RDP – remote desktop password

)[POW1iIk=@u$J.JY.Uc)H6DEfju;sX$@](mailto:POW1iIk=@u$J.JY.Uc)H6DEfju;sX$@)

P&9H0xm)SsD$&9i@L;vkolSh1ATV0&W1

AWS EC2 Linux launch and connect using ssh

1- launce linux instance

2- AWS console -> EC2 -> Instance -> launce instance

3- Name and tag 🡪linuxServer1 -> app and OS image 🡪 select Ubuntu

4- Instance type – t2.micro

5- select key pair or create new pair

6- Network setting 🡪 Default VPC -> default subnet -> auto assign public ip🡪 enable

7- Security group -> create security group -> name -> linuxServerSG -> add security group role 🡪ssh,HTTP,HTTPS -> anywhere all

8- add storage 30GB

9 – launce instance

10 – go to chrome and search for puttygen and download

21- go to chrome and search for putty and download

* Go to download -> click on puttygen -> first load and select all file (\*All ) -> select .pem file(key pair) -> save private key (.ppk file)
* Go to download-> install and open putty -> ssh -> auth -> select private (.ppk file) -> session -> hostname-> instance public-IP
* Login as -> ec2-user

Launce and instance with Ubuntu and login with command prompt

**Re**trieve metadata of amazon linux machine

1. Command prompt-> cd Download -> ssh -i "nayak-vm.pem" [ubuntu@ec2-18-188-74-79.us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-18-188-74-79.us-east-2.compute.amazonaws.com)

2 curl http://18.188.74.79/latest/meta-data/

3 curl http://69.254.169.254/latest/meta-data/

4 curl http://169.254.169.254/latest/meta-data/

5 curl http://169.254.169.254/latest/meta-data/ami-id

6 curl http://169.254.169.254/latest/meta-data/ami-launch-index

7 curl http://169.254.169.254/latest/meta-data/ami-manifest-path

8 curl http://169.254.169.254/latest/meta-data/block-device-mapping/

9 curl http://169.254.169.254/latest/meta-data/block-device-mapping/ami

10 curl http://169.254.169.254/latest/meta-data/block-device-mapping/ephemera10

11 curl http://169.254.169.254/latest/meta-data/events/

12 curl http://169.254.169.254/latest/meta-data/events/maintenance/

13 curl http://169.254.169.254/latest/meta-data/events/maintenance/history

14 curl http://169.254.169.254/latest/meta-data/hostname

15 curl http://169.254.169.254/latest/meta-data/identity-credentials/

16 curl http://169.254.169.254/latest/meta-data/identity-credentials/ec2/

17 curl http://169.254.169.254/latest/meta-data/identity-credentials/ec2/info

18 curl http://169.254.169.254/latest/meta-data/instance-action

19 curl http://169.254.169.254/latest/meta-data/instance-id

20 curl http://169.254.169.254/latest/meta-data/instance-life-cycle

21 curl http://169.254.169.254/latest/meta-data/instance-type

22 curl http://169.254.169.254/latest/meta-data/local-hostname

23 curl http://169.254.169.254/latest/meta-data/local-ipv4

24 curl http://169.254.169.254/latest/meta-data/profile

25 curl http://169.254.169.254/latest/meta-data/public-hostname

26 curl http://169.254.169.254/latest/meta-data/public-ipv4

27 curl <http://169.254.169.254/latest/meta-data/security-groups>

**How To Install the Apache Web Server on Ubuntu 20.04**

* **sudo apt update**
* **sudo apt install apache2**
* **Adjusting the firewall ->** Before testing Apache, it’s necessary to modify the firewall settings to allow outside access to the default web ports. Assuming that you followed the instructions in the prerequisites, you should have a UFW firewall configured to restrict access to your server.

During installation, Apache registers itself with UFW to provide a few application profiles that can be used to enable or disable access to Apache through the firewall.

List the ufw application profiles by typing

**sudo ufw app list**

As indicated by the output, there are three profiles available for Apache:

**Apache**: This profile opens only port 80 (normal, unencrypted web traffic)

**Apache Full**: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic)

**Apache Secure**: This profile opens only port 443 (TLS/SSL encrypted traffic)

It is recommended that you enable the most restrictive profile that will still allow the traffic you’ve configured. Since we haven’t configured SSL for our server yet in this guide, we will only need to allow traffic on port 80:

**sudo ufw allow 'Apache'**

* **sudo ufw status -> verify change**
* **sudo systemctl status apache2 -> checking web server**

As confirmed by this output, the service has started successfully. However, the best way to test this is to request a page from Apache.

You can access the default Apache landing page to confirm that the software is running properly through your IP address. If you do not know your server’s IP address, you can get it a few different ways from the command line.

Try typing this at your server’s command prompt:

* **hostname –I**

You will get back a few addresses separated by spaces. You can try each in your web browser to determine if they work.

Another option is to use the Icanhazip tool, which should give you your public IP address as read from another location on the internet:

* **curl -4 icanhazip.com**
* [**http://your\_server\_ip**](http://your_server_ip) **->** You should see the default Ubuntu 20.04 Apache web page:
* **sudo systemctl stop apache2 ->** To stop your web server
* **sudo systemctl start apache2 ->** To start the web server
* **sudo systemctl restart apache2 ->** To stop and then start
* **sudo systemctl reload apache2 ->** Apache can often reload without dropping connections
* **sudo systemctl disable apache2 ->** behavior by typing
* **sudo systemctl enable apache2 ->** To re-enable the service to start up at boot
* **Setting up virtual hosts :-**
* **sudo mkdir /var/www/your\_domain ->** Create the directory for **your\_domain**
* **sudo chown -R $USER:$USER /var/www/your\_domain ->** assign ownership of the directory with the $USER environment variable
* The permissions of your web roots should be correct if you haven’t modified your umask value, which sets default file permissions. To ensure that your permissions are correct and allow the owner to read, write, and execute the files while granting only read and execute permissions to groups and others, you can input the following command:

**sudo chmod -R 755 /var/www/your\_domain**

* **sudo nano /var/www/your\_domain/index.html ->** create a sample index.html page using nano or your favorite editor:

**<html>**

**<head>**

**<title>Welcome to Your\_domain!</title>**

**</head>**

**<body>**

**<h1>Success! The your\_domain virtual host is working!</h1>**

**</body>**

**</html>**

* In order for Apache to serve this content, it’s necessary to create a virtual host file with the correct directives. Instead of modifying the default configuration file located at /etc/apache2/sites-available/000-default.conf directly, let’s make a new one at /etc/apache2/sites-available/your\_domain.conf:

**sudo nano /etc/apache2/sites-available/your\_domain.conf**

<VirtualHost \*:80>

ServerAdmin webmaster@localhost

ServerName your\_domain

ServerAlias www.your\_domain

DocumentRoot /var/www/your\_domain

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Notice that we’ve updated the DocumentRoot to our new directory and ServerAdmin to an email that the **your\_domain** site administrator can access. We’ve also added two directives: ServerName, which establishes the base domain that should match for this virtual host definition, and ServerAlias, which defines further names that should match as if they were the base name.

Save and close the file when you are finished.

* **sudo a2ensite your\_domain.conf -**> Let’s enable the file with the a2ensite tool:
* **sudo a2dissite 000-default.conf ->** Disable the default site defined in 000-default.conf
* **sudo apache2ctl configtest ->** let’s test for configuration errors

You should receive the following output: Syntax OK.

* **sudo systemctl restart apache2**

Apache should now be serving your domain name. You can test this by navigating to http://your\_domain, where you should see something like this:

**VPC ( virtual private cloud )**

Sequence : VPC 🡪 Subnet 🡪 Internet Gateway 🡪 route tables

1. VPC 🡪create VPC 🡪 select VPC only 🡪 name – my-test-VPC 🡪 IPv4 CIDR manual input 🡪 IPv4 CIDR – 10.0.0.0/16 🡪 No IPv6 CIDR Block 🡪 tenancy –default 🡪 create VPC

10.0.0.0/26 🡪 VPC CIDR

32 – 26 = 6; 2^6 = 64; 64/4 = 16 -> 2^4 (need to borrow 4bit) 32-4 = 28 subnet mask

10.0.0.0/28 🡪 10.0.0.15/28 10.0.0.16/28 🡪 10.0.0.31/28

10.0.0.32/28 🡪10.0.0.47/28 10.0.0.48/28 🡪 0.0.0.63/28

(10.0.0.0/24 🡪 32-24=8; 2^8 = 256)

1. Creating subnet1 🡪 subnet name - testPublicSubnet-1a 🡪 AZ – us-east -2a 🡪 IPv4 VPC CIDR Block – 10.0.0.0/16 🡪 IPv4 VPC subnet Block – 10.0.0.0/28 🡪create subnet.
2. Creating subnet2 🡪 subnet name - testPublicSubnet-2b 🡪 AZ – us-east -2b 🡪 IPv4 VPC CIDR Block – 10.0.0.0/16 🡪 IPv4 VPC subnet Block – 10.0.0.16/28 🡪create subnet.
3. Creating subnet3 🡪 subnet name - testPrivateSubnet-1a 🡪 AZ – us-east -2a 🡪 IPv4 VPC CIDR Block – 10.0.0.0/16 🡪 IPv4 VPC subnet Block – 10.0.0.32/28 🡪create subnet.
4. Creating subnet3 🡪 subnet name - testPrivateSubnet-2b 🡪 AZ – us-east -2b 🡪 IPv4 VPC CIDR Block – 10.0.0.0/16 🡪 IPv4 VPC subnet Block – 10.0.0.48/28 🡪create subnet.
5. Create Internet Gateway 🡪Tag Name – test-VPC-IGW 🡪 create internet gateway 🡪 Go to created internet gateway 🡪 click on select and click action – attach to VPC – my-test-VPC1🡪 attach IGW.
6. Route table 🡪 ( by default VPC creates a rout table ) 🡪 change name – testPublicRT 🡪 save 🡪 refresh 🡪

Adding subnets:- go to testPublicRT 🡪 Subnet association 🡪 edit subnet association 🡪 add testPublicSubnet-1a and testPublicSubnet-2b 🡪 Save association.

Adding routes :- go to testPublicRT 🡪 Routes 🡪 edit routes 🡪 add routes 🡪 0.0.0.0/0 (access to internet from anywhere) – select Internet Gateway – select test-VPC-IGW 🡪 save changes.

1. Create route table 🡪 name – testPrivateRT 🡪 VPC – my-test-VPC1 🡪 Create Route table.

Go to testPrivateRT 🡪 Adding subnets:- go to testPrivateRT 🡪 Subnet association 🡪 edit subnet association 🡪 add testPrivateSubnet-1a and testPrivateSubnet-2b 🡪 Save association.

1. NAT Gateway 🡪 create NAT Gateway 🡪 name – testNATGateway -> subnet – testPublicSubnet-1a -> connectivity type – public -> Allocate Elastic IP 🡪 create NAT Gateway. (it take 1min).
2. Go to route table 🡪 testPrivateRT 🡪 Route 🡪 Routes 🡪 edit routes 🡪 add routes 🡪 0.0.0.0/0 – select NAT Gateway – testNATGateway 🡪 save changes.
3. VPC 🡪 Action 🡪 edit VPC Settings 🡪 DNS Hostname & DNS resolution – enable -> to assign the pubic hostname & public IP . 🡪 Save changes.

**Launce Public and Private Instance**

=B@d@IWy%F@$@Bhh3C7suax&KUR4md)Q

**PUBLIC INSTANCE**

1. Launce Instance 🡪 Name -> ec2-public
2. OS -> Ubuntu
3. Instance Type –> t2.micro
4. key pair -> test-VPC
5. Networking 🡪 VPC -> my-test-VPC1
6. subnet -> testPublicSubnet- 1a
7. Auto-assign public IP -> enable
8. Create security group -> Name – ec2-publicSG-> Inbound security group rules -> add – SSH & HTTP & HTTPS ( everywhere 0.0.0.0/0 )
9. Configure storage 🡪 10GB
10. Launce instance

**PRIVATE INSTANCE**

1. Launce Instance 🡪 Name -> ec2-private
2. OS -> Ubuntu
3. Instance Type –> t2.micro
4. key pair -> test-VPC
5. Networking 🡪 VPC -> my-test-VPC1
6. subnet -> testPrivateSubnet- 1a
7. Auto-assign public IP -> disable
8. Create security group -> Name – ec2-privateSG-> Inbound security group rules -> add – SSH ( everywhere 0.0.0.0/0 )
9. Configure storage 🡪 10GB
10. Launce instance

**Internet Access to Private from public Instances or subnets**

1. Public Instance login 🡪 cmd 🡪 cd Downloads 🡪 ssh -i "nayak-vm.pem" [ubuntu@ec2-18-224-33-192.us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-18-224-33-192.us-east-2.compute.amazonaws.com)
2. sudo su
3. ec2metadata
4. [www.google.com](http://www.google.com) – to check internet access
5. to access internet in ec2Privae instance 🡪.pem file of ec2Private instance must be in ec2Public instance
6. log out from ec2public instance
7. login to folder where .pem file located. ( cd Downloads )
8. to copy file from local server to remote server both used same .pem file
9. scp -i nayak-vm.pem nayak-vm.pem [ubuntu@18.224.33.192:/home/ubuntu/](mailto:ubuntu@18.224.33.192:/home/ubuntu/)
10. public instance login 🡪 ssh -i "nayak-vm.pem" [ubuntu@ec2-18-224-33-192.us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-18-224-33-192.us-east-2.compute.amazonaws.com)
11. sudo su
12. ssh -i "nayak-vm.pem" [ubuntu@10.0.0.21](mailto:ubuntu@10.0.0.21)
13. ec2metadata
14. ping www.google.com

**VPC Peering**

VPC-Ohio 🡨🡪 VPC-Mumbai

VPC-Ohio

1. Select Ohio region 🡪 create VPC
2. Name – VPC-Ohio -> CIDR Block – 10.0.0.0/16 -> create VPC
3. Action -> Edit VPC settings -> DNS Hostname- enable -> save changes.
4. Create subnet 🡪 name- subnet-Ohio1 –> subnet CIDR block -> 10.0.0.0/24 -> create subnet .
5. Create Internet Gateway 🡪 name – igw-vpc-ohio -> create -> action -> attach VPC -> save.
6. Create Route Table 🡪 name - vpc-ohio-rt 🡪 VPC – VPC-Ohio -> create route table ->

go to vpc-ohio-rt 🡪 subnet association -> edit subnet association -> add subnet-Ohio1 -> save Association.

1. Edit routes 🡪 add routes -> 0.0.0.0/0, internet gateway – igw-vpc-ohio -> save changes.
2. GO to Perring connection 🡪 create Perring Connection 🡪 name – VPC-Ohio-Mumbai -> VPC ID (requester) – VPC-Ohio -> my account -> region – another region ->VPC ID (accepter) - vpc-0569883e85303dbc9 ( VPC-Mumbai ID ) 🡪 create VPC peering.
3. Go to Route table 🡪 edit 🡪 add route 🡪 192.168.0.0/16 (VPC-Mumbai) -> peering connection -> VPC-Ohio-Mumbai 🡪 save.
4. Launch Instance 🡪 Name - EC2-Peering-Ohio 🡪 OS – Ubuntu 🡪 storage type – t2.micro 🡪 VPC – VPC-Ohio 🡪 Security group name – Peering-SG 🡪 port – SSH, HTTP, HTTPS(anywhere) 🡪 launch Instance
5. CMD login 🡪

VPC-Mumbai

1. Select Mumbai region 🡪 create VPC
2. Name – VPC-Mumbai -> CIDR Block – 192.168.0.0/16 -> create VPC
3. Action -> Edit VPC settings -> DNS Hostname- enable -> save changes.
4. Create subnet 🡪 name- subnet-Mumbai1 –> subnet CIDR block -> 192.168.0.0/24 -> create subnet .
5. Create Internet Gateway 🡪 name – igw-vpc-mumbai -> create -> action -> attach VPC -> save.
6. Create Route Table 🡪 name - vpc-mumbai-rt 🡪 VPC – VPC-mumbai -> create route table ->

go to vpc-mumbai-rt 🡪 subnet association -> edit subnet association -> add subnet-Mumbai1 -> save Association.

1. Edit routes 🡪 add routes -> 0.0.0.0/0, internet gateway – igw-vpc-Mumbai -> save changes.
2. VPC Peering connection 🡪 Action 🡪 accept request.
3. Go to Route table 🡪 edit 🡪 add route 🡪 10.0.0.0/16 (VPC-Ohio) -> peering connection -> VPC-Ohio-Mumbai 🡪 save.
4. Launch Instance 🡪 Name - EC2-Peering-Mumbai 🡪 OS – Ubuntu 🡪 storage type – t2.micro 🡪 VPC – VPC-Mumbai 🡪 Security group name – Peering-SG 🡪 port – SSH, HTTP, HTTPS(anywhere) 🡪 launch Instance

**NACL**

1. create VPC
2. Name – VPC-1 -> CIDR Block – 192.0.0.0/16 -> create VPC
3. Action -> Edit VPC settings -> DNS Hostname- enable -> save changes.
4. Create subnet 🡪 name- vpc1-Subnet –> subnet CIDR block -> 192.0.0.0/24 -> create subnet .
5. Create Internet Gateway 🡪 name – VPC-1-IGW -> create -> action -> attach VPC -> save.
6. Create Route Table 🡪 name - NACL-RT 🡪 VPC – VPC-1 -> create route table ->

go to NACL-RT 🡪 subnet association -> edit subnet association -> add VPC1-subnet-> save Association.

1. Edit routes 🡪 add routes -> 0.0.0.0/0, internet gateway – vpc1-igw -> save changes.
2. Create NACL 🡪 Name- VPC-1-NACL -> VPC- VPC-1 🡪 Create NACL 🡪 Go to VPC-1-NACL -> Subnet Association -> edit subnet association -> select – vpc1-subnet 🡪 save changes.

Edit Inbound rules 🡪 add new rule 🡪 Rule no -100 , type –SSH(22), Allow/deny- Allow 🡪 Save Changes.

Edit Outbound rules 🡪 add new rule 🡪 Rule no -100 , type –HTTP(80), Rule no -200 HTTPS(443), Allow/deny- Allow 🡪 Save Changes

1. Launch Instance 🡪 Name - EC2-NACL 🡪 OS – Ubuntu 🡪 storage type – t2.micro 🡪 VPC – VPC-1 🡪 Security group name – NACL-SG 🡪 port – SSH, HTTP, HTTPS(anywhere) 🡪 launch Instance.
2. Login to Instance 🡪 not having access to Internet.
3. Edit Inbound rules 🡪 add new rule 🡪 Rule no -200,type- custom TCP port range 1024-65535 , Allow/deny- Allow 🡪 Save Changes.
4. Login to Instance 🡪 [www.google.com](http://www.google.com) -> having access to Internet.

**VPC Endpoint**

1. Create a VPC 🡪 name- MyVPC -> CIDR Block – 10.0.0.0/16 -> create VPC
2. Action -> Edit VPC settings -> DNS Hostname- enable -> save changes.
3. Create subnet 🡪 name- Public-Subnet –> subnet CIDR block -> 10.0.0.0/28 -> create subnet .
4. Create subnet 🡪 name- Private-Subnet –> subnet CIDR block -> 10.0.0.32/28 -> create subnet .
5. Create Internet Gateway 🡪 name – myVPC-IGW -> create -> action -> attach VPC -> save.
6. Create Route Table 🡪 name - myVPC-Public-RT 🡪 VPC – myVPC-> create route table ->

go to myVPC-Public-RT 🡪 subnet association -> edit subnet association -> add Public-Subnet Public-Subnet -> save Association.

1. Edit routes 🡪 add routes -> 0.0.0.0/0, internet gateway – myVPC-IGW -> save changes.
2. Create Route Table 🡪 name- myVPC-Private-RT🡪 VPC – myVPC -> create route table ->

go to myVPC-Private-RT 🡪 subnet association -> edit subnet association -> add Private-Subnet Private-Subnet -> save Association.

1. Create Endpoints 🡪 Name- myVPC-Endpoint -> Service Category – AWS services -> Service – S3 (Gateway) -> VPC – myVPC -> Route table – myVPC-Private-RT –> policy – full access 🡪 Create Endpoint .
2. Launch Instance 🡪 name- ec2-public -> OS – amazon linux -> t2.micro -> myVPC -> Public-Subnet -> security group – Endpoint-SG -> Launch instance.
3. Launch Instance 🡪 name- ec2-private -> OS – amazon linux -> t2.micro -> myVPC -> Private-Subnet -> security group – Endpoint-SG -> Launch instance.
4. Login to ec2-public -> cd Downloads 🡪 ssh -i "nayak-vm.pem" [ec2-user@ec2-3-14-254-196.us-east-2.compute.amazonaws.com](mailto:ec2-user@ec2-3-14-254-196.us-east-2.compute.amazonaws.com) 🡪 exit
5. Move .pem file of ec2-private to remote public server
6. To copy file from local server to remote server 🡪 scp -i nayak-vm.pem nayak-vm.pem [ec2-user@3.14.254.196:/home/ec2-user/](mailto:ec2-user@3.14.254.196:/home/ec2-user/)
7. Login to ec2-public -> cd Downloads 🡪 ssh -i "nayak-vm.pem" [ec2-user@ec2-3-14-254-196.us-east-2.compute.amazonaws.com](mailto:ec2-user@ec2-3-14-254-196.us-east-2.compute.amazonaws.com) 🡪 sudo su
8. ls -> see nayak-vm.pem file copied.
9. Login ec2-private from ec2-public -> ssh -i "nayak-vm.pem" [ec2-user@10.0.0.39](mailto:ec2-user@10.0.0.39)
10. Sudo su
11. aws configure
12. Go to your AWS Account -> Security credential -> create access key and download file -> open file -> access key - AKIAZI2LD36U4EX7LGAT

secret- key - 5gT62mbnisSDcBf29A4lh0PdDYIFThOIOBVuAv98

1. aws configure
2. access key - AKIAZI2LD36U4EX7LGAT
3. secret- key - 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
4. Default region name [None]: enter
5. Default output format [None]: enter
6. aws s3 ls
7. aws s3 mb s3://yogesh1137 🡪 make bucket
8. aws s3 ls 🡪 also go to AWS account and check yogesh1137 S3 bucket created
9. aws s3 ls s3://yogesh1137 🡪 Access to AWS Service
10. aws s3 rb s3://yogesh1137

remove\_bucket failed: s3://yogesh1137 An error occurred (BucketNotEmpty) when calling the DeleteBucket operation: The bucket you tried to delete is not empty

1. go to AWS S3 and delete files in yogesh1137.
2. aws s3 rb s3://yogesh1137
3. Go to EndPoint and Delete it.
4. aws s3 mb s3://rathod1137 🡪 not able to make bucket -> this is how endpoint works and how you can access the AWS services with the private subnet.

**VPN**

1. Create a VPC 🡪 name- MyVPC -> CIDR Block – 10.0.0.0/16 -> create VPC
2. Action -> Edit VPC settings -> DNS Hostname- enable -> save changes.
3. Create subnet 🡪 name- Public-Subnet –> subnet CIDR block -> 10.0.0.0/28 -> create subnet .
4. Create subnet 🡪 name- Private-Subnet –> subnet CIDR block -> 10.0.0.32/28 -> create subnet .
5. Create Internet Gateway 🡪 name – VPN-IGW -> create -> action -> attach VPC -> save.
6. Create Route Table 🡪 name - VPN-Public-RT 🡪 VPC – myVPC-> create route table ->

go to VPN-Public-RT 🡪 subnet association -> edit subnet association -> add Public-Subnet Public-Subnet -> save Association.

1. Edit routes 🡪 add routes -> 0.0.0.0/0, internet gateway – VPN-IGW -> save changes.
2. Create Route Table 🡪 name- VPN-Private-RT🡪 VPC – myVPC -> create route table ->

go to VPN-Private-RT 🡪 subnet association -> edit subnet association -> add Private-Subnet Private-Subnet -> save Association.

1. Launch PublicEC2 🡪 name – PublicEC2 -> Browse more AMIs -> AWS Marketplace -> OpenVPN Access Server -> instance type – select t2.micro (free tier select) -> VPC – myVPC -> subnet -> Public-Subnet -> security group Name - OpenVPN Access Server , port- default ->launch instance
2. PublicEC2 login 🡪 Open puttyGen 🡪 load – nayak-vm.pem 🡪 save private key – nayakkk 🡪 open Putty 🡪 SSH 🡪 Auth 🡪 credential🡪private key file Open 🡪 Session – hostname – publicIP - 18.119.162.16 🡪 Open
3. Login as : openvpnas
4. Go with default “yes” and “enter” for all steps

Make sure you make :

Admin UI : openvpn

Password : Nayak@1137

Admin UI: https://18.119.162.16:943/admin

Client UI: <https://18.119.162.16:943/>

1. Copy client UI and paste in chrome 🡪 Advance and proceed to this 🡪 OpenVPN login
2. Admin : openvpn && password : Nayak@1137
3. Select MS windows 🡪 open downloaded file and install 🡪 Enter Password 🡪 VPN connected.
4. GO to Private EC2 🡪 connect 🡪RDP Client🡪 download RDP 🡪 get password 🡪 upload pem file 🡪 copy password 🡪 open RDP file 🡪 login
5. Add inbound rule to Private EC2 – HTTP HTTPS ALL Traffic
6. Open RDP server 🡪 Cmd 🡪 ping 8.0.0.0 🡪 internet Access

**Simple Storage Service – S3**

- S3 is a global service

- search for S3 and see the region is global

# S3 Bucket Creation

1. Create Bucket 🡪 AWS region-> US East (Ohio) us-east-2 -> Bucket Name – yogesh1137 -> Object Ownership – ACL Disabled -> Block all public access -> Bucket Versioning – Disable -> Tags – as it is -> Default encryption - Server-side encryption with Amazon S3 managed keys (SSE-S3) -> Bucket key – Enable 🡪 create Bucket
2. Open – yogesh1137 🡪Upload🡪add file🡪choose file – Image 🡪 Upload 🡪close
3. Select file 🡪 open 🡪 you will able to see image uploaded
4. Click on file 🡪 Overview 🡪 Copy object URL 🡪 past in browser 🡪 Permission denied

# Public your Bucket Object

# Go to bucket 🡪 permission 🡪 Block public access (bucket settings) 🡪 Edit 🡪 uncheck box -> save changes -> confirm

# Bucket Policy 🡪 Edit 🡪 Policy Generator 🡪

# Select type of policy – S3 Bucket Policy

# Effect – Allow

# Principle - \*

# AWS Service – Amazon S3

# Action - getObject

# Amazon Resource Name (ARN) - arn:aws:s3:::yogesh1137 ( copy ARN from your edit bucket policy )

# Add Statement

# Generate policy

# Copy policy

{

"Id": "Policy1709979820305",

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

"Statement": [

{

"Sid": "Stmt1709979814257",

"Action": [

"s3:GetObject"

],

"Effect": "Allow",

"Resource": "arn:aws:s3:::yogesh1137",

"Principal": "\*"

}

]

}

# Paste policy and edit -> arn:aws:s3:::yogesh1137/\* (allow all object access)

# Save changes

# Open Uploaded file 🡪 Object URL – copy 🡪 paste in browser 🡪 granted public access to read file.

# Upload on more file to bucket 🡪 Object URL – copy 🡪 paste in browser 🡪 granted public access to read file.

# S3 Object Properties & Metadata

# Upload file to bucket 🡪select file 🡪 properties 🡪 Tags -> key – name , value – test -> Metadata -> Type – user define, key- x-amz-meta-test, value – test 🡪 upload 🡪 close

# Go to uploaded file and notice tag and metadata and key – passport.jpg (Uploaded file)

# Go to bucket and create folder 🡪 name- courses -> upload same file

# Go to uploaded file and notice tag and metadata and key – courses/passport.jpg (Uploaded file) (that means you cannot create folders S3 bucket courses/passport.jpg it’s an object name)

# Go to bucket and create folder 🡪 name- yogesh -> upload same file

# Go to uploaded file and notice tag and metadata and key – yogesh/courses/passport.jpg (Uploaded file) (that means you cannot create folders S3 bucket yogesh/courses/passport.jpg it’s an object name)

# S3 Versioning

# Create a bucket 🡪 Name – s3versioning1137 -> ACLs Disable -> Block Public Access settings for this bucket – Allow -> Bucket versioning – Disable -> bucket key – enable 🡪 create bucket

# Open – s3versioning1137 🡪Upload🡪add file🡪choose file – task.txt 🡪 Upload 🡪close

# Edit file 🡪 tast.txt and upload again 🡪 open task.txt

# Go to object (task.txt) 🡪 versions 🡪 Enable Bucket versioning 🡪 see version ID – null

# Edit file 🡪 tast.txt and upload again 🡪 open task.txt

# Click on slow version 🡪 you will see 3 versions

# Edit file 🡪 tast.txt and upload again 🡪 open task.txt

# Go to bucket 🡪 slow version 🡪 you will see 3 versions

# Disable show version

# Delete Object 🡪 select task.taxt 🡪 Delete.--> Enable show version 🡪 you will see Delete Marker

# Now if you want to retrieve your deleted file then delete 🡪 Select –> Delete marker 🡪 Delete Object 🡪You will able to see your latest versioned file.

# If you want to delete your file permanently 🡪 select middle version of file and delete 🡪 then select other version of file and delete.

# Upload file

# Edit and upload file again

# Properties 🡪 versioning 🡪 edit 🡪 suspend

# Still you will see the version

# Static Website Hosting On S3 Bucket

# Create a bucket 🡪 Name – forms-website-1137 -> ACLs Disable -> Block Public Access settings for this bucket – Allow -> Bucket versioning – Disable -> bucket key – enable 🡪 create bucket

# Upload file 🡪 index.html and index.js

# Go to bucket 🡪 Properties 🡪 Static Website Hosting 🡪 Edit 🡪 Enable 🡪 Hosting type – Host a static website 🡪 index document – index.html 🡪 Save changes.

# Scrolled down 🡪 Static website hosting URl copy 🡪 paste in browser 🡪 403 Forbidden error

# Select index.html and open 🡪 it will work but not working with URL

# Go to bucket 🡪 permission 🡪 Bucket Policy 🡪 Edit 🡪 Policy Generator 🡪

# Select type of policy – S3 Bucket Policy

# Effect – Allow

# Principle - \*

# AWS Service – Amazon S3

# Action - getObject

# Amazon Resource Name (ARN) - arn:aws:s3::: forms-website-1137 ( copy ARN from your edit bucket policy )

# Add Statement

# Generate policy

# Copy policy

{

"Id": "Policy1709979820305",

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

"Statement": [

{

"Sid": "Stmt1709979814257",

"Action": [

"s3:GetObject"

],

"Effect": "Allow",

"Resource": "arn:aws:s3:::forms-website-1137",

"Principal": "\*"

}

]

}

# Paste policy and edit -> arn:aws:s3:::forms-website-1137/\* (allow all object access)

# Save changes

# Go to bucket properties 🡪 Static website hosting URL copy 🡪 paste in browser 🡪 <http://forms-website-1137.s3-website.us-east-2.amazonaws.com> 🡪 It Works!

# Upload another file to bucket 🡪 form1.html

# If you browse 🡪 <http://forms-website-1137.s3-website.us-east-2.amazonaws.com/form1.html> --> it will also work ! because form1.html is in bucket.

# If you are accessing yogesh.html 🡪 <http://forms-website-1137.s3-website.us-east-2.amazonaws.com/yogesh.html> --> it will throw 404 Error.

# Go to bucket 🡪 Properties 🡪 Static Website Hosting 🡪 Edit 🡪 Error Document – form1.html 🡪 save changes

# If you are accessing yogesh.html 🡪 <http://forms-website-1137.s3-website.us-east-2.amazonaws.com/yogesh.html> --> it Works! With form1.html error file.

# How to Redirect in S3 Static Website From one to another

# Create a bucket 🡪 Name – www.forms-website-1137 -> ACLs Disable -> Block Public Access settings for this bucket – Allow -> Bucket versioning – Disable -> bucket key – enable 🡪 create bucket

# Go to bucket 🡪 properties 🡪 Static Website Hosting 🡪 Edit 🡪 Enable🡪 Hosting type – Redirect request for an object -> Host name - [forms-website-1137.s3-website.us-east-2.amazonaws.com](http://forms-website-1137.s3-website.us-east-2.amazonaws.com) ( make sure Host Name only) 🡪 Save changes.

# Properties 🡪 Static website hosting 🡪 copy URL 🡪 past in browser

# Go to Bucket forms-website-1137 🡪 create a folder Name - yogesh 🡪 and upload me.jpg image in it.

# Go to browser and try to open image with domain name 🡪 <http://forms-website-1137.s3-website.us-east-2.amazonaws.com/yogesh/me.jpg> 🡪 It Works!

# How to Change Prefix in Static Website S3

# Go to Bucket forms-website-1137 🡪 properties 🡪 Static Website hosting 🡪 Edit 🡪 redirection rules -> click on  [Learn more](https://docs.aws.amazon.com/console/s3/website-redirect) documentation 🡪 scroll down and copy 🡪

# [

# {

# "Condition": {

# "KeyPrefixEquals": "yoge/"

# },

# "Redirect": {

# "ReplaceKeyPrefixWith": "yogesh/"

# }

# }

# ]

# Save changes

# copy domain name 🡪 paste in browser followed by prefix e.g.

# <http://forms-website-1137.s3-website.us-east-2.amazonaws.com/yoge/me.jpg>

# S3 Accelerated Transfer, How to Enable? How to Use Accelerated Transfer?

You can use Amazon S3 Transfer Acceleration transfer files quickly and securely over long distances between your client and an S3 bucket. You can enable Transfer Acceleration using the S3 console, the AWS Command Line Interface (AWS CLI), API, or the AWS SDKs.

This section provides examples of how to enable Amazon S3 Transfer Acceleration on a bucket and use the acceleration endpoint for the enabled bucket.

# Create Bucket 🡪 name - yogesh-youtube-test 🡪 Region(Ohio)🡪 default all 🡪 create bucket

# Create Bucket 🡪 name – testing-example.com 🡪 Region(Ohio) 🡪 default all 🡪 create bucket

# Go to yogesh-youtube-test 🡪 properties 🡪 Transfer Accelaration 🡪 Edit 🡪 transfer acceleration – enable -> accelerated Endpoint - yogesh-youtube-test.s3-accelerate.amazonaws.com 🡪 save changes

# Speed comparison test 🡪 <https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html>

# Open cmd 🡪 aws s3 ls 🡪 need to configure aws

# Download AWS CLI 64 bit

# aws configure

# enter Access Key – AKIAZI2LD36U4EX7LGAT

# Enter Secret key - 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

# Default region name : us-east-2

# Default output format : enter

# aws s3 ls

# cd Desktop

# aws s3 cp task.txt s3://yogesh-youtube-test

# Open yogesh-youtube-test bucket and check

# aws s3 ls

# Delete object 🡪 task.txt

# aws s3 ls

# aws configure set s3.addressing\_style virtual

# go to bucket yogesh-youtube-test 🡪 properties 🡪 Transfer Acceleration 🡪 accelerated Endpoint - Copy URL - yogesh-youtube-test.s3-accelerate.amazonaws.com

# cmd 🡪 aws s3 co task.txt s3:// yogesh-youtube-test –endpoint-url <https://s3-accelerated.amazonaws.com> 🡪 its Uploaded

# Same/Cross Region Replication

# Create Bucket 🡪 name - yogesh-youtube-main 🡪 Region(Ohio)🡪 default all 🡪 create bucket

# Create Bucket 🡪 name – yogesh-youtube-replica 🡪 Region(Ohio) 🡪 default all 🡪 create bucket

# Go to yogesh-youtube-main 🡪 upload file – task.txt 🡪 upload

# Open both bucket in separate tabs 🡪 enable versioning for both the buckets.

# Go to yogesh-youtube-main 🡪 Management 🡪 Replication Rule 🡪 Create Replication Rule 🡪 name – first-replication-rule-youtube -> status – enable -> source bucket – yogesh-youtube-main -> choose a rule scope – apply to all object in the bucket 🡪 Destination -> destination – choose a bucket in this account -> Bucket name -> browse S3 -> choose – yogesh-youtube-replica -> choose path🡪 destination region – default -> IAM Role – choose from existing IAM roles -> IAM role – create new role 🡪 save 🡪 Replicate existing object? – no 🡪 submit

# Go to yogesh-youtube-replica and check – no object

# Go to yogesh-youtube-main 🡪 upload file – me.jpg 🡪 upload

# Go to yogesh-youtube-replica and check – me.jpg object

# Version id also replicated check in both the buckets –> click show versions enable

# Go to yogesh-youtube-main 🡪 disable show versions -> delete me.jpg

# Show versions -> delete marker is there

# Go to yogesh-youtube-replica and check – me.jpg there (deletion of object not work on replica)

# Go to yogesh-youtube-main 🡪 management 🡪 replication rule 🡪 go to first-replication-rule-youtube 🡪 Action 🡪 Edit 🡪 Additional replication Option 🡪 select – Delete marker replication 🡪 save

# Go to yogesh-youtube-main 🡪 select delete marker 🡪 Delete

# Select me.jpd 🡪 Delete 🡪 deletes me.jpg

# Go to yogesh-youtube-replica 🡪 Show version -> me.jpg there

# Configure Logging in S3 Bucket

# You can record the actions that are taken by users, roles, or AWS services on Amazon S3 resources and maintain log records for auditing and compliance purposes. To do this, you can use server-access logging, AWS CloudTrail logging, or a combination of both. We recommend that you use CloudTrail for logging bucket-level and object-level actions for your Amazon S3 resources.

# Create Bucket 🡪 name – yogesh-youtube-s3-content 🡪 is-east-2

# Create Bucket 🡪 name – yogesh-youtube-s3-content-log 🡪 is-east-2

# Go to yogesh-youtube-s3-content 🡪 Properties 🡪Versioning-enable 🡪 Server Access Logging 🡪 Edit 🡪 server access logging - enable -> Target Bucket -> Browse S3 -> yogesh-youtube-s3-content-log 🡪 Save changes

# Go to yogesh-youtube-s3-content-log 🡪 Permissions 🡪 Policies 🡪 see grant permission for putObject

# Go to yogesh-youtube-s3-content 🡪 upload multiple files – 1.txt, 2.txt, 3.txt, 4.txt, 5.txt, 6.txt

# Open both bucket in separate tabs

# Logs will reflect in yogesh-youtube-s3-content-log bucket – It will take more than 20min.

# AWS S3 - Storage Classes - Configure Storage Class

* **S3-Standerd**
* **Glacier deep archive –** (cheapest)
* **Glacier** (long term storage)
* **Standard infrequent access or Standard IA** (cost less but you pay to access it more frequently)
* **One-Zone-IA** (infrequent Access)
* **Intelligent Tiering**

1. Create Bucket 🡪 name – s3-storage-class-demo-yogesh 🡪 us-east-2(Ohio)
2. Object upload 🡪 Select - me.jpg 🡪 Properties 🡪 Storage class – Standard
3. Object upload 🡪 Select – 1.txt, 2.txt 🡪 Properties 🡪 Storage class – Standard – IA

# Simplify Data Lifecycle Management with AWS S3

1. Create Bucket 🡪 name – s3-storage-class-demo-yogesh 🡪 us-east-2(Ohio)
2. Object upload 🡪 Select - me.jpg 🡪 Properties 🡪 Storage class – Standard
3. storage-class-demo-yogesh 🡪properties 🡪versioning - enable
4. Object upload 🡪 Select – 1.txt, 2.txt 🡪 Properties 🡪 Storage class – Standard – IA
5. s3-storage-class-demo-yogesh 🡪 Management 🡪 Create lifecycle rule 🡪 name – first-lifecycle-rule -> choose a rule scope – apply to all objects in the bucket -> Filter Type -> checked box ->

* lifecycle rule action – select move **current version** -> Choose Storage Classes ->
  1. choose storage class transition – Standard-IA, days after object creation – 30 -> add transition
  2. choose storage class transition – Glacier Instant retrieval, days after object creation – 150, add transition
  3. choose storage class transition– Glacier flexible retrieval, days after object – 240, add transition
  4. choose storage class transition – Glacier Deep archive **OR / AND**
* lifecycle rule action – select move **non - current version** -> Choose Storage Classes ->

1. choose storage class transition – Standard-IA, Days after objects become noncurrent – 30, Number of newer versions to retain – 100 -> add transition
2. choose storage class transition – Intelligent-Tiering, Days after objects become noncurrent – 60, Number of newer versions to retain -100 ->add transition
3. choose storage class transition– Glacier Instant retrieval, Days after objects become noncurrent – 90, Number of newer versions to retain -100

-> add transition

1. choose storage class transition – Glacier Deep archive, Days after objects become noncurrent – 120, Number of newer versions to retain -100

**OR / AND**

* lifecycle rule action – select expire **current version** -> days after object creation -150 days **OR / AND**
* lifecycle rule action – select permanently delete **non-current version** -> Days after objects become noncurrent- 100, Number of newer versions to retain-100 **OR / AND**
* lifecycle rule action – select delete expired **Delete marker** -> Delete expired object delete markers – yes/no, Delete incomplete multipart uploads- yes/ no

1. Create lifecycle rule

# Cost Efficiency and Performance with AWS Intelligent Tiering for S3

# S3 Intelligent-Tiering automatically stores objects in three access tiers:

* Frequent Access – Objects that are uploaded or transitioned to S3 Intelligent-Tiering are automatically stored in the Frequent Access tier.
* Infrequent Access – S3 Intelligent-Tiering moves objects that have not been accessed in 30 consecutive days to the Infrequent Access tier.
* Archive Instant Access – With S3 Intelligent-Tiering, any existing objects that have not been accessed for 90 consecutive days are automatically moved to the Archive Instant Access tier.

1. Create bucket 🡪 intelligent-tiering-yogesh
2. Upload file 🡪 storage class – intelligent-tiering
3. Go to intelligent-tiering-yogesh 🡪 Propertis 🡪 Intelligent-Tiering Archive configuration 🡪 configuration name – rule1 -> choose a configuration scope – this configuration applies to all objects in the bucket -> status – enable -> Archive rule action –> archive access tier -> Days until transition to the archive access tier – 90days -> days until transion to the deep archive access tier – 180 days 🡪create

# What is CORS and How to Enable it in S3

# Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. With CORS support, you can build rich client-side web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

# Create bucket 🡪 name – static-website-demo-yogesh 🡪 block all public access – unchecked 🡪 create bucket

# Open static-website-demo-yogesh 🡪 permission 🡪 bucket policy 🡪 edit 🡪 policy generator 🡪 copy ARN ->

# Select type of policy – S3 Bucket Policy

# Effect – Allow

# Principle - \*

# AWS Service – Amazon S3

# Action - getObject

# Amazon Resource Name (ARN) - arn:aws:s3::: static-website-demo-yogesh ( copy ARN from your edit bucket policy )

# Add Statement

# Generate policy

# Copy policy

{

"Id": "Policy1709979820305",

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

"Statement": [

{

"Sid": "Stmt1709979814257",

"Action": [

"s3:GetObject"

],

"Effect": "Allow",

"Resource": "arn:aws:s3:::forms-website-1137",

"Principal": "\*"

}

]

}

# Paste policy and edit -> arn:aws:s3::: static-website-demo-yogesh /\* (allow all object access)

**🡪** Save changes

1. Create a folder in desktop 🡪 s3-static-website (/geekster) -> sub files index.html loader.html
2. Go to bucket 🡪 upload index.html
3. Go to bucket 🡪 upload loaderpage.html
4. Properties 🡪 static website hosting – enable -> index document – index.html 🡪 save changes
5. Go to object index.html 🡪 open
6. Go to object loaderpage.html 🡪 open
7. Go to object index.html 🡪 copy hosting URL and open in browser
8. Browser 🡪 hosting URL/loaderpage.html
9. Edit index.html 🡪 add loaderpage with jquery 🡪 save
10. Go to bucket and delete index.html
11. Upload index.html
12. Go to object index.html 🡪 copy hosting URL and open in browser
13. Delete loaderpage.html

# Create bucket 🡪 name – static-website-demo-yogesh1 🡪 block all public access – unchecked 🡪 create bucket

1. Open static-website-demo-yogesh1 🡪 permission 🡪 bucket policy 🡪 edit

{

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

"Id": "Policy1710067760916",

"Statement": [

{

"Sid": "Stmt1710067759166",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::s3-website-demo-yogesh1/\*"

}

]

}

1. Upload 🡪 loaderpage.html
2. static website hosting – enable -> index document – loaderpage.html 🡪 save changes
3. Object 🡪 static URL 🡪 copy and paste in browser
4. Edit index.html 🡪 load function 🡪 paste - <https://s3-website-demo-yogesh1.s3.us-east-2.amazonaws.com/loaderpage.html>
5. static-website-demo-yogesh1 🡪 Permission 🡪 Cross-Origin Resource Sharing (CORS) 🡪 edit 🡪 cleck on learn more -> using s3 console -> configure CORS -> example 2 🡪 copy and paste ->

[

{

"AllowedHeaders": [

"\*"

],

"AllowedMethods": [

“GET”

"PUT",

"POST",

"DELETE"

],

"AllowedOrigins": [

" http://s3-website-demo-yogesh.s3-website.us-east-2.amazonaws.com"

],

"ExposeHeaders": [

"x-amz-server-side-encryption",

"x-amz-request-id",

"x-amz-id-2"

],

"MaxAgeSeconds": 3000

}

]

1. Save changes
2. Delete index.html and upload again
3. Copy URL and paste in browser 🡪 It works ! you are accessing other domain also !

**Presigned URL**

By default, all Amazon S3 objects are private, only the object owner has permission to access them. However, the object owner may share objects with others by creating a presigned URL. A presigned URL uses security credentials to grant time-limited permission to download objects. The URL can be entered in a browser or used by a program to download the object. The credentials used by the presigned URL are those of the AWS user who generated the URL.

1. Create Bucket – name – yogesh-presigned-url 🡪 create bucket
2. Upload object 🡪 me.jpg
3. Object URl -> <https://yogesh-presign-url.s3.ap-northeast-1.amazonaws.com/me.jpg> – browser – paste 🡪 not accessible
4. Select object and open 🡪 Open with presigned url
5. Object Action 🡪 Share with Presigned URL -> Time interval until the presigned URL expires – Minutes - > Number of minutes – 1 🡪 Create Presigned URL
6. Copy presigned url – paste in browser 🡪 me.jpg

**EFS – Elastic File System**

1. Create EFS 🡪 Customize
2. Step 1 🡪 File system settings

* Name – EFS1
* File system type – Regional
* Automatic backups – Enable
* Lifecycle management 🡪 Transition to IA – None days -> Transition to Archive – None days -> Transition to standard – None access
* Encryption – enable
* Performance Settings 🡪 Throughput mode – Bursting

1. Step 2 🡪 Network Access

* Network 🡪 VPC – default
* Mount targets 🡪 3 (default subnet, AZ, security Groups, IP Automatic)

1. Review and create 🡪 Next
2. Launch two ubuntu EC2 instance in different AZ with default configuration
3. Name – EFS-Instance1 & EFS-Instance2
4. Connect them with cmd with ssh

EFS- Instance1 🡪🡪

1. sudo su
2. sudo yum install -y amazon-efs-utils
3. mkdir efs
4. go to EFS1 🡪 Attach 🡪 mount via IP 🡪 AZ – us-east-2a 🡪 using the NFS client 🡪 copy
5. paste 🡪 sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport 172.31.7.83:/ efs
6. mount –s|grep efs
7. sudo chmod 777 efs
8. ll
9. cd efs
10. touch file{1..6}
11. ll
12. go to instance-2 cmd
13. touch file24
14. ll

EFS- Instance2 🡪🡪

1. sudo su
2. sudo yum install -y amazon-efs-utils
3. mkdir efs
4. go to EFS1 🡪 Attach 🡪 mount via IP 🡪 AZ – us-east-2b 🡪 using the NFS client 🡪 copy
5. Paste 🡪 sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport 172.31.29.7:/ efs
6. ll
7. cd efs
8. ll
9. go instance-1 cmd
10. ll

**AMI (Amazon Machine Image)**

1. Launch instance -> name – AMI-Instance -> OS-windows -> default-VPC -> security group – RDP -> Configure storage -> Advance -> add volume -> size – 8GB, volume type – gp2 🡪 Launch Instance
2. Connect - RDP 🡪 save password 🡪 successful
3. Go to window server 🡪 check volume at this PC -> not showing
4. Search server manager -> local server -> task at right top -> compute management -> disk management -> Disk 1 -> online -> initialized -> new simple volume -> name-demo -> next -> finish
5. Go to this PC and check disk1
6. create some folders on C drive, D drive and desktop
7. Go to instance 🡪 AMI-Instance 🡪 action 🡪 image (stop instance recommended )
8. Create image 🡪 name - AMI-Instance-Image -> description - windows webserver in ohio region -> create image
9. Go to AMI 🡪 name – ohio-image
10. Go to Snapshot 🡪 name – D drive – 8GB , name – root -30GB
11. Go to AMI-Instance -> Terminate
12. Go to AMI 🡪 select image 🡪 action 🡪 copy AMI 🡪name- AMI-Image-mumbai -> description - AMI-Image created from ohio webserver -> destination region – (asia pacific –mumbai) 🡪 copy AMI
13. Go to Mumbai region 🡪 ec2 🡪 AMI 🡪 rename – Mumbai-image
14. Select image and launch 🡪 you can change or you can use default configuration 🡪 launch instance
15. Connect with RDP 🡪 use previous rdp saved password.
16. Go to new remote server and check files created on drives and desktop.

# How to attach root volume with another EC2 Instance | How to take snapshot of EC2

# Launch Instance 🡪 name - windows-server1 -> OS – windows -> VPC – default -> subnet – us-east-2a 🡪 storage – gp2 path - /dev/sda1 --> launch instance

# Instance connect 🡪 get password 🡪 save pass for further use 🡪 HPcd0lff&.c&d6LN2y-JY9ows.yTe;86

# Go to windows server 🡪 make some folder on C drive and desktop.

# Stop instance – windows-server1

# Go to snapshot 🡪 create snapshot 🡪 select instanceID -> description – snapshot of windows-server1 ->include root volume not exclude -> Copy tags from source volume – checked -> create snapshot

# Go to snapshot 🡪 create snapshot 🡪 select volumeID 🡪 description – snapshot of root volume 🡪 create snapshot.

# Rename instance-snapshot, volume snapshot

# Delete volume snapshot

# Terminate instance

# Go to snapshot select instance-snapshot -> (you can create image/volume) 🡪 action 🡪 create volume 🡪 volume type – gp3 -> AZ - us-east-2a 🡪 create volume

# Launch Instance 🡪 name - windows-server2 -> OS – windows -> VPC – default -> subnet – us-east-2a 🡪 storage – gp2 path - /dev/sda1 --> launch instance

# Go to volumes 🡪 rename – snap-volume(available), server-volume(in-use)

# Go to instance 🡪 stop – windows-server2

# Go to volume 🡪 select server-volume 🡪 action 🡪 detach volume

# Select snap-volume 🡪 action 🡪 attach volume 🡪 AZ- us-east-2a -> instance – window-server2 -> (make sure change path given at instance creation) path - /dev/sda1 🡪 attach volume

# Start instance 🡪 windows-server2

# Connect with RDP 🡪 use windows-server1 RDP password 🡪 HPcd0lff&.c&d6LN2y-JY9ows.yTe;86 🡪Go to RDP windows server – check files are created !

# AWS Auto Scaling

# Go to EC2

# Auto scaling

# Auto scaling group

# Create auto scaling group 🡪

# Step1 🡪choose launch template -> auto scaling group name – yogesh-ASG-1 -> launch template – create a launch template 🡪

# Name-mytemplate2 -> description – auto scaling template -> auto scaling guidance- checked -> OS – ms windows -> instance type – t2.micro -> key pair ->network setting -> subnet – don’t include in launch template -> select SG -> storage – 8gb 🡪create launch template

# Step 2 🡪 choose instance launch option -> Network -> VPC- default -> availability zones and subnets – us-east-2a, us-east-2b, us-east-2c 🡪 next

# Step 3 🡪 configure advance option -> load balancing – no load balancer ->select VPC lattice service to attach – no VPC lattice Service -> health check grace period -300 sec -> monitoring- checked -> default instance warm up – checked 🡪 next

# Step 3 🡪 configure group size and scaling – Desired capacity -2 -> scaling limits – min desired capacity-1, max desired capacity-10 -> Automatic scaling -> select Target tracking scaling policy -> scaling policy name- Target tracking policy -> metric type – average CPU utilization -> Target value – 50 -> instance warmup -300sec -> instance maintenance policy – select no policy -> instance scale-in protection – check🡪 next

# Step 4 🡪 Add notifications 🡪

# Step 5 🡪 Add notifications 🡪

# Step 6 🡪 Add tags 🡪

# Step 7 🡪 Review 🡪 Create Auto scaling Group

# Go to EC2 instances 🡪 2 Instance are running 🡪 rename them as server1 and server2 🡪 connect with RDP

# Server1 RDP pass 🡪 0hExw!HnfO%RnpI=robIeYq!&-TyDzp3

# Server2 RDP pass 🡪 7wV-hiyKc!pP;PRN)&D1.cd(-8KB2CXb

# Go to server1 🡪 on desktop create a file a.txt 🡪 open a.txt – insert a.bat -> save as a.bat all file 🡪and open a.bat file multiple times

# Go to server2 🡪 on desktop create a file a.txt 🡪 open a.txt – insert a.bat -> save as a.bat all file 🡪and open a.bat file multiple times

# Wait for some time

# Go to EC2 check instances are creating

# ELASTIC LOAD BALANCER (ELB)

# Application load balancer

# 

# Launch instance 🡪 name- ELB-Server1 -> OS- MS windows -> VPC – default -> subnet – us-east-2a -> create SG – name – ELB-SG, HTTP, HTTPS, RDP -> launch instance

# Launch instance 🡪 name- ELB-Server2 -> OS- MS windows -> VPC – default -> subnet – us-east-2b -> create SG – name – ELB-SG, HTTP, HTTPS, RDP -> launch instance

# Connect ELB-Server1 with RDP 🡪 pass - UFYXa8myC5P.kcJa8.J3SOp-%29Nr?hB

# Go to ELB-Server1 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text fiel -> write in text file – SERVER AVAILABILITY ZONE A 🡪 save as 🡪 index.html , all files 🡪 save

# Connect ELB-Server2 with RDP 🡪 pass - vfQuAJv?G5ZKdrLvaT&$H\*vVu9amPKfL

# Go to ELB-Server1 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text fiel -> write in text file – SERVER AVAILABILITY ZONE B 🡪 save as 🡪 index.html , all files 🡪 save

# Go to EC2

# Load Balancer 🡪 create load balancer 🡪 select Application load balancer 🡪 create 🡪 name – Application-LB -> schemes – internet-facing -> IP address type – Ipv4 -> Network Mapping 🡪 VPC – default -> Mapping – select us-east-2a, us-east-2b (note – select all AZ in which subnet is created) -> Security Groups -> ELB-SG -> Listeners and Routing -> protocol - HTTP -> create target group -> choose a target type – Instance -> name – TG1 -> protocol:port – HTTP:80 -> address type- IPv4 -> protocol-version – HTTP1 -> health check protocol –HTTP -> Next 🡪 select ELB-Server1 & ELB-Server2 -> create Target Group 🡪 go back to listeners and routes 🡪 listeners -> default action – select – TG1 🡪 create load balancer

# Go to target groups 🡪 TG1 🡪Registerd Targets 🡪 register targets – select ELB-Server1 & ELB-Server2 🡪 register Pending targets

# Check for target -> heath status -> healthy

# Go to load balancer 🡪 Details 🡪 DNS name (Application-LB-153921349.us-east-2.elb.amazonaws.com) copy 🡪 go to chrome and paste – you will see server A -> refresh -> you will see server A -> refresh -> you will see server B

# go to browser edge and paste – you will see server B-> refresh -> you will see server A -> refresh -> you will see server B

# LOAD BALANCER SIMULTANEOSLY DISTRIBUTING LOAD SOMETIME ON SERVER A (AZ-A) AND SOMETIME ON SERVER B (AZ-B).

# Launch instance 🡪 name- ELB-Server3 -> OS- MS windows -> VPC – default -> subnet – us-east-2c -> create SG – name – ELB-SG, HTTP, HTTPS, RDP -> launch instance

# Connect ELB-Server1 with RDP 🡪 pass - MF0gwXzuX=AF;K9hcGOZlK)7LTd=q2Sy

# Go to ELB-Server1 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text fiel -> write in text file – SERVER 3 WELCOME TO THE AZ C🡪 save as 🡪 index.html , all files 🡪 save

# Go to load balancer – Application-LB 🡪 network mapping 🡪 edit subnet 🡪 edit subnets 🡪 select – us-east-2c 🡪 save

# Go to target group TG1 🡪 register targets 🡪 select ELB-Server3 -> include as pending below 🡪 register pending targets

# Go to load balancer 🡪 Details 🡪 DNS name (Application-LB-153921349.us-east-2.elb.amazonaws.com) copy 🡪 go to chrome and paste – you will see server A -> refresh -> you will see server C -> refresh -> you will see server B

# LOAD BALANCER DISTRIBUTING EQUAL TRAFFIC TO ALL SERVERS IN AZ

# Delete ELB 🡪Target Groups 🡪 Instances

# 

# 

# 2. Network Load balancer

# Launch instance 🡪 name- NLB-Server1 -> OS- MS windows -> VPC – default -> subnet – us-east-2a -> create SG – name – NLB-SG, HTTP, HTTPS, RDP -> launch instance

# Launch instance 🡪 name- NLB-Server2 -> OS- MS windows -> VPC – default -> subnet – us-east-2b -> create SG – name – NLB-SG, HTTP, HTTPS, RDP -> launch instance

# Connect NLB-Server1 with RDP 🡪 pass - bTkL(u0OZ)iSIcWJE9nu(h46oOS4Q$tR

# Go to NLB-Server1 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text fiel -> write in text file – SERVER1 WELCOMETO AVAILABILITY ZONE A 🡪 save as 🡪 index.html , all files 🡪 save

# Connect NLB-Server2 with RDP 🡪 pass - vmeZN7F=6Uxw.M$h7h\*\*(udoL%9qcq!Z

# Go to NLB-Server1 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text fiel -> write in text file – SERVER2 WELCOMETO AVAILABILITY ZONE AB🡪 save as 🡪 index.html , all files 🡪 save

# Go to EC2

# Go to load balancer 🡪 create load balancer

# Select 🡪 **Network Load Balancer 🡪 name – NLB1 -> Scheme – Internet facing -> IP address Type – Ipv4 -> Network Mapping -> VPC –default -> Mappings – us-east-2a, us-east-2b 🡪 Security Groups – NLB-SG -> Listener and Routing -> listener -> protocol – TCP, Port – 80 -> Create Target Group -> choose a target type – IP addresses -> name – NLB-TG1 -> Protocol port – TCP:80 -> IP Address type – Ipv4 -> VPC – default 🡪 Next 🡪 Choose a Network – default VPC network -> Specify IPs and Ports -> NLB-Server1 (private IP) -> add IPv4 address -> NLB-Server2 (private IP)v -> port:80 -> Include as pending below 🡪 create Target Group 🡪 Go back to load balancer tab 🡪 default action – NLB-TG1 -> Create Load Balancer**

# **Go to load balancer NLB1 – check all the configuration**

# Go to target group NLB-T1 – check health status of Targets

# Go to NLB1 -> details -> copy DNS name -> paste in chrome -> You will see SERVER1 WECOME TO AVAILABILITY ZONE A -> 1min wait & refresh -> You will see SERVER2 WECOME TO AVAILABILITY ZONE B -> paste DNS name in new tab - You will see SERVER2 WECOME TO AVAILABILITY ZONE B -> refresh -> You will see SERVER1 WECOME TO AVAILABILITY ZONE A

# NETWORK LOAD BALANCER EQUALLY DISTRIBUTING TRAFFIC TO EACH INSTANCES AVAILABLE IN DIFFERENT AZs

# How to establish a load balancer between two VPCs

# Overview Steps:

# Create two VPCs i.e. VPC1 and VPC2

# Create two subnets in VPC1 and one in VPC2

# Create internet gateway for VPC1 and VPC2

# Edit route table and add 0.0.0.0/0

# Create peering connection and update route table

# Create three EC2 instances, one in each subnets

# Install IIS and create webpage

# Create application load balancer, target type – IP

# Register private IPs of EC2 in Target Groups

# Copy DNS of load balancer and paste in web browser

# Detailed lab steps:

# Create VPC 🡪 name – VPC1 -> IPv4 CIDR – 10.0.0.0/16 🡪 create VPC 🡪 action -> edit VPC settings -> enable DNS hostname 🡪 save changes

# Create Subnets 🡪 VPC ID – VPC1 -> name – VPC1-Subnet1 -> Availability zone – us-east-2a -> IPv4 CIDR – 10.0.1.0/24

# Create Subnets 🡪 VPC ID – VPC1 -> name – VPC1-Subnet2 -> Availability zone – us-east-2b -> IPv4 CIDR – 10.0.2.0/24

# Create Internet Gateway 🡪 VPC1-IGW 🡪 create Internet Gateway 🡪 create Internet Gateway 🡪 action 🡪 attach VPC – VPC1 🡪 attach Internet Gateway

# Create Route table 🡪 name – VPC1-RT -> VPC – VPC1 🡪 create route table 🡪 edit routes 🡪 select 0.0.0.0/0 and VPC1-IGW 🡪 save changes 🡪 Subnet association 🡪 edit subnet association 🡪 select both the subnets 🡪 save changes

# Create VPC 🡪 name – VPC2 -> IPv4 CIDR – 192.168.0.0/16 🡪 create VPC 🡪 action -> edit VPC settings -> enable DNS hostname 🡪 save changes

# Create Subnets 🡪 VPC ID – VPC2 -> name – VPC2-Subnet1 -> Availability zone – us-east-2b -> IPv4 CIDR – 192.168.1.0/24

# Create Internet Gateway 🡪 VPC2-IGW 🡪 create Internet Gateway 🡪 create Internet Gateway 🡪 action 🡪 attach VPC – VPC2 🡪 attach Internet Gateway

# Create Route table 🡪 name – VPC2-RT -> VPC – VPC2 🡪 create route table 🡪 edit routes 🡪 select 0.0.0.0/0 and VPC2-IGW 🡪 save changes 🡪 Subnet association 🡪 edit subnet association 🡪 select the subnets 🡪 save changes

# Peering Connection 🡪 create Peering connection 🡪 name – VPC1-VPC2-Peering -> requester VPC ID – VPC1 -> select My Account and This Region -> Accepter VPC ID – VPC2 🡪 Create Peering Connection

# Peering Connection 🡪 VPC1-VPC2-Peering 🡪 Action 🡪 Accept request

# VPC1-RT 🡪 edit routes -> add routes -> 192.168.0.0/16 -> select peering connection -> VPC1-VPC2-Peering 🡪 save changes

# VPC2-RT 🡪 edit routes -> add routes -> 10.0.0.0/16 -> select peering connection -> VPC1-VPC2-Peering 🡪 save changes

# Launch instance 🡪 name- VPC1-Server1 -> OS- MS windows -> VPC – default -> subnet – us-east-2a -> create SG – name – VPL-LB-SG, HTTP, HTTPS, RDP -> launch instance

# Launch instance 🡪 name- VPC1-Server2 -> OS- MS windows -> VPC – default -> subnet – us-east-2b -> create SG – name – VPL-LB-SG, HTTP, HTTPS, RDP -> launch instance

# Launch instance 🡪 name- VPC2-Server1 -> OS- MS windows -> VPC – default -> subnet – us-east-2a -> create SG – name – VPL-LB-SG, HTTP, HTTPS, RDP -> launch instance

# 

# Connect VPC1-Server1 with RDP 🡪 pass - bTkL(u0OZ)iSIcWJE9nu(h46oOS4Q$tR

# Go to VPC1-Server1 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text file -> write in text file – VPC1 SERVER1 WELCOME TO AVAILABILITY ZONE A 🡪 save as 🡪 index.html , all files 🡪 save

# Connect VPC1-Server2 with RDP 🡪 pass - Zjik)xWz0&?lVT9;?cv?%6E!)pg6\*Kzv

# Go to VPC1-Server2 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text file -> write in text file – VPC1 SERVER2 WELCOME TO AVAILABILITY ZONE B 🡪 save as 🡪 index.html , all files 🡪 save

# Connect VPC2-Server1 with RDP 🡪 pass - C?26HRIN92iA?!HlAK.DRngX7L=KOQwT

# Go to VPC2-Server1 RDP 🡪 search server manager 🡪 add role and features 🡪 next 🡪 next 🡪 role-based or feature based installation 🡪 next 🡪scroll down – select web server(IIS) 🡪 add features 🡪 next up to Install 🡪 close

# Go to C drive 🡪 inetpub 🡪wwwroot 🡪 ctrl+a & delete 🡪 right click -> new text file -> write in text file – VPC2 SERVER1 WELCOME TO AVAILABILITY ZONE A 🡪 save as 🡪 index.html , all files 🡪 save

# Create Load balancer 🡪 Application load balancer 🡪 create 🡪 name – Application-LB -> scheme – Internet Facing -> IP address type -> Network Mapping -> VPC – VPC1 -> Mappings – select us-east-2a,us-east-2b -> Security Group – Create new security group -> name –APP-LB-SG -> HTTP, All ICMP – Anywhere -> Listeners and Routing -> protocol – HTTP:80 -> Create Target group 🡪 Choose a target Type – IP address -> Name – TG1 -> Protocol – port : 80 🡪 Next 🡪 choose a Network -> Network – VPC1 -> **Specify IPs and Ports -> VPC1-Server1 (private IP)** 10.0.1.143 **-> add IPv4 address -> NLB-Server2 (private IP)** 10.0.2.138 **-> Include as pending below- > Network – select other private IP address -> availability zone – us-east-2a (VPC2-Server1 private IP) -> Enter a private IP -** 172.31.15.52 **-> include as pending below 🡪 Create Target Group 🡪 load balancer tab -> default action – Select TG1 🡪 Create Load Balancer**

# **Go to load balancer Application-LB – check all the configuration**

# **Target group 🡪 check health status of targets -> 1 target unhealthy**

# Go to Application-LB -> details -> copy DNS name -> paste in chrome -> You will see VPC1 SERVER1 WECOME TO AVAILABILITY ZONE A -> 1min wait & refresh -> You will see VPC1 SERVER2 WECOME TO AVAILABILITY ZONE

# Partial implement.

# Identity and Access Management(IAM)

# How to create users and assign policies?

# Existing policy will add to user in following cases:

# USER 🡪

# Add user to group

# Copy permission from existing user

# Attach existing policies directly

# MFA (multi – factor authenticator) 🡪 (Microsoft, Google)

# On your mobile play store search – google authenticator – install – configure

# IAM – security – add MFA – QR code – scan code from mobile – enter code received on mobile

# Done

# IAM Lab

# Go to IAM

# Dashboard 🡪 AWS Account 🡪 Sign-in URL for IAM users in this account 🡪 create 🡪 alias 🡪 yogeshrathod1🡪 create alias

# Security Recommendation 🡪 Deactivate or delete access keys for root user 🡪 manage access keys 🡪 Select Access key 🡪 delete

# Create Access key 🡪 download .csv file 🡪 create key 🡪 done

# User 🡪 create user 🡪 User name – yogesh1 -> Provide user access to the AWS Management Console – checked -> I want to create an IAM user -> Custom password –Nayak@1137 -> Users must create a new password at next sign-in – Recommended –checked 🡪 Next

# Create group 🡪 User group name – IAM-User-Group 🡪 create group 🡪 go to user creation

# User groups 🡪 select IAM-User-Group 🡪 Next 🡪 create user 🡪 download .csv file (user credentials) 🡪 return to user list

# Open .csv file (user credential) 🡪 copy url 🡪 new incognito tab 🡪 paste 🡪 enter user name and password 🡪 change password

# Go to EC2 🡪 create instance 🡪 access denied

# Go to S3 🡪 create bucket 🡪 access denied

# Go to user 🡪 yogesh1 🡪 Add Permissions 🡪 Attach existing policies directly 🡪 [AmazonEC2FullAccess](https://us-east-1.console.aws.amazon.com/iam/home?region=ap-northeast-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonEC2FullAccess) 🡪 Next 🡪 add permission

# Go to yogesh1 console 🡪 create EC2 instance

# Terminate Instance

# IAM 🡪 user yogesh1 🡪 select EC2FullAccess 🡪 remove 🡪 remove policy

# Create user 🡪 name –yogesh2 🡪 Provide user access to the AWS Management Console – checked -> I want to create an IAM user -> auto generated pass -> Users must create a new password at next sign-in – checked 🡪 Next --. Add user to group 🡪 select - [IAM-User-Group](https://us-east-1.console.aws.amazon.com/iam/home?region=ap-northeast-1#/groups/details/IAM-User-Group) 🡪 Next 🡪 download .csv file (user credentials) 🡪 return to user list

# Open .csv file (user credential) 🡪 copy url 🡪 new incognito tab 🡪 paste 🡪 enter user name and password 🡪

# Uswer groups 🡪 IAM-User-Groups 🡪 add user yogesh1 -> add permission 🡪 EC2ReadOnlyAccess 🡪 add permission

# Go to yogesh1 and yogesh2 user account 🡪 check EC2 access

# IAM Group, Inline Policy and billing dashboard

# IAM

# Create user 🡪name – nayak1 🡪 I want to create IAM user 🡪 Auto generated password 🡪 attach policies directly 🡪 [AmazonEC2ReadOnlyAccess](https://us-east-1.console.aws.amazon.com/iam/home?region=ap-northeast-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonEC2ReadOnlyAccess) 🡪 Next 🡪 Create User 🡪 download .csv file

# Create user 🡪name – nayak1 🡪 I want to create IAM user 🡪 Auto generated password 🡪 attach policies directly 🡪 [AmazonEC2ReadOnlyAccess](https://us-east-1.console.aws.amazon.com/iam/home?region=ap-northeast-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonEC2ReadOnlyAccess) 🡪 Next 🡪 Create User 🡪 download .csv file

# Create group 🡪 name – S3Team 🡪 add user nayak1 and nayak2 🡪 attach permission policies – S3FullAccess 🡪 create group

# Sign-in URL for IAM users – copy 🡪 paste in new incognito tab 🡪 login nayak1 and nayak2 user with login credentials

# Go to nayak1 & nayak2 user console 🡪 check S3 bucket access

# Create group 🡪 name – BillingTeam 🡪 add user nayak1 🡪 attach permission policies – Billing 🡪 create group

# Go to account of root user 🡪 IAM user and role access to billing information 🡪 activate

# Go to nayak1 user console 🡪 check billing information 🡪 accessible

# Cross Account Access using IAM role

# Login into chrome-yogesh account

# Create one group (yogesh-group) and then create two IAM users (user1 & user2)

# Attach policy to group 🡪 EC2ReadOnlyAccess

# Login to second account i.e. firefox-yogesh

# Create one S3 bucket

# IAM 🡪 Create a role 🡪 Trusted enity type - another AWS account 🡪 insert account ID of chrome-yogesh 🡪 attach policy – S3ReadOnlyAccess 🡪 role name – S3Read

# Now login back to chrome-yogesh account 🡪 click on group 🡪 permission 🡪 inline policy 🡪 select AWS Security Token Service(STS) (provides temporary security credentials) -> action – all -> resources – all 🡪 Add statement 🡪 apply policy

# Now login as IAM user1

# Switch role (Account 🡪 beside sign out) 🡪Account ID - firefox-yogesh -> insert - ROLE-S3read paste 🡪switch role 🡪 now check, whether you are able to see the bucket of another account or not

# Repeat step 8 & step 9 with user 2

# Now login into firefox-yogesh account🡪role🡪 trusted relationship 🡪 edit🡪 paste ARN of user1🡪 replace AWS root with "AWS": "arn:aws:iam::637423312809:user/user1" 🡪 update policy

# Login again in user 1 🡪 switch role 🡪 test S3 bucket

# Now login as user2 🡪 switch role 🡪 you will get error

# How to connect window AD server to AWS

# Create one window server 2012 R2 Base

# Login to server with RDP pass - FwI3j$5X1YXp@UB;I\*vTk-1BrFV48ghn 🡪 control panel 🡪 user account 🡪 administrator 🡪 manage another account 🡪 change password (Yogesh@123)

# Server manager 🡪 local server 🡪 computer name – WIN-IQRBDJFHF1

# Server manager 🡪 add role and features 🡪 3 times next 🡪 Active directory domain services 🡪 add freature 🡪 next until install 🡪 flag at top right click 🡪 promote this server to domain controller 🡪 add new forest 🡪 domain name – guftgu.in🡪 next 🡪 pass – india@123 🡪 next until install 🡪install

# It will restart automatically

# Login with pass –Yogesh@123

# Now go to server manager 🡪 tools 🡪DNS 🡪 reverse lookup zone 🡪 right click 🡪 action 🡪new zone 🡪 next 🡪 next 🡪 next 🡪next 🡪 Network ID - 172.31.19.227 (private IP) 🡪 next 🡪 next 🡪 finish

# Now, DNS 🡪 forward lookup zone 🡪 right click 🡪 guftgu.in click 🡪 last ( win-iqrb38fgifT) click 🡪 properties 🡪 update associated pointer 🡪 apply 🡪 OK

# Home Screen 🡪search network connection Or (press window+r 🡪 ncpa.cpi 🡪ok 🡪) Ethernet 🡪 open 🡪 properties 🡪 untick IPV6 🡪 select IPV4 double click 🡪 Preferred DNS server- Private IP of instance 🡪 OK 🡪OK 🡪 close

# Cmd 🡪ns lookup 🡪perfect!

# Server manager 🡪 tools 🡪 active directory users and computers 🡪 guftgu.in 🡪 user 🡪 right click 🡪 new 🡪 user 🡪 name- madan, logon name – madan 🡪 next 🡪 india@123 🡪 next finish

# 🡪 guftgu.in 🡪 user 🡪 right click 🡪 new 🡪 user 🡪 name- vikas , logon name – vikas 🡪 next 🡪 india@123 🡪 next finish

# Now go to AWS management console, search “ Directory Services “ 🡪 set up AD connector 🡪 AD connector 🡪 next 🡪 directory size – small 🡪 next 🡪 next 🡪 DNS name -guftgu.in -> DNS IP – server private IP -> account name – administrator -> service account pass – Yogesh@123(server pass) 🡪 next 🡪 create directory take 5 -10 min

# IAM 🡪 security 🡪 IAM user and role access to billing information – enable

# Go to IAM 🡪 role 🡪 select directory services 🡪next permission 🡪 EC2FullAccess 🡪 name – EC2fullaccess1 🡪 create role

# Go to IAM 🡪 role 🡪 select directory services 🡪next permission 🡪 Billing 🡪 name – Billinguser 🡪 create role

# Go to active directory -> guftgu.in 🡪 application acess URL 🡪 create 🡪 access url name –guftgu 🡪create 🡪 AWS management console 🡪 Enable

# Directories 🡪 delegate console access 🡪 billing user 🡪 manage users nad groups for this role 🡪 add🡪 find user by 🡪 type madan 🡪 add

# Directories 🡪 delegate console access 🡪 billing user 🡪 manage users nad groups for this role 🡪 add🡪 find user by 🡪 type vikas 🡪 add

# Directory guftgu 🡪 application access url (copy)

# Paste in new incognito 🡪 url/console

# Username – madan , password – india@123

# Madan is created for billing only , try to access other services it gives access denied

# Paste in new incognito 🡪 url/console

# Username – vikas , password – india@123

# vikas is created for EC2 only , try to access other services it gives access denied

# AWS RDS

1. RDS
2. Create Database 🡪 Choose a database creation method - Standard create 🡪 Engine options – MySQL 🡪 Templates – Free-Tier 🡪 Settings 🡪 DB instance identifier - database-1 🡪 Credentials Settings 🡪 Master username – admin 🡪 Credentials management - Self managed 🡪 Master password – Nayak1137 🡪 Confirm password – Nayak1137 🡪Instance configuration 🡪 DB instance class - Burstable classes – db.t2.micro 🡪 Storage 🡪 Storage type – General Purpose SSD(gp2) 🡪 Allocated storage – 20 🡪 Connectivity 🡪 Compute resource 🡪 Don’t connect to an EC2 compute resource (You can manually set up a connection to a compute resource later) 🡪 Network type – IPv4 🡪 Virtual private cloud (VPC) – Default 🡪 DB subnet group – Default 🡪 Public access – Yes 🡪 VPC security group (firewall) – create new 🡪 New VPC security group name – RDS-SG 🡪 Availability Zone – us-east-2c🡪 Certificate authority 🡪 default 🡪 Database port – 3306 🡪 Database authentication options 🡪 Password Authentication 🡪 monitoring - enable 🡪 Additional configuration 🡪 Database options -> Initial database name – db1 🡪 DB parameter group – default 🡪 Option group – default 🡪 Backups – disable 🡪 Maintenance 🡪 Enable auto minor version upgrade – enable 🡪 Maintenance window – No Preference 🡪 create database
3. Connection Details 🡪 Master username – admin & Master password - Nayak1137
4. Go to database-1🡪security group 🡪Inbound rule🡪Edit--> MySQL/Aurora anywhere & SSH - anywhere 🡪Save
5. Launch instance 🡪 name – Linux-Server -> OS – Amazon Linux -> storage type – t2.micro -> VPC – default –> subnet – us-east-2c (you can use other also) -> auto assign public IP – enable -> Security group – RDS-SG -> storage – default 🡪 launch Instance
6. Open cmd 🡪 cd Downloads
7. ssh -i "nayak-vm.pem" [ec2-user@ec2-18-218-50-154.us-east-2.compute.amazonaws.com](mailto:ec2-user@ec2-18-218-50-154.us-east-2.compute.amazonaws.com)
8. sudo su –
9. sudo yum install -y https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm
10. sudo yum install -y mysql-community-client
11. rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
12. Amazon linux 2023
13. sudo dnf update -y
14. sudo dnf install mariadb105-server
15. mysql -u MY\_USER -p`MY\_PASSWORD` -h MY\_HOST -P 3306
16. go to database-1 🡪 copy end point url and run like this below
17. mysql -h database-1.czioemm0cdip.us-east-2.rds.amazonaws.com -u admin -p db1
18. Enter password: Nayak1137
19. Succussfully login to MySQL

# AWS RDS – Window server

1. RDS
2. Create Database 🡪 Choose a database creation method - Standard create 🡪 Engine options – MySQL 🡪 Templates – Free-Tier 🡪 Settings 🡪 DB instance identifier - database-1 🡪 Credentials Settings 🡪 Master username – admin 🡪 Credentials management - Self managed 🡪 Master password – Nayak1137 🡪 Confirm password – Nayak1137 🡪Instance configuration 🡪 DB instance class - Burstable classes – db.t2.micro 🡪 Storage 🡪 Storage type – General Purpose SSD(gp2) 🡪 Allocated storage – 20 🡪 Connectivity 🡪 Compute resource 🡪 Don’t connect to an EC2 compute resource (You can manually set up a connection to a compute resource later) 🡪 Network type – IPv4 🡪 Virtual private cloud (VPC) – Default 🡪 DB subnet group – Default 🡪 Public access – Yes 🡪 VPC security group (firewall) – create new 🡪 New VPC security group name – RDS-SG1 🡪 Availability Zone – us-east-2c🡪 Certificate authority 🡪 default 🡪 Database port – 3306 🡪 Database authentication options 🡪 Password Authentication 🡪 monitoring - enable 🡪 Additional configuration 🡪 Database options -> Initial database name – db1 🡪 DB parameter group – default 🡪 Option group – default 🡪 Backups – disable 🡪 Maintenance 🡪 Enable auto minor version upgrade – enable 🡪 Maintenance window – No Preference 🡪 create database
3. Connection Details 🡪 Master username – admin & Master password - Nayak1137
4. Go to database-1🡪security group 🡪Inbound rule🡪Edit--> MySQL/Aurora anywhere & RDP,HTTP,HTTPS - anywhere 🡪Save
5. Launch instance 🡪 name – RDP-Server -> OS – Amazon Linux -> storage type – t2.micro -> VPC – default –> subnet – us-east-2c (you can use other also) -> auto assign public IP – enable -> Security group – RDS-SG -> storage – default 🡪 launch Instance
6. Connect RDP server
7. Server manager 🡪 Local server 🡪 IE enhanced security configuration – off 🡪 window defender firewall – off 🡪 close
8. Open internet explorer 🡪 search google 🡪 search 🡪 web platform installer 🡪 install this extension 🡪 run 🡪I accept 🡪 install 🡪 finish
9. Press window key 🡪 search web platform installer 🡪 application 🡪 search MySQL 🡪MySQL 5.5 add 🡪 install 🡪 password – Nayak1137 🡪 install 🡪 finish
10. go to database-1 🡪 copy end point url and run like this below
11. Open cmd 🡪 mysql -h database-1.czioemm0cdip.us-east-2.rds.amazonaws.com -u admin -p db1
12. Enter password : nayak1137
13. Login success to MySQL !

**DynamoDB**

DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.

DynamoDB charges for reading, writing, and storing data in your DynamoDB tables, along with any optional features you choose to turn on. DynamoDB has on-demand capacity mode and provisioned capacity mode, and these modes have pricing for processing reads and writes on your tables.

Pricing 🡪 1 RCU = 4KB (Strongly consistent) , 1 RCU = 8KB (eventually consistent) & 1 WCU = 1KB

1. DynamoDB
2. Create table 🡪 Table name – student\_data -> Partition key – roll\_no -> table settings – default -> Create table
3. Go to student\_data 🡪 action 🡪 create item 🡪 create 1st item
4. Attribute name – roll\_no -> value-1 -> type –> add new attributes – string
5. Attribute name – Name -> value- yogesh -> add new attributes - number
6. Attribute name – age -> value- 24 -> add new attributes - string
7. Attribute name – hobby -> value- kabaddi -> add new attributes - string
8. Attribute name – qualification -> value- BTech
9. Go to student\_data 🡪 action 🡪 create item 🡪 create 2nd  item
10. Attribute name – roll\_no -> value-2 -> type –> add new attributes – string
11. Attribute name – Name -> value- nitin -> add new attributes - number
12. Attribute name – age -> value- 22 -> add new attributes - string
13. Attribute name – hobby -> value- cricket -> add new attributes – string
14. Attribute name – qualification -> value- LLB -> add new attributes – number
15. Attribute name – Mobile\_no -> value- 12345
16. Go to student\_data 🡪 action 🡪 create item 🡪 create 3rd item
17. Attribute name – roll\_no -> value-3-> type –> add new attributes – string
18. Attribute name – Name -> value- sudarshan -> add new attributes - number
19. Attribute name – age -> value- 21 -> add new attributes - string
20. Attribute name – hobby -> value- cricket -> add new attributes – string
21. Attribute name – education -> value- BHMS
22. Go to student\_data 🡪 action 🡪 create item 🡪 create 1st item
23. Attribute name – roll\_no -> value-4 -> type –> add new attributes – string
24. Attribute name – Name -> value- ganesh -> add new attributes - number
25. Attribute name – age -> value- 23 -> add new attributes – string
26. Scam or query item 🡪 scan or run query to get data from table according to your convenient way.

**Route 53**

1. Route 53
2. Domain registration
3. Register Domain -> search for domain - banjaranayak.com -> select -> proceed to checkout -> Registrant contact – fill all the information 🡪 make payment and resister domain .
4. DNS management 🡪 create hosted zone 🡪 Domain name – banjaranayak.com -> type – public hosted zone 🡪 create hosted zone
5. It will create 4 Name server and 1 SOA

Not Eligible on free tier perform lab later!

**CloudFront**

1. Create Bucket with public read access –> name- yogesh1137 🡪 upload object (image) – yogesh.jpg
2. cloudFront 🡪 distribution 🡪 Origin domain 🡪 select amazon S3 🡪 name – yogesh-cloudfront 🡪 Cache policy and origin request policy (recommended)🡪 Cache policy – CachingOptimized -> Origin request policy –AllViewersandCloudFrontHeaders 🡪 WAF 🡪 Do not enable security protections 🡪 for all default settings🡪 create distribution
3. Go to distribution 🡪 copy domain name 🡪 paste in browser like this add file at last - <https://d2nu2a79lxg8gy.cloudfront.net/me.jpg>
4. Done ! you are accessing data with clodfront!

**AWS SQS**

1. SQS
2. Create Queue 🡪 Type – standard -> Name – test-queue 🡪 rest all default 🡪 create queue.
3. Go to test-queue 🡪 create and receive massages 🡪 massage body – Process MP4 file 🡪 send massage
4. Go to test-queue 🡪 create and receive massages 🡪 massage body – Publish video 🡪 send massage
5. Go to lambda
6. Create Function 🡪blueprint name – process massage in an SQS queue -> use a blueprint -> Execution role - Create a new role from AWS policy templates -> Role name - myQueueRole1 -> Policy templates – Amazon SQS poller permissions -> SQS queue – test-queue -> batch size -5 🡪 create function
7. Check SQS trigger is enable
8. Go to queue 🡪 test-queue 🡪 monitoring 🡪 Number of massages sent 🡪 expand and see 2 massages sent
9. Go to queue 🡪 test-queue 🡪 monitoring 🡪 Number of massages received 🡪 expand and see 2 massages received
10. Done !

**AWS SNS**

1. SNS
2. Topic name – Covid-19 -Warning –> Next
3. Type –standard 🡪 name – Covid-19 –Warning 🡪 rest all default 🡪 create topic
4. Create subscription 🡪 protocol – email 🡪 endpoint – [yogeshrrathpd1137@gmail.com](mailto:yogeshrrathpd1137@gmail.com) 🡪create subscription
5. Open received email on your phone and confirm
6. Go to your subscription and check status is confirmed
7. Open received email on your phone and unsubscribe
8. Go to your subscription and check status is deleted
9. Open received email on your phone and confirm
10. Go to your subscription and check status is confirmed
11. Select subscription and click on publish massage 🡪 subject – Covid-19-warning -> massage structure – identical payload for all delivery protocol -> massage body - REMINDER: Due to a few infractions last week, we want to remind all employees that wearing protective equipment, including masks, is essential to ensuring a COVID Safe workplace. If you do not, this may result in the closure of our facility again. Reply STOP to opt-out 🡪 publish massage
12. Open your email inbox in mobile 🡪 see the massage. 🡪 received!
13. Click on text massaging(SMS) 🡪 publish text massage 🡪 massage type – promotional 🡪 add phone number 🡪 enter phone number and select language 🡪 verify phone number 🡪 publish massage
14. Select mobile no --> click on publish text massage 🡪🡪 massage type – promotional 🡪destination phone number 🡪 massage – stay safe! 🡪 publish text massage.
15. Checked text massage on your phone – received!

**AWS NAT Instance configuration**

1. VPC
2. Create VPC 🡪 VPC Name - myVPC1 -> CIDR – 10.0.0.0/16 🡪 create VPC 🡪 action 🡪 Edit VPC setting 🡪 DNS hostname – Enable
3. Create subnet 🡪VPC ID – myVPC1 -> name – Public-Subnet1 -> Subnet CIDR – 10.0.0.0/24 -> availability zone – us-east-2a 🡪 create subnet
4. Create subnet 🡪VPC ID – myVPC1 -> name – Private-Subnet1 -> Subnet CIDR – 10.0.1.0/24 -> availability zone – us-east-2b 🡪 create subnet
5. Create internet Gateway -> name – myVPC1-IGW -> create Internet gateway 🡪 action 🡪 attach VPC – myVPC1 🡪 attach internet Gateway
6. Create route table 🡪 name – myVPC1-Public-RT 🡪 VPC – myVPC1 🡪 create route table
7. Go to myVPC1-Public-RT 🡪 edit routes 🡪 add route – 0.0.0.0/0 select Internet Gateway igw-myVPC1 🡪 save changes 🡪 subnet association 🡪 edit subnet association – select Public-Subnet1 🡪 save changes
8. Create route table 🡪 name – myVPC1-Private-RT 🡪 VPC – myVPC1 🡪 create route table
9. Go to myVPC1-Public-RT 🡪 subnet association 🡪 edit subnet association – select Private-Subnet1 🡪 save changes
10. EC2 🡪 launch instance 🡪 public-server1 🡪 browse more AMIs 🡪 community AMIs 🡪 search NAT 🡪 select first vpc-nat AMI (amazon linux) amzn-ami-vpc-nat-2018.03.0.20220907.3-x86\_64-ebs 🡪 VPC – myVPC1 -> subnet – Public-Subnet1 🡪 public IP – enable 🡪 security group 🡪 name- NAT-Instance-SG -> port – SSH, All ICMP IPv4 anywhere 🡪 launch Instance
11. EC2 🡪 launch instance 🡪 Private-server1 OS – amazon Linux 🡪 VPC – myVPC1 -> subnet – Private-Subnet1 🡪 public IP – Disable 🡪 security group 🡪 name- NAT-Instance-SG 🡪 launch Instance
12. Connect public-server1 with CMD 🡪 sudo su - 🡪 ping 8.8.8.8
13. Exit
14. Change directory to downloads
15. Two ways to Copy file from local to remote server
16. Open NAT-key.pem file -> copy all the content -> go to remote sever -> create file using Vi editor NAT-key.pem (same name as used key in private instance) paste all the content and save.
17. scp -i NAT-key.pem NAT-key.pem ssh -i "NAT-key.pem" [ec2-user@18.179.9.89:/home/ec2-user](mailto:ec2-user@18.179.9.89:/home/ec2-user)
18. cmd login with Public-Server1 🡪 sudo su –
19. ssh -i "NAT-key.pem" [ec2-user@10.0.1.191](mailto:ec2-user@10.0.1.191) 🡪 you will be login to your private server
20. Ping [www.google.com](http://www.google.com) 🡪 not having access to internet
21. Go to route table 🡪 myVPC1-Private-RT -> edit routes -> add routes – 0.0.0.0/0 select – instance – Public-Server1 🡪 save
22. Ping [www.google.com](http://www.google.com) 🡪 not having access to internet
23. EC2 Public-Server1 🡪 action 🡪 Networking 🡪 Change Source / destination check – stop 🡪 Save
24. Ping [www.google.com](http://www.google.com) 🡪 having access to internet, Done!

**AWS site to site VPN demo**

1. Select Mumbai Region
2. Create VPC 🡪 VPC Name – VPC-1 -> CIDR – 10.0.0.0/16 🡪 create VPC 🡪 action 🡪 Edit VPC setting 🡪 DNS hostname – Enable
3. Create subnet 🡪VPC ID – VPC-1 -> name – Public-Subnet1 -> Subnet CIDR – 10.0.0.0/24 -> availability zone – ap-south-1a 🡪 create subnet
4. Create internet Gateway -> name – myVPC1-IGW -> create Internet gateway 🡪 action 🡪 attach VPC – VPC-1 🡪 attach internet Gateway
5. Create route table 🡪 name – VPC1-Public-RT 🡪 VPC – VPC-1 🡪 create route table
6. Go to VPC1-Public-RT 🡪 edit routes 🡪 add route – 0.0.0.0/0 select Internet Gateway igw-AWS-side-IGW 🡪 save changes 🡪 subnet association 🡪 edit subnet association – select Public-Subnet1 🡪 save changes
7. Select Singapore Region
8. Create VPC 🡪 VPC Name – CustomerEndVPC -> CIDR – 10.1.0.0/16 🡪 create VPC 🡪 action 🡪 Edit VPC setting 🡪 DNS hostname – Enable
9. Create subnet 🡪VPC ID – CustomerEndVPC -> name – Public\_Subnet1 -> Subnet CIDR – 10.1.0.0/24 -> availability zone – ap-southeast-1a 🡪 create subnet
10. Create internet Gateway -> name – Customer-IGW-> create Internet gateway 🡪 action 🡪 attach VPC – VPC-1 🡪 attach internet Gateway
11. Create route table 🡪 name – Customer-Public-RT 🡪 VPC – CustomerEndVPC 🡪 create route table
12. Go to Customer-Public-RT 🡪 edit routes 🡪 add route – 0.0.0.0/0 select Internet Gateway igw- Customer-IGW🡪 save changes 🡪 subnet association 🡪 edit subnet association – select Public\_Subnet1 🡪 save changes
13. Launch instance (Singapore region) 🡪 name- customer-server -> OS – Amazon Linux -> VPC – CustomerEndVPC -> subnet –Public\_Subnet1 -> Auto assign IP – enable -> Security group – VPN-SG -> SSH, ALL TCP, ALL ICMP – anywhere -> launch Instance
14. Come back to Mumbai Region
15. Virtual Private Gateway 🡪 create virtual private gateway 🡪 name – AWS-Side-GW 🡪 Autonomous System Number (ASN) - Amazon default ASN 🡪 create virtual private gateway 🡪 action 🡪 attach to VPC - VPC-1 🡪 attach
16. Customer Gateway 🡪 create Customer Gateway 🡪 name – CG-AWS-Side 🡪 IP address – (paste IP Address of Singapore EC2 instance i.e. Customer-Server) 🡪 create Customer Gateway
18. Site-To-Site VPN connection 🡪 create VPN connection 🡪 name - mumbai-singapore-vpn -> target – virtual private gateway -> select virtual private gateway - AWS-Side-GW -> customer Gateway – existing -> customer Gateway ID – select – CG-AWS-Side -> routing options – static -> Static IP prefixes – 10.1.0.0/24 (Singapore subnet) 🡪 create VPN connection
19. Select mumbai-singapore-vpn 🡪 download configuration 🡪 vendor – generic 🡪 download
20. Route table 🡪 VPC1-Public-RT 🡪 route propagation 🡪 edit route propagation 🡪 enable 🡪 save
22. Go to Singapore
23. Connect with cmd customer-server 🡪 sudo su –
24. Install vpn using command 🡪 yum install openswn –y

**Lambda Function**

1. IAM 🡪 Roles
2. Create role 🡪 AWS service 🡪 Use Case – Lambda 🡪 Next 🡪 Permission policy – DynamoDBFullAccess 🡪 Next 🡪 Role name – lambda-for-dynamodb 🡪 Create Role
3. Lambda 🡪 Create Function 🡪 Author from Scratch 🡪 Function name – Lambda1 🡪 Runtime 🡪 Python 3.9 🡪 Execution role – choose an existing role – lambda-for-dynamodb 🡪 Create Function
4. Lambda1 🡪 Code 🡪 remove existing code and paste this code 🡪

import boto3

from uuid import uuid4

def lambda\_handler(event, context):

s3 = boto3.client("s3")

dynamodb = boto3.resource('dynamodb')

for record in event['Records']:

bucket\_name = record['s3']['bucket']['name']

object\_key = record['s3']['object']['key']

size = record['s3']['object'].get('size', -1)

event\_name = record ['eventName']

event\_time = record['eventTime']

dynamoTable = dynamodb.Table('newtable')

dynamoTable.put\_item(

Item={'unique': str(uuid4()), 'Bucket': bucket\_name, 'Object': object\_key,'Size': size, 'Event': event\_name, 'EventTime': event\_time})

1. File 🡪 Save
2. S3 🡪 create bucket 🡪 name – yogesh1137 (public) 🡪 create bucket
3. Lambda1 🡪 Add trigger 🡪 S3 🡪 bucket – yogesh1137 🡪 Event type – All object create events 🡪 Checked box -> Add
4. DynamoDB 🡪 Create Table 🡪 Table name – newtable (same as in the code) 🡪 Partition key – unique (same as in the code) 🡪 Create Table
5. Go to newtable 🡪 Item 🡪 check if there is any items
6. S3 🡪 yogesh1137 🡪 upload any file 🡪 upload
7. DynamoDB 🡪 Go to newtable 🡪Explore Item 🡪 check if there is any items 🡪 Yes, Done !
8. S3 🡪 yogesh1137 🡪 upload any file 🡪 upload
9. DynamoDB 🡪 Go to newtable 🡪Explore Item 🡪 check if new there is any items 🡪 Yes, Done !