EC2 (Definition:- It is scalable)

→ We can connect in different ways to this server; with the help of linux server, power shell, git bash, putty.

Connection methods:-

① Linux server to linux server:-

→ Create a key in the linux server

```
cat > intelliq.pem
```

→ Paste the key in the above file and save.

→ change file permissions as

```
chmod 400 intelliq.pem
```

→ Now connect your server.

eg:-
```
ssh -i intelliq.pem ec2-user@public ip
```

```
ssh -i keypair-name.pem user_name@public ip
```

② Powershell to linux server

→ Go to the folder in which the pem file is present, in powershell

```
cd downloads
```

then

```
ssh -i intelliq.pem ec2-user@public-ip
```

③ Git bash

→ open git bash

→ go to the folder where keypair is present

→ `ssh -i intelliq.pem ec2-user@public-ip`

④ Putty

→ It requires only ppk file to connect the server.

pem → privacy enhanced mail

ppk → putty private key

→ Convert .pem file to .ppk file with the help of puttygen.

→ open puttygen → click on load → select the pem file → Save private key

→ open putty
  → Hostname (paste public ip)
    → Right hand side click on SSH
      → Then click on Auth
        → Then click on Credentials
          → Browse private key → open.

Interview Questions

Errors :-

① No supported authentication error.

→ This is due to if you dont add key pair.

→ If you forget to attach the key pair you will get authentication issue.

Solution:-

→ Attach key pair pair while connecting.

(2). Timed out error (or) Connection timed out

→ This is due to if you don't add the port no. 22 in Security group.

→ $ prompt :- We are running user comands

→ # prompt :- We are running root comands

→ To switch from normal user to root user

`sudo -i` (or) `sudo su -`

→ To find current user name

`whoami`

(*) How to check kernel version ?

`uname -r`

(*) To check linux version name

`cat /etc/redhat-release` → redhat

`lsb_release -a` → ubuntu

release

→ It shows current version. let's say Version is 9.3

then I want install application on 9.4
then update the version as

dnf/yum update -y

→ To check the pending updates

yum check-update

Note:- We can use yum/dnf

Scenario:

* Install httpd (webserver) on Redhat linux server:

→ dnf/yum install -y httpd

→ To check the version

httpd -v

→ To check whether the service is running or not:

systemctl status httpd

→ To start a service systemctl start httpd

`Systemctl status/start/stop/enable/restart service name`

enable → It is used to keep the service in running state automatically when the server is rebooted.

(or)

to make the service persistence.

→ To host a webpage in httpd.

→ go to location

`cd /var/www/html`

the load the index.html file.

(*) To check the binary name of ifconfig

`yum provides ifconfig`

`yum provides package_name`

→ when you are checking ifconfig, it is not showing, it shows command not found.

→ then install it `yum install ifconfig`

→ but it is showing not matched with the package.

→ therefore to install ifconfig, we must have it binary name..

`yum provides ifconfig`

→ It shows binary names as `net-tools`

first install net-tools and then ifconfig

→ To check port number

☆ `netstat -pantl` (or) `netstat`

☆ → If the service is not responding we can troubleshoot through logs.

→ The httpd log-files are under the location

`cd /var/log/httpd`

syntax:- `cd /var/log/service_name`

→ In that location we have two logs, they are

access_log | error_log

access_log :- It maintains the, who are visiting the service (or) website

error_log :- It maintains the service log files (at what time it is stopped, started restarted)

cat access_log and cat error_log

→ To open the file continously:

tail -f access_log

→ INSTANCE : Virtual Compute Environment

→ TAGS :- Meta data known as Tags (used for identification and use it in automation)

→ AMI :- Pre-configured Images (Pre-config Os + additional s/w's)

→ EBS :- Persistent storage for data.

→ INSTANCE TYPE :- Configurations of various CPU/memory/storage/networking.

→ KEYPAIR :- secure login information for your instance

→ SECURITY GROUPS :- Firewall rules using Security Groups.

EC2 Purchasing

1) ON-DEMAND :- You pay on what you use.

2) SAVING PLANS :-
3) RESERVED PLANS : } → both are same the commitment would be 2 or 3 years.

eg :- If your project is 2 or 3 years, you take this plan. Then you can save upto 60% of the amount.

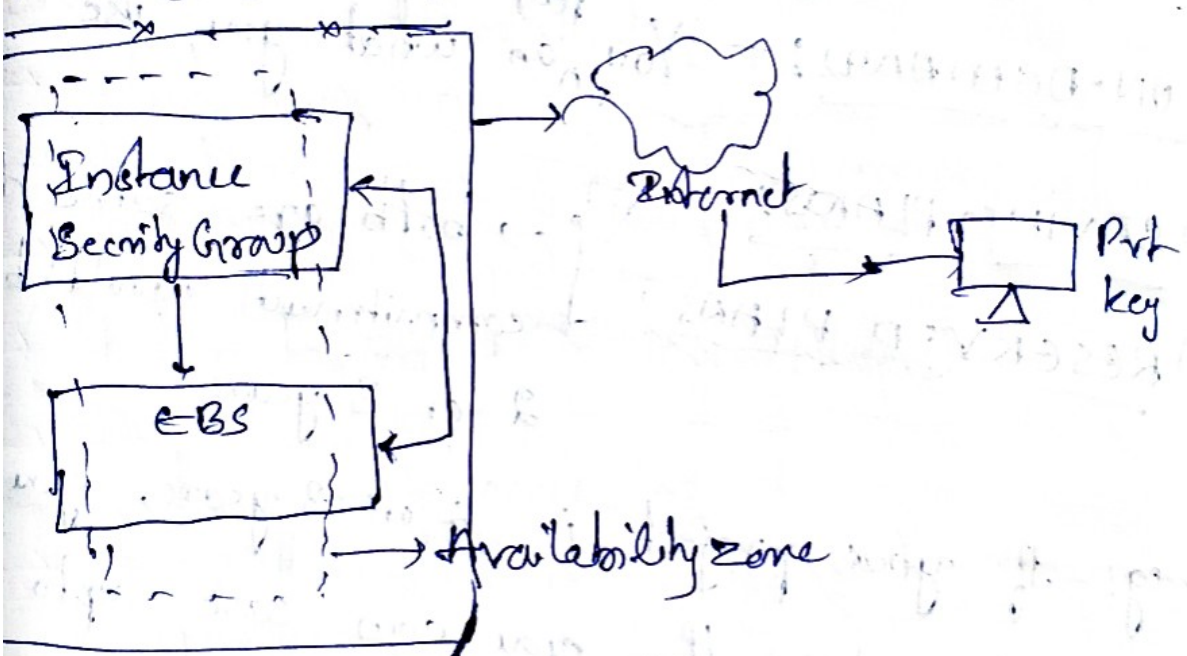
4) SPOT INSTANCES :- The pricing is based on demand.

5) DEDICATED INSTANCES :- Particulary dedicated for you only. eg :- ola cars

EC2 Instance Types:-

1) General purpose instances
2) Compute optimized
3) Memory optimized
4) Accelerated computed
5) storage optimized
6) HPC optimized (High Performance Compute)

EC2 overview:-



Instance Life cycle

launch
AMI → pending
running → stopping → stopped
terminate
shutting down
terminated

How to launch EC2 windows server and install httpd

Login Username : Administrator

Port : (RDP) : 3389

Remote Desktop Protocol

Connect the window server

→ Select connect
→ Select RDP client
→ Click on Get Password
→ Then upload your private key
→ Clicle on Decrypt.
→ Copy password
→ In the search option (Rdp)

RDP
computer : paste DNS or public IP

→ Clicle on connect
→ Click on more choices
→ Select Use different account
Username : Administrator

Webserver :- To host webpages.

Webserver - IIS :- (Internet Information Service) role in windows server provides a secure, easy to manage, modular and extensible platform for reliably hosting websites, services and applications.

HTTP → Hyper Text Transfer Protocol : 80.

→ Select windows
→ Select Server manager
→ Add role and features
→ Select webserver(IIS) from drop down
→ Next and install.
→ Now take the public ip and access from the browser.
→ To host website → go to C drive on the server.
→ Then click on inetpub → wwwroot → in that default pages are there, remove them and copy your pages

→ Create text document as save as index.html

```
<html>
<body bgcolor=yellow text=blue>
<marquee><H1> WELCOME </H1></marquee>
</body>
</html>
```

→ After creating the index.html file remove text file.

How to generate custom key pairs

→ We can generate with the help of Puttygen

→ Click on generate.

→ while generating move the cursor on the prompt only.

→ Then save public and private key.

→ Then in AWS import public key.

→ In AWS account → right hand side, there is keypair → click on Actions → Import key pair.

Note:- Keep private key in your laptop and export public key in Aws account.

How to create Security Group

→ It is used to protect the instance
→ It is like firewall.
→ But every SGs maintain two rules they are inbound and outbound
→ By default SGs supports allow rules, it means statefull i.e we can write only inbound rules not the outbound rules.

→ Security Groups
→ Create SG
→ In inbound section → click on add → write select the ports whatever you want for the application.

Note:- Tags are also used for automation purpose.

Elastic Network Interface :-

→ It is like ethernet cards,

→ Every instance has one ENI., so the users able to connect the instance over the internet.

→ Every instance has one IP address, the IP address is attached to interfaces.

→ In real time we need to attach multiple interfaces

Usecase :-

→ One application like oracle is running on the server with (172.32.10.2)

→ If I want to use tomcat in the same server then create ENI tomcat (172.32.10.5)

→ If we want to run multiple applications on the same server use ENI

Interview Q. **Note :-** We can't delete the network interfaces, one interface is mandatory

→ We can delete additionally added network interfaces.

→ To check no. of ENI's

ifconfig

enX0, enX1 ---- for amazon linux

eth0, eth1 ---- for other linux server

② Use case:

→ To increase the speed

→ Suppose eth0 (1Gbps) eth1 (1Gbps)

→ If we bond them, the speed increases

How to create ENI

→ In Network & Security

→ Click on Network interfaces

→ Create Network interface.

Note:- It must be create in the same zone of where the instance is running.

→ Auto assign

(or)

→ custom (if you have a ip address)

→ select SG

→ Tags

→ Create.

→ Then select the interface → actions → attach to the instance.

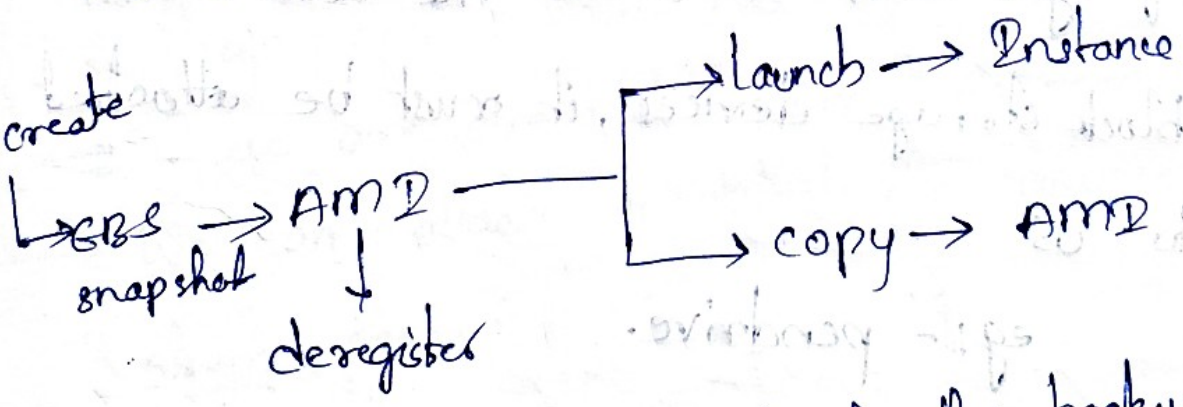
→ To check, then give `ifconfig`

→ To clear the history in the server

`history -c`

AMI :-

→ ami life cycle



→ AMI creates snapshot for the backup in the backend. automatically.

EBS (Elastic Block Storage)

→ Block storage is a technology that is used to store datafiles on storage area networks (SANs) or cloud based storage environments.

→ It breaks up data into blocks and then stores those blocks as separate pieces, each with unique identifiers.

→ It offers an impressive level of flexibility because it can be accessed by different OS's as mounted drive volumes and has the ability to use OS specific file system.

→ If you want to access the data from block storage devices, it must be attached to OS

eg:- pendrive.

Volumes are two types

1) Instance store volumes (temporary)
2) Block store volumes (permanent)

① Instance store :-

→ These volumes are virtual devices whose underlying hardware is physically attached to the host computer that is running the instance

→ These are 'Ephemeral data'

→ Once the instance is "stopped" or "shutdown" data is erased.

② Block store volumes

→ These are permineut

→ The data is persistent.

→ These are highly available and reliable.

→ These are created in multiple zones.

→ We can attach or detach the volume from the instance.

EBS performance :-

→ It measure i/p/output operations in IOPS

→ AWS measures IOPS in 256 kb chunks

EBS volume types

① SOLID STATE DRIVE (SSD):-

→ Best option of IOPS-intensive use cases such as transactional workloads, databases & boot volumes.

② HARD DISK DRIVE (HDD):

→ For throughput-intensive use cases like storage, Mapreduce and log processing.

③ PREVIOUS GENERATION:-

→ Workloads with small datasets where data is accessed infrequently and performance is not of primary importance.

SSD

④ General purpose SSD (gp2/gp3):-

→ Use for Dev/Test environments and smaller DB instances

④ Provisioned IOPS SSD (io1/io3)

→ Used for mission critical applications, large database workloads

→ Volume size of 4 GiB to 16 GiB

HDD

① Throughput optimised HDD: (st1)

→ low cost volumes designed for frequently accessed

→ Volume size of 500 Gb to 16 Tb

eg:- Big data, data warehouses, log processing.

② COLD HDD (sc1)

→ low cost volumes designed for less frequently accessed

→ The volume size of 500 Gb to 16 Tb

③ Magnetic (standard)

→ Lodo cost, low performance, data is infrequently accessed

→ Volume size min 1 Gb to 1 Tb.

→ To check the volumes

`lsblk` or `fdisk -l`

→ Every volume has located under /dev

→ The vender name is xvd

→ /dev/xvda

a → the position of volume.

b

c

:

z

Create a volume and attach to a instance

→ first check the instance's zone.

→ Then create the volume in the same zone

→ Go to EBS → volumes → create volume

→ volume type

→ volume size, let's say 1GB

→ Availability zone

→ Create.

→ Now select that volume and attach to the instance.

How to increase 3GB volume to 5GB volume

→ Select the 3GB volume → actions → modify volume.

To store the data in the volume

→ format :- It means building file system
→ mount :- attaching a folder to the volume

file systems in Windows

1) fat
2) fat32 → universal file system.
3) NTFS (New Technology File System) → ~~universal~~

linux file systems

→ xfs
→ file system will decide what kind of data to store.

→ To store what kind of data in the volume is decided by file system and it done by formating and mounting.

format

```
mkfs.xfs -f /dev/xvdb
```

Mount

```
mkdir /devops
```

```
mount /dev/xvdb /devops
```

Snapshot :-

→ It is nothing but point in time backup that are stored in s3

→ Snapshots are incremental in nature, it stores the changes since the most recent snapshot, thus reducing costs.

→ To take backup of volume,

→ select the volume → actions → create snapshot

→ Again to use the snapshot as a volume

→ first convert it into volume and attach to a server.

→ go to snapshot → select snapshot → actions
→ create volume from the snapshot

Interview

(*) when to create image from the snapshot and when to create volume from snapshot?

→ when snapshot contains root volume information (i.e OS information) create image and when snapshot contains regular data create volume and attach it.

How to create automate backups

→ Go into cloudwatch
→ Events → rules → create rule →
→ Name AUTO-SNAP
→ click on Schedule → continue to create rule
→ select (A schedule that runs at a regular rate, such as every minutes)
→ rate 1 min (for example) → next →
→ select target (EBS snapshot)
→ paste volue id
→ Next.

→ It will create multiple snapshots automatically.

→ I don't want to keep all these snapshots and want to retain latest 2 snapshots and delete older ones automatically for this we use DLM

DLM (Data Life cycle Manager)

→ It provides a simple, automated way to backup data stored on AWS EBS volumes.

→ It can define backup and retention schedules for EBS snapshots by creating lifecycle policies based on tags.

→ Right hand side in EBS

→ select Life cycle manager → Next

→ Target resources
- ◉ volume
- ○ instance

→ tags

Name	my-snap

→ add

→ Policy description

auto-delete-snaps

→ fill the time details → create

S3 (Simple storage Service)

Block storage	Object storage
1) It maintains file systems (eg: xfs, ntfs, fat, fat32)	1) It doesn't maintain any filesystems
2) To access the block storage, it must be attached to OS	2) It doesn't requires OS to access the storage.
3) The data gaurantee is not 100% because it has filesystem. if the file system is corrupted, then data is lost	3) It gaurantee your data. ie (99.999999%)

→ S3 is a object storage Service.

→ We can access the S3 bucket globally

→ Buckets are the main storage containers for objects

→ Buckets must have unique name globally.

→ There is unlimited storage.

→ 100 buckets per an account.

→ Objects are static files contain meta-data information.

→ All objects are private by default.

→ Object can be as small as 0 byte and as large as 5TB

→ Be made publicly available via URL.

S3 security:-

→ All buckets are created in Private mode

→ Bucket access controlled by:

(i) bucket policies

(ii) Access Control list

→ Object access controlled by ACL

→ Bucket activity tracking.

(i) logs

(ii) events

storage class ← It represents object availability, durability and cost.

storage classes:

1. standard → Frequently accessed data.
2. S3 Express One Zone → Single digit millisecond response time for the most frequently accessed data.
3. Intelligent Tiering → data with changing or unknown access patterns
4. standard - IA → Infrequently accessed data
5. One Zone - IA → Infrequently accessed data stored in a single zone.
6. Glacier Instant Retrieval → long-lived archive data accessed once a quarter.
7. Glacier Flexible Retrieval → long-lived data accessed once a year
8. Glacier deep Archieve → long lived archive data less than once a year with retrieval of hours
9. Reduced Redundancy → Frequently accessed data with millisecond access

S3 Server accey logging

→ It provides detailed records for the request that are made to a bucket.

→ Server accey logs are useful for many applications.

eg:- who is accesing the bucket, modifying, deleting, etc, I want to maintain all these logs.

→ Under properties → server accey logs → enable

AWS CLOUD TRAIL DATA EVENTS

→ It is the AWS API Auditing Service.

→ Object-level logging allows you to incorporate s3 object accey to your central auditing and logging in cloud trail.

→ Everything about the event is recorded in cloud trail.

→ Integrate this S3 bucket with cloud trail, then only it will record everything.

→ Not only S3, you can integrate any service on AWS with cloud trail

Creation: Select bucket → properties → aws cloud trail → data events → configure in cloud trail

Event notifications :-

→ It sends notification to owner if any event is occuring, with the service called SNS

S3 static Website Hosting :-

→ It can host entire static website.

→ It requires no virtual machines/instances

→ It is static only, can't serve database-driven content.

Creation :-

→ Create a S3 bucket, with the name of

your website (eg:- www.flipkart.com)

→ enable the ACL's

→ Unblock the public access

→ Create bucket

→ Now open the bucket, under properties section

enable the ☑ static website hosting.

→ index.html

→ error.html

→ Save changes

→ Now upload the html.index, error.html in the s3 bucket.

→ click on "add files", then upload.

→ If the website is not ~~opp~~ opening then make it public by writing policies or any other way.

→ let me use policy.

→ In google type "s3 bucket policy for static website"

→ copy that code and paste

→ In the bucket under "permissions" section

→ in bucket policy, paste the code and

makes changes, as bucket name.

S3 Management

→ An object life cycle policy is a set of rules that automate the migration of an objects storage class to a different storage class, based on specific 'Time Intervals'

→ Minimum time interval is 30 days

Writing policy:-

→ write rule name

→ select ☑ move noncurrent objects (i.e older objects or older versions)

→ Add class and days

→ create.

S3 Cross Region Replication

→ Automatic and Asynchronous copying of objects ~~across~~ across buckets in different AWS regions

→ Require versioning.

eg:- I am a developer, wanted access the bucket of other developer in the ~~same~~ ~~region~~ or other region or other account.

Scenario:-

Create one bucket in one region and another bucket in another region.

→ Create one bucket in N. Virginia
→ Create another bucket in Ohio region.
→ Asynchronous (ie one way only ie source to destination)
→ Now open one bucket, in that management section
→ In that "Replication rules"
→ Create replication rule
→ Name, my_bucket_name, to another bucket name
→ In Destination → choose the bucket
→ Destination storage → select One Zone-IA only. → Save.

S3 GLACIER

→ It is also one seperate service in AWS.

→ It is a storage service optimised for 'infrequently used data' or 'cold data'

→ It is a secure, durable and extremely low cost for data archiving and long-term backup

→ Minimum 90 days

→ Objects can be sent to Glacier using lifecycle management policies.

Note!— It is mainly designed for backups only i.e cold data.

→ Before uploading into this class, archive the data and upload

Vaults! —

→ Glacier uses "vaults" as containers to store Archives (Zip file)

→ An archive can be any data such as photo, video, document etc.

33 GLACIER

→ It is also one separate service in AWS.
→ It is a storage service optimised for 'infrequently used data' or 'cold data'
→ It is a secure, durable and extremely low cost for data archiving and long-term backup
→ Minimum 90 days
→ objects can be sent to Glacier using lifecycle management policies.

Note:- It is mainly designed for backups only i.e cold data.

→ Before uploading into this class, archive the data and upload

Vaults:-
→ Glacier uses "vaults" as containers to store Archives (zip files)
→ An archive can be any data such as photo, video, document etc.

→ Vault size is unlimited

→ object size is max 5 Tb

→ Archive is TAR or ZIP file whose size is max 40 Tb

Creation:

→ S3 Glacier

→ Create vault

→ Name → BACKUP-DATA

→ Create

→ We don't have option to ~~create~~ upload the objects

→ We can use third party tool to upload or download data

→ In google, type Glacier tool → then download → install

→ In AWS account create Access keys and security access keys.

→ Then select the region, in which the glacier is present and select that and upload files

AWS CLI

→ It is a unified tool to manage AWS service
→ It can control multiple AWS services from the command line and automate them through scripts.

PIP Installs Python (Pip)

→ It is recommended method of installing AWS CLI on linux which is python based tool helps in install, ~~expo~~ upgrade and remove python packages.

Requirements:

→ Python
→ Aws cli tool
→ Windows, Linux/Unix, macOs.

Aws cli installation on windows

→ first install python
→ Install aws cli on windows from google type "aws cli on windows"

→ Check in powershell as "aws --version"

→ aws ec2 help or aws s3 help, then use those commands.

(or)

In google aws cli reference.

→ aws configure

→ enter access keys and secret access keys.

→ After configuring, all the user details are found in .aws directory

```
cd ~/.aws
```

→ ls

→ You will see config and credentials.

eg:-

① aws ec2 describe-regions --output=table

or

aws ec2 describe-regions.

② To get key pairs.

aws ec2 describe-key-pair

aws ec2 describe-key-pairs --key-name your

3) To create a bucket

```
aws s3 mb s3://chaleram-bucket
```

aws cli on linux

→ go to google and take the comand and paste
→ aws configure
→ paste credentials

IAM (Identity Acces Management)

→ It provides to securely control access to AWS services and resources for your users and groups.
→ It manage users & groups and use permissions to allow & deny their access to AWS resources
→ All IAM users are beside the root user, users must be created with proper permissions
→ Permissions are governed by Policies

Policies:-

→ A policy is a document that formally sta
one or more permissions.

→ By default an explicit deny always overri
and explicit allow.

→ AWS provides pre-built policy template to
assign to users and groups.

IAM Groups:-

→ It is a collection of multiple IAM users

→ It specifies the permissions for multiple users, it is easy to manage users by allocating permissions.

IAM users

→ An IAM user is a resource in IAM that has associated credentials and permissions.

→ It represents a person or an application that uses its credentials to make AWS requ

→ IAM user permissions are not same as root user permissions

IAM roles:-

→ An IAM role is an IAM entity that defines a set of permissions for making AWS service requests

→ It is not associated with specific user or group

→ IAM roles for EC2 instances

→ An EC2 instance can only have ONE role attached at a time

Interview

(*) Instead of aws cli, we can also do this roles, this is alternative to aws cli

→ The policy which is attached to the role, only that service comands only can execute

eg:- S3fullaccess role, then we can list S3 servie only.

Entry

① what is IAM role ?

→ Role contains policies, but every role makes service request, but one role can be attached to only one instance, and role is not for users and groups, It is only for instances

Create custom policy

```
{
  "version": "2012-10-17",
  "statement": {
    "Effect": "Allow/Deny",
    "Action": "S3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

→ We can create by Visual/json/

→ visual is more easy, no need of coding

→ It will be present in customer managed

→ We can also import policy for the service

Inline policy :- It is working with the identity i.e user.

→ Click on any user → inline policy
→ attach the policy

MFA (Multi factor Authentication)

→ It is the second layer for your password.

eg:- password is 1st layer
otp is second layer

→ Add MFA for a user
→ click on user
→ security credentials
→ Assign MFA device
→ google authenticator

AWS AUDIT, NOTIFICATION, PERFORMANCE & MONITORING

(i) Cloud Trail

(ii) SNS

(iii) SQS

(iv) Cloudwatch

CLOUD TRAIL

→ It provides governance, compliance, operational and risk auditing of audit for your AWS account

→ It tracks your activity and API usage

→ This event simplifies:

- → Security analysis
- → Resource change tracking
- → Troubleshoot operational issues

→ It is simply a "auditing service"

→ It records every activity on AWS account

→ It is global service

→ Trail is a configuration that enables delivery of cloudtrail events to S3 bucket, Cloudwatch logs/events.

→ Only one trail is free.

→ You must integrate your trail with any storage device eg:- S3 bucket.

Creation:-

→ Search cloudTrail in AWS

→ create trail

→ select S3 bucket/create new S3 bucket

→ click on management events

→ ☒ Read ☑ write

→ Next

SQS (Simple Queue Service)

→ It is message queuing service.

→ It sends, store, receive messages between s/w components at any volume without loosing messages.

eg:- If you buy a product from flipkart.

→ It sends messages like.

① Your product has been purchased.

② " " " shipped

③ " " " ready to deliver

④ " " " out for delivery.

→ In this way the messages has been queued.

→ It will be sent in order.

→ It has two kind of SQS; FIFO & standard.

FIFO :- First in first out

standard :- without order, it delivers message

Creation :-

→ Search SQS

→ Create queue

→ select standard/FIFO

→ create.

→ integrate with SNS

SNS

→ It enables applications, end-users & devices to instantly send and receive notifications.

→ It is attached to cloudwatch, which monitors the environments and ~~notifies~~ notifies the administrators of alerts, capacity issues, downtime, changes in the environment

SNS components

TOPIC: The group of subscriptions that you send a message to (like subject) (or) display name

SUBSCRIPTION: - An endpoint that a message is sent

PUBLISHER: - The entity that triggers the sending of a message.

ENDPOINTS: - HTTP, HTTPS, EMAIL, SQS, LAMBDA, SMS ... etc

CLOUD WATCH

→ It is a monitoring service for compled. stack (applications, infrastructure and services)

→ It is used to collect and track metrics, log files, set alarms and automatically react to changes aws resources

→ CloudWatch alarms can be used as "Triggers".

Monitoring Levels

① Basic monitoring

→ Data is available automatically in 5-min period at no charge.

→ By default, instances enabled with basic monitoring.

② Detailed monitoring

→ Data is available in 1-min period at an additional cost

→ Free tier allows us to have 10 detailed monitoring metrics

→ The above two methods can be enabled while launching instance also, i.e

in Advanced details → Detailed Cloudwatch monitoring → enable / disable

(or)

select the instance → actions → monitor and troubleshoot → manage detailed monitoring.

* Get instance screenshot! -

→ It shows the backend current state of a server.

→ It is used to troubleshoot, when server is not successfully booted up

* Get system log! -

→ To get the system logs,

Actions → monitor and troubleshoot → get system logs

2/2 checks passed! -

→ 2/2 → the server is healthy, no issues

from cloud side and client side

→ 1/2 ; 0/2 : the system is facing issues from the cloud side.

→ 9/0 , 9/1 : then problem is our side, then we can trouble shoot

→ Mostly the problem is on our side.

STATUS CHECKS

(1) System status checks: - (things that outside of your control)

→ Loss of system power and n/w connectivity

→ H/w and s/w issues on the physical host

How to solve? -

→ Generally stopping and starting the instance will fix the issues.

(2) Instance status checks: (s/w issues that we do control)

→ Mis configured networking or startup-configuration

→ Exhausted memory
→ corrupted file system
→ Incompatible kernel.

How to solve! –

→ Generally, a reboot or resolving the file system configuration issues.

ALARMS

→ Alarms are used to triggers notification of any metric.

→ Metric alarm state are: OK

~~[redacted]~~ INSUFFICIENT DATA

Actions → monitor and troubleshoot → manage cloud watch alarms

Alarm action: - It is used to take the action based on the alarm

→ to stop.
recover
reboot / terminate

Dashboards:-

→ It is the collection of alarms.

→ Create dashboard and add alarms to it.

→ go to alarms → select alarms → actions → add alarms to dashboard

→ 3 dashboards are free (upto 50 metrics)

** LOG GROUPS

→ Logs to monitor, store, access your log files from EC2 instance, Cloudtrail, Route53 etc.

→ To send logs to cloudwatch, make sure IAM permissions are correct.

→ Logs can use filter expressions

→ Logs can will be displayed in cloudwatch.

→ To overcome manually going into the location and seeing log files such as access_logs and error_logs.

How to integrate logs in cloudwatch

Scenario:

(1) Install httpd and see the logs in cloud watch.

→ Launch Amazon linux server.

→ Install httpd in it.

→ first install awslogs package

```
yum install awslogs -y
```

→ edit the awslogs.conf

```
vim /etc/awslogs/awslogs.conf
```

→ In the last of that file, there are some lines, copy those lines and paste beneth it and edit as shown below, the lines are

```
[/var/log/messages]
datetime_format = %b %d %H:%M:%S
file = /var/log/messages
buffer_duration = 5000
log_stream_name = {instance_id}
initial_position = start_of_file
```

log_group_name = /var/log/messages.

Copy the above lines and paste below and edit as shown

[/var/log/httpd/access_log]

datetime_format = %b %d %H:%M:%S

file = /var/log/httpd/access_log → It is the file location.

buffer_duration = 5000

log_stream_name = {webserver}

initial_position = start_of_file

log_group_name = ~~/var/log~~ ACCESS_LOG_FILE

↓

to display the name in cloudwatch.

→ Do same for error_logs

→ This is instance and cloudwatch is a different service, so service request is need, ie create role & cloudwatch full access or configure with Access key and secret access key, in the server.

i.e aws configure

. paste Access_key :
Secret_key :

→ systemctl restart awslogsd

→ systemctl enable awslogsd

** EVENTS

→ Events delivers a near real time stream of system events that describe changes in AWS resources.

→ It schedules automated actions that self-trigger at certain times using cron & rate expressions.

→ eg:- I want to launch a server at particular time, stop a server at particular time, schedule the time and it will be launched at that time.

Rate expressions:-

→ It starts when you create the scheduled event rule, and then runs on its defined schedule.

→ It has two required fields and separated by white space.

SYNTAX:-

rate (value unit)

Cloudwatch → Events → Rules → create rule →
Name → ⊙ schedule → continue to create rule

→ Rate expression

rate 

→ Next → target types → Next

Cron expressions:-

→ Cron expressions have six fields,

SYNTAX:-

cron (fields)

Field	Values
min	0-59
Hours	0-23
Day-of-month	1-31
month	1-12 or JAN-DEC
Day-of-week	1-7 or SUN-SAT
Year	1970-2199

Events → Rules → create rule → Name → ⊙ schedule → continue to create rule → Cron.

59	23	31	12	?	*
min	Hours	Day of month	month	Day of week	Year

* → every year

? → if you don't want to specify any value use it

*/01 → every one minute the action take place

01 → at every one hr at one minute

→ 2024 → it will execute in this year only

year

→ To refer the cron formats with examples

google → [cron expression reference]

15 → Run at 12:15 every day

0/15 → Run at every 15 minutes.

Aws Serverless (Lambda) :

→ It is a new paradigm in which developers don't have to manage servers anymore.

→ They just deploy code and deploy functions.

→ It means Function as a Service (FAAS)

→ Serverless does not mean there are no servers ... it means you just don't manage or provision hardware.

Traditional webhosting	Serverless webhosting
→ Provision capacity	→ Run On-Demand
→ How much Server capacity ? CPU, Memory, RAM	→ Unlimited capacity Pay for code execution
→ Scaling (Pay too much)	→ scales automatically (pay for what you use)

→ update OS + s/w

→ Prevent security issues

→ Runs on managed AWS infrastructure

→ code runs upto date and secure environment

→ Lambda is a compute service that lets you to run the code without provisioning or managing servers

→ You pay only for the compute time you consume

→ There is no charge when your code is not running.

→ It runs your code on high-availability compute infrastructure and performs all the administration of the compute resources

Language Support:

→ Node.js (JavaScript)

→ Python

→ Java

→ C#, (.NET Core)

→ Golang

→ C# / Powershell

Lambda function

→ The code you run on AWS Lambda is uploaded as a "Lambda function".

Note:- Lambda automatically creates default code for the function.

Advantages:-

→ Easy pricing.
→ Integrated with the whole AWS stack.
→ " " many programming languages.
→ Easy monitoring with cloudwatch
→ Increasing RAM will also improve CPU and n/w.
→ per function upto 3GB of RAM

Lambda limits:-

→ ~~Lambda~~

Execution :-

→ Memory allocation : 128mb - 3008mb (64mb increments)
→ Max execution time : 5 min

→ Disk capacity in the "function container" (in /tmp) : 512MB

→ Concurrency limits : 1000

DEPLOYMENT :-

→ Lambda function deployment size (compressed .zip) : 50MB

→ uncompressed deployment (code + dependencies) : 250MB

→ Can use the /tmp directory to load other files at startup

→ Size of environment variables : 4KB

Scenario :-

* Run the Prod-Server - from (Mon-Friday) and stop the server on Saturday and sunday, Automate this task with lambda function and using python :-

step-1 :- Create a policy using visual editor.

→ Go to IAM policies

→ click on visual editor

→ select EC2

→ under write → select stop instance and start instance

→ Resource → all → Next → policy name →

start_stop_instance → create policy.

step-2:- Create a new role with policy stop and start policy for Lambda

step-3:- Create Lambda functions for start instance

→ Go to Lambda

→ Create function

→ Name: start instance

We don't know python code for start instance

then, in google type

start instance lambda py

then copy the code and paste and make required changes

→ And create another lambda function for stop instance, copy the code and make

changes

→ Then deploy the functions.

step-IV :-

In cloudwatch event rules, create rules for the functions start and stop

→ under event → create ~~role~~ → schedule →

cron → stop instance

59	23	*	*	Fri	*
min	Hours	Day of month	month	Day of week	year

start instance

00	06	*	*	? (or) (MON, FRI)	*

VPC

→ It is a logically isolated n/w environment to launch AWS resources in to a virtual n/w.

→ It closely resembles like traditional n/w that you would in your own datacenter.

→ In AWS we have default VPC, it is a pre configured setup.

→ Each instance in the default VPC has Private ip and Public ip.

→ Default VPC CIDR block is 172.31.0.0/16

→ Private subnets are used in real time. we have to launch instances and databases on private subnet only.

CIDR :- Classless Inter Domain Routing

→ It is a set of Internet protocol (IP) standards that is used to create unique identifiers for n/w

and individual devices.

eg:- 10.10.0.0/16

Imp
VPC Flow logs

→ It capture information about the IP traffic going to and from n/w interfaces in your VPC.

→ Flow log data can be published to cloud-watch logs and S3.

Creation:-

→ Create one S3 bucket.

→ Now copy the bucket ARN

→ Click on Flowlogs → create flowlog.

→ Filter → all

→ Destination → Send to S3 bucket

→ Past S3 bucket ARN

→ Create.

SUBNET:-

→ It is a range of IP addresses in your VPC, typically a LAN

→ A subnet lives within a Single Availability Zone in VPC.

eg:- 10.10.10.0/24
10.10.15.0/24

→ For high availability we create subnet in different zones.

→ We can create flowlogs for each subnet also

Enable public ip for all the instances launched in Public subnet

→ Select Public subnet
→ Actions
→ Auto assign IP settings
☑ enable

Route table:-

→ It contain a set of rules, called route that are used to determine where n/w traffic is directed.
→ It direct n/w traffic between instances

inside a subnet.

→ It has two main components

(i) Destination: CIDR block range of target

(ii) Target: A name identifier of where the data is being routed to.

Internet Gate Way :- (IGW)

→ It is a n/w node that connects two different n/w that allows communication between instances in your VPC and the internet.

Egress - only Internet gateway :-

→ A stateful gateway to provide egress only access for IPv6 traffic from the VPC to the internet.

Scenario :-

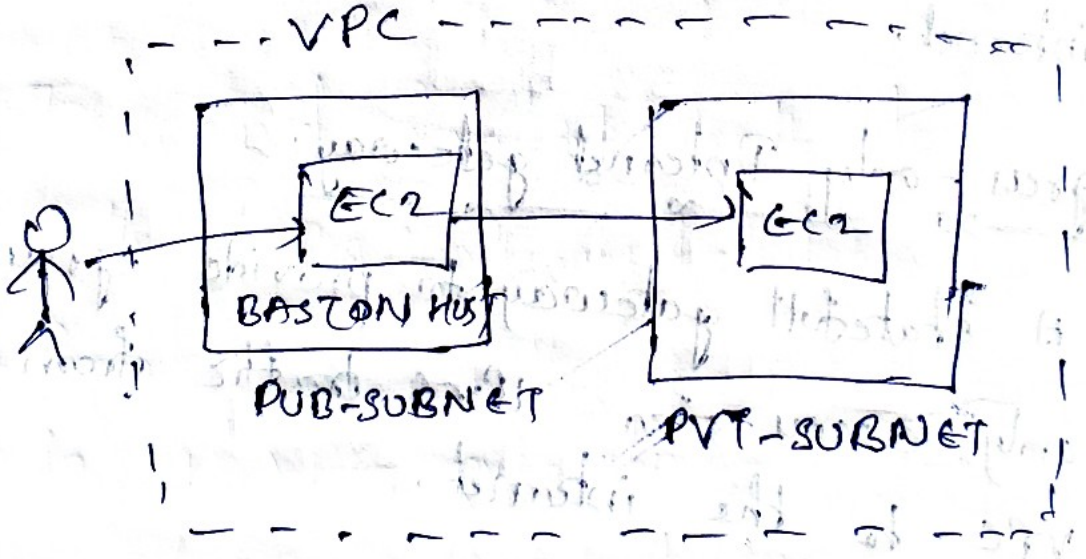
→ Launch instances in Public subnet and Private subnet and install httpd in private subnet instance.

BASTON HOST

→ Baston hosts are instances that sit within public subnet and are typically accessed using SSH or RDP.

→ Once remote connectivity has been established then it acts as a "jump" server within VPC.

→ It essentially acts as a bridge to your private instances via the internet.



→ To connect pvt instance from pub-instance then keep the key pair of pvt instance in public instance.

→ connect public instance

```
cat > mykey.pem
```

paste the key here and press one enter

ctrl + d → it will save and come out.

→ then change permissions as

```
chmod 400 mykey.pem
```

Now connect the pvt instance as

```
ssh -i mykey.pem ec2-user@pvt_ip
```

of pvt instance

→ To check whether the pvt server is working or not.

```
ping ip_address of pvt server
```

Enter

(*) To enable the ping communication, which protocol is used?

→ icmp

→ In pvt instance security group → allow icmp protocol

Internet gateway

→ These are not secure.

→ These provide internet to public subnet

Nat gateway

→ These are secure.

→ These provides internet to pvt subnets.

→ Create NAT gateway in public subnet and configure in pvt subnet route table,

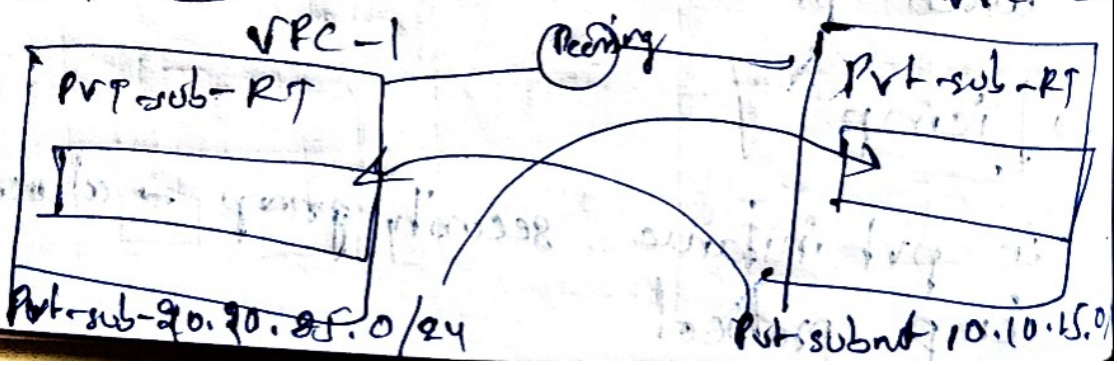
→ NAT gateways are chargeable.

(*) To connect from pvt-subnet-instance to another VPC pvt-subnet-instance.

→ Peering

→ In 1st VPC, pvt-subnet-RT → add second VPC pvt-subnet CIDR

→ And do vice versa



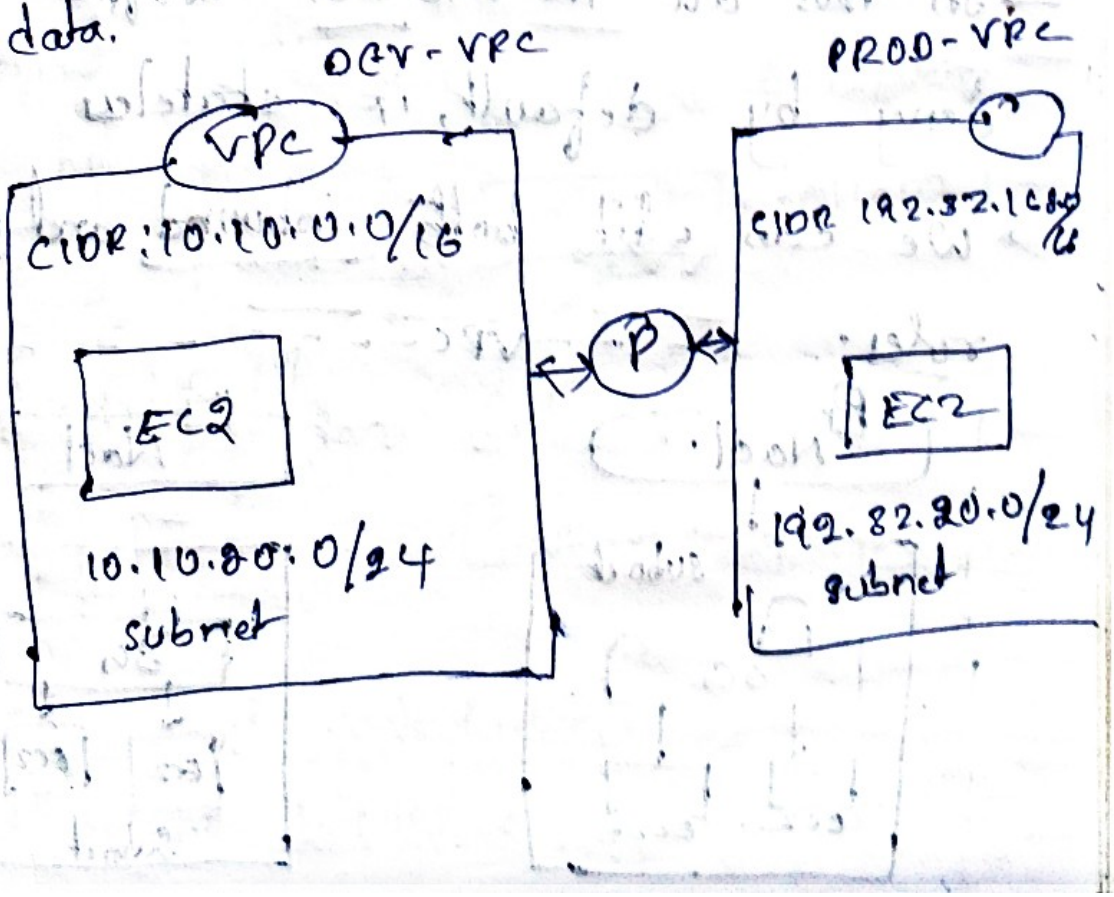
→ This will connect all the instances in that subnet

→ To connect only one instance, then keep their privat ip addresses in the routetables as 10.192.63.20.5/24 vice versa

Peering connection

→ It is a networking connection between two VPC's

→ It helps you to faciltate the transfer of data.



Entered ✓ ☆ VPC : SECURITY

Security Groups (SG's)

→ Security groups are act as virtual firewa for controlling traffic at the instance leve

→ These support only allow rules. ie stateful

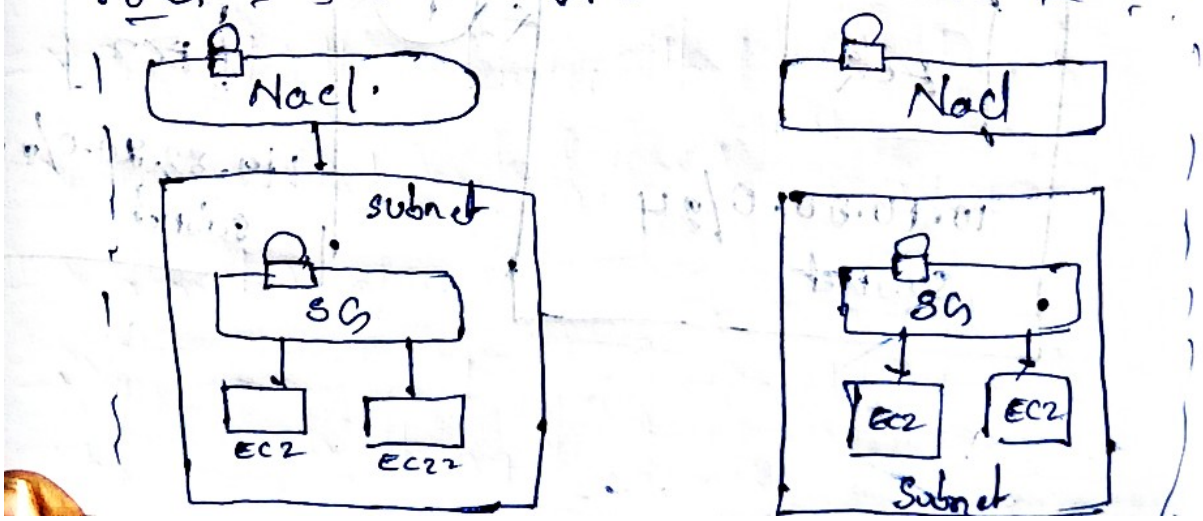
☆→ In this we can edit only inbound rul

Network Access Control list (NACLs)

→ These act as virtual firewalls for controlling the traffic at the subnet leve

→ In this all IN and OUT traffic is deny by default, ie stateless

→ We can edit both inbound and outbou rules.



Creation of Nacl

→ Left hand side → Security
→ Network ACL's
→ Create Network ACL.
→ Select VPC
→ Create
→ Now seled the Nacl
→ In subnet associations → edit
→ then attach it to the subnet.

To allow the traffic in the instance.

→ Select the Nacl
→ Inbound → edit

Rule no	Type	Source	Allow/Deny
101	SSH(22)	0.0.0.0/0	Allow

→ Outbound → edit

		Port range	
101		32768 - 65000	Allow

→ A n/w acces control list (ACL) allows or denies specific inbound or outbound traffic

at the subnet level.

→ Ephemeral port range of 32768 - 65535.

→ The range varies depending on the client OS.

→ Many Linux kernel use ports 32768 - 61000.

→ Elastic load balancer " " 1024 - 65535.

→ Window OS 2003 " " 1025 - 5000

→ Windows server 2008 " 49152 - 65535.

→ NAT gateway uses " 1024 - 65535.

→ Aws Lambda " " 1024 - 65535.

→ ACL acts as a two layer security.

→ SG " first layer "

VPN Connection

→ By default, instances that you launch into VPC can't communicate with your own (remote) n/w.

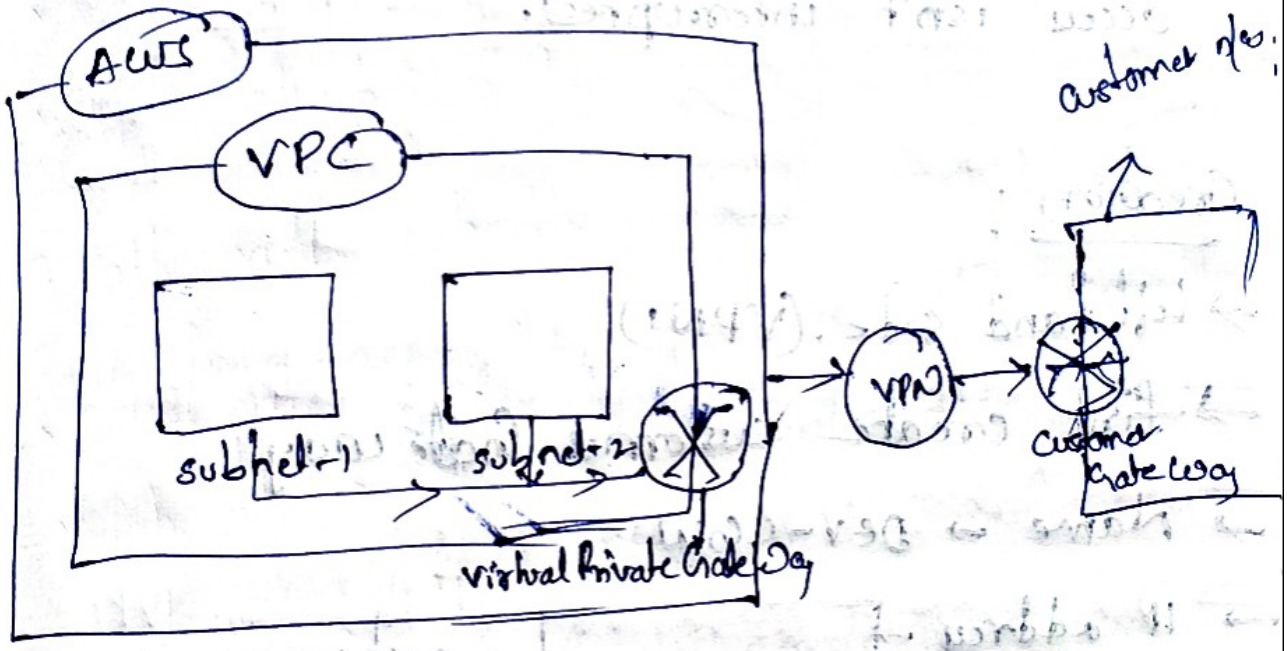
→ You can connect your VPC to remote n/w by using a VPN connection.

Customer Gateway :-

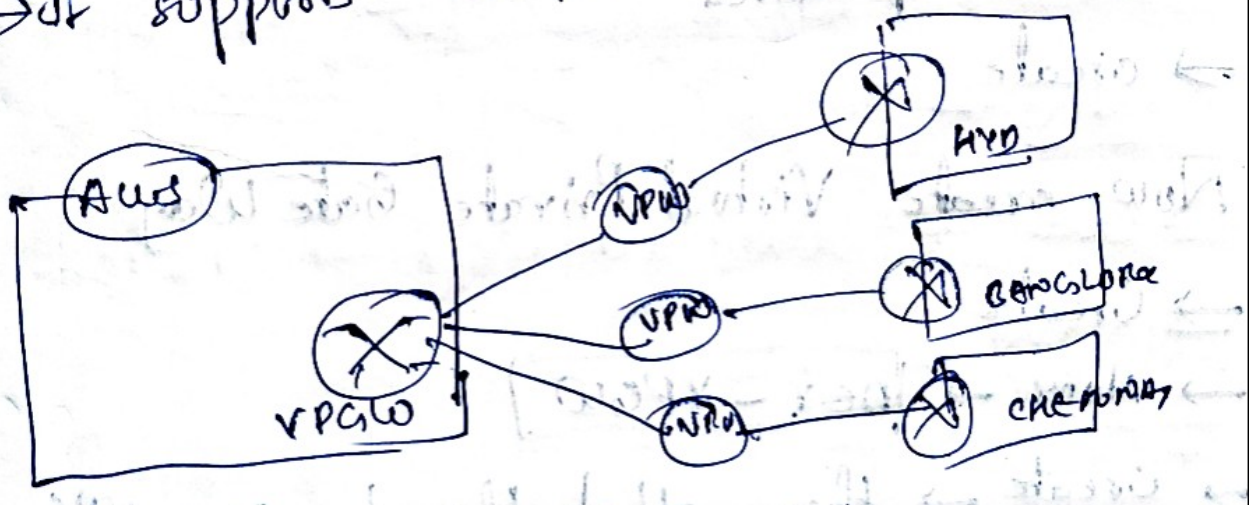
→ It is a physical device or s/w application on your side of the VPN connection.

Virtual Private Gateway :-

→ It is the VPN concentrator on the Amazon side of the site-to-site VPN connection



→ It supports multiple VPN connections



VPN Tunnel

→ VPN connection consists of two tunnels to provide increased availability for the VPC service.

→ If there is a device failure within AWS your VPN connection automatically fails over to the second tunnel so that your access isn't interrupted.

Creation:

→ Left hand side. (VPN's)

→ first create customer Gateway

→ Name → DEV-CGW

→ IP address → [ ] → customer side VPN device ip address

→ create

Now create Virtual Private Gate Way

→ Create

→ Name → DEV-VPGW

→ Create → then attach this to AWS VPC (DEV-VPC)

→ Select VPC(s) → Actions → Attach to VPC

Now create site-to-site VPN connection

→ Create VPN connection
→ Name → DEV-VPNC
→ Create.

Now download the VPN configuration

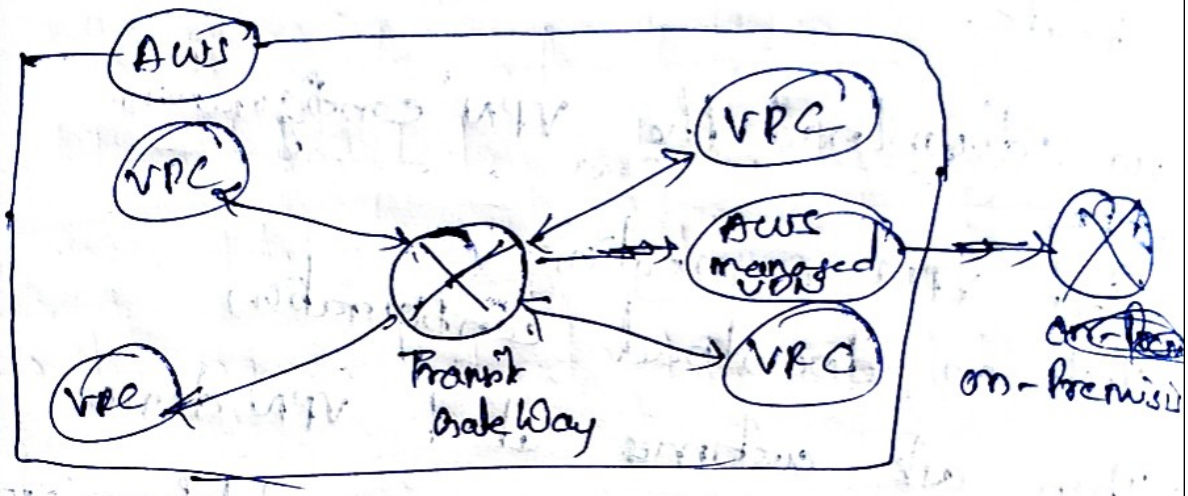
→ Select VPN connection
→ Click on download configuration
→ Then ask customer about VPN device company name, s/w version, platform etc and edit them → download.
→ Then mail that file to the customer
→ Then customer starts configuring this VPN device based on that file's information on.

TRANSIT GATEWAY

→ Transit gateway is a transit hub that you can use to interconnect your VPC

and on-premises n/w's to a single gate way'.

→ It acts as a hub that controls how traffic is routed among all the connected n/w's which act like spokes.



→ One VPN connection offers only one VPC only.

→ To connect more VPC's through VPN we use Transit Gate Way.

Creation:-

→ Click on transit gate way → create

→ Name [ ]

→ Create

Now transit Gate Way attachment

→ Create transit gateway attachment
→ Name [DEV-VPC]
→ Attachment type [▽]
→ select VPC
→ attachment

VPC Limits

→ 5 Elastic IP addresses
→ 5 Internet Gateways
→ 5 VPC's per region
→ 50 VPN connections per region
→ 50 customer gateways per region
→ 200 route tables per region
→ 500 Security groups per VPC
→ 50 rules per security group

Note:- It can be increased upon request.

RDS

→ RDS is a service that makes it easier to setup, operate and scale a relational database in the cloud.

→ It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups.

→ It organise stored data into tables

Feature:-

→ Automatic installation
→ Automated backup/recovery
→ Manage db administrative tasks
→ Automatic upgrades/patches
→ Automated log management
→ High scalability
→ Security
→ Monitor and view database metrics

RDS free tier

→ 750 hours of RDS usage in single-AZ for

db.t3.micro, db.t2.micro instances, every month for one year.

→ 20gb of database storage : any combination of general purpose (ssd) or Magnetic storage.

→ 20gb of backup storage.

★ Databases are running only on Pvt subnets

★ RDS supports only SSD's and Magnetic storage volumes.

★ We can connect the databases which are running on the pvt subnets through jump servers.

antaV

⊛ Once the database is created on VPC, we can't change the VPC name, subnets, and that db from one VPC to another VPC

→ We can modify the database but not the VPC.

MySql → mysql workbench

(or)

from instance install mysql client.

In linux servers

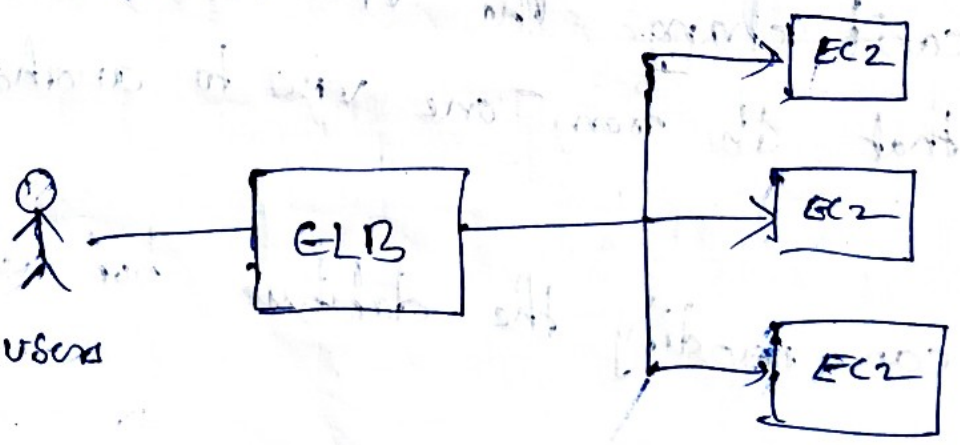
→ first install mysql

```
yum install -y mysql
```

To connect the db from instance

```
mysql -u username -h endpoint
-P 3306 -p
```

Elastic Load Balancer



→ It automatically distributes incoming application traffic across multiple EC2 instances. This increases the fault tolerance of your application.

→ A load balancer serves as the single point of contact for clients.

→ An ELB has its own DNS record set that allows for direct access from the open internet access.

Health checks! –

→ ELB will automatically stop serving traffic to an instance that becomes unhealthy.

→ They enable the load balancer to know if instances it forwards traffic to are available to reply to requests.

→ If the response is not 200 (OK), then the instance is unhealthy.

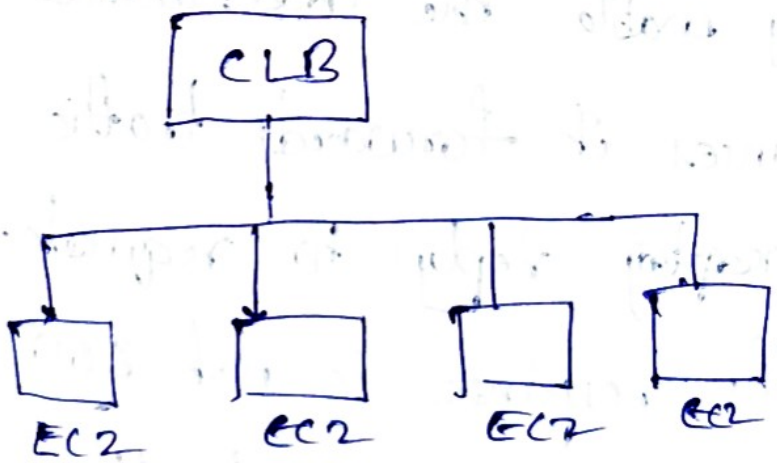
Types of ELB

1. Classic Load Balancer (CLB)
2. Application " (ALB)
3. Network " (NLB)
4. Gateway " (GWLB)

Classic Load Balancer (CLB)

→ A classic ELB is designed for simple balancing of traffic to multiple EC2 instances

☆ It is used when all the instances contain the same data



```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "Welcome to AWS" >> /var/www/html/index.html
echo "This is DEV!!" >> /var/www/html/dev.html
```

com test.html

Application Load Balancer (ALB)

→ It is designed for complex balancing of traffic to multiple EC2 instances using "Content-based" rules.

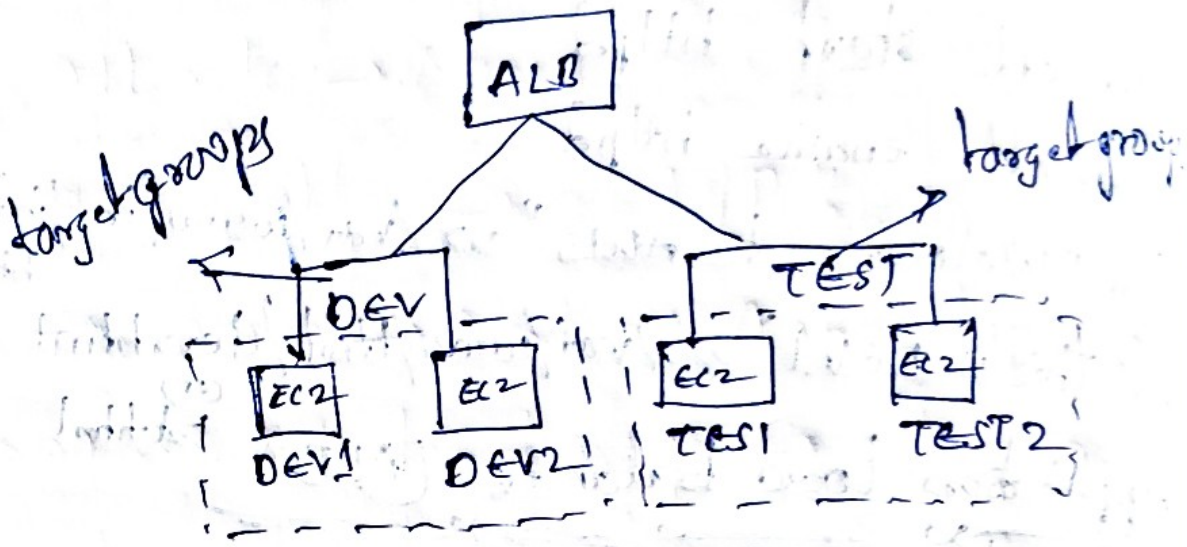
→ It is a seventh layer (Application Layer) of the Open System Interconnection (OSI) model

→ The traffic route based on the user i/p

→ Suppose I have 3 environments i.e DEV, TES, PROD, and all these environments has server with same application. If I want to access the DEV, then ALB's endpoint/Dev, Endpoint/Test ... etc.

→ These all are called "Target groups"

→ ALB works on target groups



Creation :-

→ Create 4 instances with different content in the instances (2 for DEV) (2 for TEST

→ Now create target groups DEV and TEST and add 2 servers in DEV target group and 2 servers in TEST target group.

→ Left hand side → under Load balancers → select Target Groups.

→ Name → DEV

→ Health check → Health check path

/index.html → Next

→ Register targets → select the DEV instance

→ click on "include as pending below" →

create target group.

→ And create TEST (target group) and follow above steps.

→ Now add these target groups in ALB

→ Create ALB

→ Name [myALB]

→ select availability zones

→ select security group.

→ Listeners and routing

Default action [TEST]

{ in this select the target group, if we select TEST, by default traffic routes to TEST target group only }

→ Create load balancer

→ It works only for TEST target groups and display that content only

→ Now I want to add DEV target group to ALB, then

→ In Listeners and rules → Manage rules →

Add rule → Name [DEV] → Next →

Add condition → select [Path] is [*dev]

→ confirm → Next → Action types →

Forward to target groups → select [DEV]

→ priority [1] (or use any value) →

Next → create.

Now access from browser

[endpoint/dev.html] it shows the content in dev server.

→ In this way we can access the content based on users request

eg:- www.flipkart.com/mobiles / clothes → target group

→ You can modify the default target gro

→ select that target group → Actions → edit rules →
select the target group name → save changes

Network Load Balancer (NLB)

→ It functions at the fourth layer (Transport layer) of the OSI model.
→ It is mostly used for "extreme performance"
→ It can handle "millions of requests per second"
→ It support for static IP / Elastic IP
→ Overall the creation process is same as the "Application Load Balancer"
→ Use it for "Prod environment"

GATEWAY Load Balancer (GWLB)

→ It is a third layer (n/w layer) of OSI model
→ It is used to deploy, scale and manage third-party virtual appliances such as security appliances (firewalls)

EC2 : Auto Scaling

→ It helps you to maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define.

→ It will increase & decrease the no. of instances based on chosen CloudWatch metrics.

Features and benefits :-

→ Maintain you EC2 instance availability

→ Automatically scale in and out

IntraV

(*) what is vertical scaling and horizontal scaling ?

Vertical scaling :-

→ Increase instance size (scale up/down)

From : t2.nano - 0.5G of RAM, 1 VCPU

To : t2.medium - 4G of RAM, 2 VCPUs

Horizontal Scaling :-

→ Increase no. of instances (scale out/in)
→ Autoscaling group.
→ Load Balancer.

High availability :-

→ Run instances for the same application across multi-AZ
→ Autoscaling Group multi-AZ
→ Load Balancer multi-AZ

→ To increase the CPU utilization,

```
yes >>/dev/null &
```

Route 53

→ It is a domain management service (Cons hashing solution)
→ It is a system that translates IP-address into Human-Readable Domain Names

→ A reliable and cost effective way to route end users to internet applications.

Advantages:-

→ It provides high available and scalable DNS, domain name registration and health-checking web services.

→ It is commonly used with an ELB to direct traffic from the domain to the ELB.

Key features:-

① Domain Registration:-

→ It is an organisation that manages the reservation of internet domain names.

② Domain Name System Service

→ It translates friendly domain names like www.cloudkubes.com into numeric IP like 192.168.10.200.

(3) Health Checks:-

→ It sends automated requests over the internet to your application to verify that's it reachable, available and functional.

(4) Traffic Flow:-

→ Easy to use and cost effective global traffic management;

→ Route end users to the best endpoint for your application based on geoproximity, latency, health and other considerations.

Hosted Zones:-

→ A hosted zone stores DNS records for your Domain.

→ Basically, it contains all the rules (Record sets) that tells Route53 what do to with DNS request.

→ There are both Public and Private hosted zones

Record Sets :-

① A (Address) : Point a domain to an IPV4 address

② AAAA (IPV6 Address) : Point a domain to an IPV6 address.

③ CNAME (Canonical Name) : Translate one domain name to another.

④ MX (Mail Exchange) : Route Email.

⑤ NS (Name Server) : Nodes that hold information about a given name.

⑥ PTR (Pointer) : Reverse DNS records (opposite to what A record does)

⑦ SOA (start of Authority) : Manager of DNS. Overall information about domain.

⑧ TTL (Time to live) : Storage time for cached results

Routing Policies :-

① Simple Routing :- Route all traffic to one endpoint.

② Weighted Routing :- Route traffic to multiple endpoints (manual load Balancing)

③ Latency :- Route traffic to an endpoint based on the user's latency to various endpoints.

④ Failover :- Route traffic to a "Secondary" endpoint if the "Primary" endpoint is unavailable.

⑤ Geolocation :- Route traffic to an endpoint based on the geographical location of the user.

⑥ Multivalue Answer :- Route traffic to DNS queries with up to eight healthy records selected at random.

⑦ Geoproximity :- Route traffic based on the location of your resources.

☆ → To know the ip address of any website the command is

nslookup website_name	→ universal

eg:- nslookup www.facebook.com

→ dig command is also used

dig www.facebook.com	→ only for unix platforms

Elastic Beanstalk

→ It makes it even easier for developers to quickly deploy and manage applications.

→ Just the application code is the responsibility of the developer.

Three architecture models!

→ Single instance deployment: Good for Dev

→ LB + ASG: Great for production or pre-production web applications.

→ AWS only :- Great for non-web apps in production (workers, etc).

Beanstalk supported platforms

→ Go, Java with tomcat, Node.js, Python, Packer builder, multicontainer Docker, Java SE, .Net on windows Server with IIS, PHP, Ruby, Single container Docker, Preconfigured docker.

Note :- If not supported, you can write your custom platform (advanced).

reading :-

→ first create a role (attach ElasticBeanstalk webTier, ElasticBeanstalk WorkerTier)

→ Now create ElasticBeanstalk → webserver env

→ Create.

Cloudformation

→ Cloudformation is the pure definition of infrastructure as code.

→ It helps you model and setup AWS resources, so that you can spend less time managing the resources and more time focusing on your application that run on AWS.

→ For creating templates use a JSON or YAML script.

Advantages :-

→ Simplify infrastructure management.

→ Quickly replicate your infrastructure.

→ Easily control and track changes to your infrastructure.

Components :-

i) Templates
ii) Stacks
iii) Stackset
iv) Cloudformer.

(i) Template :- It is JSON/YAML formatted text file that describes your AWS infrastructure.

→ It is like blueprint for building your AWS resources.

→ It includes several major sections.

- Format version
- Description
- Metadata
- Parameters
- Rules
- Mappings
- Conditions
- Transform
- Resources
- Output

(ii) Stack :-

→ It is the implementation of your template

eg:- Creating S3 bucket code template

```
Resources:
    MyS3bucket:
        Type: "AWS::S3::Bucket"
        Properties: { }
```

eg:- To give public access & change bucket name

Resources:

MyS3bucket:

Type: "AWS::S3::Bucket"

Properties:

AccessControl: PublicRead

BucketName: "my-bucket-123"

...

step-2:-

→ Navigate to cloudformation

→ Create stack.

→ Choose an exiting template.

→ Give stackname

→ Next.

→ submit.

Note:- If you delete the template it will delete all the infrastructure created by that template.

→ Template code is available in google and modify according to your requirem

(iii) Stacksets :-

→ It is a container for Aws cloudformation stacks that lets you provision stacks across Aws accounts and regions by using single template.

→ You can create infrastructure in multiple accounts, multiple regions at a time.

Creation :-

→ left hand side click on stacksets

→ upload your template.

→ Deployment locations

⊙ Deploy stacks in accounts ⊙ Deploy stacks in organisational unit

Account numbers

A32U5G21, 7U23U12

...etc.

→ Next.

Cloud formed :-

→ It is a tool for creating templates Aws

cloud templates from existing resources in your AWS account.

→ That means it capture and redeploy applications you already have running.

→ It creates a template in S3 bucket.

Cloudformation designer

→ AWS Cloudformation Designer is a graphic tool for creating, viewing and modifying AWS Cloudformation templates.

→ You can diagram your template resources using a drag and drop interface and edit details using the integrated JSON or YAML editor.

Note :- Now it is service in AWS