IAM:-

Policy

1. Create a policy to access all objects in a specific bucket.

   Step1: - create a 2 bucket
   Step2: - create a user
   Step3: - create a policy
   IAM-→policy→create→service(s3)→list, read, write all check→
   add bucket ARN→next→name→create policy
   step4: - Attach policy to user
   step5: - user log in and check s3

##

2. Create a policy to access only 2 objects in a specific bucket.

   Step 1: - Create Bucket and put Object
   Step 2: - Create IAM User and attach Json Policy

```
        {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "NotResource": [
                "arn:aws:s3:::my-yash-bucket/Object-1.jpeg",
                "arn:aws:s3:::my-yash-bucket/Object-2.jpeg"
            ]
        },
        {
            "Effect": "Deny",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::my-yash-bucket/Object-3.jpeg",
                "arn:aws:s3:::my-yash-bucket/Object=4.jpeg"
            ]
        }
    ]
```

   Step 3: - Make Bucket & Object Public
   Step 4: - Login IAM user and go through s3 bucket
   https://s3.console.aws.amazon.com/s3/buckets/bucket-name
   Step 5: - Go through Object and Check Object open or Download

##

3. Create a policy to deny the access of Specific bucket.

Step: - Create user in S3 full access
Step: - Create a Bucket
Step: - Create Bucket Policy
S3 → bucket → permission → Bucket Policy → Edit → Policy Generator → Select Policy type s3 → Effect (Deny) → Principal (User name or ARN) → Action (All select) → ARN (bucket arn) → save → Generate policy
Copy json code and paste bucket policy
Step: - Login user and check the bucket

##########

4. Create policy to create the user only and can attach policy to them.

Step: - cerate a policy to that specific user

```
        {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateUser",
        "iam:CreatePolicy",
        "iam:AttachUserPolicy",
        "iam:ListAttachedUserPolicies"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListPolicies",
        "iam:ListUsers",
        "iam:GetPolicy",
        "iam:GetUser",
        "iam:ListAccessKeys",
        "iam:ListGroups"
      ],
      "Resource": "*"
    }
  ]
}
```

Step: - Create a IAM user and attach the policy to that specific user
Step: - Login to this user

Step: - Then check the user has been able to create user and attach to policy to that created user.

5. Create a policy to place the user in a group only.

Step: - Create a one IAM user

Step: - **Create a policy with following permission**

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"iam:AddUserToGroup",
"iam:GetGroup",
"iam:ListGroups",
"iam:GetUser",
"iam:ListGroupsForUser",
"iam:GetGroupPolicy",
"iam:ListAttachedGroupPolicies",
"iam:ListGroups",
"iam:ListGroupPolicies",
"iam:ListUsers",
"iam:RemoveUserFromGroup"
],
"Resource": "*"
}
]
}
```

Step: - attach above policy to that user

Step: - Login with that user which policy has been attached

Step: - Check with adding some members in already created group

Note: - This user is not permission to create a group they can edit already created group.

6. Policy to get "read" access of all users, groups but not policy.

Step: - create a policy with following json description whose user can have only group and user read access

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers",
        "iam:GetUser",
        "iam:ListGroups",
        "iam:GetGroup",
        "iam:ListGroupPolicies",
        "iam:ListAttachedGroupPolicies",
        "iam:GetGroupPolicy",
```

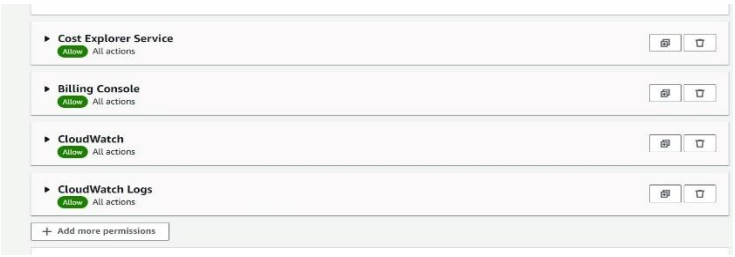
```
        "iam:ListGroupsForUser",
        "iam:ListAccessKeys",
        "iam:ListSSHPublicKeys",
        "iam:ListServiceSpecificCredentials",
        "iam:ListSigningCertificates",
        "iam:ListMFADevices",
        "iam:GetLoginProfile",
        "iam:GetServiceLastAccessedDetails"
      ],
      "Resource": "*"
    }
  ]
}
```

Step: - Attach above policy to that specific user
Step: - Then login with that user and check with creating group and user

7. Policy to get access to billing, ec2 and cloudwatch.

Step: - Create a policy
Step: - Attach following json able to access EC2, Billing, CloudWatch

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "aws-portal:*",
        "ce:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "logs:*"
      ]
      "Resource": "*"
    }
  ]
}
```



Step: - Attach above policy to that specific user and login with console for check the specified permissions to that user

8. How to give access only to northern virginia.

   Add 4 users >> 2 groups (devops & cloud) >> cloud member will access their password will only no & devops group there no will password standards
   >> cloud group SE3 full access % other EC2 full access >> Devops EC2 full access in north virginia.

   Step: - Create 4 empty permission user
   Step: - Create 2 groups (Devops, Cloud)
   Step: - Create a policy for devops group

```
      {
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "ec2:*",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "aws:RequestedRegion": "us-east-1"
    }
   }
  }
 ]
}
```

   Step: - Create a policy for Cloud group

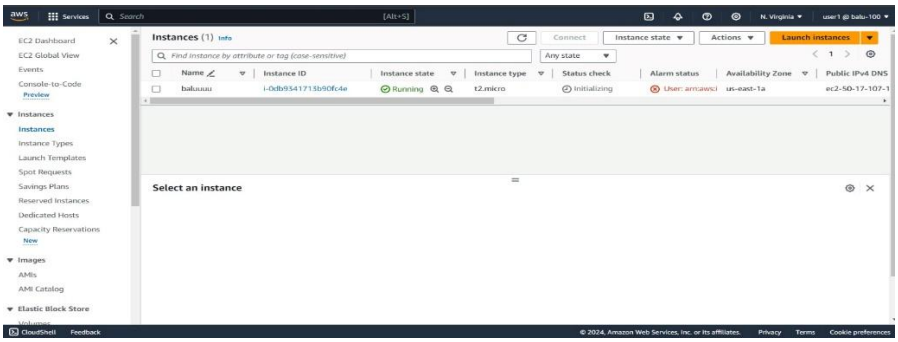
```
  {
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "s3:*"
   ],
   "Resource": "*"
  },
  {
```

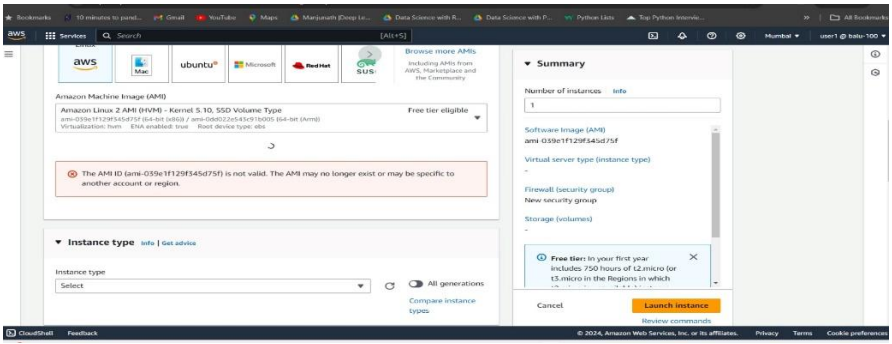
```
      "Effect": "Allow",
      "Action": [
        "ec2:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:Region": "us-east-1"
        }
      }
    }
  ]
}
```

Step: - Create a 1 policy attach all group for standard password.

```
        {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "iam:ChangePassword",
      "Resource": "*"
    }
  ]
}
```

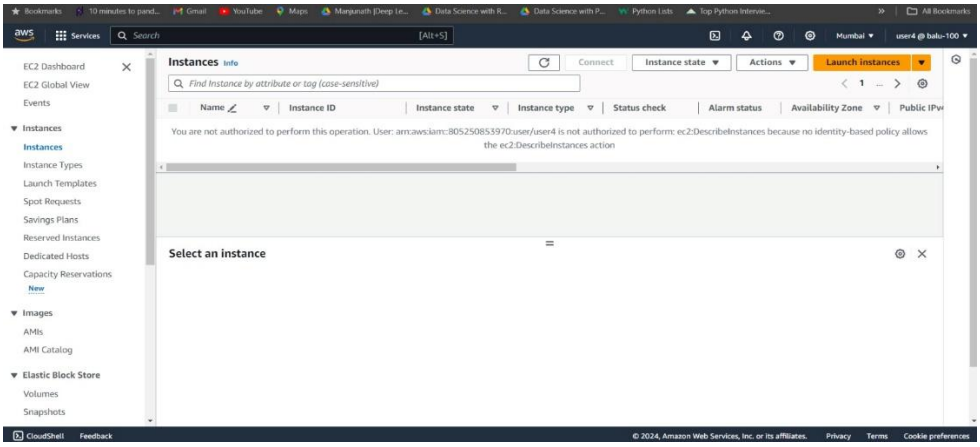
Step: - Login with devops group user

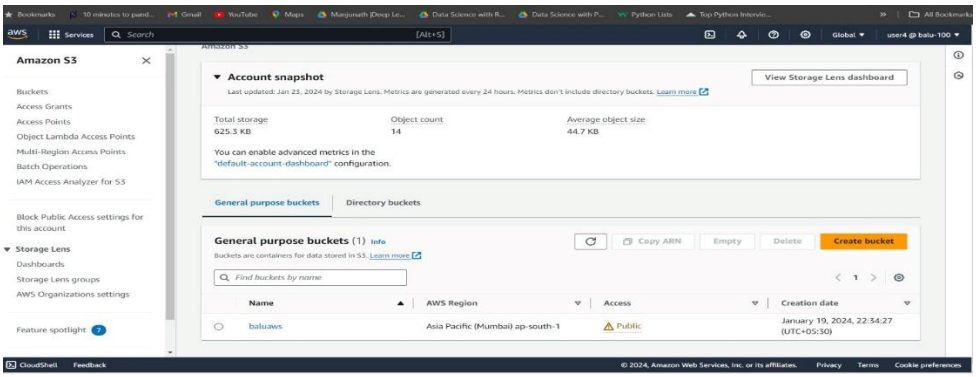




<u>NOTE</u>: - This image shows the devops group member access the Ec2 service with N. Virginia only and second image shows the other region not access.

Step: - Login with cloud group user





NOTE: -The first image shows the EC2 other region not having access to cloud group members excepts N. Virginia and second image shows the having s3 full access.

9. Create a policy like when users are login into console, Without MFA not a single IAM user has able to access any AWS kind of aws services, he gots an permission denied error.
10. Create a resource based policy and attach to S3_B16 Bucket & only sunny user can able to access that bucket.

User:-

1. Add four different users in the organisation account.

Step: - Create 4 different user in organisation account.

IAM → User → create user → name → password → next → permission → save

2. Add one user without permission.

IAM → User → create user → name → password → next → save

3. Add user with console & **programmatic user**.

IAM → User → create user → name → password → next → save

- aws iam create-login-profile --user-name YourUserName --password YourPassword

4. If IAM users are created w/o password, then how can you set the password of that IAM user as an Admin. (W/O deleting user)

Step: - create user w/o permission
Step: - create user Access key and secret access key
Step: - Create Ec2 instance and take SSH
Step: - awscli configure and set password to this command

- aws iam create-login-profile --user-name YourUserName --password YourPassword

5. Add User Without Any permission (permission denied)

IAM → User → create user → name → password → next → save

6. S3 Read Only Access

Step: - Create IAM user
Step: - create policy to s3 only read access
Step: - Attach the policy to this user
Step: - Login to this user and check the s3 bucket

7. Jarvis Admin Access

Step: - Create Jarvis name user
Step: - Attach Administrator policy to this user
Step: - Login to this user and check this user can use many services like as admin.

8. Add User with Programmatic Access

- aws iam create-user --user-name YourUserName

- aws iam create-login-profile --user-name YourUserName --password YourPassword

##

S3:-

1. Implement MFA on Bucket when users are going to delete any objects, then he needs an Approval from MFA Code.

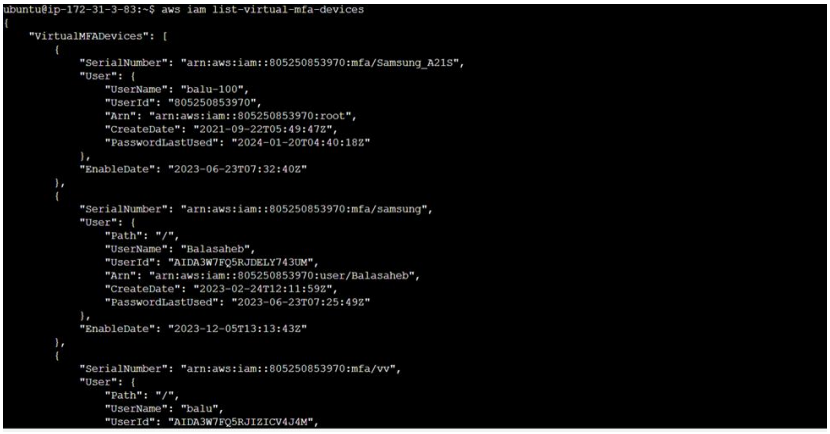
   Note: - Required Root AWS account & MFA enable

   Step 1: - Create Root Access key
   → Go through AWS Right side Console security credential Create Access Key
   Step 2: - Create s3 bucket and put object and Bucket versioning enable.
   Step 3: - Launch Ec2 Instance
   Step 4: - Create Role and attach to instance
   - IAM → role → create → Select Ec2 Service → s3 full access permission → name → save
   - Select instance → Action → Security → modify IAM role → select role → save

   Step 5: - Take SSH and AWS configure
   - Sudo -i
   - apt update
   - apt install awscli -y
   - aws configure

   paste Access key → Secret Access Key → enter → enter

   Step 6: - list virtual device
   - aws iam list-virtual-mfa-devices



Step 7: - To turn on MFA delete, run the [put-bucket-versioning](#) command

```
aws s3api put-bucket-versioning --bucket mybucketname --versioning-configuration
MFADelete=Enabled,Status=Enabled --mfa "arn:aws:iam::(accountnumber):mfa/root-
account-mfa-device (pass)"
```

Example: -

aws s3api put-bucket-versioning --bucket baluaws --versioning-configuration MFADelete=Enabled,Status=Enabled --mfa "arn:aws:iam:: 805250853970:mfa/Samsung 557245"

Step 8: - Check Bucket versioning

- aws s3api get-bucket-versioning –bucket (bucket-name)

Example: -

- aws s3api get-bucket-versioning --bucket my-yash-bucket

Step 9: - Make sure Enable the Show Versions, If you used normal cli command to delete bucket object then object will delete but object version will create in bucket, if you want to delete proper object then use this command.

```
      aws s3api delete-object --bucket mybucketnme --key myobjectkey --version-id
3HLkqCxf3vjVBH40Nrjkd --mfa "arn:aws:iam::(accountnumber):mfa/root (pass)"
```

Example: -

aws s3api delete-object --bucket baluaws --key Object-1.jpeg --version-id null --mfa "arn:aws:iam:: 805250853970:mfa/samsung 634495"

Note: - If you want MFADelete=Disable then use this command

aws s3api put-bucket-versioning --bucket YOUR_BUCKET_NAME --versioning-configuration Status=Suspended,MFADelete=Disabled --mfa "{serialNumber}(space){tokenCode}"

Example: -

aws s3api put-bucket-versioning --bucket my-yash-bucket --versioning-configuration Status=Suspended,MFADelete=Disabled --mfa "arn:aws:iam:: 805250853970:mfa/Samsung 524458"

##########

2. Create a Bucket with cli commands & upload the object from **cli mode.**

Create a Bucket with cli command:

- aws S3 mb S3://Bucket-name

Upload the object from cli mode:

- aws s3 cp file-name s3://Bucket-name/file-name

##########

3. Complete one project with server less static website hosting, with deployment & after deployment management of the website in backend side. Make a proper document of that project.

Step: - Create a s3 bucket
Step: - Download CSS template and extract
Step: - Upload the CSS template in s3 bucket
Step: - make static website hosting enable
Step: - make a Bucket ACL enable and block all public access OFF
Step: - Go through ACL and Everyone (public access) Read, List permission
Step: - Copy CSS template Object and paste this bucket
Step: - Select all object and make public using ACL

Note: - If your all steps are complete then your static website hosted and copy URL and check the website.

AWS Cloud Trail:-

1. Configure and enable the trail for multi region in your both staging and production account.
   - **Step: - Create Trail Staging account.**
   - Cloud trail → cerate → name → select bucket → KMS encryption → next →select events → select data event → next → create
   - **Step: - Create trail as production account same as staging trail step.**
   - **Step: - Check s3 bucket logs will created or not.**

2. Configure and enable the trail for the only Mumbai region in your both staging and production account.
3. Create a trail for one service S3 by default it enables all logging features, you guys disable some features of logging.

AWS SNS:-

1. Configure SNS with s3 static website.

   Step: - Create SNS topic
   SNS → select type (Standard) → name → create
   Step: - Create Subscription of this topic
   SNS → Go through this topic →Create Subscription → Select Protocol (Email) → Endpoint (type email id) → Create

   GO Gmail index and confirm subscription

   Step: - Configure SNS
   S3 → bucket → Properties → Create Event Notification → name → select event → select SNS → select topic name → save

2. Configure SNS with Email server level notification with Load Balancer configuration.

   Step: - Create SNS topic
   SNS → select type (Standard) → name → create
   Step: - Create Subscription of this topic
   SNS → Go through this topic →Create Subscription → Select Protocol (Email) → Endpoint (type email id) → Create

   GO Gmail index and confirm subscription

   Step: - Create load balance and attach SNS topic.

3. Configure the SNS with SMS level.

   Step: - Create SNS topic
   SNS → select type (Standard) → name → create
   Step: - Create Subscription of this topic
   SNS → Go through this topic →Create Subscription → Select Protocol SMS → Add phone no. → type phone no. → select language → add → inter verification code → verify → select topic → select endpoint (phone no.) → create
   Step: - Configure SNS
   S3 → bucket → Properties → Create Event Notification → name → select event → select SNS → select topic name → save
   NOTE: - If you any put and get in bucket then send you a notification on your mobile.

AWS EC2:-

1. Create an Application Load Balancer with mobile, laptop, tablet, clothes, shoes pages?

Step 1:- Create a servers with the appropriate user data.

Server-App Homepage

```
#!/bin/bash
sudo -i
yum update all -y
yum install httpd -y
echo "hello world, welcome to my Homepage $HOSTNAME" > /var/www/html/index.html
systemctl start httpd.service
systemctl enable httpd.service
systemctl status httpd.service
```

Server- App Tshirt

```
#!/bin/bash
sudo -i
yum update all -y
yum install httpd -y
sudo mkdir -p /var/www/html/tshirt/
```

echo "hello world, welcome to my tshirt web-page $HOSTNAME" >
/var/www/html/tshirt/index.html
systemctl start httpd.service
systemctl enable httpd.service
systemctl status httpd.service

Server- App Mobile

#!/bin/bash
sudo -i
yum update all -y
yum install httpd -y
mkdir -p /var/www/html/mobile/
echo "hello world, welcome to my mobile web-page $HOSTNAME" >
/var/www/html/mobile/index.html
systemctl start httpd.service
systemctl enable httpd.service
systemctl status httpd.service

Server- App earphones
#!/bin/bash
sudo -i
yum update all -y
yum install httpd -y
mkdir -p /var/www/html/earphones/
echo "hello world, welcome to my earphones web-page $HOSTNAME" >
/var/www/html/earphones/index.html
systemctl start httpd.service
systemctl enable httpd.service
systemctl status httpd.service

Step 2:- Create the all Target Groups based on there web pages & associate the all including servers in target group.

TG-Homepage

Step 3: - Create a Application Load Balancer

Step 4: - Configure the Listner Rule in Listners (Protocol 80 ---> Rules 3)

Step 5: - Check the all TG are Healthy or not

Step 6: - Hit the DNS of ALB and try Path based routing

2. Create a Simple Application Load Balancer with Auto scaling and host any free css template.

Step: - Create a Launch template with user data

```
#!/bin/bash
Sudo -i
apt upadate
apt install nginx -y
systemctl start nginx.service
systemctl start nginx.service
echo " hallo this is cbz &HOSTNAME" > /var/www/html/index.nginx-debian.html
systemctl restart nginx.service
```

Step: - Create Auto scaling group

EC2 → ASG → name → select template → next → Network (VPC, availability zone select) → next → next → Group size (desired-1) → scaling (min, max) → Automatic scaling (target tracking scaling policy) → matrix type (average cpu, target value '80', warmup '1s') → next → Add notification → next → create

Note: - If auto scaling create then create a by default desired instance.

Step: - Create application load balance

Configure the Listner Rule in Listners (Protocol 80 ---> Rules 3)

Step: - Attach application load balance to auto scaling group

Ec2 → ASG select → Action → Edit → select Application load balance → save

Step: - Take instance SSH and configure 'stress' command

Step: - Copy ALB DNS and hit.

3. Create a simple classic Load Balancer.

Step 1: - Create a instance

Server-1

```
#!/bin/bash
sudo -i
sudo apt install nginx -y
sudo systemctl start nginx.service
sudo systemctl enable nginx.service
sudo systemctl status nginx.service
sudo echo "Hello CDECB2025 People look my IP $HOSTNAME from server-1" > /var/www/html/index.nginx-debian.html
sudo systemctl restart nginx.service
```

Server-2

```
#!/bin/bash
sudo -i
sudo apt install nginx -y
sudo systemctl start nginx.service
sudo systemctl enable nginx.service
sudo systemctl status nginx.service
sudo echo "Hello CDECB2025 People look my IP $HOSTNAME from server-2" > /var/www/html/index.nginx-debian.html
sudo systemctl restart nginx.service
```

==========

Server-3

```
#!/bin/bash
sudo -i
sudo apt install nginx -y
sudo systemctl start nginx.service
sudo systemctl enable nginx.service
sudo systemctl status nginx.service
sudo echo " Hello CDECB2025 People look my IP $HOSTNAME from server-3" > /var/www/html/index.nginx-debian.html
sudo systemctl restart nginx.service
```

Step 2: - Create a Classic Laod Balancer and check sequrity group to add http port.

Step 3: - Copy CLB DNS and hit.

4. Create an AMI Image, with name(tomcat-ami-v1) from an existing running tomcat server.

Step: - Now the Existing running tomcat server

Step: - Create an AMI image

EC2 → Select instance → action → image & templet → create image → name (tomcat-ami-v1) → tag image and snapshot together → create

Note: - If you create a AMI image that means you create a instance all data backup.

5. Host any customised HTML web page in Ubuntu server.

Step: - Create a EC2 instance with user data.

```
#!/bin/bash
Sudo -i
apt update
apt install nginx -y
systemctl start nginx.service
systemctl start nginx.service
echo “ hallo this is cbz &HOSTNAME” > /var/www/html/index.nginx-debian.html
systemctl restart nginx.service
```

Step: - Copy instance public Ip and hit the show the web page.

6. Host any customised HTML web page in RedHat 9 server.

Step: - Create a EC2 instance with user data

```
#!/bin/bash
sudo -i
yum update
yum install httpd -y
echo "hello world, welcome to my earphones web-page $HOSTNAME" >
/var/www/html/index.html
systemctl start httpd.service
systemctl enable httpd.service
systemctl restart httpd.service
```

Step: - Copy instance public Ip and hit the show the web page.

7. Host any customised HTML web page in windows 11 server.

Step: - Create instance select windows image

Step: - Go through instance and connect through RDP client

Select instance → Connect → RDP client → Get password → Upload selected instance pem key → Decrypt password → Download remote desktop file → copy password → open downloaded file and paste password

Step: - Go through downloaded file → connect → paste password → connect

Step: - Go through internet explorer block popup window to use go through server manager →turn off IE enhance security configuration

Step: - server manager → Dashboard → tools → Add roles and feature → next → next → next → server roles → click on web server (IIS) → next → next → install → close

Step: - click on dash board → tools → internet information services manager → click on sites → click on right side explore →

Step: - Download free CSS template and extract

Step: - Copy all data into /localdisk(C)/interpub/wwwroot
Step: - Browse http to hit the web page

8. Create a one EC2 Instance with debian family, create an 1000 no of jarvis.txt files and mount the existing S3 bucket into the same server and do upload all the jarvis.txt files into s3 bucket using aws command line tool.

   Step 1: - Create IAM user with s3 full access
   Step 2: - Create user Access key & Secret Access key
   Step 3: - create s3 bucket and put object
   Step 4: - Create ec2 Instance and take SSH
   Step 4: - Create Role and attach Ec2 Instance

   - IAM → role → create → Select Ec2 Service → s3 full access permission → name → save
   - Select instance → Action → Security → modify IAM role → select role → save

   Step 5: -

   - Sudo -I
   - apt-get update
   - apt-get install s3fs -y

   Step 6: - Redirect USER Access key & Secret Access Key

   - echo "ACCESS_KEY:SECRET_KEY" > $HOME/.passwd-s3fs
   - chmod 600 $HOME/.passwd-s3fs

   Step 7: - Create Directory and Mount s3 bucket in this Directory

   - sudo s3fs YOUR_S3_BUCKET_NAME /mnt/s3 -o passwd_file=$HOME/.passwd-s3fs

   Step 8 : - Show all bucket object in mounted folder and you can upload the file in bucket In cli command

   - cp <object-name> <mount directory>

##

9. Do task number 1 in **AWS EC2** in scripting. Automate the whole server level configuration.

   If you Instance Launch then type User Data

   #!/bin/bash

Sudo -i
apt update
apt install nginx
Systemctl start nginx.service
Systemctl enable nginx.service
cd /var/www/html/
Echo " hi this is cbz $HOSTNAME " > index.html
Systemctl restart nginx.service

##

10. If your server lost the key-pair, and you wanna take a ssh of that server, then how can you recover the key and take a ssh. (Want the practical, with documentation)

Step 1: - Create Instance (Old-instance)
Step 2: - Take SSH make a directory in / ex: - mkdir /yash touch /yash/file{1..100}
Step 3: - Create new instance (new-instance)
Step 4: - old instance key delete and stop the instance
Step 5: - old server volume detach and the attach to this volume in new server
Step 6: - Take SSH in New instance and mount the volume
Lsblk
Lsblk -lt
mount -t ext4 /dev/xvdf1 /mnt
cat /home/ubuntu/.ssh/authorized_keys >> /home/ubuntu/.ssh/authorized_keys
umount /mnt
Step 7: - Stop new instance and old instance volume detach and attach to old Instance (change Device name " /dev/sda1") and save
Step 8: - take SSH in old server in new key and check data.

##

AWS VPC:-

1. Create a 5 VPC with different networks.

Step 1: - Create a 5 VPC in different region

VPC → Create VPC → VPC only → name → Ipv4 CIDR manual input → 192.168.0.0/26 → Create VPC

(VPC only – create manually, VPC& more – AWS create automatically VPC)

Step 2: - Create Subnet

VPC → Subnet → Create → Select VPC Id → name → Availability zone select → CIDR Block (192.168.0.0/28) → Save

Step 3: - Create gateway and attach to VPC

VPC → Internet gateway → create → name → save

Select VPC → Action → Attach to VPC → Select VPC → Attach

Step 4: - Attach route table to gateway

VPC → Route table → select table → Action → Edit route → Add route → (Destination – 0.0.0.0/0, Target – Internet gateway) → select gateway → save

Step 5: - Create Instance and take SSH to cli and check Network proper work or not. (Security group → All ICMP-Ipv4 → 0.0.0.0/0 → save)

2. Create a customised VPC and enable the DHCP & DNS option set.
3. Enable the flow logs of newly created custom VPC.

Select VPC → Action → Create Flow Log → Name → Filter (All) → Maximum aggregation interval (10 min) → Destination (AWS s3 Bucket) → Type s3 Bucket ARN → log record Format (AWS "default" format) → Log File Format (Text Default) → Partition logs by time (Every 24 Hours) → Create

4. Create a Network Load Balancer (Need proper flow diagram, with explanation) *

Step: - Create Ec2 3 Instance with web page

Step: - Create Network load balance

EC2 → LB → create → select NLB → name → select vpc → ****

5. Create a 3 VPC & do the peering between 3 VPC.

Step 1: - Create 3 VPCs

Step 2: - Create subnet in All VPCs

Step 3: - Create Particular VPCs Internet Gateway and Attach to VPCs

Step 4: - Attach internet gateway to Route Table

Step 5: - Create Transit Gateway

VPC → Transit gateway → Create → Name → Type ASN (45000000000) → Select Auto Accept Share Attachment Box → Create

Step 6: - Create Transit Gateway Attachment in VPC1 VPC2 VPC3

VPC → Transit Gateway Attachment → Name → Select transit gateway ID → Attachment type Select (VPC) → Select VPC ID →Create

Step 7: - Create 3 Instance in Different VPCs

Step 8: - Add Routes in VPCs Route table

- **VPC-1 → RT-1** → Add →VPC 2 & VPC 3 CIDR → Select Destination (Transit Gateway) → Save
- **VPC-2 → RT-2** → Add →VPC 1 & VPC 3 CIDR → Select Destination (Transit Gateway) → Save
- **VPC-3 → RT-3** → Add →VPC 1 & VPC 2 CIDR → Select Destination (Transit Gateway) → Save

Step 9: - Add Inbound Rule in Instances Security Group

- **Instance 1 → SG** → Add → Select (Custom ICMP – IPV4) → Add VPC 2
  - VPC 3 CIDR → Save
- **Instance 1 → SG** → Add → Select (Custom ICMP – IPV4) → Add VPC 2
  - VPC 3 CIDR → Save
- **Instance 1 → SG** → Add → Select (Custom ICMP – IPV4) → Add VPC 2
  - VPC 3 CIDR → Save

Step 10: - Take SSH and PING the Connection and Jump the Private Server And PING, the Connection.

6. Access the all S3 Buckets in Particular single customised VPC w/o internet, so how can we achieve it?

Step 1: - Create VPC
Step 2: - Create 2 Subnet (1 Public, 1 Private)
Step 3: - Create gateway and attach to VPC
Step 4: - Attach default route table to gateway
Step 5: - Create 2 Instances (1 Public, 1 Private)
NOTE: - If Instances Create then go through Security Group and Add →All ICMP Ipv4 → 0.0.0.0/0 →Save
Step 6: - Create Route Table (Ex: - RT -2)
Step 7: - Copy 'pem' Key data and take a SSH to Public Server and Create a new 'pem' file and paste the data and take a SSH to Private Server. (Jump public server To Private Server)
Step 8: - Create NAT Gateway use to Public Subnet
VPC → NAT Gateway → Name →Select Public Subnet → Select Connectivity Type Public → Elastic Ip Allocate → Create
Step 9: - Private Subnet Associate with the Route Table (RT -2)
VPC → RT -2 → Subnet Associate → Edit → Select Private Subnet → Save
Step 10: - NAT Gateway Attach to the Route table (RT -2)
VPC → Route Table → Edit Route → Add → 0.0.0.0/0 →Select NAT gateway → Save
Step 11: - Download AWS Cli Package and Configure

- curl -O AWS URL
- Sudo apt install unzip
- Unzip aws Package
- Sudo ./aws/install

Step 12: - Delete NAT Gateway and Route Table (RT -2)
Step 13: - Create Role use of EC2 and access the S3 Bucket in Cli
Iam → Role → Create → Select AWS Service → Select use case 'EC2' → next → Select Permission Policy 'S3 Full access' → Role name → Create

Step 14: - Attach Iam Role to Private Server
EC2 → Select Private Server → Action → Security → Modify Iam Role →Select Role → save

Step 15: - Create Endpoint in S3 Gateway
VPC → Endpoint → Create → Name → Select AWS Service → Select S3 Service Gateway → Select VPC → Select Route Table 'Default' → Create

Step 16: - Check to Cli S3 Bucket (aws S3 ls)