2. Click on **VPCs** in the left-hand menu.
* **Create VPC:**
  1. Click on **Create VPC**.
  2. **Name Tag:** Enter vprofile-vpc.
  3. **IPv4 CIDR Block:** Enter 172.20.0.0/16.
  4. **Tenancy:** Select default.
  5. Click on **Create VPC**.

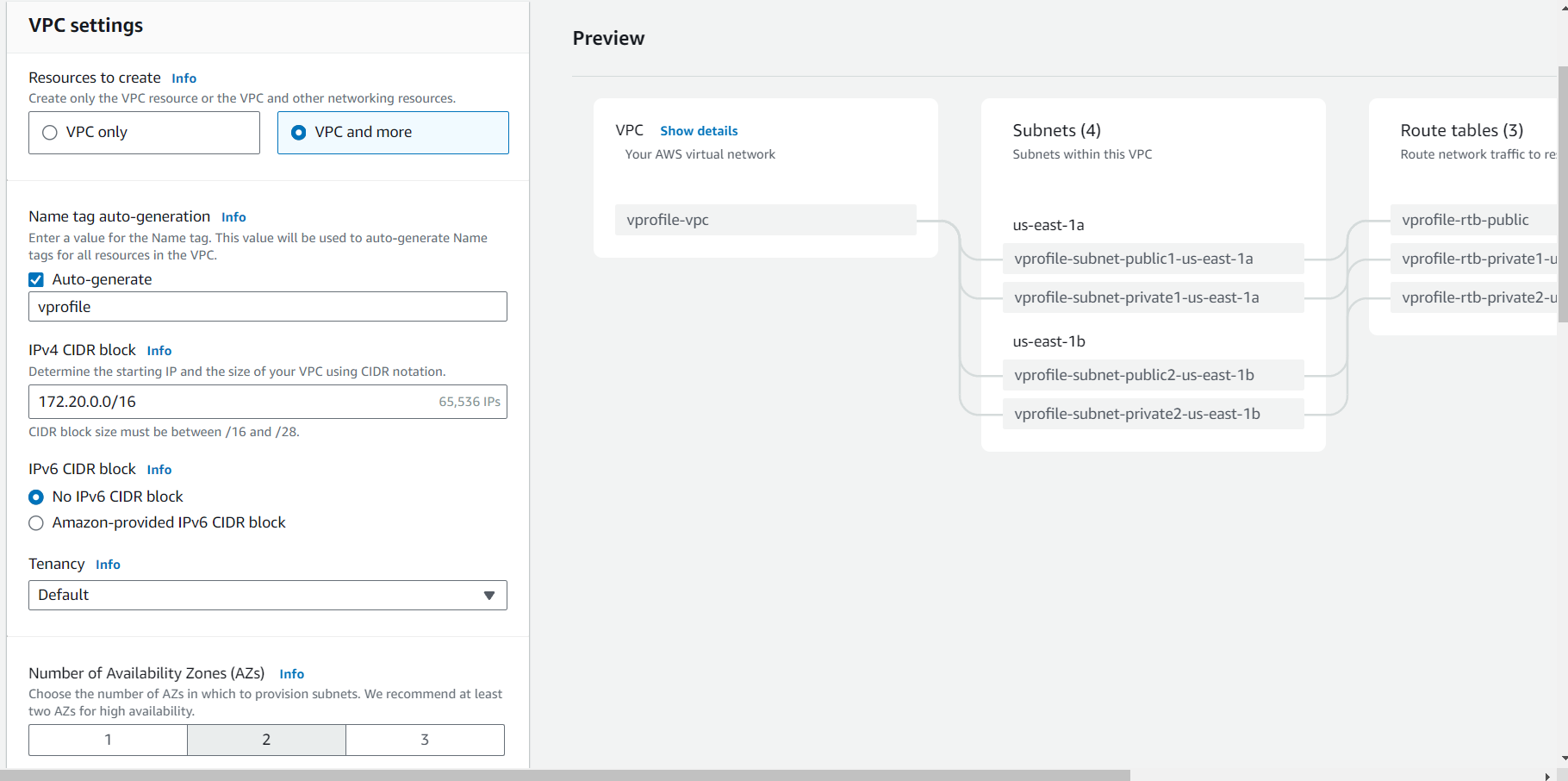
#### 2. **Confirm VPC Creation**

* **Verify VPC:**
  1. Ensure that your VPC (vprofile-vpc) is listed under the VPCs section.
  2. Note that a default route table is automatically created but will not be used in this configuration.

### Summary

You have successfully created a VPC with the CIDR block 172.20.0.0/16. This VPC will serve as the foundation for your network infrastructure. In the next steps, you will create subnets, an internet gateway, route tables, and other necessary components to complete your VPC setup.

## VPC and more



It will create the all compounds (subnet, route table & network connection)

But in this class we going to create by individually

## VPC only

In create 4 subnets (2 for public, 2 for private)

Vpro-pubsub-1 172.20.1.0/24 us-west-1a

Vpro-pubsub-2 172.20.2.0/24 us-west-1b

Vpro-pubsub-1 172.20.3.0/24 us-west-1a

Vpro-pubsub-2 172.20.4.0/24 us-west-1b

### Step-by-Step Guide to Creating Subnets in AWS VPC

#### 1. **Create Subnets**

* **Navigate to Subnets:**
  1. Go to the **VPC Dashboard** in the AWS Management Console.
  2. Click on **Subnets** in the left-hand menu.
* **Create Subnet:**
  1. Click on **Create subnet**.
  2. **VPC:** Select your VPC (e.g., vprofile-VPC).

#### 2. **Add Subnets Details**

* **Add New Subnets:**
  1. Click on **Add new subnet** until you have four subnets.

#### 3. **Enter Subnet Information**

##### First Public Subnet

1. **Name Tag:** vpro-pubsub-1
2. **Availability Zone:** Select us-east-1a (or your preferred AZ)
3. **IPv4 CIDR Block:** 172.20.1.0/24

##### Second Public Subnet

1. **Name Tag:** vpro-pubsub-2
2. **Availability Zone:** Select us-east-1b (or your preferred AZ)
3. **IPv4 CIDR Block:** 172.20.2.0/24

##### First Private Subnet

1. **Name Tag:** vpro-privsub-1
2. **Availability Zone:** Select us-east-1a (or your preferred AZ)
3. **IPv4 CIDR Block:** 172.20.3.0/24

##### Second Private Subnet

1. **Name Tag:** vpro-privsub-2
2. **Availability Zone:** Select us-east-1b (or your preferred AZ)
3. **IPv4 CIDR Block:** 172.20.4.0/24

#### 4. **Review and Create Subnets**

* **Review Subnet Details:**
  1. Ensure that each subnet has the correct name, availability zone, and CIDR block.
  2. Double-check that the subnets are created in the correct VPC (vprofile-VPC).
* **Create Subnets:**
  1. Click on **Create subnet**.

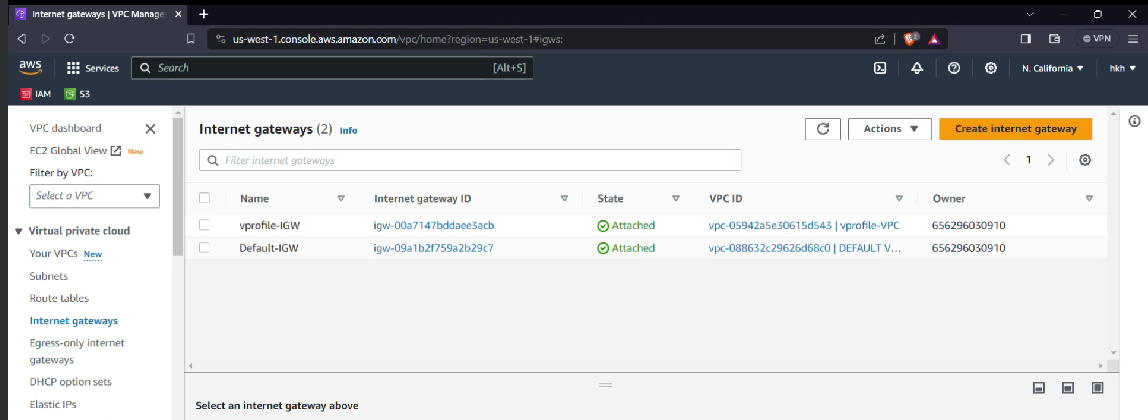
### Summary

You have successfully created four subnets within your VPC: two public subnets and two private subnets. However, these subnets are currently not functional for internet access. In the next steps, you will configure route tables and associate them with the subnets, and set up an internet gateway to enable internet connectivity for the public subnets.

# INTERNET GATEWAY

Internet gateway 🡪 create internet gateway

Attach internet gateway to vpc(vprofile)



### Step-by-Step Guide to Creating and Attaching an Internet Gateway in AWS VPC

#### 1. **Create an Internet Gateway**

* **Navigate to Internet Gateways:**
  1. Go to the **VPC Dashboard** in the AWS Management Console.
  2. Click on **Internet Gateways** in the left-hand menu.
* **Rename Default Internet Gateway:**
  1. Select the existing internet gateway for the default VPC.
  2. Click on **Actions** and select **Edit name**.
  3. Rename it to Default-IGW.
* **Create a New Internet Gateway:**
  1. Click on **Create internet gateway**.
  2. **Name:** Enter vprofile-IGW.
  3. Click on **Create internet gateway**.

#### 2. **Attach Internet Gateway to VPC**

* **Attach to VPC:**
  1. Select the newly created internet gateway (vprofile-IGW).
  2. Click on **Actions** and select **Attach to VPC**.
  3. **VPC:** Select your VPC (e.g., vprofile-VPC).
  4. Click on **Attach internet gateway**.

#### 3. **Verification**

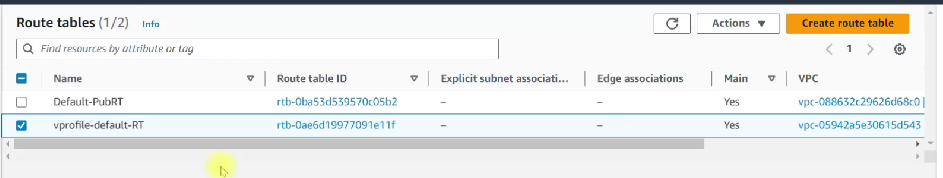
* **Check Internet Gateway Status:**
  1. Ensure that the state of the internet gateway is now attached.

### Summary

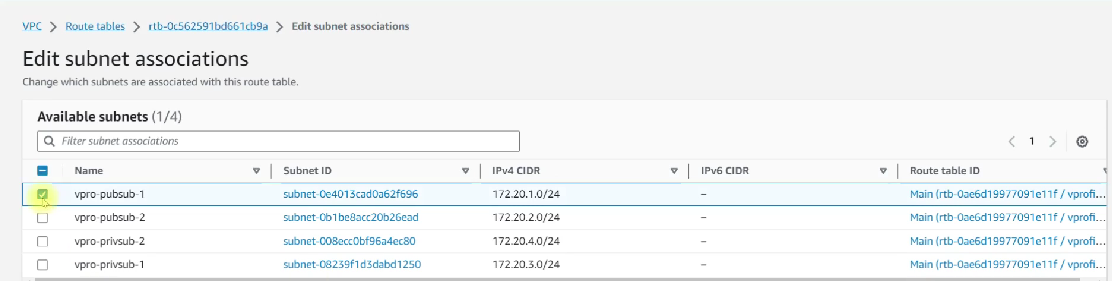
You have successfully created and attached an internet gateway to your VPC. This setup will allow instances in your VPC to communicate with the internet. In the next steps, you will associate this internet gateway with your public subnets through route tables to enable internet access for instances in those subnets.

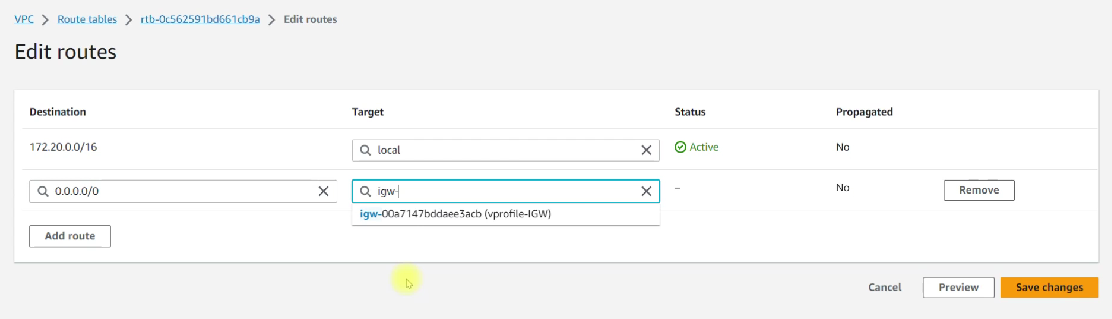
# Route Table

When you create a VPC, it automatically create one route table



But create a new route table and add the public subnets and edit route add 0.0.0.0/0 🡪 this makes it public





### Step-by-Step Guide to Configuring Route Tables and Enabling Public IP Assignment in AWS VPC

#### 1. **Overview**

* **Objective:** Connect Internet Gateway to subnets via route tables and enable automatic public IP assignment for instances in the public subnets.
* **Components:** VPC, Internet Gateway, Public Subnets, Route Tables.

#### 2. **Rename Route Tables for Clarity**

* **Navigate to Route Tables:**
  1. Go to the **VPC Dashboard** in the AWS Management Console.
  2. Click on **Route Tables** in the left-hand menu.
* **Rename Default Route Tables:**
  1. Identify the route table for the default VPC and rename it to Default-PubRT.
  2. Identify the automatically created route table for your VPC (e.g., vprofile-VPC) and rename it to vprofile-default-RT.

#### 3. **Create a Route Table for Public Subnets**

* **Create New Route Table:**
  1. Click on **Create route table**.
  2. **Name:** Enter vpro-public-RT.
  3. **VPC:** Select your VPC (e.g., vprofile-VPC).
  4. Click on **Create route table**.

#### 4. **Associate Route Table with Public Subnets**

* **Subnet Associations:**
  1. Select the newly created route table (vpro-public-RT).
  2. Go to the **Subnet associations** tab.
  3. Click on **Edit subnet associations**.
  4. Select your public subnets (e.g., vpro-public-subnet-1 and vpro-public-subnet-2).
  5. Click on **Save**.

#### 5. **Add Route to Internet Gateway**

* **Edit Routes:**
  1. With the vpro-public-RT route table selected, go to the **Routes** tab.
  2. Click on **Edit routes**.
  3. Click on **Add route**.
  4. **Destination:** Enter 0.0.0.0/0.
  5. **Target:** Select the Internet Gateway for your VPC (e.g., vprofile-InternetGateway).
  6. Click on **Save changes**.

#### 6. **Enable Automatic Public IP Assignment**

* **Navigate to Subnets:**
  1. Go to the **VPC Dashboard**.
  2. Click on **Subnets** in the left-hand menu.
* **Enable Auto-Assign Public IP for Public Subnets:**
  1. Select the first public subnet (e.g., vpro-public-subnet-1).
  2. Click on **Actions** and select **Edit subnet settings**.
  3. Check the box for **Enable auto-assign public IPv4 address**.
  4. Click on **Save**.
  5. Repeat for the second public subnet (e.g., vpro-public-subnet-2).

#### 7. **Verification**

* **Public Subnets Configuration:**
  1. Ensure that the public subnets have the route table associated (vpro-public-RT) with a route to the Internet Gateway.
  2. Confirm that the setting for auto-assigning public IPs is enabled for both public subnets.

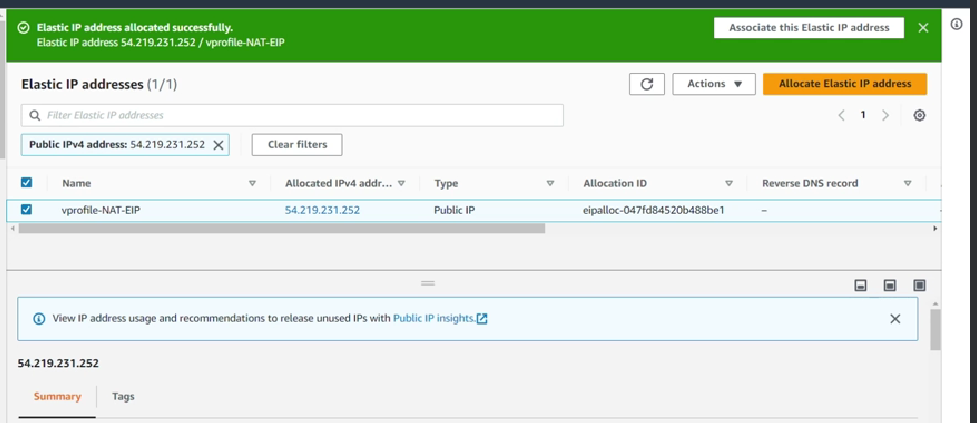
### Summary

You have successfully connected your public subnets to the Internet Gateway via route tables and enabled automatic public IP assignment. This setup allows instances launched in the public subnets to access the internet and be accessible via their public IPs.

In the next steps, you will configure the NAT Gateway and route tables for the private subnets to complete your VPC setup.

# NAT Gateway

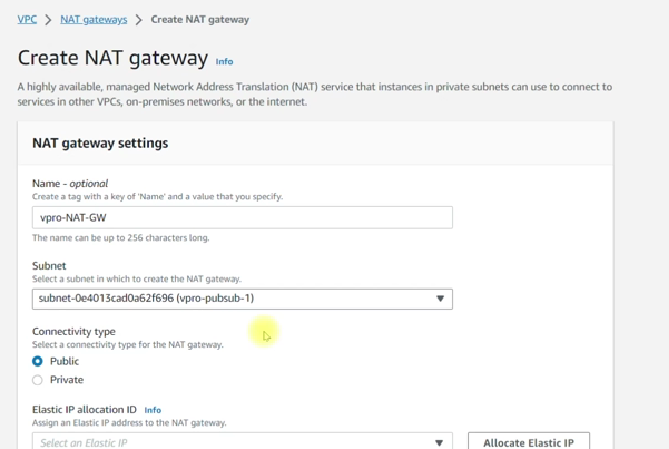
* First create Elastic IP

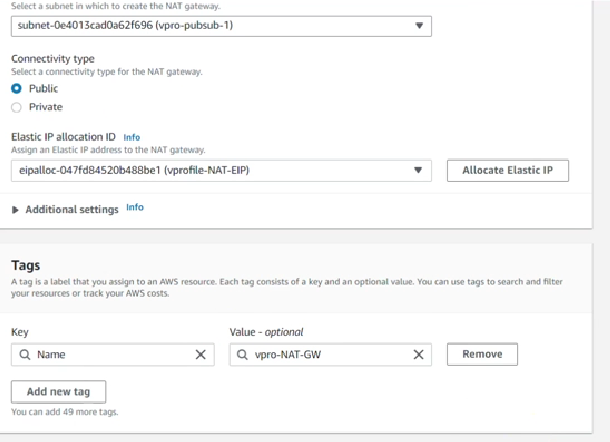


* Create a nat gateway

Nat gateway is connected to private subnets but it lives in public subnet(place in pubsub)

Add the allocated elastic IP which we created before





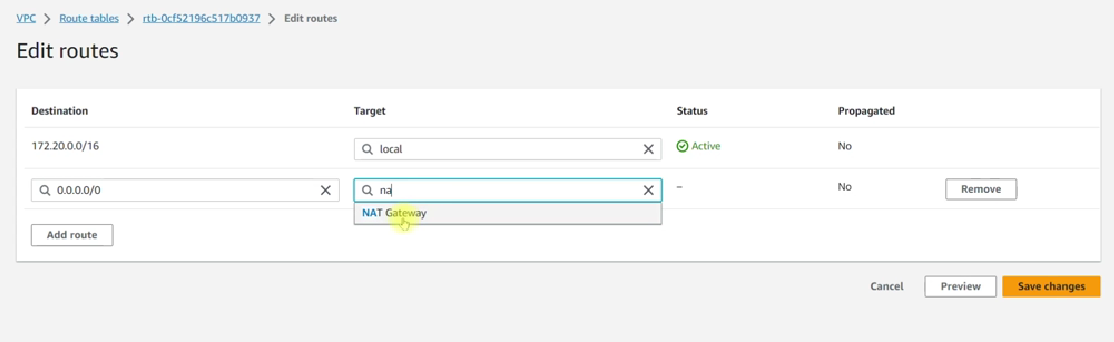
In meantime

* Create a route table for private IP

Create route table(vprofile-priv-RT)🡪 attach to vprofile-vpc

route table(vprofile-priv-RT)🡪subnets associations 🡪 edit subnets associations 🡪 add the private subnets 🡪 save associations

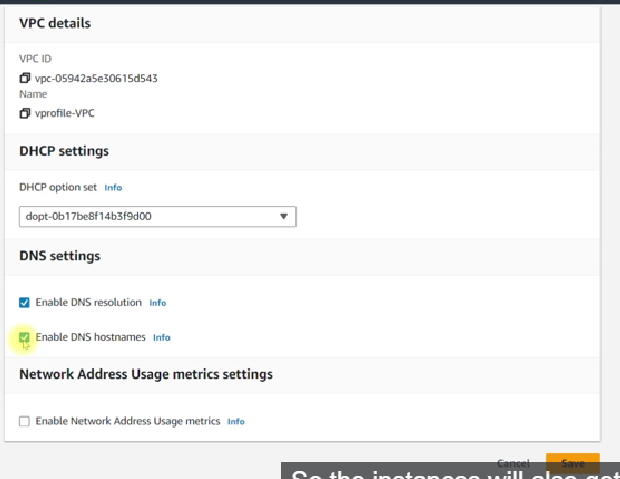
route table(vprofile-priv-RT)🡪 route 🡪 add the route 0.0.0.0/0 🡪 nat gateway



## Host names

We can enable the host names (but it not a default),

Goto VPC🡪find your VPC(vprofile)🡪 Action🡪 edit VPC setting🡪enable DNS Host names



### Step-by-Step Guide to Creating a NAT Gateway in AWS VPC

#### 1. **Allocate Elastic IP**

* **Navigate to Elastic IPs:**
  1. Go to the AWS Management Console.
  2. Navigate to the **VPC Dashboard**.
  3. Click on **Elastic IPs** in the left-hand menu.
* **Allocate a New Elastic IP:**
  1. Click on **Allocate Elastic IP address**.
  2. (Optional) Add a name tag for easy identification, e.g., vprofile-NAT-ElasticIP.
  3. Click on **Allocate**.

#### 2. **Create a NAT Gateway**

* **Navigate to NAT Gateways:**
  1. In the VPC Dashboard, click on **NAT Gateways** in the left-hand menu.
* **Create a NAT Gateway:**
  1. Click on **Create NAT Gateway**.
  2. **Name:** Enter a name, e.g., vpro-NAT-Gateway.
  3. **Subnet:** Select a public subnet where the NAT Gateway will reside (e.g., vpro-public-subnet-1).
  4. **Elastic IP:** Select the Elastic IP you allocated in the previous step.
  5. Click on **Create NAT Gateway**.
* **Wait for the NAT Gateway to become available:**
  1. The status will initially show as **pending**. It will become **available** after a few minutes.

#### 3. **Create a Route Table for Private Subnets**

* **Navigate to Route Tables:**
  1. In the VPC Dashboard, click on **Route Tables** in the left-hand menu.
* **Create a New Route Table:**
  1. Click on **Create route table**.
  2. **Name:** Enter a name, e.g., vpro-private-route-table.
  3. **VPC:** Select your VPC (e.g., vprofile-VPC).
  4. Click on **Create route table**.

#### 4. **Associate Route Table with Private Subnets**

* **Subnet Associations:**
  1. Select the route table you just created (vpro-private-route-table).
  2. Go to the **Subnet associations** tab.
  3. Click on **Edit subnet associations**.
  4. Select the private subnets (e.g., vpro-private-subnet-1 and vpro-private-subnet-2).
  5. Click on **Save**.

#### 5. **Create Route to NAT Gateway**

* **Add Route:**
  1. Select the route table (vpro-private-route-table).
  2. Go to the **Routes** tab.
  3. Click on **Edit routes**.
  4. Click on **Add route**.
  5. **Destination:** Enter 0.0.0.0/0 (this route sends all traffic to the internet that does not match a more specific route).
  6. **Target:** Select the NAT Gateway you created earlier.
  7. Click on **Save changes**.

#### 6. **Enable DNS Hostnames for the VPC**

* **Navigate to VPC:**
  1. In the VPC Dashboard, click on **Your VPCs** in the left-hand menu.
* **Edit VPC Settings:**
  1. Select your VPC (e.g., vprofile-VPC).
  2. Click on **Actions** and select **Edit VPC settings**.
  3. Scroll down to **DNS hostnames** and check **Enable**.
  4. Click on **Save**.

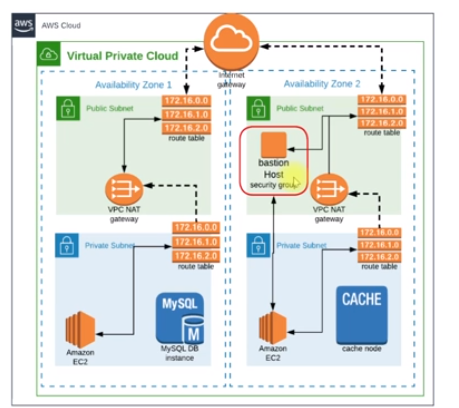
### Summary

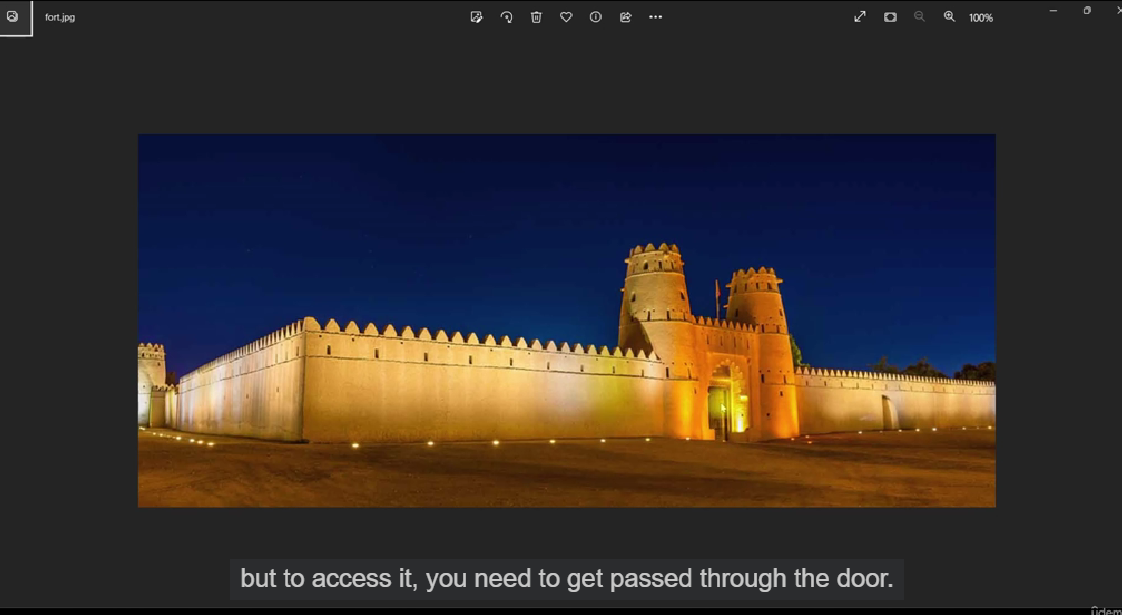
You have successfully set up a VPC with public and private subnets, configured a NAT Gateway, and updated route tables. This setup allows instances in private subnets to access the internet via the NAT Gateway, while instances in public subnets can directly access the internet through an Internet Gateway.

In the next lecture, you will learn how to use this VPC by creating instances (bastion hosts) in the private subnets and configuring a load balancer.

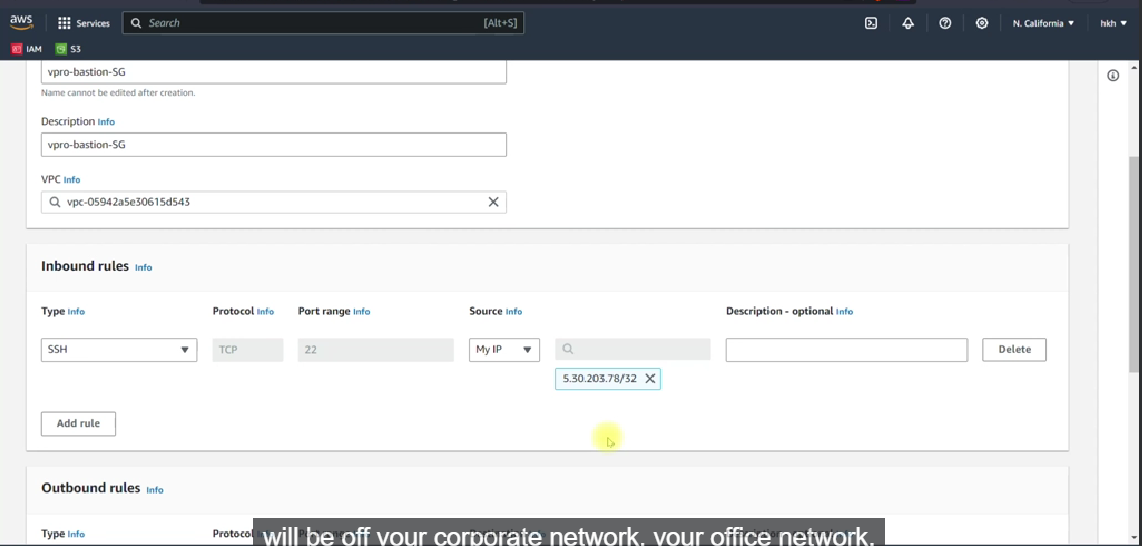
# Bastion host/Jump server

Jump server 🡪it is not a VPC team, it is general networking term assigned to any secure network, host or computer through which you can access the private subnet or resources in the private subnet.

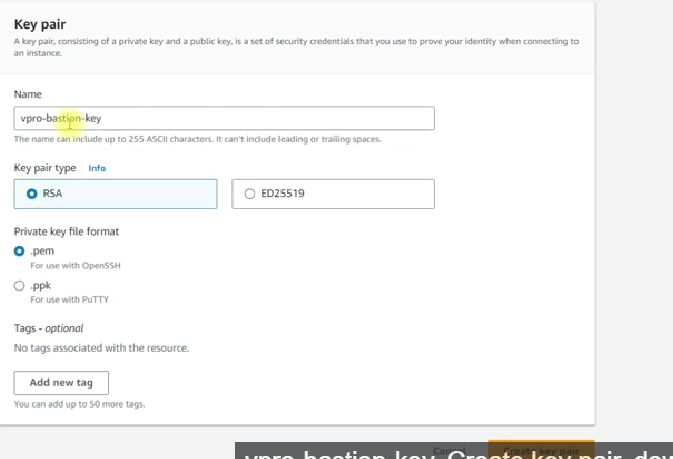




Create a security group:



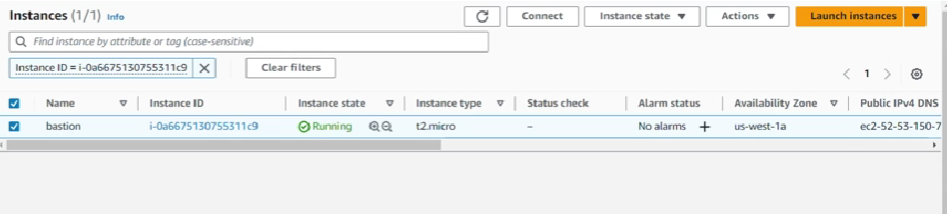
Create a key-pair



AMI:

So far we are using the free AMI, **But for the real time (bastion host), you should go for AMI that as tested for vulnerability 🡪 ex : CIS (central security information system)**

* **Create a ubuntu instances for this**

****

## Summary of Creating a Bastion Host

In this lecture, we learned how to create a bastion host, also known as a jump server. This host is a crucial part of a secure network setup, particularly in a Virtual Private Cloud (VPC) environment. Here are the key steps and concepts covered:

### What is a Bastion Host?

* A bastion host is a secure network host that provides access to a private subnet.
* It allows secure access to resources in the private subnet, such as instances without public IPs.
* Access to these resources requires connecting first to the bastion host, which is placed in the public subnet.

### Setting Up the Bastion Host

#### Step 1: Create a Security Group

1. **Navigate to the EC2 dashboard** on AWS.
2. **Create a new security group** for the bastion host:
   * Name it vpro-bastion-sg.
   * Assign it to your specific VPC (vprofile-VPC).
   * Add a rule to allow SSH access from your IP address (or your corporate network IP).

#### Step 2: Create a Key Pair

1. **Create a new key pair**:
   * Name it vpro-bastion-key.pem.
   * Download and keep the private key safe.

#### Step 3: Launch the Bastion Host Instance

1. **Choose an appropriate Amazon Machine Image (AMI)**:
   * For real-time use, consider using a tested AMI from the AWS Marketplace, such as those provided by the Center for Internet Security (CIS).
   * For learning purposes, you can use a free tier eligible AMI like Ubuntu Server 22.
2. **Configure the instance**:
   * Select t2.micro as the instance type.
   * Use the previously created key pair.
   * Ensure the instance is placed in the public subnet of your VPC.
   * Attach the security group (vpro-bastion-sg).
3. **Launch the instance** and wait for it to be in the running state.

### Connecting to the Bastion Host

1. **Copy the public IP of the instance** from the EC2 dashboard.
2. **Use an SSH client** (such as Git Bash or terminal) to connect to the bastion host:
   * Use the command: ssh -i /path/to/vpro-bastion-key.pem ubuntu@<public-ip>.
   * Accept the connection by typing yes when prompted.

### Verification

* Successfully SSH-ing into the instance confirms that the public subnet setup is correct.

### Next Steps

* In the next lecture, we will:
  + Launch an instance in the private subnet.
  + Access the private instance through the bastion host.
  + Set up a website on the private instance and access it via a load balancer.

By following these steps, you ensure that your private resources are securely accessed through a well-guarded entry point, maintaining the integrity and security of your network.

# Website in VPC

let's follow the steps outlined in your lecture to set up an EC2 instance in a private subnet, configure a bastion host, and set up a load balancer to route traffic to the private instance.

### Steps to Set Up the Website in Private Subnet with Load Balancer

1. **Create Key Pair and Copy to Bastion Host**:
   * Create a new key pair named web-key and download it.
   * Use scp to copy the web-key.pem to the bastion host.

sh

Copy code

scp -i vprofile-bastion-key.pem ~/Downloads/web-key.pem ubuntu@<BastionHostPublicIP>:/home/ubuntu/

1. **Launch EC2 Instance in Private Subnet**:
   * Go to the EC2 dashboard and launch a new instance.
   * Use Amazon Linux 2, instance type t2.micro.
   * Select the web-key key pair.
   * Place the instance in the private subnet of the vprofile-VPC.
   * Create a new security group web01-sg allowing SSH from the bastion host security group and HTTP from the load balancer security group.
2. **SSH into Private Instance via Bastion Host**:
   * SSH into the bastion host.

sh

Copy code

ssh -i vprofile-bastion-key.pem ubuntu@<BastionHostPublicIP>

* + From the bastion host, SSH into the private instance using the private IP and the web-key.pem.

sh

Copy code

chmod 400 /home/ubuntu/web-key.pem

ssh -i /home/ubuntu/web-key.pem ec2-user@<PrivateInstanceIP>

1. **Set Up the Web Server on the Private Instance**:
   * Install HTTPD and other necessary packages.

sh

Copy code

sudo yum install httpd wget unzip -y

* + Download and set up a template website.

sh

Copy code

wget https://www.tooplate.com/zip-templates/2107\_pop.zip

unzip 2107\_pop.zip

sudo cp -r 2107\_pop/\* /var/www/html/

sudo systemctl start httpd

sudo systemctl enable httpd

1. **Create a Target Group for the Load Balancer**:
   * Go to the EC2 dashboard and create a target group for your instances.
   * Select instances as the target type, port 80, and the vprofile-VPC.
   * Register your private instance.
2. **Create a Security Group for the Load Balancer**:
   * Create a security group web-elb-sg allowing HTTP (port 80) from anywhere.
3. **Create an Application Load Balancer**:
   * Go to the EC2 dashboard and create a new Application Load Balancer.
   * Name it, select the vprofile-VPC, and select public subnets.
   * Attach the web-elb-sg security group.
   * Configure the listener to route to the target group created earlier.
4. **Update Security Group for Private Instance**:
   * Edit the inbound rules of web01-sg to allow HTTP (port 80) from the load balancer security group (web-elb-sg).

sh

Copy code

Inbound rule:

Type: HTTP

Protocol: TCP

Port Range: 80

Source: web-elb-sg

1. **Test the Setup**:
   * Wait until the load balancer is active and the target group shows the instance as healthy.
   * Access the website using the load balancer's DNS name.

### Clean Up Resources

1. **Delete Load Balancer**:
   * Go to the EC2 dashboard, select the load balancer, and delete it.
2. **Terminate Instances**:
   * Terminate both the bastion host and private instances.
3. **Delete NAT Gateway and Release Elastic IP**:
   * Delete the NAT gateway and then release the associated Elastic IP.

### Additional Notes

* Ensure that the security group rules are configured correctly to allow necessary traffic.
* Verify that the private instance's web server is running and accessible through the load balancer.
* If issues persist, review network ACLs, route tables, and security groups for any misconfigurations.

By following these steps, you should be able to successfully set up a private instance with a bastion host and a public-facing load balancer.

# VPC peering

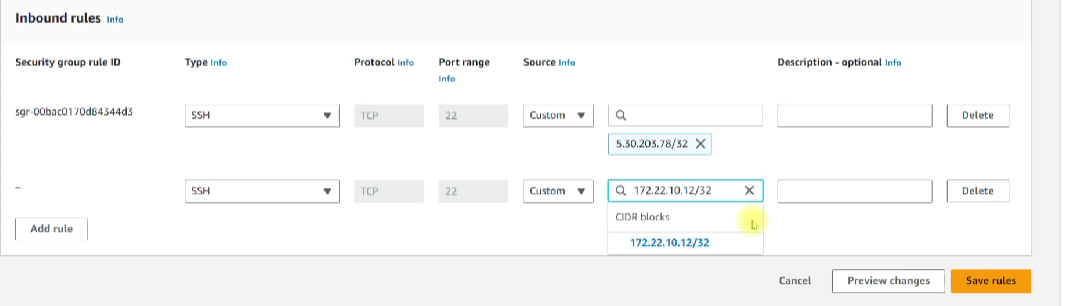
### Implementing VPC Peering Between Two VPCs in Different Regions

“you have seen everything in the VPC and the purpose of creating VPC is, our own custom VPC is so we get security through private subnet and high availability through multiple zones. And we distribute our infrastructure in multiple zones through the subnets. But when you're working in a large-scale environment, you will not have just one single VPC. You may have one VPC for the web infrastructure, one maybe for API, one for the database. You may have multiple VPCs. And in order to connect your resource from one VPC to other VPC, you need to do VPC peering.””

To connect two VPCs in different regions using VPC peering, follow these steps. We'll create a new VPC in a different region and set up VPC peering between this new VPC and an existing one.

### Step-by-Step Guide

1. **Create a New VPC in a Different Region**
   * Navigate to the AWS Management Console.
   * Switch to the region where you want to create the new VPC (e.g., Oregon).
   * Go to the VPC Dashboard and click on "Create VPC".
   * Name it vpro-db and set the CIDR block to 172.21.0.0/16 (ensure it doesn't overlap with the existing VPC).
2. **Create Peering Connection**
   * Switch back to the region of the existing VPC (e.g., North California).
   * Go to the VPC Dashboard and click on "Peering Connections".
   * Click on "Create Peering Connection".
   * Name it vpro-nc.
   * Select the requester VPC (vprofile-VPC in North California).
   * For the accepter, select "Another region" and choose the region (e.g., Oregon).
   * Enter the VPC ID of vpro-db in the Oregon region.
   * Click "Create Peering Connection".
3. **Accept the Peering Connection**
   * Switch to the Oregon region.
   * Go to the VPC Dashboard and select "Peering Connections".
   * You should see the pending peering request. Select it and click "Actions" -> "Accept Request".
4. **Update Route Tables**
   * Go back to the North California region.
   * Navigate to the Route Tables section.
   * Update the route tables for subnets that need to communicate with the new VPC.
     + For example, select the private route table.
     + Click "Edit Routes".
     + Add a new route with the destination CIDR block of the new VPC (172.21.0.0/16) and target as the peering connection ID.
   * Repeat the same for the Oregon region:
     + Update the route tables to route traffic destined for 172.20.0.0/16 to the peering connection.
5. **Update Security Groups**
   * Security groups in the peered VPCs need to allow traffic from the CIDR blocks of the other VPC.
   * Go to the security group settings in the North California region.
     + Add inbound rules to allow traffic (e.g., SSH on port 22, HTTP on port 80) from 172.21.0.0/16.
   * Do the same in the Oregon region:
     + Update the security groups to allow traffic from 172.20.0.0/16.



### Cleaning Up

1. **Delete Peering Connection**
   * Navigate to the VPC Dashboard in both regions.
   * Go to "Peering Connections".
   * Select the peering connection and delete it.
2. **Delete the VPC**
   * Ensure the peering connection is deleted before trying to delete the VPC.
   * Navigate to the VPC Dashboard in the Oregon region.
   * Select the vpro-db VPC and delete it.
3. **Terminate Instances and Other Resources**
   * Terminate any instances, and delete the NAT gateway, route tables, and other resources you created for the demo.
   * Ensure you release any Elastic IPs to avoid unnecessary charges.

### Summary

By following these steps, you should be able to establish a peering connection between two VPCs in different regions and configure them to communicate securely. This setup is crucial for scenarios where you need to ensure secure and seamless connectivity between different parts of your infrastructure spread across multiple regions. Practice and familiarity with these concepts will significantly enhance your skills as a DevOps engineer.

# EC2 LOG

### Log Management on AWS

Welcome to this session on log management in AWS. Today, we will discuss general log handling, archiving, and various AWS solutions for managing logs. Here’s a structured approach to handling logs for web servers, particularly focusing on AWS services.

#### Setting Up a Web Server with Amazon Linux 2

1. **Create and Configure Web Server:**
   * Use Amazon Linux 2 AMI.
   * Install HTTPd service.
   * Download and use a template from tooplate.com.
2. **Log Generation:**
   * Access logs generated by HTTPd service located at /var/log/httpd/access\_log.

#### Archiving Logs to S3

1. **Create an S3 Bucket:**
   * Create a bucket named wave-web-logs-<unique-id>.
2. **Archive and Move Logs:**
   * Archive logs using tar czvf wave-web01-httpdlogs-<timestamp>.tar.gz \*.
   * Move the archive to a temporary directory and clean the log files:

bash

Copy code

mkdir /tmp/logs-wave

mv wave-web01-httpdlogs-<timestamp>.tar.gz /tmp/logs-wave/

> /var/log/httpd/access\_log

> /var/log/httpd/error\_log

1. **Install and Configure AWS CLI:**
   * Install AWS CLI using yum install awscli.
   * Configure AWS CLI with an IAM user that has S3 access.
2. **Copy Logs to S3:**

bash

Copy code

aws s3 cp /tmp/logs-wave/wave-web01-httpdlogs-<timestamp>.tar.gz s3://wave-web-logs-<unique-id>/

#### Streaming Logs to CloudWatch

1. **Use IAM Roles for EC2 Instance:**
   * Create a role with S3 full access and CloudWatch Logs full access.
   * Attach the role to the EC2 instance.
2. **Install and Configure CloudWatch Logs Agent:**
   * Install the CloudWatch Logs agent:

bash

Copy code

yum install awslogs

* + Edit the configuration file /etc/awslogs/awslogs.conf to specify the log files to stream:

ini

Copy code

[/var/log/httpd/access\_log]

log\_group\_name = wave-web

log\_stream\_name = web01-httpd-access

file = /var/log/httpd/access\_log

* + Start and enable the awslogs service:

bash

Copy code

systemctl start awslogsd

systemctl enable awslogsd

1. **Verify Logs in CloudWatch:**
   * Check CloudWatch Logs in the AWS console for the new log groups and streams.

#### Using Load Balancer Logs

1. **Enable Access Logs for Load Balancer:**
   * Create a folder in the S3 bucket for load balancer logs (elb-wave).
   * Enable access logs in the load balancer settings and configure the S3 bucket and folder.
2. **Set Up Bucket Policies:**
   * Edit the bucket policy to allow the load balancer to write logs:

json

Copy code

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::<elb-account-id>:root"

},

"Action": "s3:PutObject",

"Resource": "arn:aws:s3:::wave-web-logs-<unique-id>/elb-wave/AWSLogs/<aws-account-id>/\*"

},

{

"Effect": "Allow",

"Principal": {

"Service": "delivery.logs.amazonaws.com"

},

"Action": "s3:PutObject",

"Resource": "arn:aws:s3:::wave-web-logs-<unique-id>/elb-wave/AWSLogs/<aws-account-id>/\*"

},

{

"Effect": "Allow",

"Principal": {

"Service": "delivery.logs.amazonaws.com"

},

"Action": "s3:GetBucketAcl",

"Resource": "arn:aws:s3:::wave-web-logs-<unique-id>"

}

]

}

1. **Verify Log Delivery:**
   * After enabling, check the S3 bucket for the presence of load balancer log files.

### Conclusion

By following these steps, you can efficiently manage and archive logs from your web server to AWS S3, stream logs to CloudWatch for real-time monitoring, and handle load balancer logs, ensuring comprehensive log management in your AWS infrastructure.