

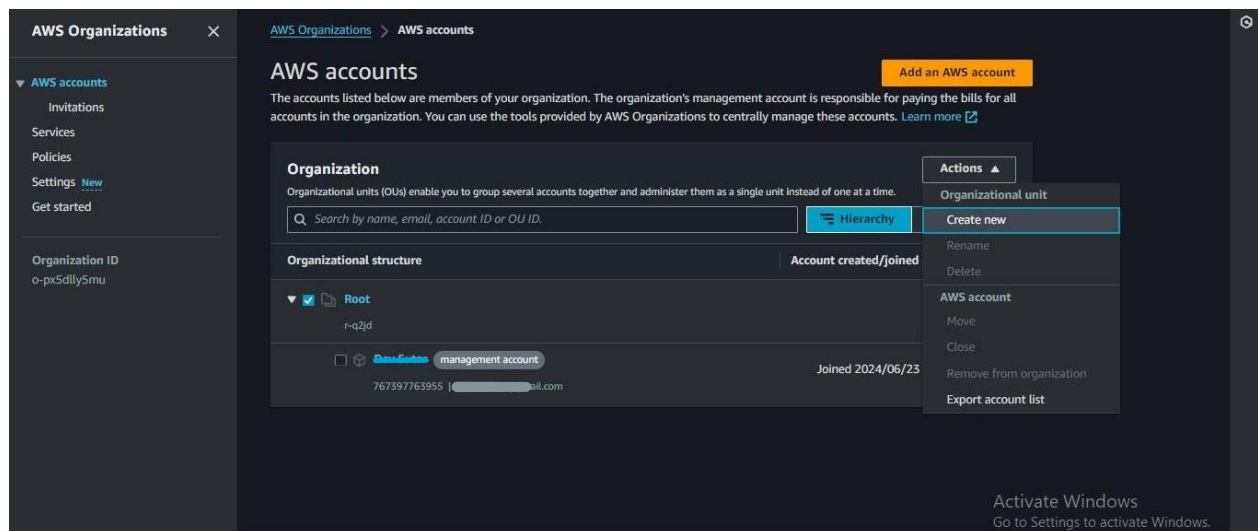
Task 1: Set up AWS Organizations for managing multiple AWS accounts.

In AWS Organization, with the help of organization we will be able to create a multiple account within AWS.

Suppose there are two Account that is DEVELOPING Account and TESTING account, if developer is working on the Developing account and doing some developing work, so will not effect on Testing Account.

And also same for Tester if tester is working on the Testing account and doing some work related to testing so will not effect on Developing Account

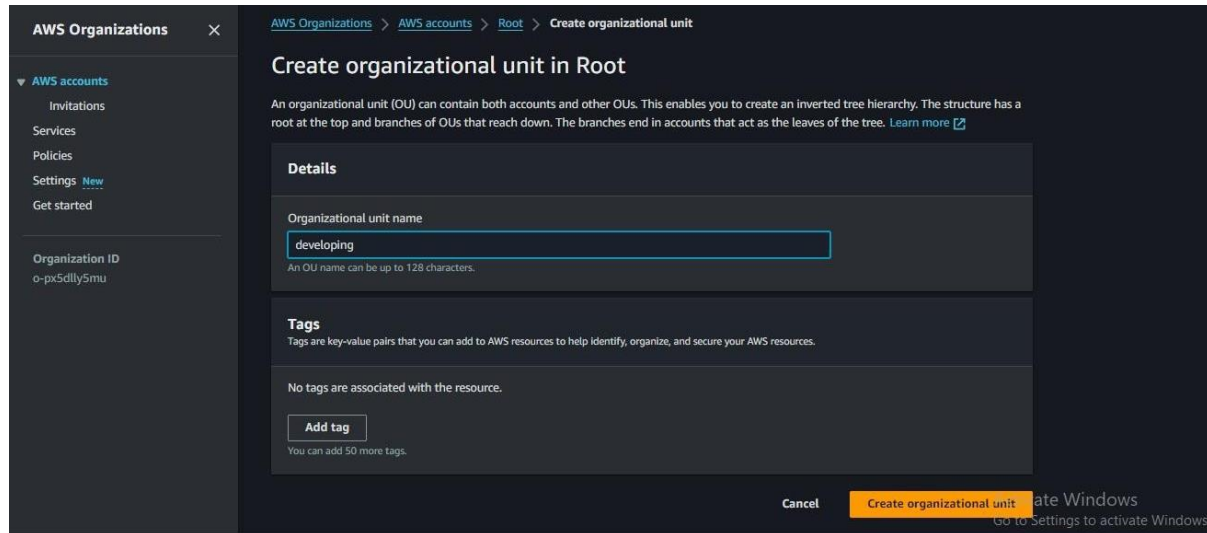
STEPS:-



- go to AWS Organizations select Root and click on (Action) after that also click on (create new)

Name : - Gaurav Patkari , Batch:- 15 Jan

AWS Assessment – Set B



AWS Organizations ×

[AWS Organizations](#) > [AWS accounts](#) > [Root](#) > [Create organizational unit](#)

Create organizational unit in Root

An organizational unit (OU) can contain both accounts and other OUs. This enables you to create an inverted tree hierarchy. The structure has a root at the top and branches of OUs that reach down. The branches end in accounts that act as the leaves of the tree. [Learn more](#)

Details

Organizational unit name

developing

An OU name can be up to 128 characters.

Tags

Tags are key-value pairs that you can add to AWS resources to help identify, organize, and secure your AWS resources.

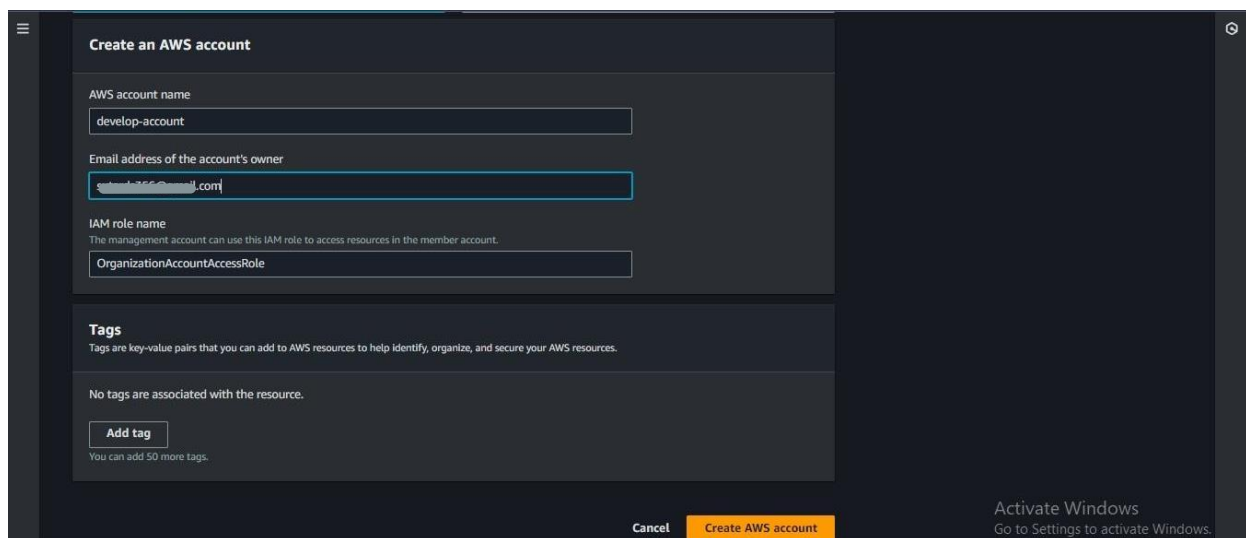
No tags are associated with the resource.

[Add tag](#)

You can add 50 more tags.

[Cancel](#) [Create organizational unit](#) [Activate Windows](#)
Go to Settings to activate Windows.

- click on (create organization unit), after that we created developing organization unit



Create an AWS account

AWS account name

develop-account

Email address of the account's owner

gaurav.patkari@gmail.com

IAM role name

The management account can use this IAM role to access resources in the member account.

OrganizationAccountAccessRole

Tags

Tags are key-value pairs that you can add to AWS resources to help identify, organize, and secure your AWS resources.

No tags are associated with the resource.

[Add tag](#)

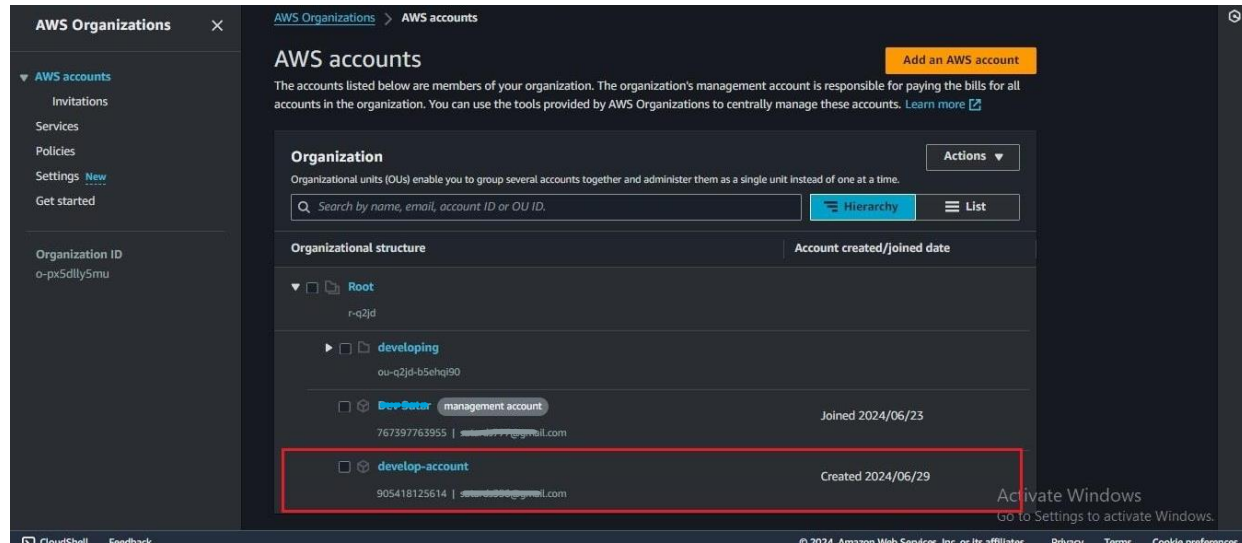
You can add 50 more tags.

[Cancel](#) [Create AWS account](#) [Activate Windows](#)
Go to Settings to activate Windows.

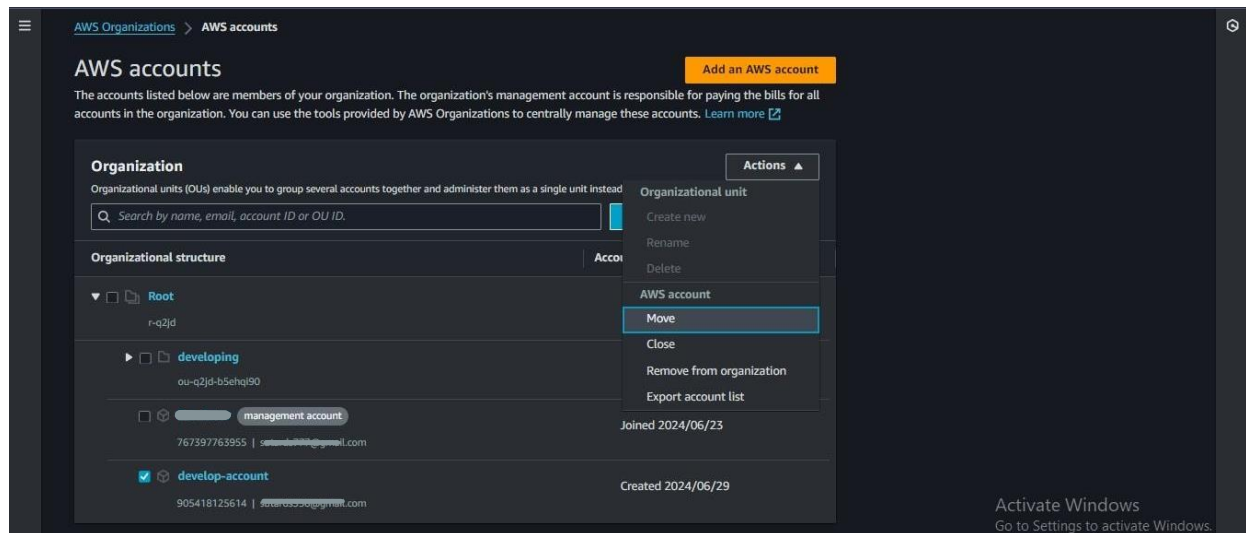
- Now select developing account, and click on top (add aws accounts), after that you will be able to see this page. fill up information and click on (Create AWS account).

Name : - Gaurav Patkari , Batch:- 15 Jan

AWS Assessment – Set B



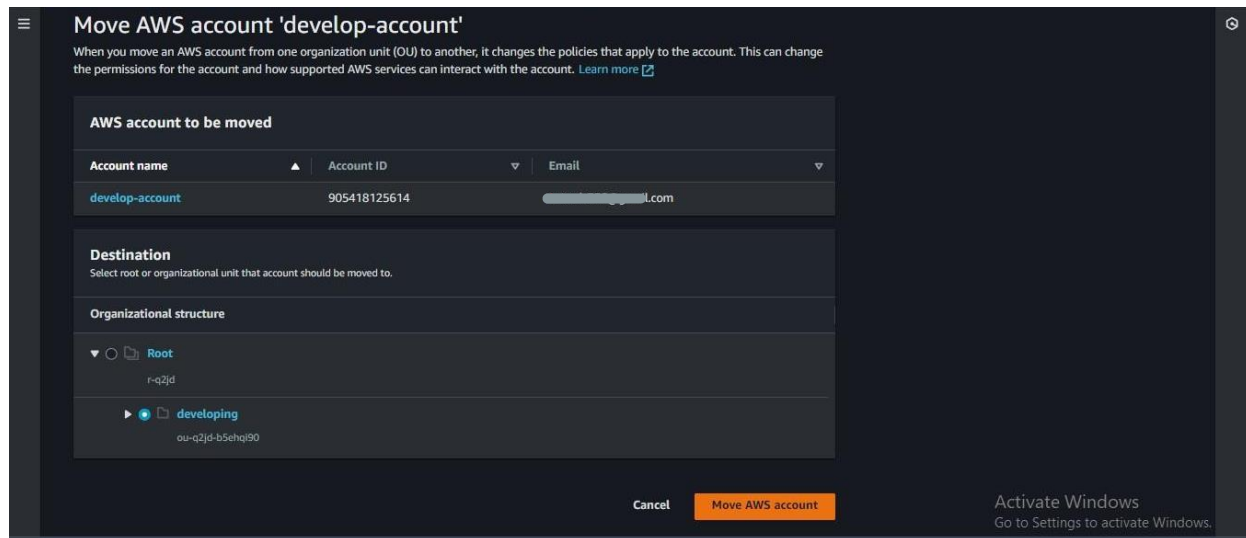
- **Successfully Develop-account account created.**



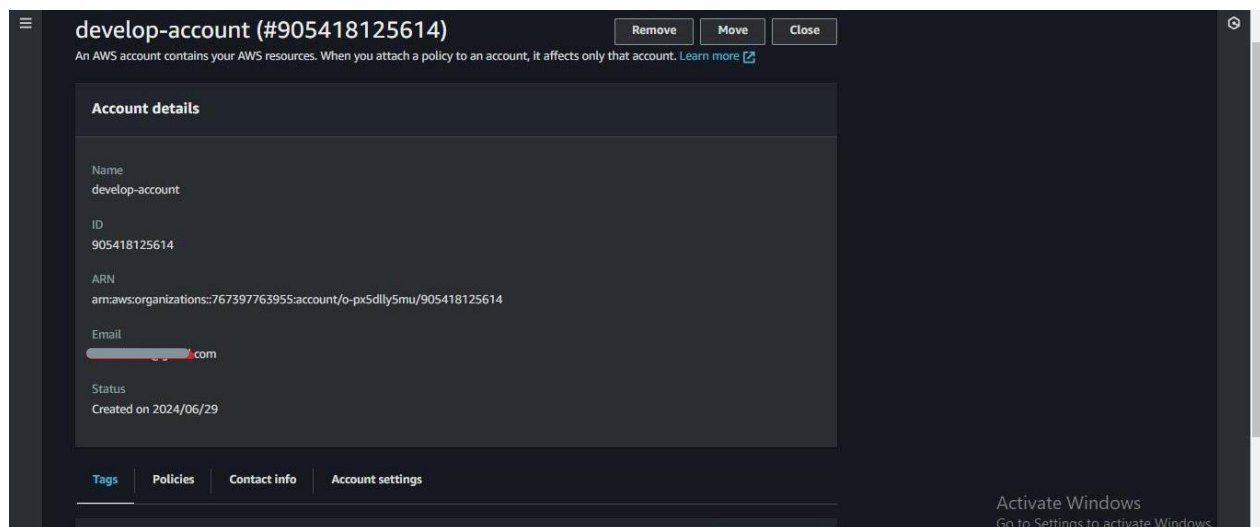
- **Now select develop-account and go to Action and click on (Move)**

Name : - Gaurav Patkari , Batch:- 15 Jan

AWS Assessment – Set B



- **Successfully we moved account in to developing unit.**



- **Successfully we created account, next logout form your root account and login with this account , put your provided email id and reset a password.**

Task 2: What is the purpose of use of AWS-CLI? What are the key requirements to access AWS account using AWS-CLI.

- a) AWS-CLI long form is Amazon Web Services Command Line Interface.
- b) The AWS CLI is a powerful tool that enables you to interact with and manage your AWS resources directly from your command line.
- c) Execute tasks quickly and efficiently without navigating through a web interface. Create scripts to automate repetitive AWS tasks, streamlining your workflow.
- d) We can create a multiple resources by using commands such as create ec2 , create s3 bucket , create key pairs, create a security group, assign port number to security group (ingress)or (egress).

Key Requirements to Access AWS Account Using AWS CLI: -

- 1) AWS Account - We required a AWS account to perform a CLI tasks.
- 2) AWS CLI Installation - Download and install the AWS CLI for your operating system.
- 3) IAM User Credentials - For security purposes, to create an IAM user with specific permissions for the tasks you'll be performing. Avoid using your root account credentials directly. You'll need the following IAM user credentials:

- **Access Key ID:** A unique identifier for your IAM user.
- **Secret Access Key:** A secret key associated with your IAM user. This should be kept confidential.

- 4) Configuration - Once installed, configure the AWS CLI.

STEPS: -

- 1) Open a PowerShell and Write Following commands One By One: -
 - (1) aws --
 - (2) aws -- configure
 - (i) put your ACCESS_KEY and SECRET_KEY.
 - (3) Default region name: us-east-1 (example)
 - (4) Default output format: json (example)

By using this steps you can access AWS-CLI.

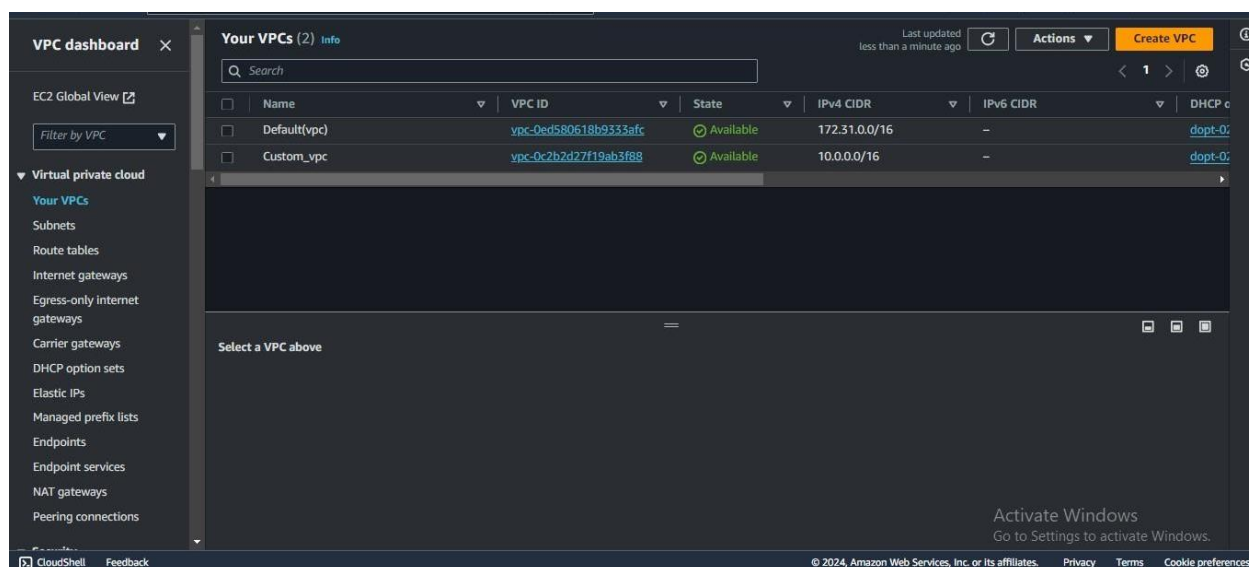
Name : - Gaurav Patkari , Batch:- 15 Jan

AWS Assessment – Set B

Create a working custom VPC. Launch one server in each subnet & connect private server with bastion host using CLI.

```
Windows PowerShell
PS C:\Users\admin> aws configure
S Access Key ID [*****LRCA]: AKIA3FLOYTNZ2JKOLRCA
AWS Secret Access Key [*****LRCA]: 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
Default region name [us-east-1]: us-east-1
Default output format [json]: json
PS C:\Users\admin> aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=Custom_vpc}]'
{
  "Vpc": {
    "CidrBlock": "10.0.0.0/16",
    "DhcpOptionsId": "dopt-0263b9a9780f5e2ff",
    "State": "pending",
    "VpcId": "vpc-0dc0c300b08e5067f",
    "OwnerId": "767397763955",
    "InstanceTenancy": "default",
    "Ipv6CidrBlockAssociationSet": [],
    "CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-07cbdc953498ed78",
        "CidrBlock": "10.0.0.0/16",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ],
    "IsDefault": false,
    "Tags": [
      {
        "Key": "Name",
        "Value": "Custom_vpc"
      }
    ]
  }
}
```

- VPC command by AWS-CLI



- Custom VPC Created.

AWS Assessment – Set B

```
Windows PowerShell
PS C:\Users\admin> aws ec2 create-subnet --vpc-id vpc-0dc0c3060b0e5067f --cidr-block 10.0.0.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=public_subnet}]'
{
  "Subnet": {
    "AvailabilityZone": "us-east-1a",
    "AvailabilityZoneId": "usel-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.0.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0fae96fc577ab356d",
    "VpcId": "vpc-0dc0c3060b0e5067f",
    "OwnerId": "767397763955",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "Tags": [
      {
        "Key": "Name",
        "Value": "public_subnet"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-east-1:767397763955:subnet/subnet-0fae96fc577ab356d",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
      "HostnameType": "ip-name",
      "EnableResourceNameDnsARecord": false,
      "EnableResourceNameDnsAAAARecord": false
    }
  }
}

PS C:\Users\admin> aws ec2 create-subnet --vpc-id vpc-0dc0c3060b0e5067f --cidr-block 10.0.15.0/24 --availability-zone us-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=private_subnet}]'
{
  "Subnet": {
    "AvailabilityZone": "us-east-1a",
    "AvailabilityZoneId": "usel-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.15.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0fe5138a02ecccbeb",
    "VpcId": "vpc-0dc0c3060b0e5067f",
    "OwnerId": "767397763955",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "Tags": [
      {
        "Key": "Name",
        "Value": "private_subnet"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-east-1:767397763955:subnet/subnet-0fe5138a02ecccbeb",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
      "HostnameType": "ip-name",
      "EnableResourceNameDnsARecord": false,
      "EnableResourceNameDnsAAAARecord": false
    }
  }
}
```

- Public and private subnet are created by AWS-CLI.

```
PS C:\Users\admin> aws ec2 create-internet-gateway --tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name,Value=myigw}]'
{
  "InternetGateway": {
    "Attachments": [],
    "InternetGatewayId": "igw-0a9489f3f31e0fe80",
    "OwnerId": "767397763955",
    "Tags": [
      {
        "Key": "Name",
        "Value": "myigw"
      }
    ]
  }
}
```

- Internet gateway created by AWS-CLI.



The screenshot shows the AWS VPC console interface. On the left, there's a sidebar with 'VPC dashboard' and 'EC2 Global View'. The main area is titled 'Internet gateways (2) info'. It contains a table with the following data:

Name	Internet gateway ID	State	VPC ID	Owner
myigw	igw-0a9489f3f31e0fe80	Detached	-	767397763955
Default_igw	igw-0aea319a60fd7a53	Attached	vpc-0ed580618b9333afc Default(vpc)	767397763955

- Internet gateway

```
PS C:\Users\admin> aws ec2 create-route-table --vpc-id vpc-0dc0c3060b0e5067f --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=private_route_table}]'
{
  "RouteTable": {
    "Associations": [],
    "PropagatingVgws": [],
    "RouteTableId": "rtb-00238740d67214a6f",
    "Routes": [
      {
        "DestinationCidrBlock": "10.0.0.0/16",
        "GatewayId": "local",
        "Origin": "CreateRouteTable",
        "State": "active"
      }
    ],
    "Tags": [
      {
        "Key": "Name",
        "Value": "private_route_table"
      }
    ],
    "VpcId": "vpc-0dc0c3060b0e5067f",
    "OwnerId": "767397763955"
  },
  "ClientToken": "dad12001-35e6-4a4b-903b-f6e407e6ab33"
}
```

- Created a route table by AWS-CLI.

Name : - Gaurav Patkari , Batch:- 15 Jan

AWS Assessment – Set B

```
PS C:\Users\admin> aws ec2 run-instances --image-id ami-01b799c439fd5516a --instance-type t2.micro --count 1 --key-name shellkey --subnet-id subnet-0fae96fc577ab356d --tag-specifications 'ResourceType=instance,tags=[{Key=Name,Value=public_server}]'
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-01b799c439fd5516a",
      "InstanceId": "i-012ed98c4a0ce30d4",
      "InstanceType": "t2.micro",
      "KeyName": "shellkey",
      "LaunchTime": "2024-06-29T07:46:42+00:00",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "us-east-1a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-10-0-0-44.ec2.internal",
      "PrivateIpAddress": "10.0.0.44",
      "ProductCodes": [],
      "PublicDnsName": "",
      "State": {
        "Code": 0,
        "Name": "pending"
      },
      "StateTransitionReason": "",
      "SubnetId": "subnet-0fae96fc577ab356d",
      "VpcId": "vpc-0dc8c3060b08e5067f",
      "Architecture": "x86_64",
      "BlockDeviceMappings": [],
      "ClientToken": "9ab8768e-8204-4494-9e2d-bc19ae071fe1",
      "EbsOptimized": false,
      "EnaSupport": true,
      "Hypervisor": "xen",
      "NetworkInterfaces": [
        {
          "Attachment": {
            "AttachTime": "2024-06-29T07:46:42+00:00",
            "AttachmentId": "eni-attach-0a5205b8fa1ff7a6f",
            "DeleteOnTermination": true,
            "DeviceIndex": 0,
            "Status": "attaching",
            "NetworkCardIndex": 0
          },
          "Description": "",
          "Groups": [

```

- Created a public server(ec2) by AWS-CLI.

```
Windows PowerShell
PS C:\Users\admin> aws ec2 run-instances --image-id ami-01b799c439fd5516a --instance-type t2.micro --count 1 --key-name shellkey --subnet-id subnet-0fe5138a02ecccbeb --tag-specifications 'ResourceType=instance,tags=[{Key=Name,Value=private_server}]'
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-01b799c439fd5516a",
      "InstanceId": "i-0606a15ab2bb7cae4",
      "InstanceType": "t2.micro",
      "KeyName": "shellkey",
      "LaunchTime": "2024-06-29T07:50:28+00:00",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "us-east-1a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-10-0-15-19.ec2.internal",
      "PrivateIpAddress": "10.0.15.19",
      "ProductCodes": [],
      "PublicDnsName": "",
      "State": {
        "Code": 0,
        "Name": "pending"
      },
      "StateTransitionReason": "",
      "SubnetId": "subnet-0fe5138a02ecccbeb",
      "VpcId": "vpc-0dc8c3060b08e5067f",
      "Architecture": "x86_64",
      "BlockDeviceMappings": [],
      "ClientToken": "7eabc790-069d-43a1-b24a-2a2faf497a37",
      "EbsOptimized": false,
      "EnaSupport": true,
      "Hypervisor": "xen",
      "NetworkInterfaces": [
        {
          "Attachment": {
            "AttachTime": "2024-06-29T07:50:28+00:00",
            "AttachmentId": "eni-attach-00d1def910a8825be",
            "DeleteOnTermination": true,
            "DeviceIndex": 0,
            "Status": "attaching",
            "NetworkCardIndex": 0
          },
          "Description": "",
          "Groups": [

```

- Created a private server(ec2) by AWS-CLI.

AWS Assessment – Set B

Instances [1/9] Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input type="checkbox"/>	Kube-Master	i-04f8ad83ef114c96	Stopped	t2.small	-	View alarms +	us-east-1a	-
<input checked="" type="checkbox"/>	public_server	i-061149df41466d570	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-
<input type="checkbox"/>	live-server	i-01e9869367e557bb9	Stopped	t2.micro	-	View alarms +	us-east-1b	-
<input type="checkbox"/>	Terraform	i-048C27279d2713d02	Stopped	t2.micro	-	View alarms +	us-east-1b	-
<input type="checkbox"/>	MINIKUBE	i-021d014e94See4ee4	Stopped	t2.medium	-	View alarms +	us-east-1e	-
<input type="checkbox"/>	kube-Worker	i-07b99cf1010cd29ee	Stopped	t2.micro	-	View alarms +	us-east-1b	-
<input type="checkbox"/>	Jenkins	i-04d2a54ae8a425f73	Stopped	t2.micro	-	View alarms +	us-east-1b	-
<input type="checkbox"/>	liveserverr	i-0a40c659a189fd64a	Stopped	t2.micro	-	View alarms +	us-east-1b	-
<input type="checkbox"/>	private_server	i-05bc24cac78ecf274	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-

i-061149df41466d570 (public_server)

- Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary Info

Instance ID i-061149df41466d570 (public_server)	Public IPv4 address 54.163.135.243 open address ↗	Private IPv4 address 10.0.14.131 Go to Settings to activate Windows.
---	---	--

- Two server launched in Custom VPC.

[illegible]

- **Successfully connected private server with bastion host using CLI.**

Task 3: Write a short note on S3. What is difference between Static website and Dynamic website?

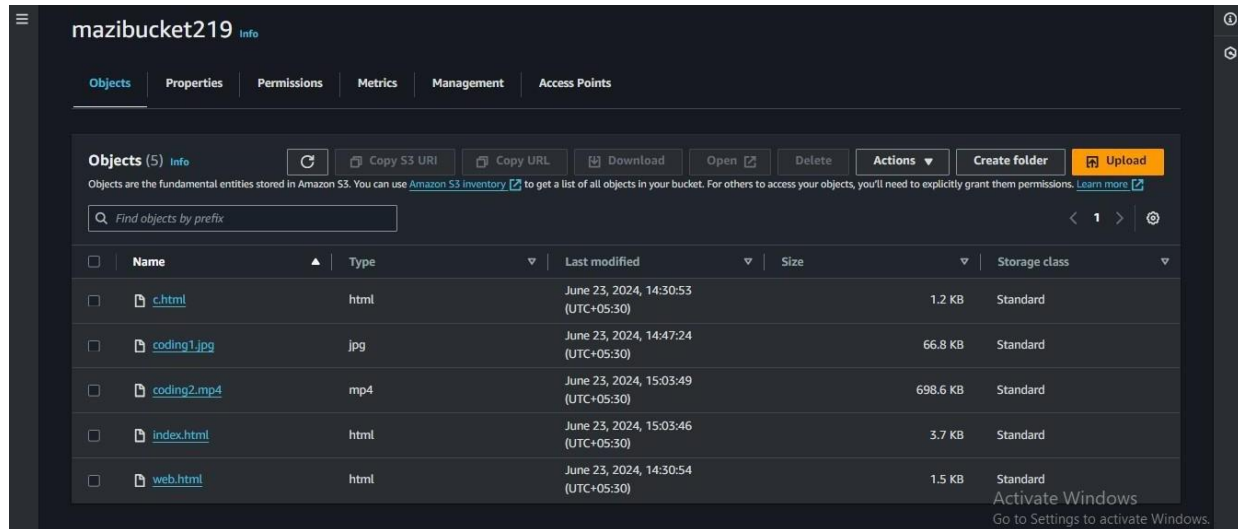
Create a static website using S3 with minimum 3 pages having images and videos.

- a) S3, or **Amazon Simple Storage Service**, is a scalable object storage service offered by Amazon Web Services (AWS). It allows you to store any type of data, from documents and images to backups and big data archives.
- b) In S3 we got unlimited storage and it is a pay as you go .
- c) S3 it is a managed by AWS . it have a High Availability 99.99% data you will get , it will create 3 copies of data and Durability of 99.999999% you will gwt.
- d) S3 service is a regional but we can use it global.
- e) S3 bucket name required globally different, name must be a unique.
- f) It have a naming rule , so name not in IP format.
- g) In S3 bucket we can upload 16GB file through Console.
- h) Its store user content.
- i) If you created an ACL enabled Bucket so you can give only limited permission to the object such as read and write. But you created an ACL disabled Bucket you can able to give multiple permission to the object.
- j) Its consist a storage classes such as 1) standard 2) standard Infrequent Access 3) Intelligent tier4) One –zone 5) Glacier Instant Retrieval 6) Glacier Flexible Retrieval 7) Glacier Deep Archive.
- k) It consist a Property like Bucket Versioning, in this you can store multiple object with same in same bucket, if e delete file it will store in show versions. Also property have Object Lock in which consist two type such as 1) Governance mode 2) Compliance mode .
- l) Common Storage we can use.
- m) We can able to upload 5TB single file at one time in S3 bucket, by using coding /CLI.
- n) Data stored in object format. file data, meta data, key
Meta data: - data about data, key : - location of a file , path.

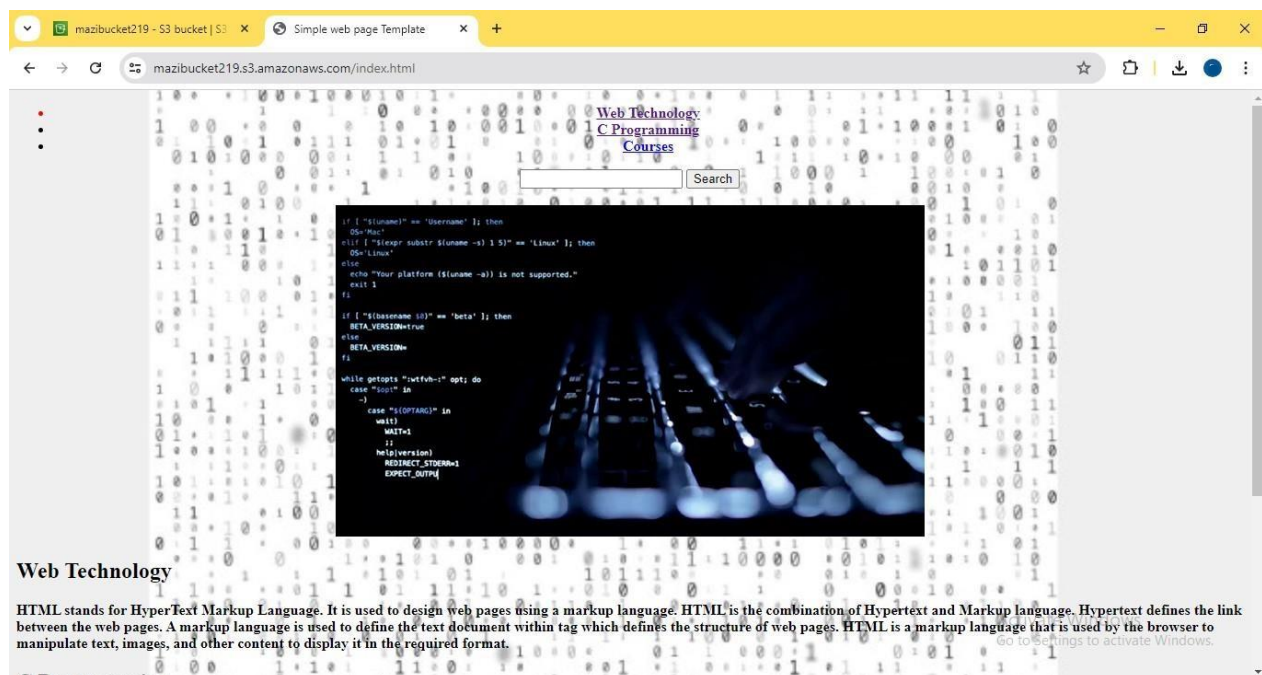
What is difference between Static website and Dynamic website?

Static website	Dynamic website
i. Built with HTML, CSS (think basic building blocks).	I. Uses server-side languages (PHP, Python) for on-the-fly content.
ii. Updates require manual code edits.	II. Content can change based on user actions or data.
iii. Lightweight and ideal for simple displays	III. Offers user interaction, forms, and dynamic elements.
iv. Portfolios, landing pages, student projects.	IV. E-commerce, social media, complex applications.
v. A restaurant brochure with its menu.	V. An online store where you can search and buy products.

Create a static website using S3 with minimum 3 pages having images and videos.



- Pages created with images and videos



- Video is running in this page.

AWS Assessment – Set B



Web Technology can be Classified into the Following Sections:

- 1] World Wide Web (WWW): The World Wide Web is based on several different technologies: Web browsers, Hypertext Markup Language (HTML), and Hypertext Transfer Protocol (HTTP).
- 2] Web Browser: The web browser is an application software to explore www (World Wide Web). It provides an interface between the server and the client and requests to the server for web documents and services.
- 3]Web Server: Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).
- 4]Web Pages: A webpage is a digital document that is linked to the World Wide Web and viewable by anyone connected to the internet has a web browser.
- 5]Web Development: Web development refers to the building, creating, and maintaining of websites. It includes aspects such as web design, web publishing, web programming, and database management. It is the creation of an application that works over the internet i.e. websites.

Activate Windows
Go to Settings to activate Windows.

- Second webpage



Web Technology can be Classified into the Following Sections:

- 1] World Wide Web (WWW): The World Wide Web is based on several different technologies: Web browsers, Hypertext Markup Language (HTML), and Hypertext Transfer Protocol (HTTP).
- 2] Web Browser: The web browser is an application software to explore www (World Wide Web). It provides an interface between the server and the client and requests to the server for web documents and services.
- 3]Web Server: Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).
- 4]Web Pages: A webpage is a digital document that is linked to the World Wide Web and viewable by anyone connected to the internet has a web browser.
- 5]Web Development: Web development refers to the building, creating, and maintaining of websites. It includes aspects such as web design, web publishing, web programming, and database management. It is the creation of an application that works over the internet i.e. websites.

Activate Windows
Go to Settings to activate Windows.

- Third webpage.

Task 4: What is Instance Refresh? Demonstrate the use of Instance Refresh.

- Instance Refresh in AWS EC2 Auto Scaling provides a powerful and automated way to update instances within your Auto Scaling Group
- You define the desired configuration changes, like a new AMI with updated software or a more powerful instance type.
- Instance Refresh takes care of replacing existing instances with the new configuration in a controlled manner.
- By replacing instances in batches, it ensures a certain number of healthy instances remain operational throughout the refresh process.
- This minimizes downtime for your application and maintains service availability.
- Instance Refresh automates the update process, eliminating the need for manual instance termination and launch. This saves you time and effort compared to managing updates individually.

How it Works:

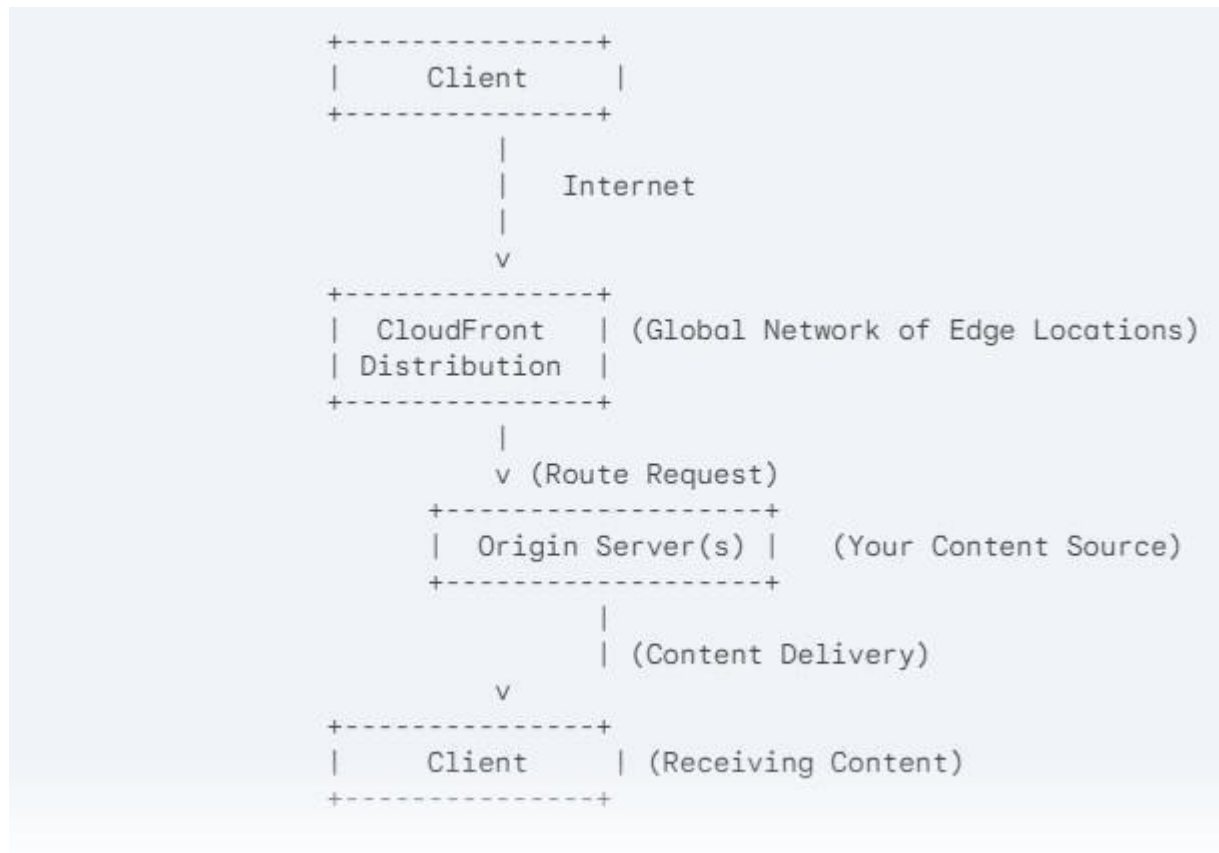
1. **Define Desired Configuration:** The first step is to create a new launch template or update an existing one. This template specifies the desired changes, such as:
 - **New AMI:** Update to an AMI containing a newer software version.
 - **Instance Type:** Upgrade to a more powerful instance type for better performance.
 - **User Data Script:** Apply new configuration settings through a user data script.
 - **Launch Template Version:** If using a launch template, specify the desired version with the updates.
2. **Initiate Instance Refresh:** Once the desired configuration is defined, use the AWS Management Console, AWS CLI, or AWS SDK to trigger the Instance Refresh process for your ASG.

3. **Controlled Rollout:** Instance Refresh replaces existing instances in a controlled manner:

Benefits of Instance Refresh:

- **Reduced Downtime:** Updates occur in batches, minimizing application downtime and ensuring service continuity.
- **Improved Efficiency:** Automating updates saves significant time and effort compared to manual instance replacement.
- **Controlled Updates:** The minimum healthy percentage ensures application availability during the refresh process.
- **Consistent Configuration:** Ensures all instances in the ASG adhere to the desired configuration.
- **Simplified Management:** Streamlines the process of updating Auto Scaling Groups.

Task 5: Create a architectural diagram of uses of cloud front.



Components:

- **Client:** Represents the end user accessing your content (website, application, etc.)
- **CloudFront Distribution:** A configuration within CloudFront that specifies.
 - **Origin:** The source of your content (e.g., S3 bucket, EC2 instance, etc.)
 - **Edge Locations:** A global network of servers that cache and deliver your content.

Data Flow:

1. **Client Request:** The user makes a request for content through their browser or application.
2. **Route Request:** The request is routed to the nearest CloudFront Edge Location based on the user's location for optimal performance.

Benefits:

- **Reduced Latency:** Users experience faster content delivery due to geographically distributed Edge Locations.
- **Scalability:** CloudFront automatically scales to handle traffic spikes without impacting performance.
- **Security:** CloudFront can be integrated with AWS WAF (Web Application Firewall) for added protection against cyberattacks.
- **Reduced Cost:** By offloading traffic from your origin server, CloudFront can help optimize origin server costs.

Use Cases:

- **Website Content Delivery:** Deliver static website content (HTML, CSS, JavaScript) efficiently.
- **Media Streaming:** Stream audio and video content with low latency and high availability.
- **API Gateway Caching:** Cache API responses for faster performance and reduced load on your backend services.
- **Secure Content Delivery:** Integrate CloudFront with WAF to protect your content from malicious attacks