Project report

Name: Ashish Gajanan Ingle

Email: [ingle13agi@gmail.com](mailto:ingle13agi@gmail.com)

Phone no: 9322476735

Address: Malkapur

Maharashtra

Project: Resilient and Scalable Web Application Deployment on AWS

Date:18-05-2025

Index

|  |  |  |
| --- | --- | --- |
| SR.N | TOPIC | PAGE NO |
| 1 | PROJECT | 3 |
| 2 | CORE AWS SERVICES UTILISATION | 4 |
| 3 | INRODUCTION | 8 |
| 4 | PREREQUISITES | 9 |
| 5 | ARCHITECTURE OVERVIEW | 10 |
| 6 | AWS SERVICE CONFIGURATION(ASC) | 11 |
| 7 | ASC 1.VPC | 11 |
| 8 | ASC2. EC2 | 14 |
| 9 | ASC3. AUTOSCALING | 16 |
| 10 | ASC4. EFS | 18 |
| 11 | ASC5. APP LOAD BALANCER | 20 |
| 12 | ASC6. ROUTE53 | 21 |
| 13 | APPLICATION DEPLOYMENT | 24 |
| 14 | CONFIGURATION MANAGEMENT | 27 |
| 15 | SECURITY CONFIGURATION | 29 |
| 16 | BEST PRACTICES | 31 |
| 17 | MONITORING AND LOGGING | 32 |
| 18 | SETUP & MANAGE LOGS | 34 |
| 19 | BACKUP AND DISASTER RECOVERY | 36 |
| 20 | RECOVERY PLAN | 38 |
| 21 | TROUBLESHOOTING | 40 |
| 22 | SUPPORT | 42 |
| 23 | ADDITONAL RESOURSES LINKS | 43 |
| 24 | THANK YOU | 44 |

Project Title: Resilient and Scalable Web Application Deployment on AWS

Project Description This project aims to design and implement a highly available and scalable infrastructure for a web application on AWS. The architecture will leverage AWS services to ensure the web application can efficiently handle varying loads and maintain high availability across multiple Availability Zones.

Objectives

High Availability: Achieve minimal downtime by utilising multiple Availability Zones. Scalability: Use AWS Auto Scaling to adjust resources automatically in response to traffic changes. Security: Implement security measures focusing on security groups and secure communication. Resilience: Develop an application setup that can withstand failures and traffic spikes without manual intervention.

Core AWS Services Utilisation

Virtual Private Cloud (VPC): Provides an isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define.

Elastic File System (EFS): Offers a simple, scalable, elastic file storage for use with AWS Cloud services and on-premises resources.

Elastic Compute Cloud (EC2): Provides scalable computing capacity in the AWScloud. It allows developers to scale up or down based on demand.

AWSAutoScaling: Monitors applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

Application Load Balancer (ALB): Automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses.

Route 53: A highly available and scalable cloud Domain Name System (DNS) web service, designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications.

Architecture Overview

High-Level Design: ● Diagram: Include a cloud architecture diagram showing the relationship between VPC, EFS, EC2 instances, Auto Scaling, ALB, and Route 53.

● VPC Configuration: Detail the setup of the VPC, including subnets across multiple Availability Zones for high availability. ● Security: Describe the security groups, IAM roles, and policies to ensure secure access and communication.

Detailed Component Design

VPC

● Purpose: Isolation and provision of a secure network for AWS resources.

● Configuration: Subnet creation across multiple AZs for resilience and high availability.

EFS

● Purpose: Provide a scalable file storage system accessible by EC2 instances.

● Configuration: Set up to ensure it is mounted on all EC2 instances for shared storage needs.

EC2

● Purpose: Host the web application and handle compute tasks.

● Configuration: Instance types, AMIs, and initial bootstrapping scripts.

AWS Autoscaling

● Purpose: Dynamically adjust the number of EC2 instances based on traffic.

● Configuration: Define scaling policies based on metrics like CPU utilisation and request rates.

ALB

● Purpose: Distribute incoming traffic across multiple targets to increase the availability and fault tolerance of your application.

● Configuration: Listener and rule setup for routing traffic to the appropriate target groups.

Route 53

● Purpose: Connect user requests to the infrastructure running in AWS.

● Configuration: DNS management, health checks, and routing policies to ensure high availability and traffic management.

Security Measures

● Security Groups: Define inbound and outbound traffic rules for EC2 instances and other services.

● IAM Roles: Secure API calls from EC2 instances.

● Data Encryption: At rest (EFS) and in transit using TLS. Scalability and High Availability Strategy

● Autoscaling: Setup and configuration details, including scaling policies.

● Multi-AZ Deployment: Ensure the application is deployed across multiple AZs to achieve high availability.

Conclusion

This document outlines the initial design for deploying a resilient and scalable web application on AWS. It emphasises high availability, scalability, security, and resilience, leveraging AWS services to achieve these objectives. The next steps include detailed planning, implementation, testing, and deployment phases, ensuring the web application meets the outlined objectives.

Introduction

● Purpose: This project aims to design and implement a highly available and scalable infrastructure for a web application on AWS. The architecture will leverage AWS services to ensure the web application can efficiently handle varying loads and maintain high availability across multiple Availability Zones.

● Scope:

1 This project involves designing and implementing a highly available and scalable web application infrastructure on AWS.

2 The architecture will leverage AWS services to ensure:

a) Fault tolerance,

b) Load balancing,

c) Secure user access

3 The core of the project is to deploy a web application that can handle varying loads efficiently and maintain high availability across multiple Availability Zones (AZs).

Prerequisites

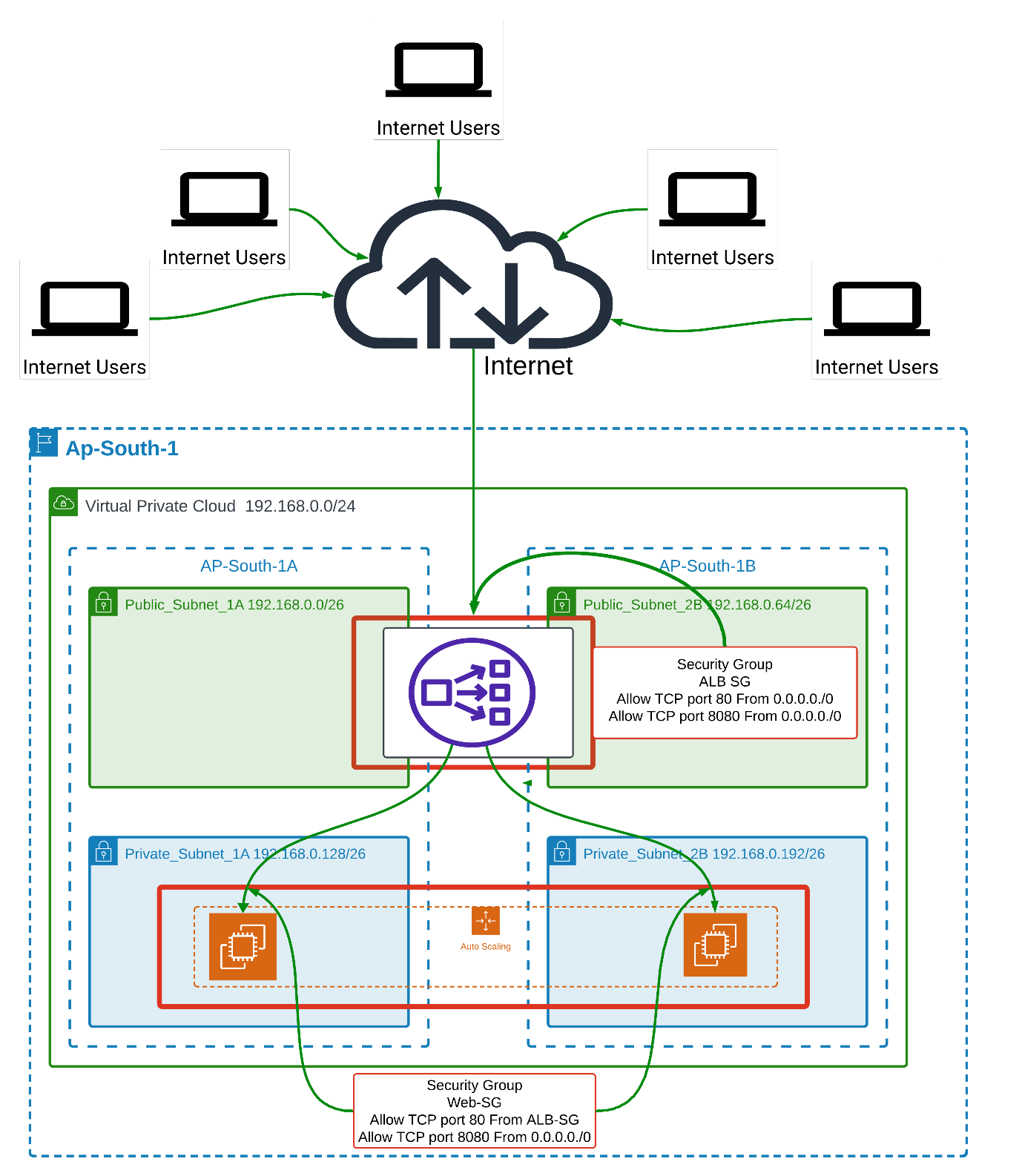
● AWS Account Ensure you have an active AWS account.

● IAMRoles: Define the IAMroles required for the implementation.

● Knowledgebase: knowledge of Virtual Private Cloud (VPC), Elastic File System (EFS), Elastic Compute Cloud (EC2), AWS Autoscaling

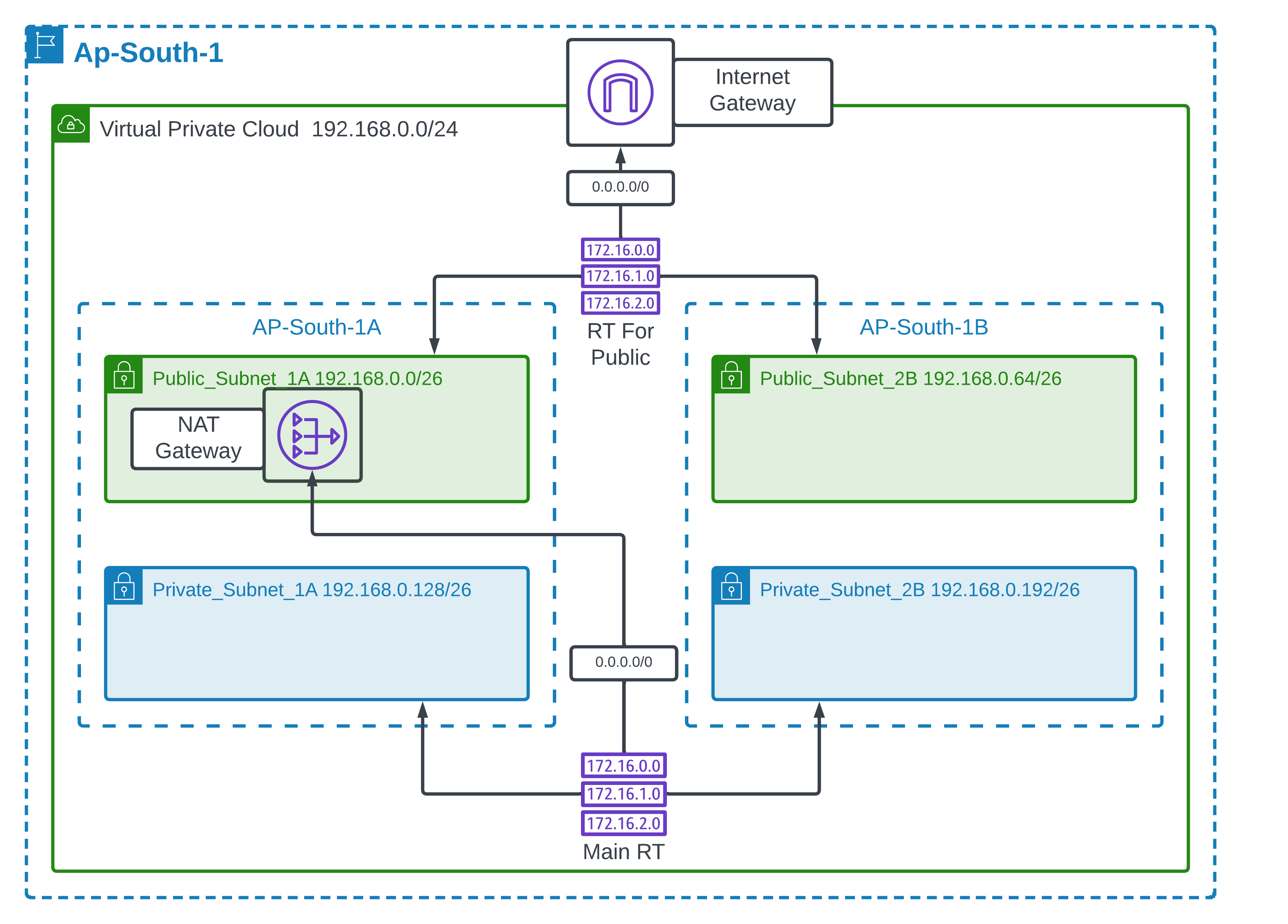
Architecture Overview

● Architecture Diagram: Include a diagram of the architecture.

● Description: Describe the high-level architecture and how different components interact

AWS Services Configuration(ASC)

ASC 1.)Virtual Private Cloud (VPC)



Step 1: Create the VPC

Goto the VPC Dashboard in the AWS Management Console. Click on "Create VPC".

In the "Name tag" field, input a name for your VPC (e.g., "Ap-South-1"). For the IPv4 CIDR block, enter 192.168.0.0/24.

Skip the IPv6 CIDR block unless you need it.

Select "No" for Tenancy (unless you require a Dedicated Instance).

Click "Create".

Step 2: Create Subnets

In the VPC Dashboard, click on "Subnets".

Click on "Create subnet".

Select the VPC you created from the dropdown list.

For the first subnet:

● Nametag:"Public\_Subnet\_1A".

● Availability Zone: Select "ap-south-1a".

● IPv4CIDRblock: 192.168.0.0/26.

Click "Create" to create the first subnet. Repeat the steps for the additional subnets with the following details:

● Forthesecond subnet:

● Nametag:"Private\_Subnet\_1A".

● Availability Zone: "ap-south-1a".

● IPv4CIDRblock: 192.168.0.128/26.

● Forthethird subnet:

● Nametag:"Public\_Subnet\_2B".

● Availability Zone: "ap-south-1b".

● IPv4CIDRblock: 192.168.0.64/26.

● For the fourth subnet:

● Nametag:"Private\_Subnet\_2B".

● Availability Zone: "ap-south-1b".

● IPv4CIDRblock: 192.168.0.192/26.

Step 3: Set Up Internet Gateway In the VPC Dashboard,

click on "Internet Gateways".

Click "Create internet gateway". Provide a name tag (e.g., "IGW-Ap-South-1").

Click "Create". Select the newly created internet gateway and click on "Actions".

Click "Attach to VPC" and select the VPC you have created. Click "Attach".

Step 4: Create Route Tables

In the VPC Dashboard, go to "Route Tables".

Click "Create route table". Enter a name for the route table (e.g., "Main RT").

Select your VPC from the dropdown list.

Click "Create". After creation, select the new route table and click on "Routes".

Click "Edit routes" and add the following route to allow internet access:

● Destination: 0.0.0.0/0.

● Target: Select the Internet Gateway you created.

Click "Save routes".

Step 5: Associate Route Tables with Subnets

Select the route table, click on "Subnet Associations". Click "Edit subnet associations". Select the public subnets ("Public\_Subnet\_1A" and "Public\_Subnet\_2B"). Click "Save".

Step 6: Create NAT Gateway (for Private Subnets)

Goto "NAT Gateways" in the VPC Dashboard.

Click "Create

NAT gateway". Select one of the public subnets (e.g., "Public\_Subnet\_1A").

Allocate an Elastic IP by clicking "Allocate Elastic IP".

Click "Create a NAT Gateway". Once created, go back to "Route Tables", and create a new route table for your private subnets.

Follow a similar process as before to add a route pointing to the NAT Gateway for internet access from the private subnets.

ASC.2) Elastic Compute Cloud (EC2)

**Step 1: Log in to AWS Console**

* Go to the AWS Management Console.
* Navigate to **EC2 Dashboard**.

**Step 2: Launch an EC2 Instance**

* Click **Launch Instance**.
* Choose an **Amazon Machine Image (AMI)** (e.g., Ubuntu, Amazon Linux).
* Select an **Instance Type** (e.g., t2.micro for free-tier eligibility).
* Click **Next: Configure Instance Details**.

**Step 3: Configure Instance Details**

* Choose the **VPC and Subnet**.
* Enable **Auto-assign Public IP** if needed.
* Configure **IAM Role** for access permissions.
* Click **Next: Add Storage**.

**Step 4: Add Storage**

* Define **root volume size** (e.g., 30GB for general use).
* Add **additional volumes** if required.
* Click **Next: Add Tags**.

**Step 5: Configure Security Group**

* Create a new **Security Group** or select an existing one.
* Allow necessary ports:
  + **SSH (22)** for remote access.
  + **HTTP (80)** and **HTTPS (443)** for web applications.
  + **Custom ports** for databases or applications.

Click **Next: Review and Launch**.

**Step 6: Review and Launch**

* Verify all configurations.
* Click **Launch**.
* Select or create a **Key Pair** for SSH access.
* Click **Launch Instance**.

**Step 7: Connect to Your Instance**

* Open a terminal and use SSH:

ssh -i /path/to/key.pem ubuntu@your-instance-ip

For Windows, use **PuTTY** to connect.

ASC.3) AutoScaling: Configuration steps for auto-scaling groups

**Step 1: Create a Launch Template**

1. Go to the **AWS EC2 Dashboard**.
2. Click **Launch Templates** → **Create Launch Template**.
3. Provide a **name** and **description**.
4. Choose an **AMI (Amazon Machine Image)** for your instances.
5. Select an **instance type** (e.g., t2.micro for free-tier eligibility).
6. Configure **network settings** (VPC, subnets, security groups).
7. Attach **storage volumes** if needed
8. Click **Create Launch Template**.

**Step 2: Create an Auto Scaling Group**

1. Navigate to **Auto Scaling Groups** in the AWS Console.
2. Click **Create Auto Scaling Group**.
3. Select the **Launch Template** created earlier.
4. Choose a **VPC and subnets** for instance placement.
5. Define the **desired, minimum, and maximum number of instances**.
6. Click **Next: Configure Load Balancer**.

**Step 3: Attach a Load Balancer (Optional)**

1. Select **Application Load Balancer (ALB)** or **Classic Load Balancer**.
2. Choose a **Target Group** to route traffic.
3. Click **Next: Configure Scaling Policies**.

**Step 4: Configure Scaling Policies**

1. Choose **Dynamic Scaling** or **Scheduled Scaling**.
2. Set up **CloudWatch alarms** to trigger scaling actions.
3. Define **CPU utilization thresholds** (e.g., scale up when CPU > 70%).
4. Click **Next: Review and Create**.

**Step 5: Review and Launch**

1. Verify all configurations.
2. Click **Create Auto Scaling Group**.
3. Monitor instances in the **EC2 Dashboard**.

ASC.4 Elastic File System (EFS)

**Step 1: Create an EFS File System**

1. **Navigate to the EFS Dashboard** in the AWS Management Console.
2. Click on **"Create file system"**.
3. Choose the **VPC** where you want to deploy EFS.
4. Select the **performance mode** (General Purpose or Max I/O).
5. Choose the **storage class** (Regional or One Zone).
6. Enable **automatic backups** if needed.
7. Click **"Create"**.

**Step 2: Configure Security Groups**

1. Ensure the **EFS security group** allows inbound traffic on **port 2049 (NFS)**.
2. Modify the **EC2 security group** to allow NFS traffic from the EFS security group.

**Step 3: Create Mount Targets**

1. In the **EFS dashboard**, go to **"Network settings"**.
2. Select the **Availability Zones** and **subnets** for mount targets.
3. Assign **security groups** to control access.

**Step 4: Mount EFS on an EC2 Instance**

1. **Install NFS client** on your EC2 instance:

sudo apt-get update

sudo apt-get install -y nfs-common

**Create a mount point**:

sudo mkdir /mnt/efs

**Mount the EFS file system**:

sudo mount -t nfs4 -o nfsvers=4.1 <file-system-id>.efs.<region>.amazonaws.com:/ /mnt/efs

Replace <file-system-id> and <region> with your actual values.

**Step 5: Verify the Mount**

1. Check if the mount was successful:

ls /mnt/efs

Try adding a file:

echo "Hello, EFS!" | sudo tee /mnt/efs/hello.txt

ASC.5) Application Load Balancer (ALB)

**Step 1: Create a Target Group**

1. Open the **EC2 Dashboard** in the AWS Management Console.
2. Navigate to **Target Groups** and click **"Create target group"**.
3. Choose the **target type** (Instances, IP addresses, or Lambda function).
4. Provide a **name** for the target group.
5. Select the **protocol** (HTTP or HTTPS) and **port**.
6. Choose the **VPC** where your targets are located.
7. Configure **health checks** (path, interval, success codes).
8. Click **"Create"**.

**Step 2: Register Targets**

1. Select the **target group** you created.
2. Click **"Register targets"**.
3. Choose the **EC2 instances** you want to include.
4. Click **"Include as pending"** and then **"Register"**.

**Step 3: Create an Application Load Balancer**

1. Navigate to **Load Balancers** in the EC2 Dashboard.
2. Click **"Create Load Balancer"** and select **Application Load Balancer**.
3. Provide a **name** for the ALB.
4. Choose the **scheme** (Internet-facing or Internal).
5. Select the **IP address type** (IPv4 or Dualstack).
6. Choose the **VPC** and at least **two public subnets**.
7. Click **"Next"**.

**Step 4: Configure Listeners and Routing**

1. Add a **listener** (HTTP or HTTPS).
2. For HTTPS, upload an **SSL certificate**.
3. Select the **target group** created earlier.
4. Click **"Next"**.

**Step 5: Configure Security Groups**

1. Create or select a **security group**.
2. Allow inbound traffic on **port 80 (HTTP) and 443 (HTTPS)**.
3. Click **"Next"**.

**Step 6: Review and Create**

1. Review all configurations.
2. Click **"Create Load Balancer"**.
3. Wait for the ALB to be provisioned.

ASC.6) Route 53

**Step 1: Create a Hosted Zone**

1. Open the **AWS Route 53 Console**.
2. Click **"Create hosted zone"**.
3. Enter your **domain name** (e.g., example.com).
4. Choose the **type**:
   * **Public hosted zone** (for internet-facing domains).
   * **Private hosted zone** (for internal AWS resources).
5. Click **"Create hosted zone"**.

**Step 2: Add DNS Records**

1. In the **hosted zone**, click **"Create record"**.
2. Choose the **record type**:
   * **A Record** (maps domain to an IPv4 address).
   * **CNAME Record** (maps domain to another domain).
   * **MX Record** (for email routing).
   * **TXT Record** (for verification and security).
3. Enter the **record name** (e.g., www.example.com).
4. Provide the **value** (e.g., IP address or another domain).
5. Click **"Create record"**.

**Step 3: Configure Domain Registration (Optional)**

1. If you don’t have a domain, register one in **Route 53**.
2. Navigate to **"Domains" > "Register domain"**.
3. Search for your desired domain name.
4. Complete the registration process.

**Step 4: Update Domain Name Servers**

1. If your domain is registered outside AWS, update the **Name Server (NS) records**.
2. Copy the **NS records** from Route 53.
3. Go to your domain registrar and update the **DNS settings**.

**Step 5: Configure Routing Policies**

1. Choose a **routing policy**:
   * **Simple Routing** (directs traffic to a single resource).
   * **Weighted Routing** (distributes traffic based on assigned weights).
   * **Latency-based Routing** (routes users to the lowest-latency region).
   * **Failover Routing** (redirects traffic to a backup resource).
   * **Geolocation Routing** (routes users based on their geographic location).

Apply the routing policy to your records.

**Step 6: Set Up Health Checks (Optional)**

1. Navigate to **"Health Checks"** in Route 53.
2. Click **"Create health check"**.
3. Enter the **IP address or domain** to monitor.
4. Configure **failure thresholds** and **notification settings**.

Application Deployment

**Step 1: Create IAM Roles**

1. Open the **AWS IAM Console**.
2. Create a role for **EC2** with the policy AmazonEC2RoleforAWSCodeDeploy.
3. Create a role for **CodeDeploy** with the policy AWSCodeDeployRole.
4. Attach the **EC2 role** to your instance.

**Step 2: Launch an EC2 Instance**

1. Open the **EC2 Dashboard**.
2. Click **"Launch Instance"**.
3. Choose an **Amazon Linux 2 AMI** or **Ubuntu**.
4. Select an **instance type** (e.g., t2.micro).
5. Attach the **IAM role** created earlier.
6. Configure **security groups** to allow SSH (port 22) and HTTP (port 80).
7. Click **"Launch"**.

**Step 3: Install CodeDeploy Agent on EC2**

1. Connect to your EC2 instance via SSH.
2. Run the following commands to install the **CodeDeploy agent**:
3. sudo yum update -y
4. sudo yum install -y ruby wget
5. cd /home/ec2-user
6. wget https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install
7. sudo chmod +x ./install
8. sudo ./install auto
9. sudo service codedeploy-agent start

**Step 4: Create an Application in AWS CodeDeploy**

1. Open the **AWS CodeDeploy Console**.
2. Click **"Create application"**.
3. Provide an **application name**.
4. Select **Compute platform** as **EC2/On-premises**.
5. Click **"Create"**.

**Step 6: Prepare the Application Code**

1. Create a appspec.yml file in your application directory:

version: 0.0

os: linux

files:

- source: /

destination: /var/www/html

hooks:

BeforeInstall:

- location: scripts/install\_dependencies.sh

timeout: 300

runas: root

1. Zip your application files: zip -r myapp.zip .

**Step 7: Upload Code to S3**

1. Open the **S3 Console**.
2. Create a **bucket** and upload myapp.zip.
3. Copy the **S3 URL** of the uploaded file.

**Step 8: Create a Deployment**

1. In **CodeDeploy**, click **"Create deployment"**.
2. Select the **application** and **deployment group**.
3. Provide the **S3 URL** of the application package.
4. Click **"Deploy"**.

‘Configuration Management: Best practices for managing application configuration.

**1. Use AWS AppConfig for Dynamic Configuration**

* AWS AppConfig allows **centralized storage and management** of configurations.
* Enables **automatic rollbacks** to prevent faulty deployments.
* Supports **multi-environment and multi-tenant architectures**.

**2. Store Configuration Securely**

* Use **AWS Systems Manager Parameter Store** for **hierarchical storage** of configuration data.
* Encrypt sensitive data like **API keys and database credentials** using **AWS Secrets Manager**.
* Implement **IAM policies** to restrict access to configuration settings

**3. Version Control and Auditing**

* Maintain **version history** for configuration changes to allow rollbacks.
* Use **AWS CloudTrail** to log configuration changes for auditing.
* Implement **fine-grained access control** to prevent unauthorized modifications

**4. Automate Configuration Deployment**

* Integrate **AWS AppConfig** with **CI/CD pipelines** for automated updates.
* Use **feature flags** to enable or disable functionalities dynamically.
* Implement **environment-specific configurations** for development, staging, and production.

**5. Ensure High Availability and Scalability**

* Store configurations in **Amazon S3 or DynamoDB** for scalable access.
* Use **AWS Lambda** to dynamically retrieve and apply configurations.
* Implement **failover strategies** to ensure application resilience.

Security Configuration

**Step 1: Create a Security Group**

1. Open the **AWS Management Console** and navigate to **EC2 Dashboard**.
2. Click on **"Security Groups"** under **Network & Security**.
3. Click **"Create Security Group"**.
4. Provide a **name** and **description**.
5. Select the **VPC** where the security group will be used.
6. Click **"Create"**.

**Step 2: Define Inbound Rules**

1. Select the **security group** you created.
2. Click **"Inbound rules"** and then **"Edit inbound rules"**.
3. Click **"Add rule"** and define:
   * **Protocol** (e.g., TCP, UDP, ICMP).
   * **Port range** (e.g., 22 for SSH, 80 for HTTP, 443 for HTTPS).
   * **Source** (specific IP, CIDR block, or another security group).
4. Click **"Save rules"**.

**Step 3: Define Outbound Rules**

1. Click **"Outbound rules"** and then **"Edit outbound rules"**.
2. Click **"Add rule"** and define:
   * **Protocol** (e.g., TCP, UDP).
   * **Port range** (e.g., 0-65535 for all traffic).
   * **Destination** (specific IP, CIDR block, or another security group).
3. Click **"Save rules"**.

**Step 4: Associate Security Group with EC2 Instances**

1. Navigate to **EC2 Dashboard**.
2. Select the **instance** you want to secure.
3. Click **"Networking" > "Change security groups"**.
4. Select the **security group** you created.
5. Click **"Save"**.

**Best Practices**

* **Follow the principle of least privilege**—grant only necessary permissions.
* **Use managed policies** instead of inline policies for better scalability.
* **Regularly review and update policies** to remove unused permissions.
* **Enable multi-factor authentication (MFA)** for added security.
* **Monitor IAM activity** using **AWS CloudTrail**.

Monitoring and Logging

**Step 1: Enable CloudWatch Metrics**

1. Open the **AWS CloudWatch Console**.
2. Navigate to **Metrics** and select the AWS service you want to monitor.
3. Choose the **metric namespace** (e.g., EC2, Lambda, RDS).
4. Select the **specific metric** (e.g., CPU utilization, memory usage).
5. Click **"Create dashboard"** to visualize metrics.

**Step 2: Set Up CloudWatch Alarms**

1. In the **CloudWatch Console**, go to **Alarms**.
2. Click **"Create alarm"**.
3. Select a **metric** (e.g., EC2 CPU utilization).
4. Define a **threshold** (e.g., trigger alarm if CPU usage exceeds 80%).
5. Choose an **action** (e.g., send an SNS notification or auto-scale EC2).
6. Click **"Create alarm"**.

**Step 3: Configure CloudWatch Logs**

1. Navigate to **Logs** in the CloudWatch Console.
2. Click **"Create log group"** and provide a name.
3. Click **"Create log stream"** to start logging.
4. Install the **CloudWatch Agent** on your EC2 instance:

sudo yum install -y amazon-cloudwatch-agent

sudo systemctl enable amazon-cloudwatch-agent

sudo systemctl start amazon-cloudwatch-agent

Configure the agent using the **CloudWatch Agent Configuration Wizard**.

**Step 4: Enable CloudWatch Events**

1. Go to **Events** in the CloudWatch Console.
2. Click **"Create rule"**.
3. Choose an **event source** (e.g., EC2 instance state change).
4. Define a **target** (e.g., Lambda function or SNS topic).
5. Click **"Create rule"**.

**Step 5: Set Up CloudWatch Dashboards**

1. Navigate to **Dashboards** in CloudWatch.
2. Click **"Create dashboard"**.
3. Add **widgets** for metrics, logs, and alarms.
4. Customize the dashboard layout.
5. Click **"Save dashboard"**.

**Step 6: Monitor and Optimize**

1. Use **CloudWatch Insights** to analyze logs.
2. Set up **automated scaling** based on CloudWatch metrics.
3. Regularly review **CloudWatch reports** for optimization.

Logging: Define how to set up and manage logs in aws

**Step 1: Enable CloudWatch Logs**

1. Open the **AWS CloudWatch Console**.
2. Navigate to **Logs** and click **"Create log group"**.
3. Provide a **name** for the log group.
4. Set **retention policies** to manage log storage.
5. Click **"Create"**.

**Step 3: Enable Logging for AWS Services**

* **EC2 Instances**: Install and configure the **CloudWatch Agent**:

sudo yum install -y amazon-cloudwatch-agent

sudo systemctl enable amazon-cloudwatch-agent

sudo systemctl start amazon-cloudwatch-agent

 **S3 Buckets**: Enable **Server Access Logging** in the S3 console.

 **VPC Flow Logs**: Capture network traffic logs for security analysis.

**Step 4: Set Up AWS CloudTrail for API Logging**

1. Open the **AWS CloudTrail Console**.
2. Click **"Create trail"**.
3. Choose **"Apply to all regions"** for global logging.
4. Select an **S3 bucket** for log storage.
5. Click **"Create"**.

**Step 5: Configure Log Analysis and Alerts**

1. Use **CloudWatch Insights** to query logs.
2. Set up **CloudWatch Alarms** for log-based alerts.
3. Integrate with **AWS Lambda** for automated log processing.

**Step 6: Optimize Log Storage and Retention**

1. Enable **log rotation** to prevent excessive storage usage.
2. Archive logs in **Amazon S3** for long-term storage.
3. Use **AWS Glue** for log data transformation.

Backup and Disaster Recovery

**Step 1: Enable AWS Backup**

1. Open the **AWS Backup Console**.
2. Click **"Settings"** and enable the services you want to back up.
3. Click **"Confirm"**.

**Step 2: Create a Backup Vault**

1. Navigate to **Backup Vaults** and click **"Create backup vault"**.
2. Provide a **name** for the vault.
3. Choose **encryption settings** (AWS-managed or customer-managed keys).
4. Click **"Create backup vault"**.

**Step 3: Create a Backup Plan**

1. Go to **Backup Plans** and click **"Create backup plan"**.
2. Choose **"Build a new plan"** or **"Use an existing template"**.
3. Define **backup frequency** (e.g., daily, weekly).
4. Set **retention period** for backups.
5. Click **"Create plan"**.

**Step 4: Assign Resources to the Backup Plan**

1. Click **"Assign resources"** in the backup plan.
2. Select **resource types** (EC2, RDS, EFS, etc.).
3. Use **tags** to automatically include resources.
4. Click **"Assign"**.

**Step 6: Restore Data from Backup**

1. Go to **Backup Vaults** and select the backup.
2. Click **"Restore"**.
3. Choose the **restore destination** (e.g., EC2 instance, RDS database).
4. Click **"Restore backup"**.

Recovery Plan: Outline the disaster recovery plan.

**1. Define Recovery Objectives**

* **Recovery Time Objective (RTO)**: Maximum acceptable downtime.
* **Recovery Point Objective (RPO)**: Maximum acceptable data loss.
* **Business Impact Analysis**: Identify critical workloads and dependencies.

**2. Choose a Disaster Recovery Strategy**

AWS offers **four DR strategies**2:

1. **Backup and Restore**: Periodic backups stored in **Amazon S3 or AWS Backup**.
2. **Pilot Light**: Minimal infrastructure running in standby mode.
3. **Warm Standby**: A scaled-down version of production, ready for failover.
4. **Multi-Site Active/Active**: Fully redundant systems across multiple AWS regions.

**3. Implement Backup and Replication**

* **Use AWS Backup for automated backups.**
* **Enable Amazon S3 versioning for object recovery.**
* **Replicate databases using Amazon RDS Multi-AZ or AWS DMS.**
* **Store snapshots in Amazon S3 Glacier for long-term retention.**

**4. Set Up Failover Mechanisms**

* **Configure Route 53 DNS failover for automatic redirection.**
* **Use Elastic Load Balancing (ELB) to distribute traffic.**
* **Implement Auto Scaling to handle increased load during recovery.**

**5. Automate Recovery with Infrastructure as Code**

* **Use AWS CloudFormation or AWS CDK for rapid infrastructure deployment.**
* **Maintain Amazon Machine Images (AMIs) for quick instance restoration.**

**Troubleshooting**

**Common Issues: List common issues and their solutions, while deploying a web application in aws**

**1. Incorrect IAM Permissions**

* **Issue: Lack of proper IAM roles and policies can prevent services from interacting correctly.**
* **Solution: Ensure IAM roles have the necessary permissions. Use AWS IAM policies to grant least privilege access.**

**2. Misconfigured Security Groups**

* **Issue: Security groups may block necessary inbound/outbound traffic.**
* **Solution: Configure security groups to allow required ports (e.g., 80 for HTTP, 443 for HTTPS, and 22 for SSH).**

**3. Improper VPC and Subnet Configuration**

* **Issue: Incorrectly configured VPCs and subnets can prevent instances from communicating.**
* **Solution: Set up public and private subnets correctly, attach an Internet Gateway for public access, and use NAT Gateway for private subnets.**

**4. Application Load Balancer Misconfiguration**

* **Issue: Load balancer may not route traffic correctly.**
* **Solution: Ensure target groups are correctly associated, health checks are properly configured, and listener rules are set up.**

**5. Database Connectivity Issues**

* **Issue: Web application fails to connect to the database.**
* **Solution: Verify security group rules, ensure the database is in the correct subnet, and check credentials.**

**6. Auto Scaling and Elastic Load Balancing Issues**

* **Issue: Instances may not scale properly under load.**
* **Solution: Configure Auto Scaling policies correctly and ensure the load balancer is distributing traffic evenly.**

**7. Incorrect Route 53 DNS Configuration**

* **Issue: Domain name does not resolve correctly.**
* **Solution: Ensure Route 53 records are correctly set up and point to the right resources.**

**Support: Information on how to get help or support.**

**1. AWS Support Plans**

* **AWS offers different support plans, including Basic, Developer, Business, and Enterprise. You can check details and subscribe to a plan that fits your needs here.**

**2. AWS Documentation & Tutorials**

* **AWS provides extensive documentation and step-by-step guides for deploying web applications. You can explore tutorials like this one for building a basic web application.**

**3. AWS Forums & Community**

* **Engage with the AWS community through AWS Forums, Stack Overflow, and Reddit to get insights from experienced developers.**

**4. AWS Support Center**

* **You can create a support case through the AWS Support Center for technical assistance.**

**5. AWS Well-Architected Framework**

* **Follow AWS best practices for deploying applications using the AWS Well-Architected Framework.**

**Additional Resources ● Documentation: Links to AWS documentation and other helpful resource**

* [**https://www.cloudfolkshub.com/t/u/activeCourses**](https://www.cloudfolkshub.com/t/u/activeCourses)
* [**AWSTrainingLive – Twitch**](https://www.twitch.tv/awstraininglive)
* [**Welcome to AWS Documentation**](https://docs.aws.amazon.com/)
* [**AWS Educate - Cloud Skills for Education- AWSAWS Educate - Cloud Skills for Education- AWS**](https://aws.amazon.com/education/awseducate/)

Thank you

