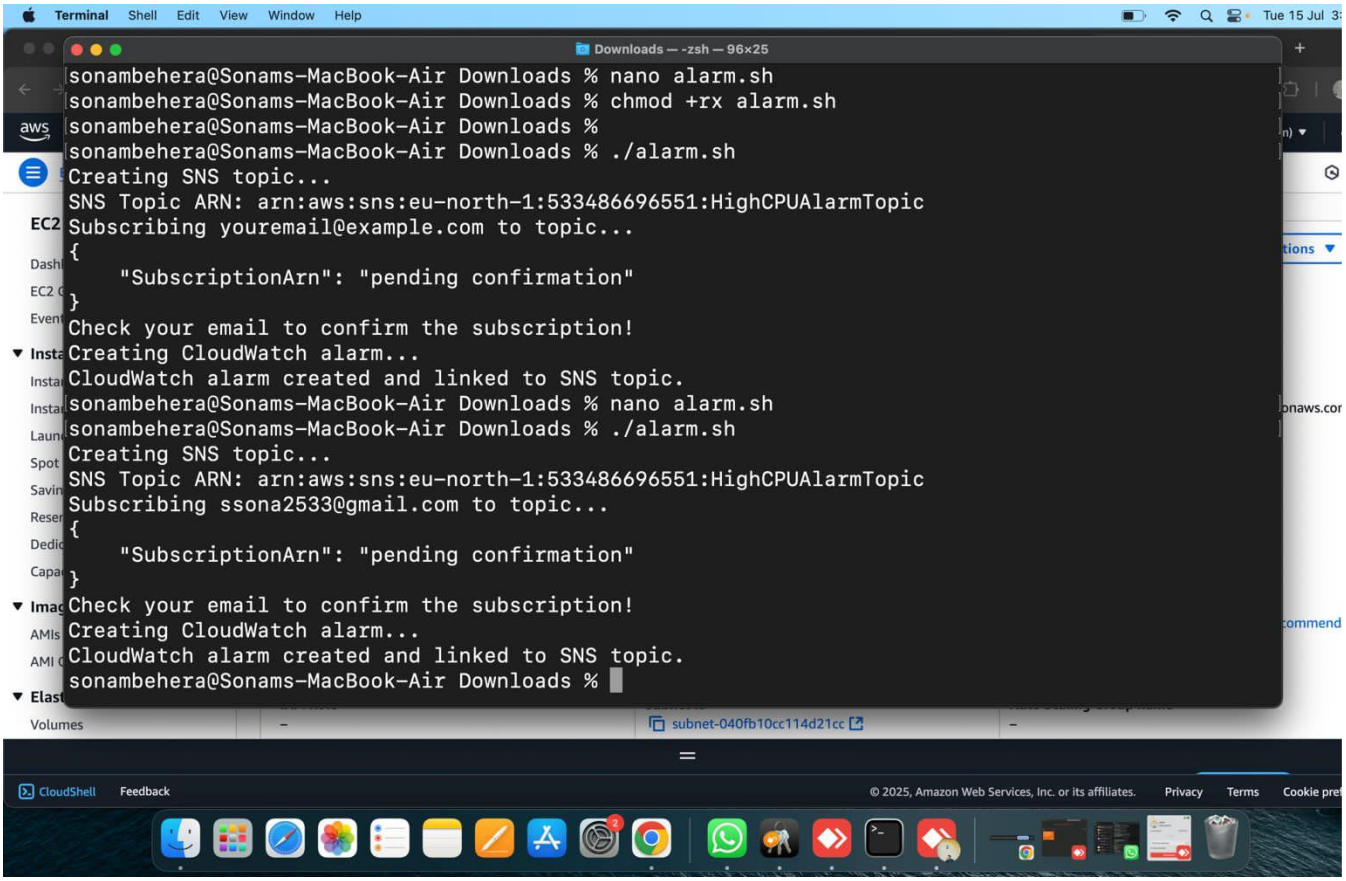
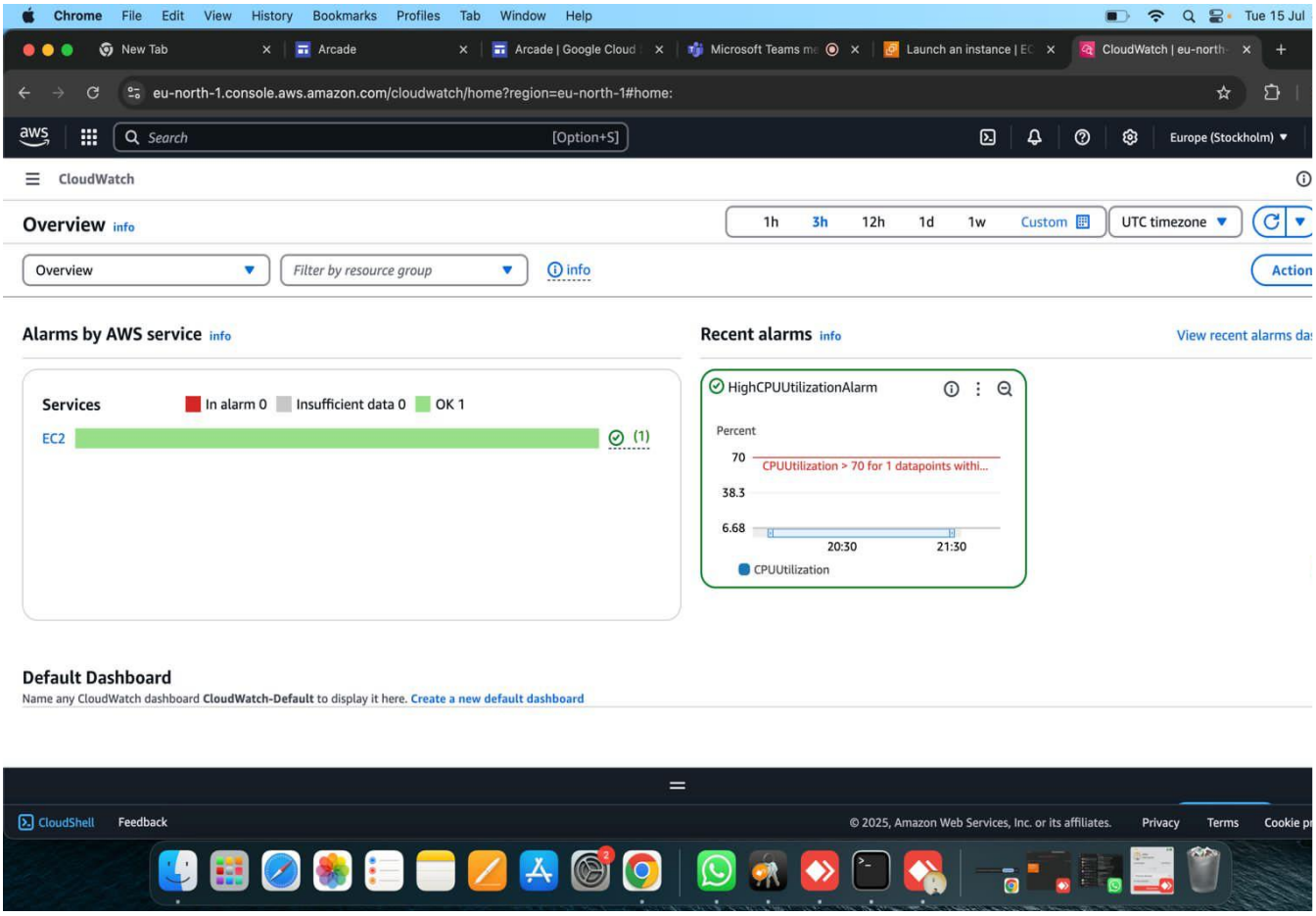
Q1.creates an alarm for cputilization which trigger the threshold is greater than to create CNS topic and attach to the alaram to the instance created

```
REGION="eu-north-1"
INSTANCE_ID="i-04b2cd6ac16bf5ab6"
EMAIL="ssona2533@gmail.com"
TOPIC_NAME="HighCPUAlarmTopic"
ALARM_NAME="HighCPUUtilizationAlarm"

# Step 1: Create SNS topic
echo "Creating SNS topic..."
TOPIC_ARN=$(aws sns create-topic \
  --name $TOPIC_NAME \
  --region $REGION \
  --query 'TopicArn' \
  --output text)
echo "SNS Topic ARN: $TOPIC_ARN"

# Step 2: Subscribe email to SNS topic
echo "Subscribing $EMAIL to topic..."
aws sns subscribe \
  --topic-arn $TOPIC_ARN \
  --protocol email \
  --notification-endpoint $EMAIL \
  --region $REGION
echo "Check your email to confirm the subscription!"

# Step 3: Create CloudWatch Alarm
echo "Creating CloudWatch alarm..."
aws cloudwatch put-metric-alarm \
  --alarm-name $ALARM_NAME \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Average \
  --period 300 \
  --threshold 70 \
  --comparison-operator GreaterThanThreshold \
  --dimensions Name=InstanceId,Value=$INSTANCE_ID \
  --evaluation-periods 1 \
  --alarm-actions $TOPIC_ARN \
  --region $REGION
echo "CloudWatch alarm created and linked to SNS topic."
```

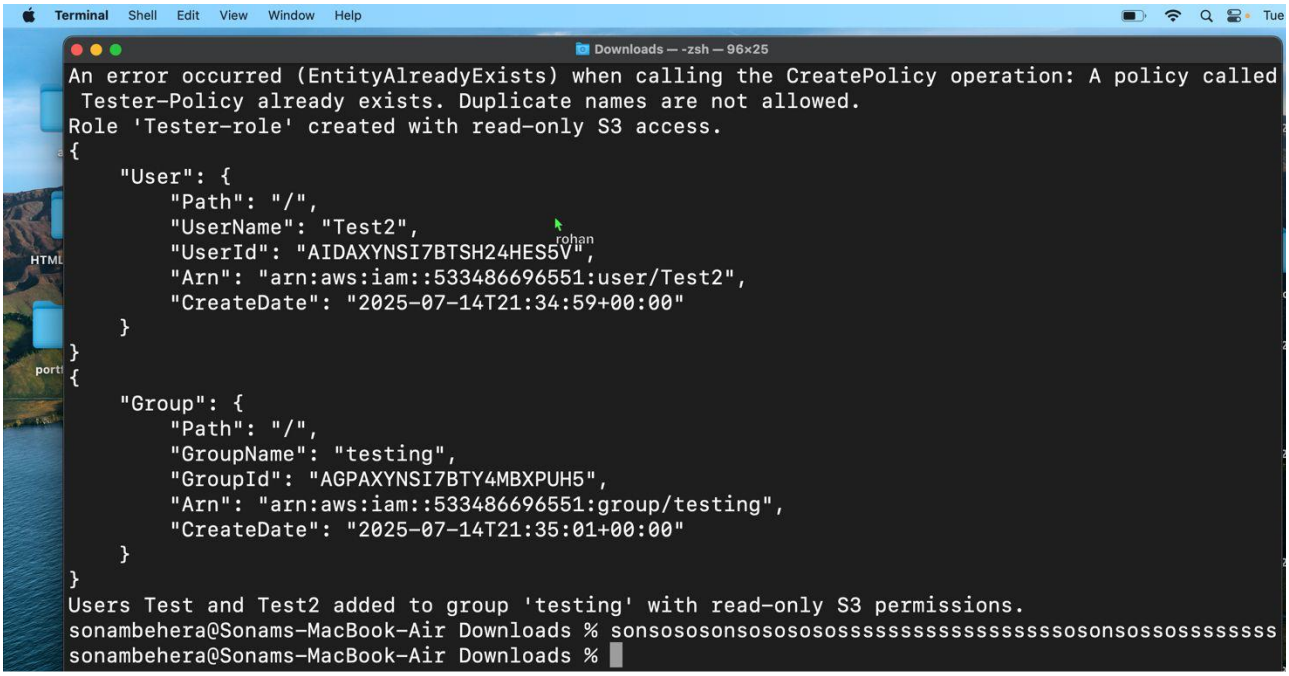

Terminal Shell Edit View Window Help
Tue 15 Jul
Downloads — -zsh — 96×25
sonambehera@Sonams-MacBook-Air Downloads % nano alarm.sh
sonambehera@Sonams-MacBook-Air Downloads % chmod +rx alarm.sh
sonambehera@Sonams-MacBook-Air Downloads %
sonambehera@Sonams-MacBook-Air Downloads % ./alarm.sh
Creating SNS topic...
SNS Topic ARN: arn:aws:sns:eu-north-1:533486696551:HighCPUAlarmTopic
Subscribing youremail@example.com to topic...
{
"SubscriptionArn": "pending confirmation"
}
Check your email to confirm the subscription!
Creating CloudWatch alarm...
CloudWatch alarm created and linked to SNS topic.
sonambehera@Sonams-MacBook-Air Downloads % nano alarm.sh
sonambehera@Sonams-MacBook-Air Downloads % ./alarm.sh
Creating SNS topic...
SNS Topic ARN: arn:aws:sns:eu-north-1:533486696551:HighCPUAlarmTopic
Subscribing ssona2533@gmail.com to topic...
{
"SubscriptionArn": "pending confirmation"
}
Check your email to confirm the subscription!
Creating CloudWatch alarm...
CloudWatch alarm created and linked to SNS topic.
sonambehera@Sonams-MacBook-Air Downloads %
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms

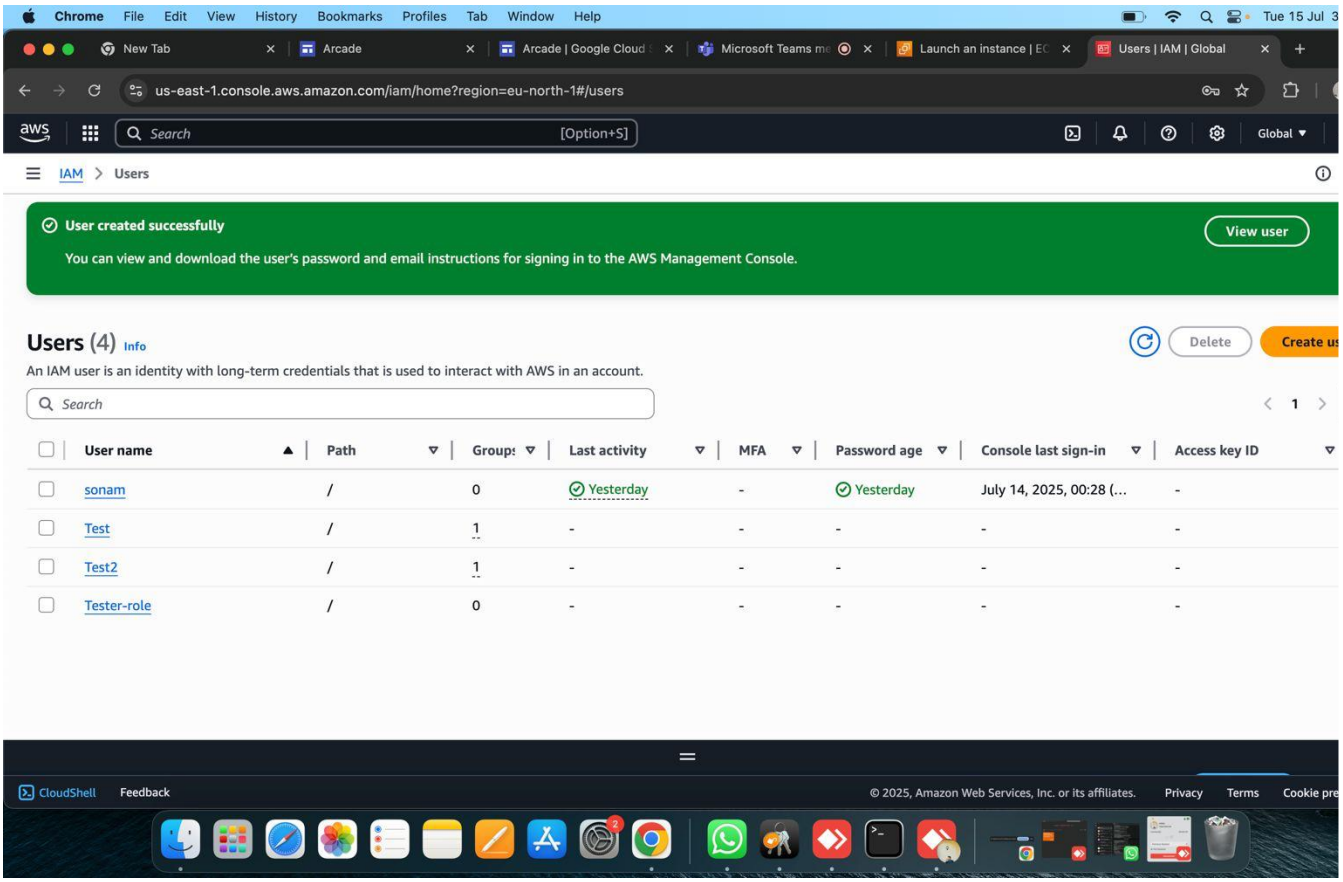

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue 15 Jul
New Tab
Arcade
Arcade | Google Cloud
Microsoft Teams
Launch an instance | EC
CloudWatch | eu-north-
eu-north-1.console.aws.amazon.com/cloudwatch/home?region=eu-north-1#home:
Search
[Option+S]
Europe (Stockholm)
CloudWatch
Overview info
1h
3h
12h
1d
1w
Custom
UTC timezone
Overview
Filter by resource group
info
Action
Alarms by AWS service info
Services
In alarm 0
Insufficient data 0
OK 1
EC2
(1)
Recent alarms info
View recent alarms da
HighCPUUtilizationAlarm
Percent
70
CPUUtilization > 70 for 1 datapoints withi...
38.3
6.68
20:30
21:30
CPUUtilization
Default Dashboard
Name any CloudWatch dashboard CloudWatch-Default to display it here. Create a new default dashboard
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms

Q2.Write a Bash script to create an IAM user named Test.Add Adminstrator access to the IAM user.
Q3.Write a Bash script to create an IAM role named Tester-role.In the same script create a Access policy for Tester-Role named Tester-policy.The policy should provide readonly access to s3.Attach the role to the policy.
Q4.Extend the above bash script to create another user named Test2.Add Test1 and Test2 to a user group named testing (which should also be created through the script).Attach Tester Role to all users in the user group testing.

```
#!/bin/bash

# 1. Create IAM user "Test" and attach AdministratorAccess
aws iam create-user --user-name Test

aws iam attach-user-policy \
  --user-name Test \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

echo "User 'Test' created with Administrator access."

# 2. Create IAM role "Tester-role" with trust policy for EC2
TRUST_POLICY='{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "ec2.amazonaws.com" },
      "Action": "sts:AssumeRole"
    }
  ]
}'

aws iam create-role \
  --role-name Tester-role \
  --assume-role-policy-document "$TRUST_POLICY"

# Create inline policy JSON for read-only S3 access
cat > s3-readonly-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
```

```
  ]
}
EOF

# Create policy and attach it to the role
aws iam create-policy \
  --policy-name Tester-Policy \
  --policy-document file://s3-readonly-policy.json

# Get ARN of policy
POLICY_ARN=$(aws iam list-policies --query "Policies[?PolicyName=='Tester-Policy'].Arn" --
output text)

aws iam attach-role-policy \
  --role-name Tester-role \
  --policy-arn "$POLICY_ARN"

echo "Role 'Tester-role' created with read-only S3 access."

# 3. Create user Test2
aws iam create-user --user-name Test2

# Create group 'testing'
aws iam create-group --group-name testing

# Add users Test and Test2 to the group
aws iam add-user-to-group --user-name Test --group-name testing
aws iam add-user-to-group --user-name Test2 --group-name testing

# Attach Tester-role policy to group (can't attach roles to groups directly)
# So we attach the same S3-readonly policy to the group
aws iam attach-group-policy \
  --group-name testing \
  --policy-arn "$POLICY_ARN"

echo "Users Test and Test2 added to group 'testing' with read-only S3 permissions."
```

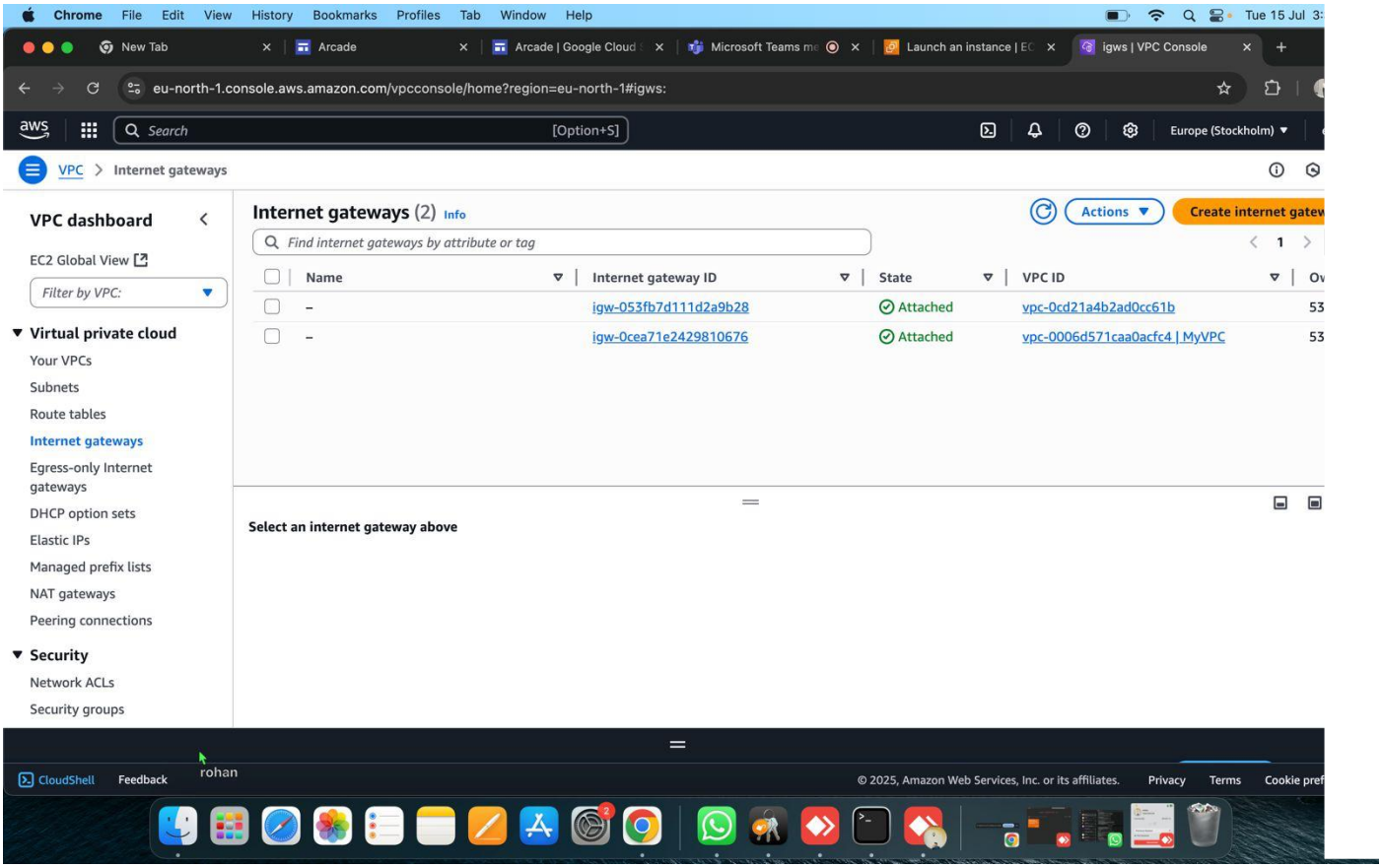




Q5.Write a bash script to create a vpc .create two subnet inside the vpc .create an internet gateway attach the internet gateway to the vpc.create a routing table attach the routing table to public subnet .edit routing table open a routing table with destination 0.0.0./0 attach it to the IG.

```
!/bin/bash

REGION="eu-north-1"
AMI_ID="ami-0914547665e6a707c"   # Ubuntu 22.04 LTS in eu-north-1
INSTANCE_TYPE="t3.micro"
KEY_NAME="mykey"
SECURITY_GROUP_NAME="public-sg"
TAG_NAME="default"

# Step 1: Fetch VPC ID
VPC_ID=$(aws ec2 describe-vpcs \
  --region "$REGION" \
  --query 'Vpcs[0].VpcId' \
  --output text)

echo "VPC ID: $VPC_ID"

# Step 2: Get first subnet ID in the VPC (assume public if VPC has internet gateway setup)
SUBNET_ID=$(aws ec2 describe-subnets \
  --region "$REGION" \
  --filters "Name=vpc-id,Values=$VPC_ID" \
  --query "Subnets[0].SubnetId" \
  --output text)

echo "Subnet ID: $SUBNET_ID"

# Step 3: Launch EC2 instance
INSTANCE_ID=$(aws ec2 run-instances \
  --region "$REGION" \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-groups "$SECURITY_GROUP_NAME" \
  --subnet-id "$SUBNET_ID" \
  --associate-public-ip-address \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG_NAME}]" \
  --query 'Instances[0].InstanceId' \
  --output text)

echo "Subnet ID: $SUBNET_ID"

# Step 3: Launch EC2 instance
INSTANCE_ID=$(aws ec2 run-instances \
  --region "$REGION" \
  --image-id "$AMI_ID" \
  --count 1 \
```

```
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-groups "$SECURITY_GROUP_NAME" \
  --subnet-id "$SUBNET_ID" \
  --associate-public-ip-address \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$TAG_NAME}]" \
  --query 'Instances[0].InstanceId' \
  --output text)

echo "Instance ID: $INSTANCE_ID"

# Step 4: Fetch public IP
PUBLIC_IP=$(aws ec2 describe-instances \
  --region "$REGION" \
  --instance-ids "$INSTANCE_ID" \
  --query "Reservations[0].Instances[0].PublicIpAddress" \
  --output text)

echo "Public IP: $PUBLIC_IP"
echo "To SSH: ssh -i ~/Downloads/$KEY_NAME.pem ubuntu@$PUBLIC_IP"
```

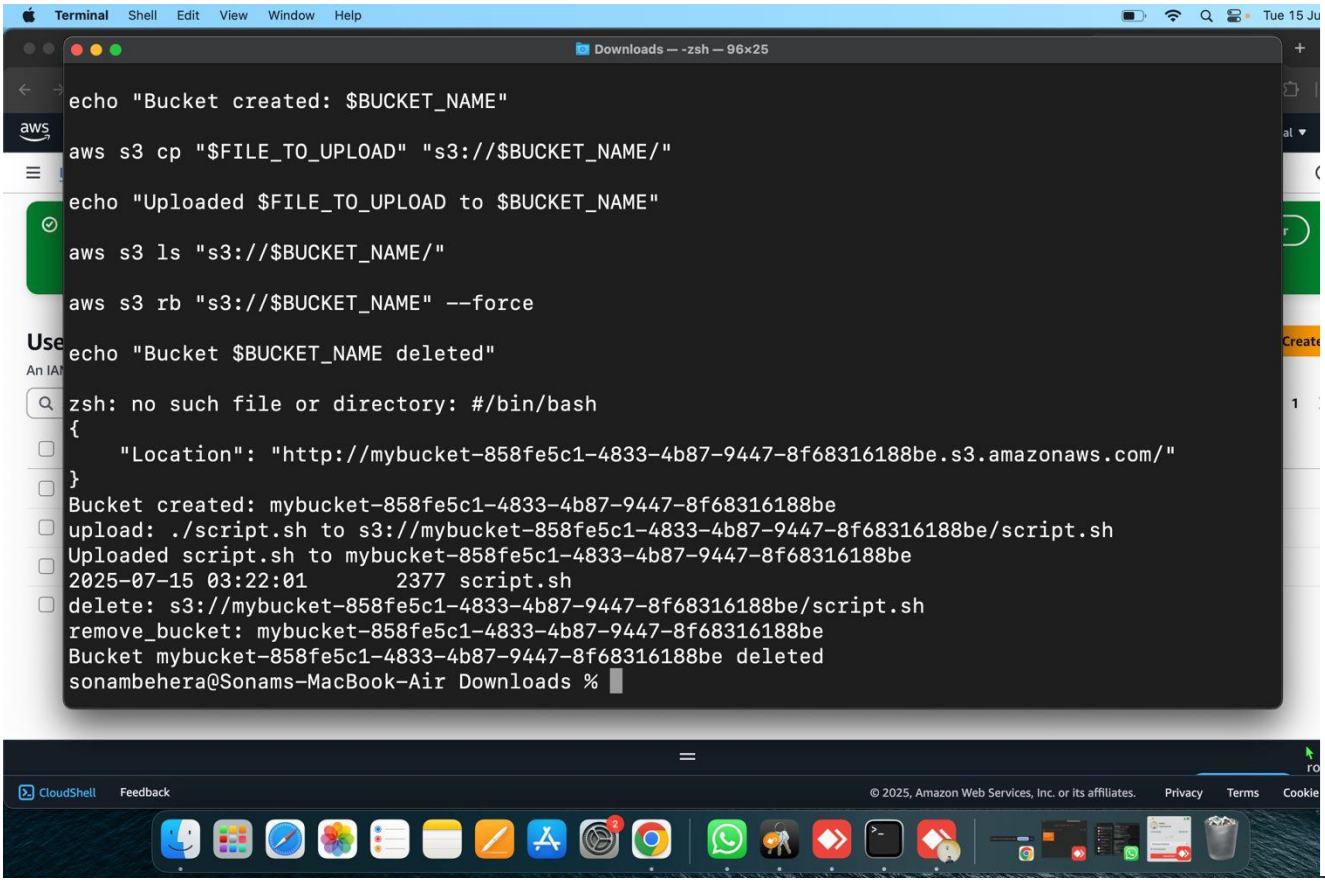


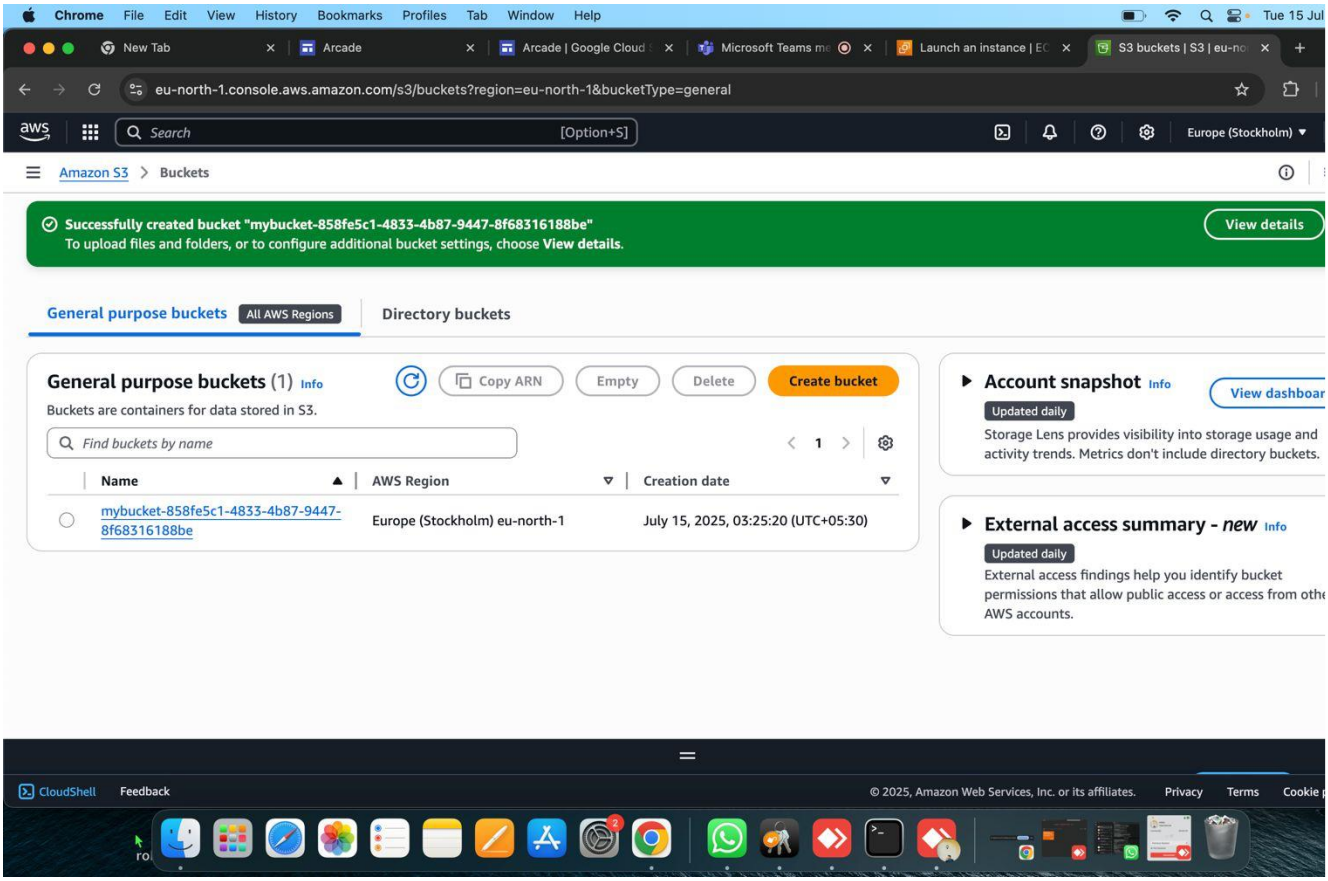
Q6.Write a bash script to create a s3 bucket.The bucket name should be created using a random function where you have no idea of the name of the bucket .use some commands to retrieve the bucket name and cp a script file to it.Display the contents of the bucket and delete it.
Q7.Write a bash script to modify the access control policy of an existing bucket.
Q8.write a bash script to create a bucket and host a static website in it.is the website accessible through curl and through browser?
Q9.Using AWS CLI create an EC2 instance.Use some command to display the public dns address of the ec2 instance.Using the public dns ssh into the instance through the script.Write another remote script that should be copied to the ec2 instance.This script should contain steps to create a bucket and attach an access point to it.

```
#!/bin/bash

# Configuration
REGION="eu-north-1"
KEY_NAME="ubuntu-key"
KEY_FILE="$KEY_NAME.pem"
SECURITY_GROUP_NAME="ubuntu-sg"
INSTANCE_TYPE="t3.micro"
SCRIPT_FILE="setup.sh"

# Step 1: Create Key Pair
echo "Creating key pair: $KEY_NAME"
aws ec2 create-key-pair --region "$REGION" \
  --key-name "$KEY_NAME" \
  --query 'KeyMaterial' \
  --output text > "$KEY_FILE"

chmod 400 "$KEY_FILE"
echo "Key saved to $KEY_FILE"

# Step 2: Create Security Group
echo "Creating security group: $SECURITY_GROUP_NAME"
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --region "$REGION" \
  --group-name "$SECURITY_GROUP_NAME" \
  --description "Security group for Ubuntu instance" \
  --query 'GroupId' --output text)

echo "Security group created: $SECURITY_GROUP_ID"

# Step 3: Authorize SSH access (port 22)
echo "Authorizing SSH access..."
aws ec2 authorize-security-group-ingress \
  --region "$REGION" \
  --group-id "$SECURITY_GROUP_ID" \
```

```
  --protocol tcp \
  --port 22 \
  --cidr 0.0.0.0/0

# Step 4: Get latest Ubuntu 22.04 AMI ID
echo "Fetching latest Ubuntu 22.04 AMI ID..."
AMI_ID=$(aws ec2 describe-images \



AMI_ID=$(aws ec2 describe-images \
  --region "$REGION" \
  --owners 099720109477 \
  --filters \
    "Name=name,Values=ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*" \
    "Name=architecture,Values=x86_64" \
    "Name=virtualization-type,Values=hvm" \
    "Name=root-device-type,Values=ebs" \
  --query 'Images[*].[ImageId,CreationDate]' \
  --output text | \
  sort -k2 -r | \
  head -n 1 | \
  awk '{print $1}')

echo "Latest AMI ID: $AMI_ID"

# Step 5: Launch EC2 instance
echo "Launching EC2 instance..."
INSTANCE_ID=$(aws ec2 run-instances \
  --region "$REGION" \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --query 'Instances[0].InstanceId' \
  --output text)

echo "Instance launched with ID: $INSTANCE_ID"

# Step 6: Fetch public IP
echo "Fetching public IP..."
PUBLIC_IP=$(aws ec2 describe-instances \
  --region "$REGION" \
  --instance-ids "$INSTANCE_ID" \
  --query "Reservations[0].Instances[0].PublicIpAddress" \
  --output text)
```

```
echo "Public IP: $PUBLIC_IP"
echo "Connect using: ssh -i $KEY_FILE ubuntu@$PUBLIC_IP"
# Step 7: Copy script to EC2 instance
if [[ -f "$SCRIPT_FILE" ]]; then
  echo "Sending $SCRIPT_FILE to EC2 instance..."
  scp -i "$KEY_FILE" "$SCRIPT_FILE" ubuntu@"$PUBLIC_IP":~
else
  echo "No $SCRIPT_FILE found. Skipping SCP."
fi
```





Q10.Write a Bash script to create an EC2 instance and ssh into it from the script.
Q11.Write a script to create an EC2 instance and ssh into it and install nginx.
Q12.Write a script to create a key-pair, security group, and create the instance using the key-pair and security group.

```
#!/bin/bash

# Configuration
REGION="${REGION:-$(aws configure get region)}"
KEY_NAME="mykey"
KEY_FILE="$KEY_NAME.pem"
SECURITY_GROUP_NAME="ubuntu-sg"
INSTANCE_TYPE="t3.micro"
SCRIPT_FILE="setup.sh"

# Step 1: Create key pair
echo "Creating key pair: $KEY_NAME"
aws ec2 create-key-pair --region "$REGION" --key-name "$KEY_NAME" --query
'KeyMaterial' --output text > "$KEY_FILE"
chmod 400 "$KEY_FILE"
echo "Key saved to $KEY_FILE"

# Step 2: Create security group
echo "Creating security group: $SECURITY_GROUP_NAME"
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
```

```
  --region "$REGION" \
  --group-name "$SECURITY_GROUP_NAME" \
  --description "Security group for Ubuntu instance" \
  --query 'GroupId' \
  --output text)
echo "Security group created: $SECURITY_GROUP_ID"

# Step 3: Allow SSH access
echo "Authorizing SSH access (port 22)"
aws ec2 authorize-security-group-ingress \
  --region "$REGION" \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr 0.0.0.0/0

# Step 4: Find latest Ubuntu 22.04 AMI ID
echo "Fetching latest Ubuntu 22.04 LTS AMI ID..."
AMI_ID=$(aws ec2 describe-images \
  --region "$REGION" \
  --owners 099720109477 \
  --filters \

--group-name "$SECURITY_GROUP_NAME" \
  --description "Security group for Ubuntu instance" \
  --query 'GroupId' \
  --output text)
echo "Security group created: $SECURITY_GROUP_ID"

# Step 3: Allow SSH access
echo "Authorizing SSH access (port 22)"
aws ec2 authorize-security-group-ingress \
  --region "$REGION" \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr 0.0.0.0/0

# Step 4: Find latest Ubuntu 22.04 AMI ID
echo "Fetching latest Ubuntu 22.04 LTS AMI ID..."
AMI_ID=$(aws ec2 describe-images \
  --region "$REGION" \
  --owners 099720109477 \
  --filters \

"Name=name,Values=ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"
\
```

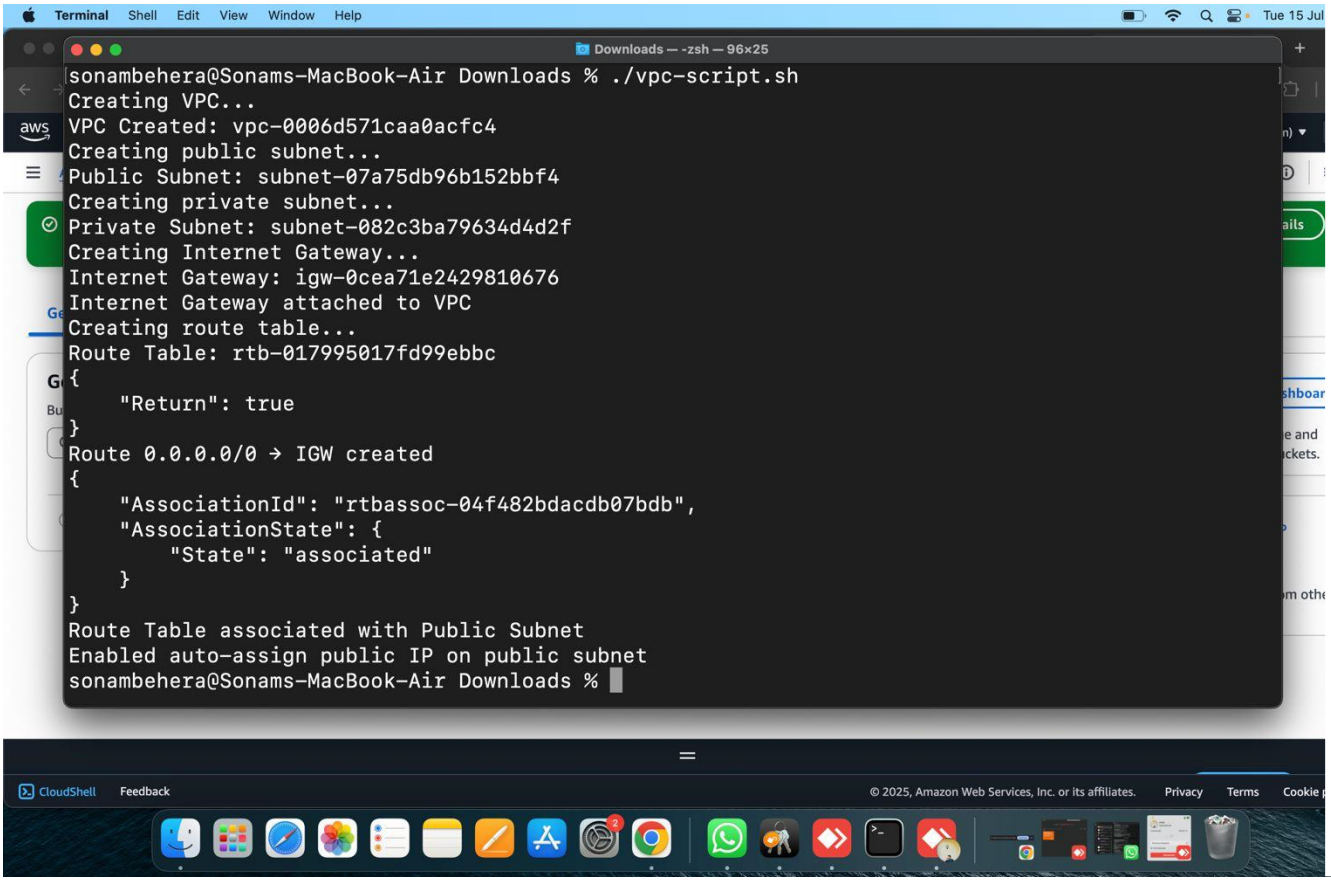
```
    "Name=architecture,Values=x86_64" \
    "Name=virtualization-type,Values=hvm" \
    "Name=root-device-type,Values=ebs" \
  --query 'Images[*].[ImageId,CreationDate]' \
  --output text | \
  sort -k2 -r | \
  head -n 1 | \
  awk '{print $1}')

echo "Latest Ubuntu AMI: $AMI_ID"

# Step 5: Launch instance
echo "Launching EC2 instance..."
INSTANCE_ID=$(aws ec2 run-instances \
  --region "$REGION" \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type "$INSTANCE_TYPE" \

--key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --query "Instances[0].InstanceId" \
  --output text)
echo "Instance launched: $INSTANCE_ID"

# Step 6: Get public IP
echo "Fetching public IP..."
PUBLIC_IP=$(aws ec2 describe-instances \
  --region "$REGION" \
  --instance-ids "$INSTANCE_ID" \
  --query "Reservations[0].Instances[0].PublicIpAddress" \
  --output text)

echo "Public IP: $PUBLIC_IP"
echo "Connect using: ssh -i $KEY_FILE ubuntu@$PUBLIC_IP"

# Step 7: Copy setup file to instance
echo "Sending $SCRIPT_FILE to EC2 instance..."
scp -i "$KEY_FILE" "$SCRIPT_FILE" ubuntu@"$PUBLIC_IP":~
```

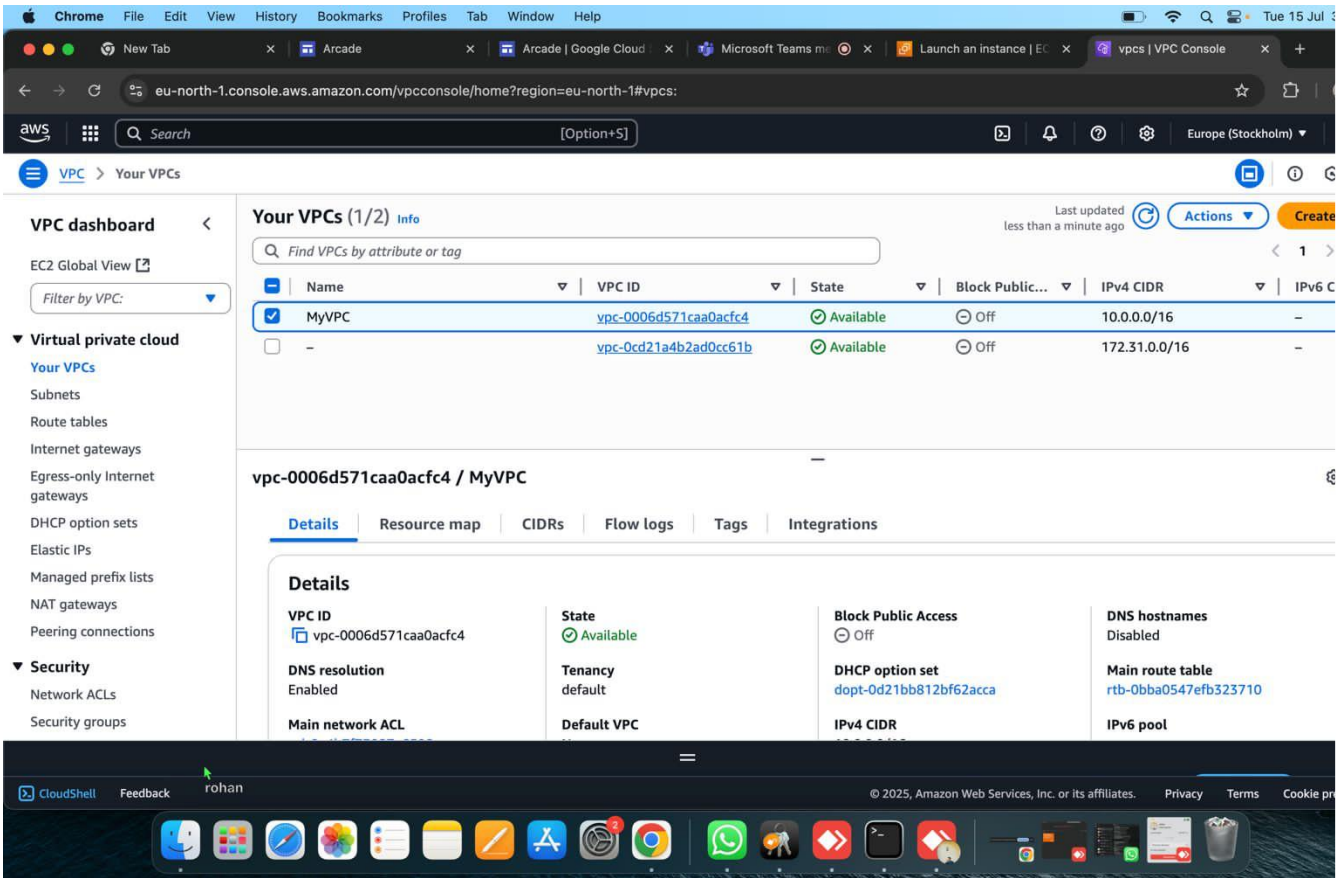
q11.Write a bash script to create a vpc create two subnet inside the vpc.create an internet gateway attch the internet gateway attach the internetgateway to the vpc.create a routing table attach the routing table open a routing table with destination 0.0.0./0 attach to the IG

```
#!/bin/bash

REGION="eu-north-1"
CIDR_BLOCK="10.0.0.0/16"
```

```
PUBLIC_SUBNET_CIDR="10.0.1.0/24"
PRIVATE_SUBNET_CIDR="10.0.2.0/24"

# Create VPC
echo "Creating VPC..."
VPC_ID=$(aws ec2 create-vpc \
  --cidr-block $CIDR_BLOCK \
  --region $REGION \
  --query 'Vpc.VpcId' \
  --output text)
echo "VPC Created: $VPC_ID"

# Tag the VPC
aws ec2 create-tags \
  --resources $VPC_ID \
  --tags Key=Name,Value=MyVPC \
  --region $REGION

# Create Subnets
echo "Creating public subnet..."
PUBLIC_SUBNET_ID=$(aws ec2 create-subnet \
  --vpc-id $VPC_ID \
  --cidr-block $PUBLIC_SUBNET_CIDR \
  --availability-zone "${REGION}a" \
  --region $REGION \
  --query 'Subnet.SubnetId' \
  --output text)
echo "Public Subnet: $PUBLIC_SUBNET_ID"

echo "Creating private subnet..."
PRIVATE_SUBNET_ID=$(aws ec2 create-subnet \
  --vpc-id $VPC_ID \
  --cidr-block $PRIVATE_SUBNET_CIDR \
  --availability-zone "${REGION}b" \
  --region $REGION \
  --query 'Subnet.SubnetId' \
  --output text)
echo "Private Subnet: $PRIVATE_SUBNET_ID"
# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway \
  --region $REGION \
  --query 'InternetGateway.InternetGatewayId' \
  --output text)
echo "Internet Gateway: $IGW_ID"

# Attach IGW to VPC
```

```
aws ec2 attach-internet-gateway \
  --internet-gateway-id $IGW_ID \
  --vpc-id $VPC_ID \
  --region $REGION
echo "Internet Gateway attached to VPC"

# Create Route Table
echo "Creating route table..."
RT_ID=$(aws ec2 create-route-table \
  --vpc-id $VPC_ID \
  --region $REGION \
# Add Route to Internet via IGW
aws ec2 create-route \
  --route-table-id $RT_ID \
  --destination-cidr-block 0.0.0.0/0 \
  --gateway-id $IGW_ID \
  --region $REGION
echo "Route 0.0.0.0/0 → IGW created"

# Associate Route Table with Public Subnet
aws ec2 associate-route-table \
  --subnet-id $PUBLIC_SUBNET_ID \
  --route-table-id $RT_ID \
  --region $REGION
echo "Route Table associated with Public Subnet"

# Optional: Enable auto-assign public IP on public subnet
aws ec2 modify-subnet-attribute \

--subnet-id $PUBLIC_SUBNET_ID \
  --map-public-ip-on-launch \
  --region $REGION
echo "Enabled auto-assign public IP on public subnet"
```


Terminal Shell Edit View Window Help
Tue 15 Jul
Downloads — -zsh — 96×25
sonambehera@Sonams-MacBook-Air Downloads % ./vpc-script.sh
Creating VPC...
VPC Created: vpc-0006d571caa0acfc4
Creating public subnet...
Public Subnet: subnet-07a75db96b152bbf4
Creating private subnet...
Private Subnet: subnet-082c3ba79634d4d2f
Creating Internet Gateway...
Internet Gateway: igw-0cea71e2429810676
Internet Gateway attached to VPC
Creating route table...
Route Table: rtb-017995017fd99ebbc
{
"Return": true
}
Route 0.0.0.0/0 → IGW created
{
"AssociationId": "rtbassoc-04f482bdacdb07bdb",
"AssociationState": {
"State": "associated"
}
}
Route Table associated with Public Subnet
Enabled auto-assign public IP on public subnet
sonambehera@Sonams-MacBook-Air Downloads %
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue 15 Jul
New Tab
Arcade
Arcade | Google Cloud
Microsoft Teams
Launch an instance | EC
vpcs | VPC Console
eu-north-1.console.aws.amazon.com/vpcconsole/home?region=eu-north-1#vpcs:
Search [Option+S]
Europe (Stockholm)
VPC > Your VPCs
VPC dashboard
EC2 Global View
Filter by VPC:
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only Internet gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Peering connections
Security
Network ACLs
Security groups
Your VPCs (1/2) Info
Last updated less than a minute ago
Actions
Create
Find VPCs by attribute or tag
Name
VPC ID
State
Block Public...
IPv4 CIDR
IPv6 C
MyVPC
vpc-0006d571caa0acfc4
Available
Off
10.0.0.0/16
–
vpc-0cd21a4b2ad0cc61b
Available
Off
172.31.0.0/16
vpc-0006d571caa0acfc4 / MyVPC
Details
Resource map
CIDRs
Flow logs
Tags
Integrations
Details
VPC ID
vpc-0006d571caa0acfc4
State
Available
Block Public Access
Off
DNS hostnames
Disabled
DNS resolution
Enabled
Tenancy
default
DHCP option set
dopt-0d21bb812bf62acca
Main route table
rtb-0bba0547efb323710
Main network ACL
Default VPC
IPv4 CIDR
IPv6 pool
CloudShell Feedback
rohan
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms