Name: Reena Qureshi

Reg No: 2023-BSE-052

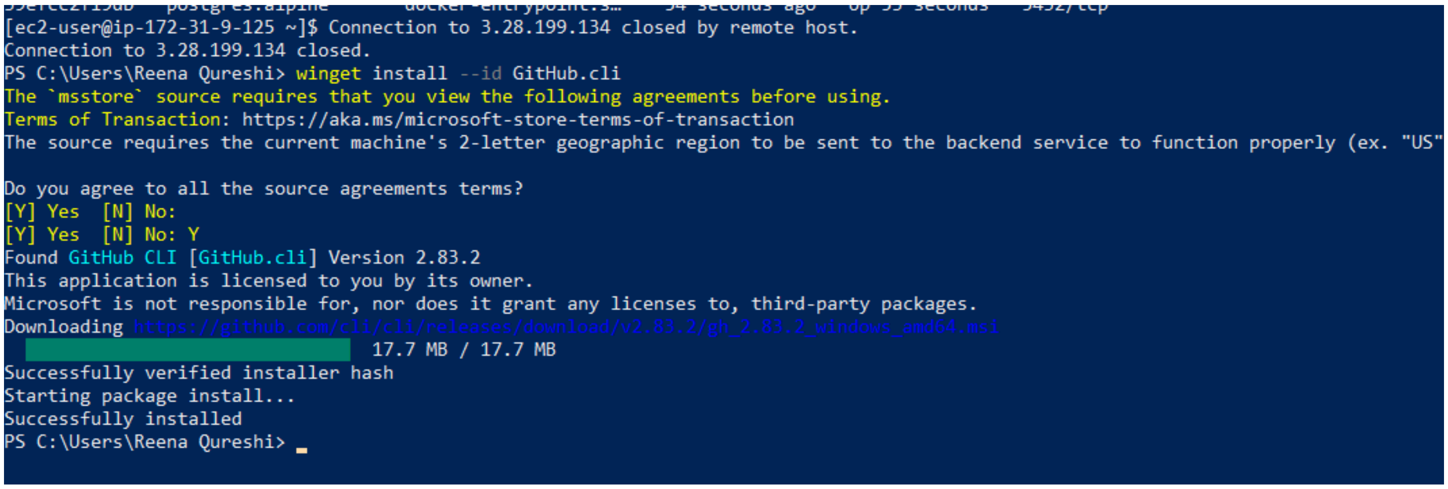
Section: V-B

LAB 9

Task 1 — GitHub CLI, Codespace setup and authentication

(Local desktop) Install GitHub CLI (Windows example via winget):

winget install --id GitHub.cli



```
[ec2-user@ip-172-31-9-125 ~]$ Connection to 3.28.199.134 closed by remote host.
Connection to 3.28.199.134 closed.
PS C:\Users\Reena Qureshi> winget install --id GitHub.cli
The `msstore` source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to the backend service to function properly (ex. "US"

Do you agree to all the source agreements terms?
[Y] Yes  [N] No:
[Y] Yes  [N] No: Y
Found GitHub CLI [GitHub.cli] Version 2.83.2
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://github.com/cli/cli/releases/download/v2.83.2/gh_2.83.2_windows_amd64.msi
  ██████████████████████████████  17.7 MB / 17.7 MB
Successfully verified installer hash
Starting package install...
Successfully installed
PS C:\Users\Reena Qureshi> _
```

(Local) Authenticate GH CLI for Codespaces:

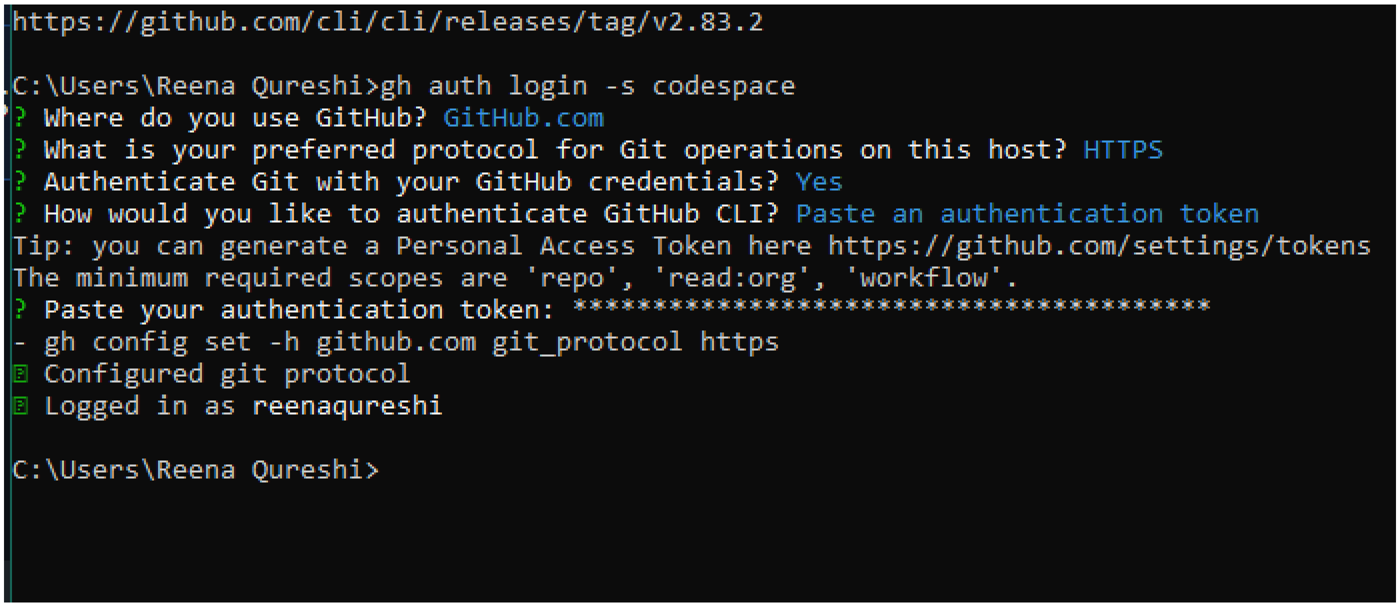
gh auth login -s codespace

When prompted, generate a GitHub Access Token (classic) in the browser with the following Token scopes:

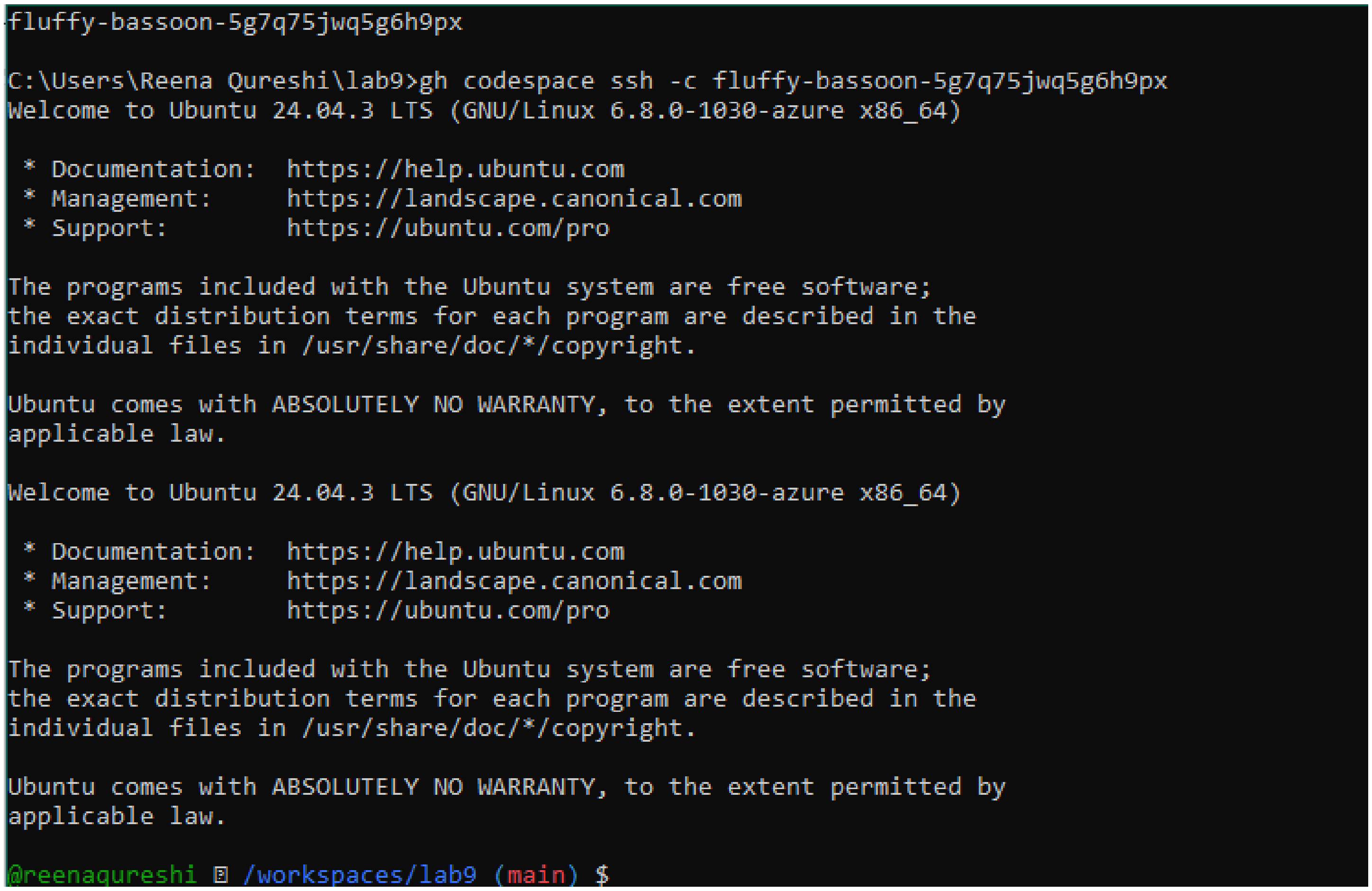
admin:org

codespace

repo


```
https://github.com/cli/cli/releases/tag/v2.83.2

C:\Users\Reena Qureshi>gh auth login -s codespace
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: ****************************************
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as reenaqureshi

C:\Users\Reena Qureshi>
```

(Local) Create or connect to a Codespace. To create a new codespace from the current repo:

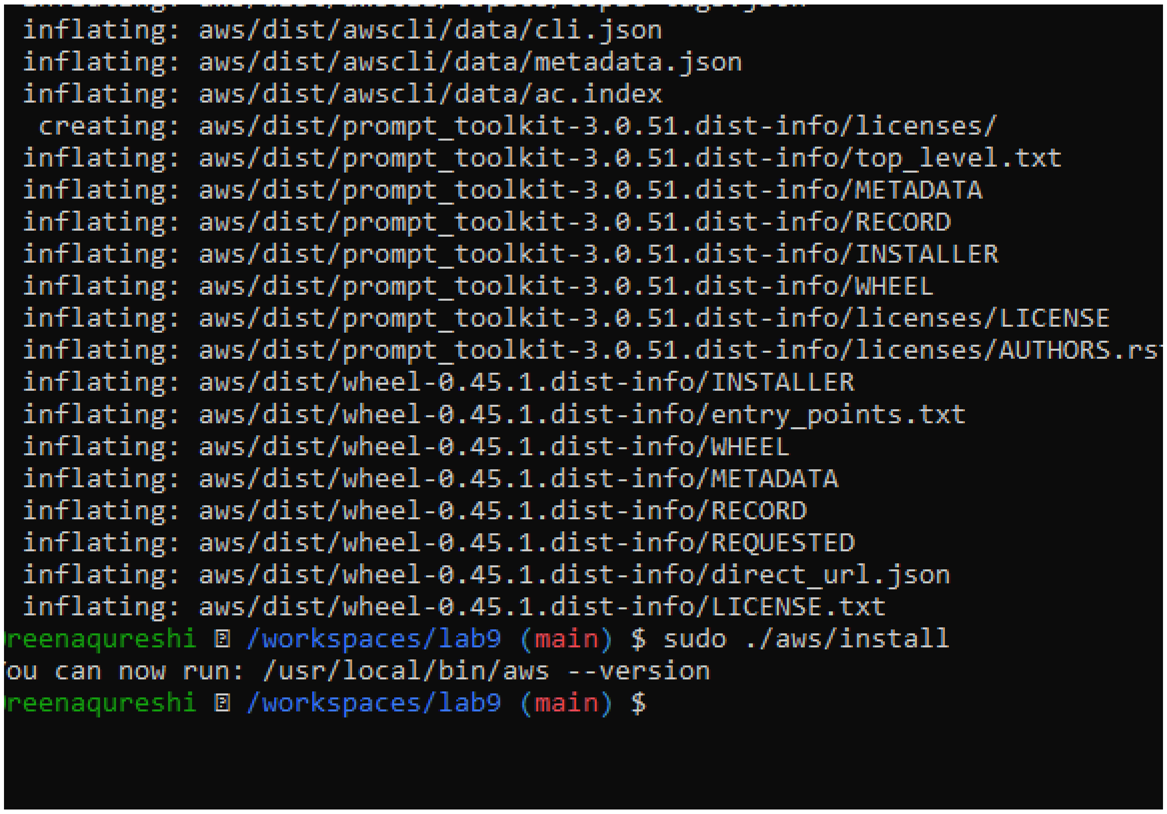

```
fluffy-bassoon-5g7q75jwq5g6h9px

C:\Users\Reena Qureshi\lab9>gh codespace ssh -c fluffy-bassoon-5g7q75jwq5g6h9px
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-1030-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-1030-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

@reenaqureshi ➜ /workspaces/lab9 (main) $
```

Task 2 — Install AWS CLI inside the Codespace and configure it

Goal: Install the AWS CLI in the Codespace and configure it to interact with your AWS account.

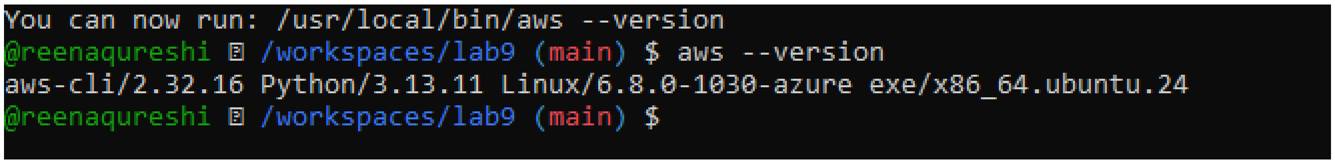
Inside the Codespace shell run:

Download and install AWS CLI:



```
 inflating: aws/dist/awscli/data/cli.json
 inflating: aws/dist/awscli/data/metadata.json
 inflating: aws/dist/awscli/data/ac.index
  creating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/
 inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/top_level.txt
 inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/METADATA
 inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/RECORD
 inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/INSTALLER
 inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/WHEEL
 inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/LICENSE
 inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/AUTHORS.rs
 inflating: aws/dist/wheel-0.45.1.dist-info/INSTALLER
 inflating: aws/dist/wheel-0.45.1.dist-info/entry_points.txt
 inflating: aws/dist/wheel-0.45.1.dist-info/WHEEL
 inflating: aws/dist/wheel-0.45.1.dist-info/METADATA
 inflating: aws/dist/wheel-0.45.1.dist-info/RECORD
 inflating: aws/dist/wheel-0.45.1.dist-info/REQUESTED
 inflating: aws/dist/wheel-0.45.1.dist-info/direct_url.json
 inflating: aws/dist/wheel-0.45.1.dist-info/LICENSE.txt
reenaqureshi ➜ /workspaces/lab9 (main) $ sudo ./aws/install
ou can now run: /usr/local/bin/aws --version
reenaqureshi ➜ /workspaces/lab9 (main) $
```

Verify installation:



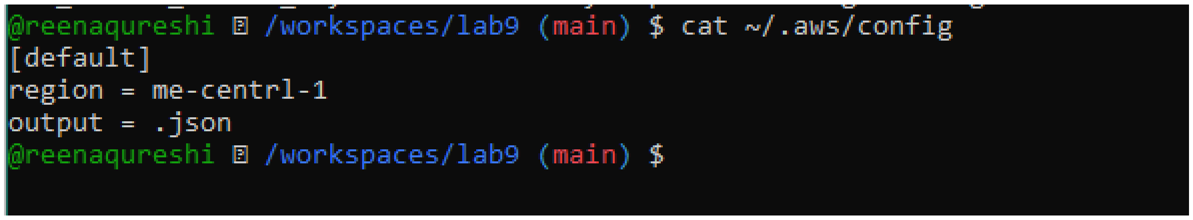
```
You can now run: /usr/local/bin/aws --version
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws --version
aws-cli/2.32.16 Python/3.13.11 Linux/6.8.0-1030-azure exe/x86_64.ubuntu.24
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

Configure the AWS CLI (you will provide Access Key ID and Secret Access Key for a user with permissions, or use root/IAM user you prepared for the lab):

Verify credentials/config files:

cat ~/.aws/credentials

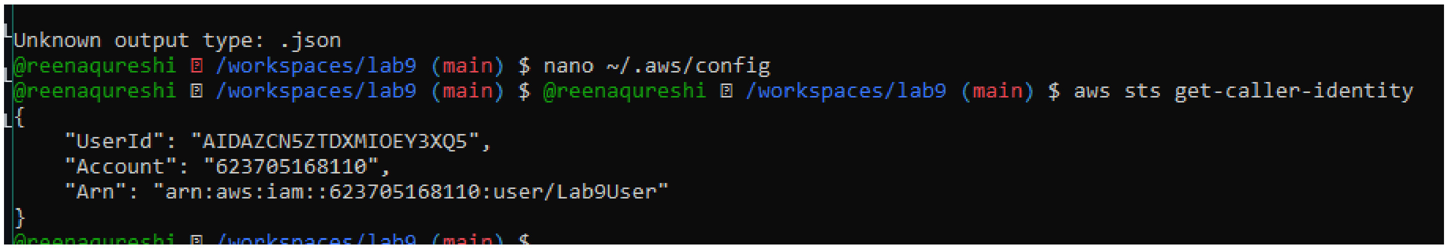
cat ~/.aws/config



```
@reenaqureshi ➜ /workspaces/lab9 (main) $ cat ~/.aws/config
[default]
region = me-centrl-1
output = .json
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

Verify connectivity (you should see a JSON result showing your caller identity):

aws sts get-caller-identity



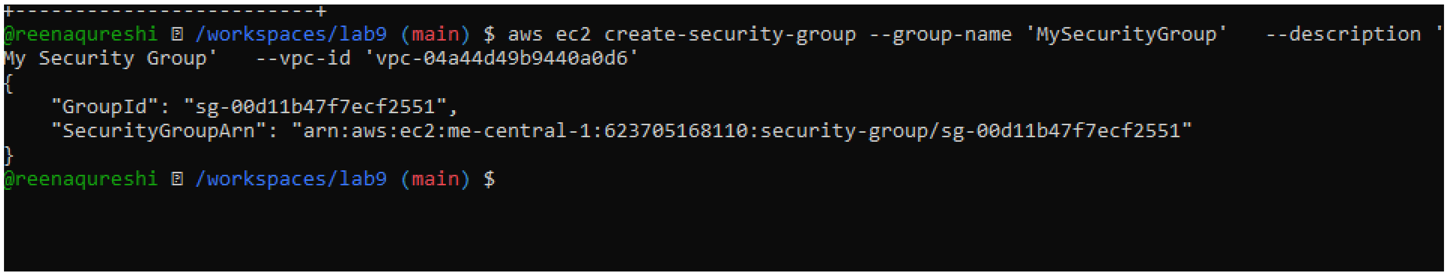
```
Unknown output type: .json
@reenaqureshi ➜ /workspaces/lab9 (main) $ nano ~/.aws/config
@reenaqureshi ➜ /workspaces/lab9 (main) $ @reenaqureshi ➜ /workspaces/lab9 (main) $ aws sts get-caller-identity
{
    "UserId": "AIDAZCN5ZTDXMIOEY3XQ5",
    "Account": "623705168110",
    "Arn": "arn:aws:iam::623705168110:user/Lab9User"
}
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

Task 3 — Create security group and add ingress rules using Codespace IP

Goal: Create an EC2 security group, inspect it, add an SSH rule for your Codespace public IP, and verify.

Steps (run in the Codespace shell):

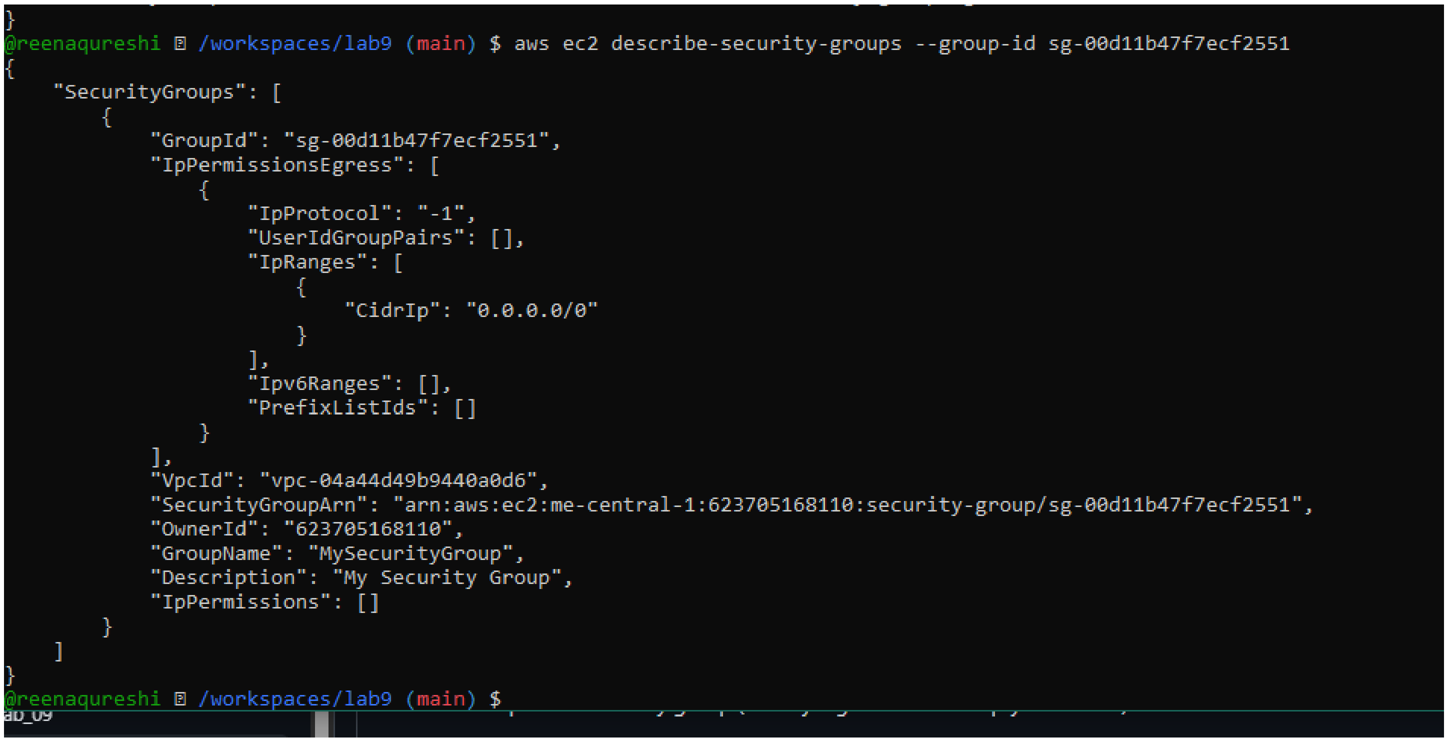
Create a security group (substitute your VPC id):



```
+-------------------------+
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 create-security-group --group-name 'MySecurityGroup'   --description '
My Security Group'   --vpc-id 'vpc-04a44d49b9440a0d6'
{
    "GroupId": "sg-00d11b47f7ecf2551",
    "SecurityGroupArn": "arn:aws:ec2:me-central-1:623705168110:security-group/sg-00d11b47f7ecf2551"
}
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

Inspect the security group (initially ingress will be empty or default):

aws ec2 describe-security-groups --group-ids sg-EXAMPLE1234567890



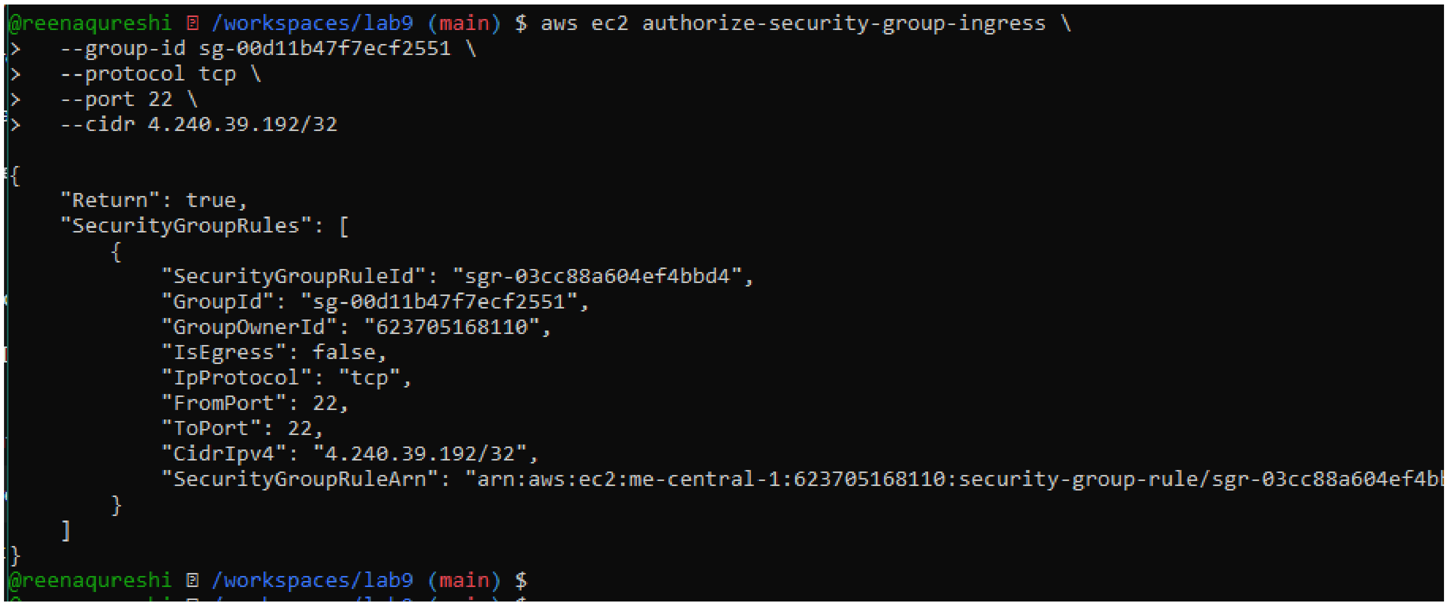
```
}
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-security-groups --group-id sg-00d11b47f7ecf2551
{
    "SecurityGroups": [
        {
            "GroupId": "sg-00d11b47f7ecf2551",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-04a44d49b9440a0d6",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:623705168110:security-group/sg-00d11b47f7ecf2551",
            "OwnerId": "623705168110",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
            "IpPermissions": []
        }
    ]
}
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

Get your Codespace public IP (from inside the Codespace):

curl icanhazip.com

Authorize SSH inbound on port 22 from your Codespace IP:

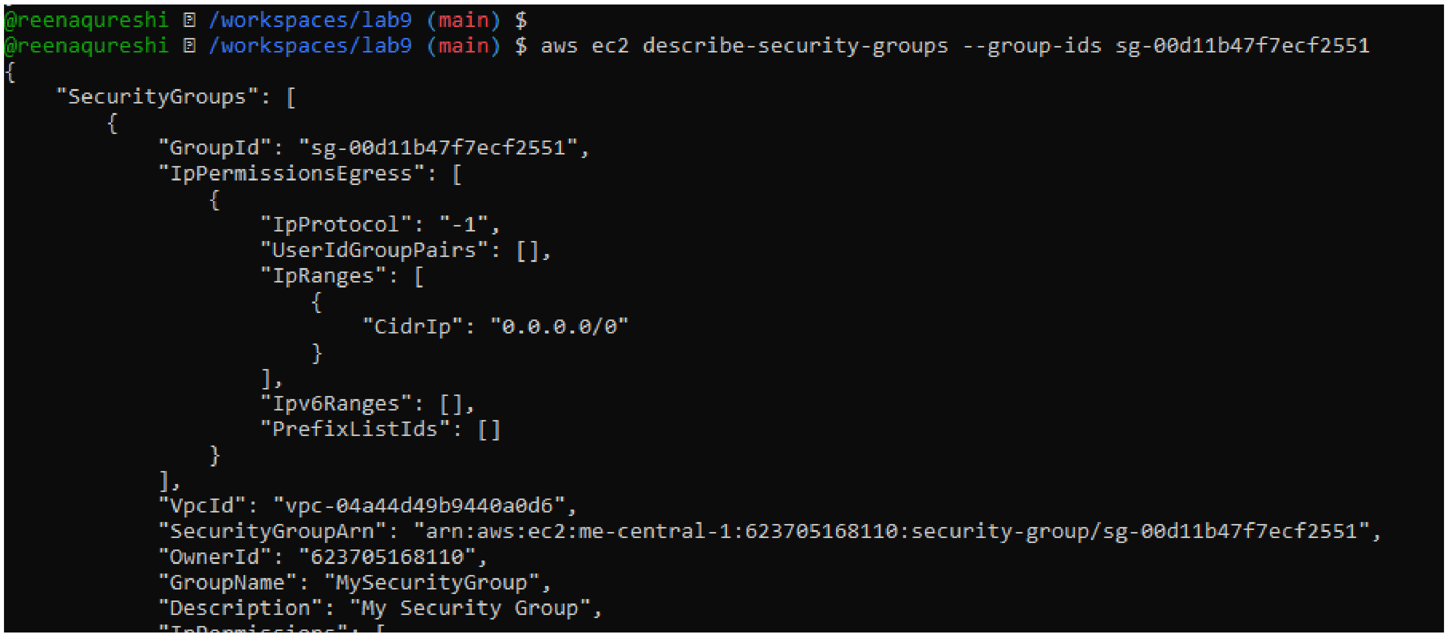
```
aws ec2 authorize-security-group-ingress \
  --group-id sg-EXAMPLE1234567890 \
  --protocol tcp \
  --port 22 \
  --cidr <XXX.XXX.XXX.XXX>/
```



```
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-00d11b47f7ecf2551 \
>   --protocol tcp \
>   --port 22 \
>   --cidr 4.240.39.192/32

{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-03cc88a604ef4bbd4",
            "GroupId": "sg-00d11b47f7ecf2551",
            "GroupOwnerId": "623705168110",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "4.240.39.192/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:me-central-1:623705168110:security-group-rule/sgr-03cc88a604ef4bb
        }
    ]
}
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

Verify ingress rule appears:

aws ec2 describe-security-groups --group-ids sg-EXAMPLE1234567890



```
@reenaqureshi ➜ /workspaces/lab9 (main) $
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-security-groups --group-ids sg-00d11b47f7ecf2551
{
    "SecurityGroups": [
        {
            "GroupId": "sg-00d11b47f7ecf2551",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-04a44d49b9440a0d6",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:623705168110:security-group/sg-00d11b47f7ecf2551",
            "OwnerId": "623705168110",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
```

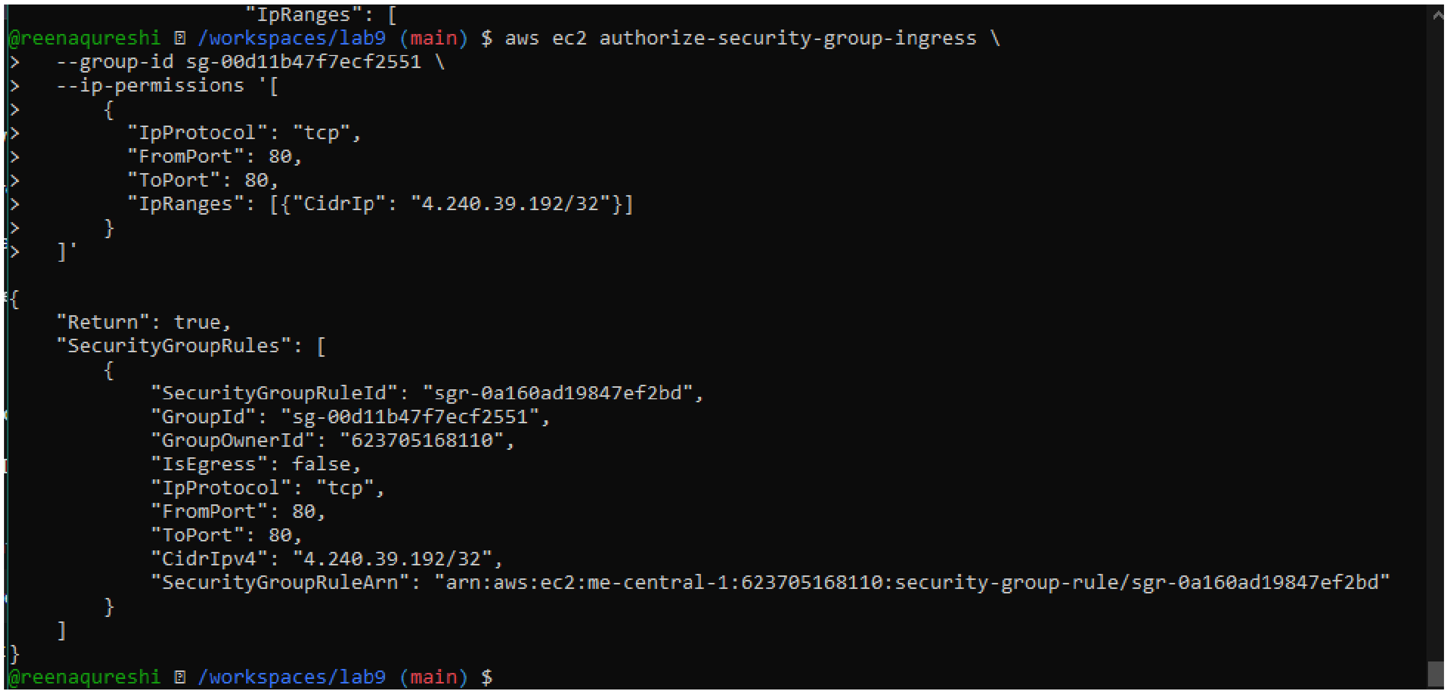
Add an HTTP rule (port 80) using ip-permissions JSON:

aws ec2 authorize-security-group-ingress \

  --group-id 'sg-EXAMPLE1234567890' \

  --ip-permissions '{"FromPort":80,"ToPort":80,"IpProtocol":"tcp","IpRanges":[{"CidrIp":"<XXX.XXX.XXX.XXX>/32"}]}'

Verify both ingress rules are present:



```
                    "IpRanges": [
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-00d11b47f7ecf2551 \
>   --ip-permissions '[
>       {
>         "IpProtocol": "tcp",
>         "FromPort": 80,
>         "ToPort": 80,
>         "IpRanges": [{"CidrIp": "4.240.39.192/32"}]
>       }
>   ]'

{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0a160ad19847ef2bd",
            "GroupId": "sg-00d11b47f7ecf2551",
            "GroupOwnerId": "623705168110",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "4.240.39.192/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:me-central-1:623705168110:security-group-rule/sgr-0a160ad19847ef2bd"
        }
    ]
}
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

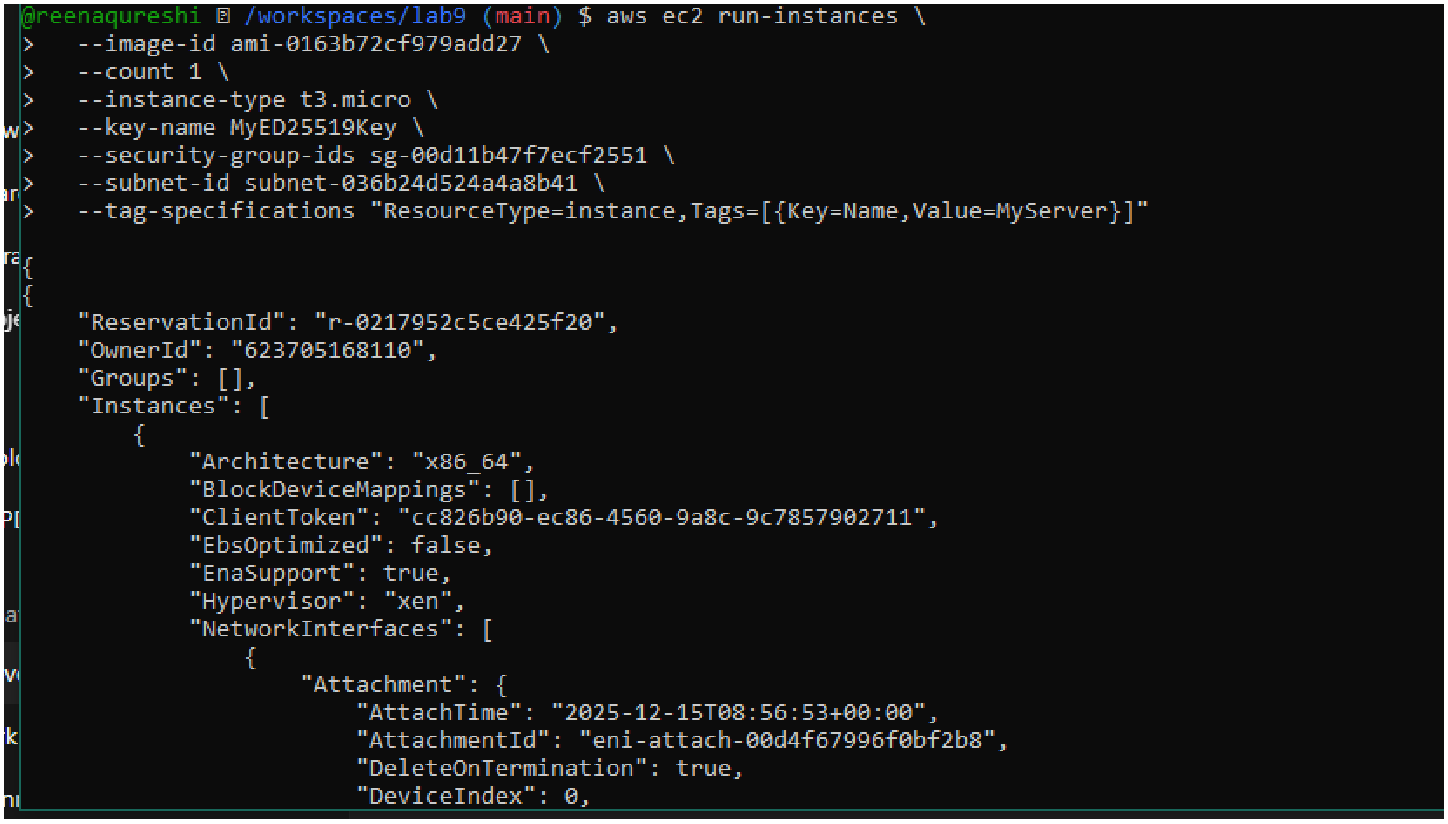
Task 4 — Create a key pair, describe key pairs, and launch EC2 instance

Goal: Create an ED25519 key pair, view it, and launch an EC2 instance using the key pair and the security group created earlier.

Steps:

Create the key pair and save the PEM file into the Codespace workspace:

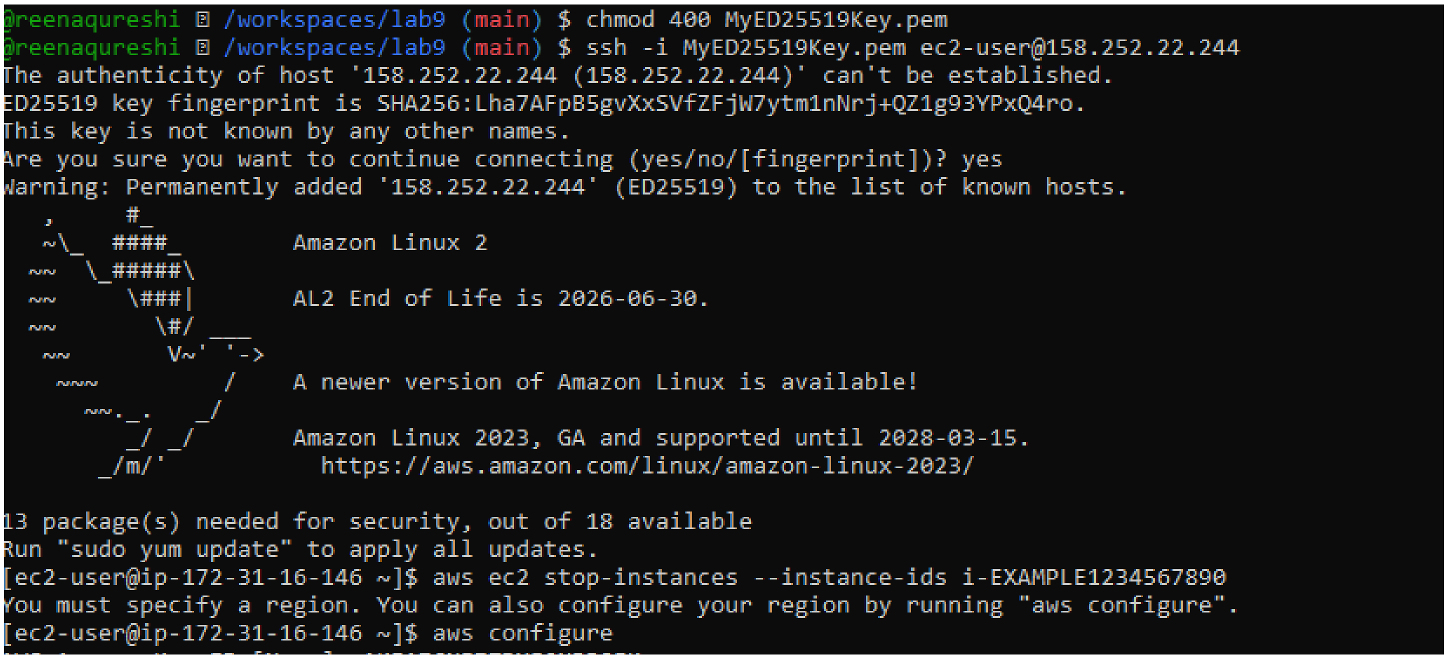
```
aws ec2 create-key-pair \
  --key-name MyED25519Key \
  --key-type ed25519 \
  --key-format pem \
  --query 'KeyMaterial' \
  --output text > MyED25519Key.pem
```



```
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 run-instances \
>   --image-id ami-0163b72cf979add27 \
>   --count 1 \
>   --instance-type t3.micro \
>   --key-name MyED25519Key \
>   --security-group-ids sg-00d11b47f7ecf2551 \
>   --subnet-id subnet-036b24d524a4a8b41 \
>   --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=MyServer}]"

{
{
    "ReservationId": "r-0217952c5ce425f20",
    "OwnerId": "623705168110",
    "Groups": [],
    "Instances": [
        {
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "cc826b90-ec86-4560-9a8c-9c7857902711",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2025-12-15T08:56:53+00:00",
                        "AttachmentId": "eni-attach-00d4f67996f0bf2b8",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
```

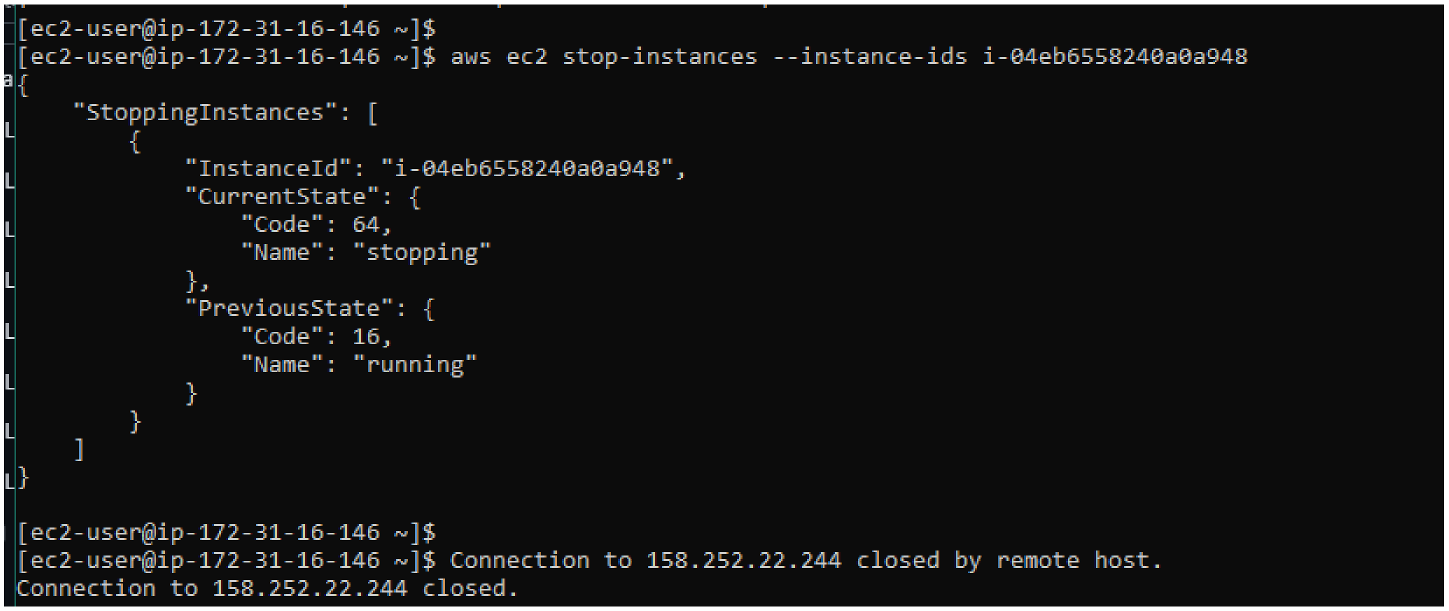
Attempt SSH into the instance from the Codespace or from a machine whose IP is allowed in the security group:

ssh -i MyED25519Key.pem ec2-user@<public-ip-address>

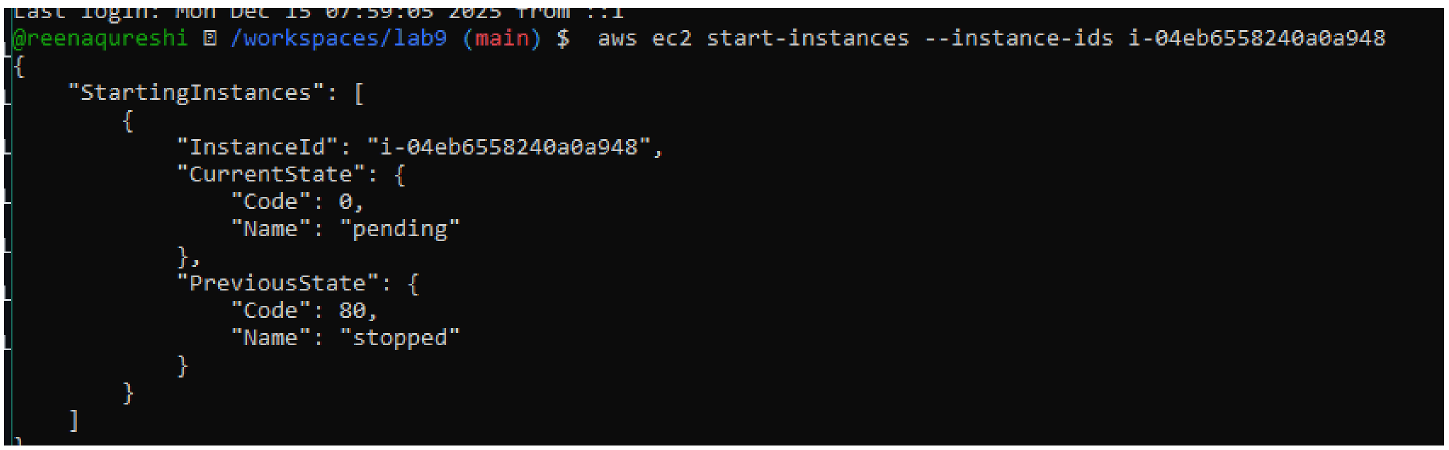


```
@reenaqureshi ➜ /workspaces/lab9 (main) $ chmod 400 MyED25519Key.pem
@reenaqureshi ➜ /workspaces/lab9 (main) $ ssh -i MyED25519Key.pem ec2-user@158.252.22.244
The authenticity of host '158.252.22.244 (158.252.22.244)' can't be established.
ED25519 key fingerprint is SHA256:Lha7AFpB5gvXxSVfZFjW7ytm1nNrj+QZ1g93YPxQ4ro.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '158.252.22.244' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

13 package(s) needed for security, out of 18 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-16-146 ~]$ aws ec2 stop-instances --instance-ids i-EXAMPLE1234567890
You must specify a region. You can also configure your region by running "aws configure".
[ec2-user@ip-172-31-16-146 ~]$ aws configure
```

aws ec2 stop-instances --instance-ids i-EXAMPLE1234567890

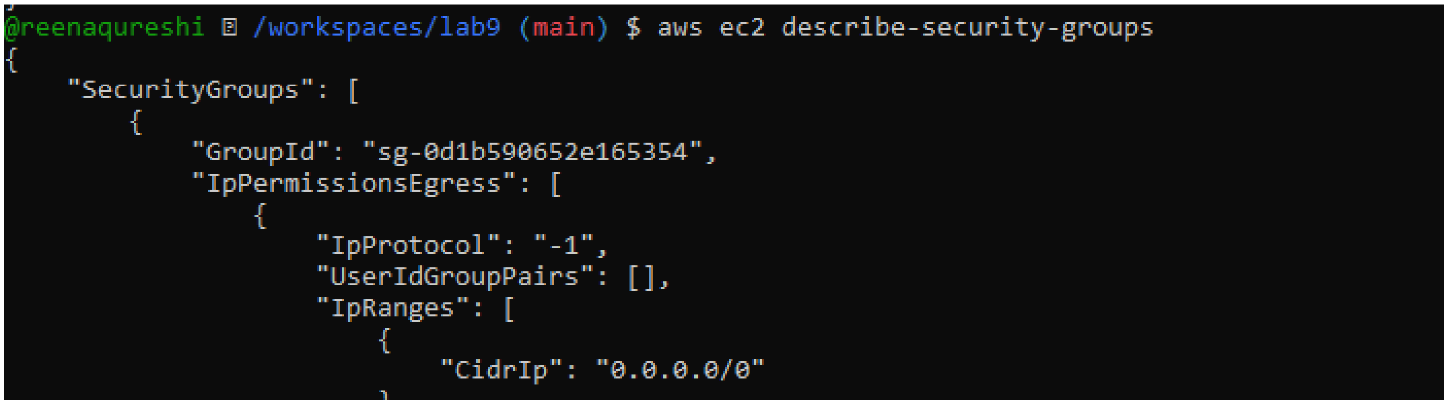

```
[ec2-user@ip-172-31-16-146 ~]$
[ec2-user@ip-172-31-16-146 ~]$ aws ec2 stop-instances --instance-ids i-04eb6558240a0a948
{
    "StoppingInstances": [
        {
            "InstanceId": "i-04eb6558240a0a948",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}

[ec2-user@ip-172-31-16-146 ~]$
[ec2-user@ip-172-31-16-146 ~]$ Connection to 158.252.22.244 closed by remote host.
Connection to 158.252.22.244 closed.
```

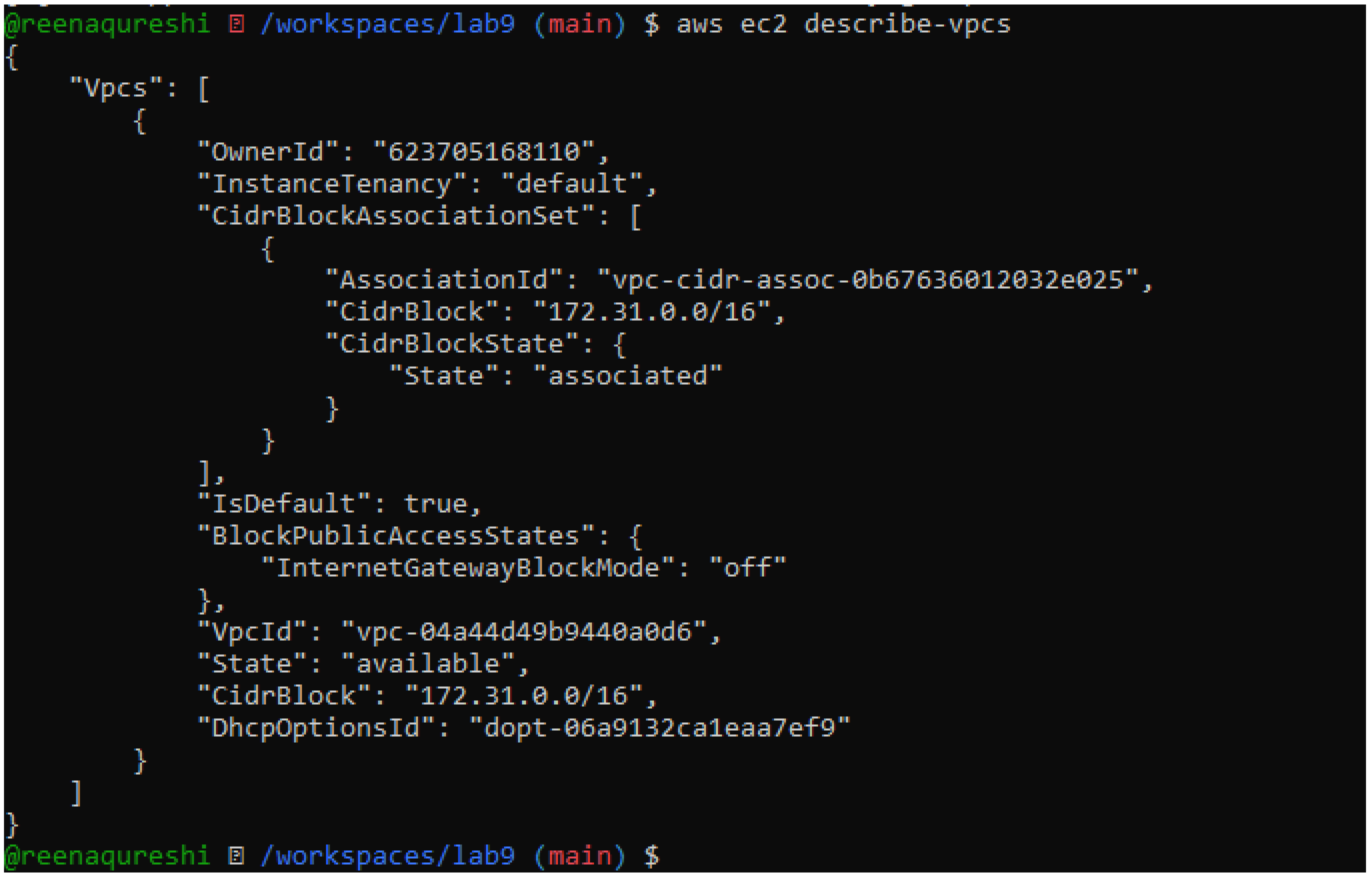
aws ec2 start-instances --instance-ids i-EXAMPLE1234567890


```
@reenaqureshi ➜ /workspaces/lab9 (main) $  aws ec2 start-instances --instance-ids i-04eb6558240a0a948
{
    "StartingInstances": [
        {
            "InstanceId": "i-04eb6558240a0a948",
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
```

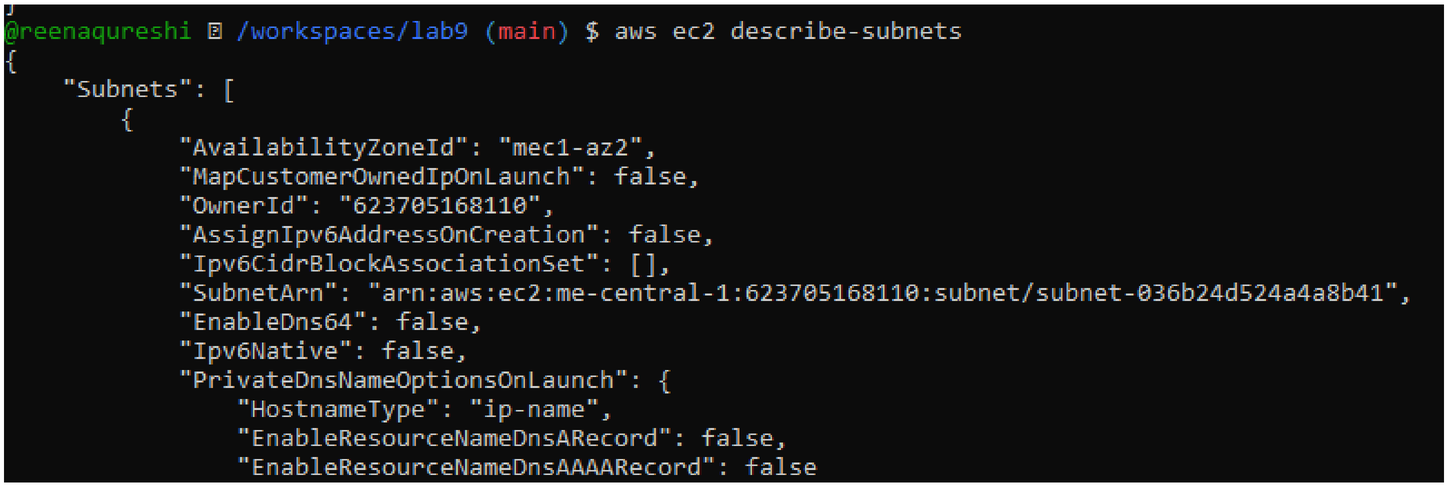
Task 5 — Understand AWS describe-* commands

Run and understand these commands (run each, then capture screenshot immediately after):

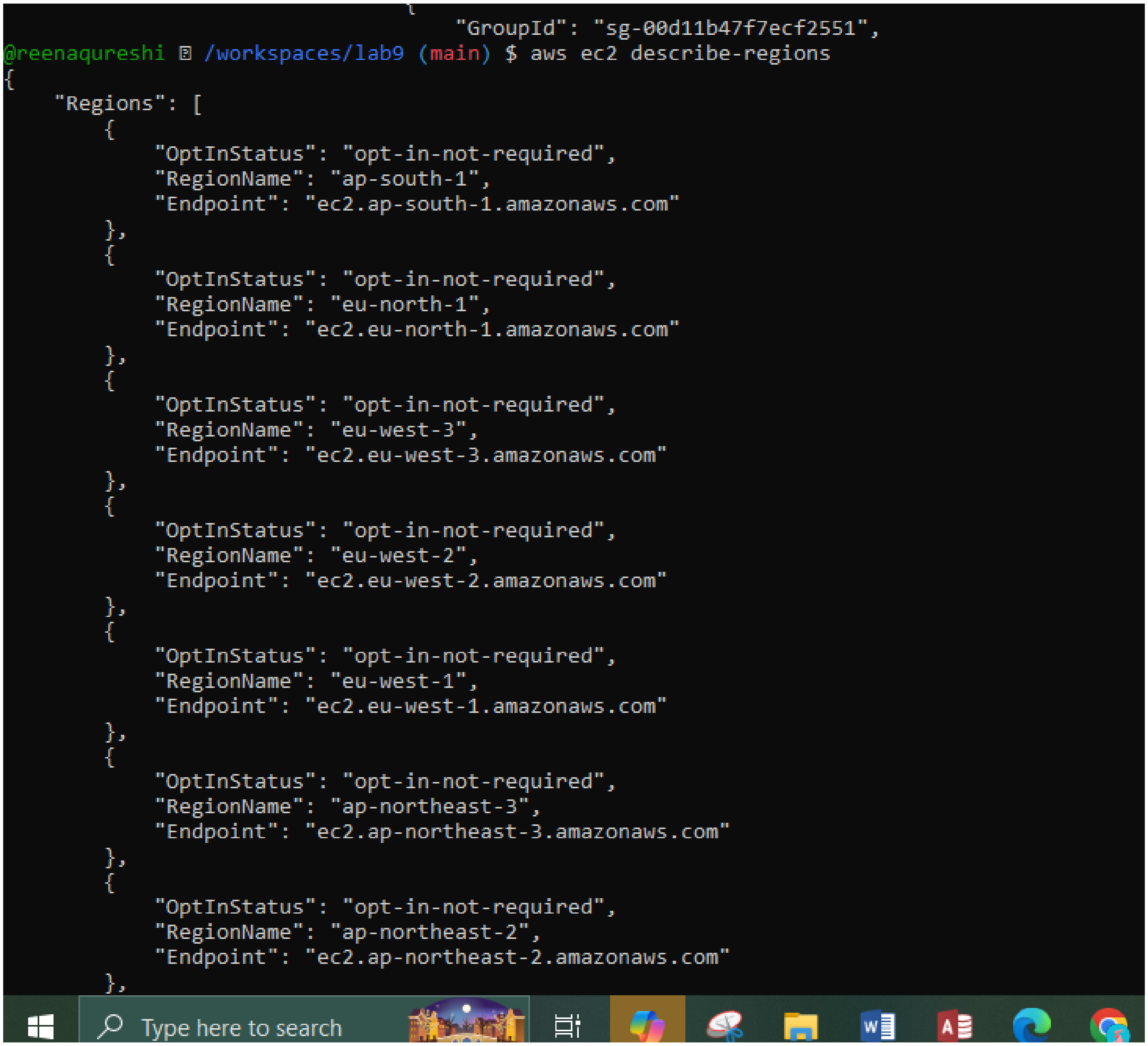

```
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-security-groups
{
    "SecurityGroups": [
        {
            "GroupId": "sg-0d1b590652e165354",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
```



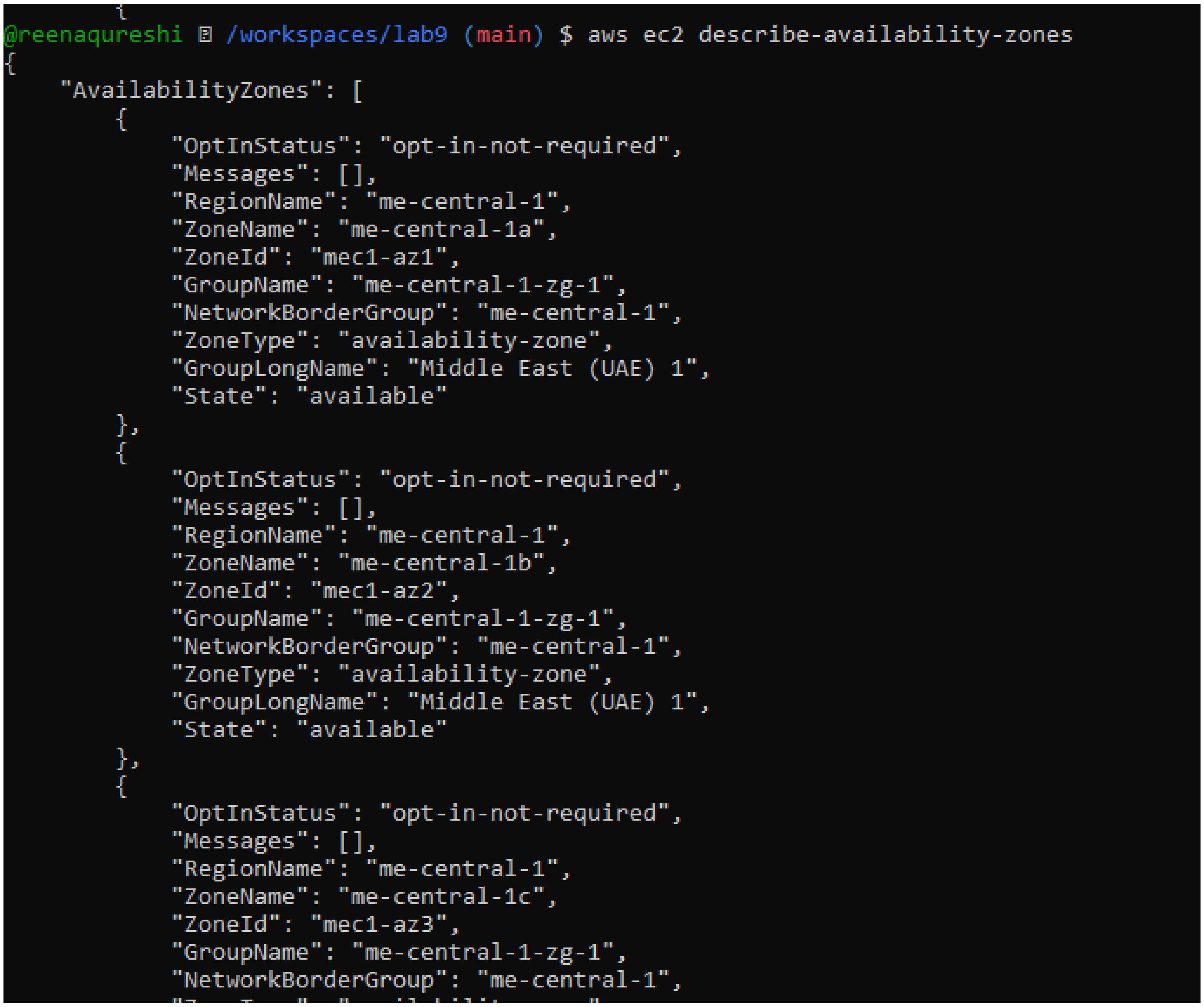
```
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-vpcs
{
    "Vpcs": [
        {
            "OwnerId": "623705168110",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-0b67636012032e025",
                    "CidrBlock": "172.31.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": true,
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "VpcId": "vpc-04a44d49b9440a0d6",
            "State": "available",
            "CidrBlock": "172.31.0.0/16",
            "DhcpOptionsId": "dopt-06a9132ca1eaa7ef9"
        }
    ]
}
@reenaqureshi ➜ /workspaces/lab9 (main) $
```



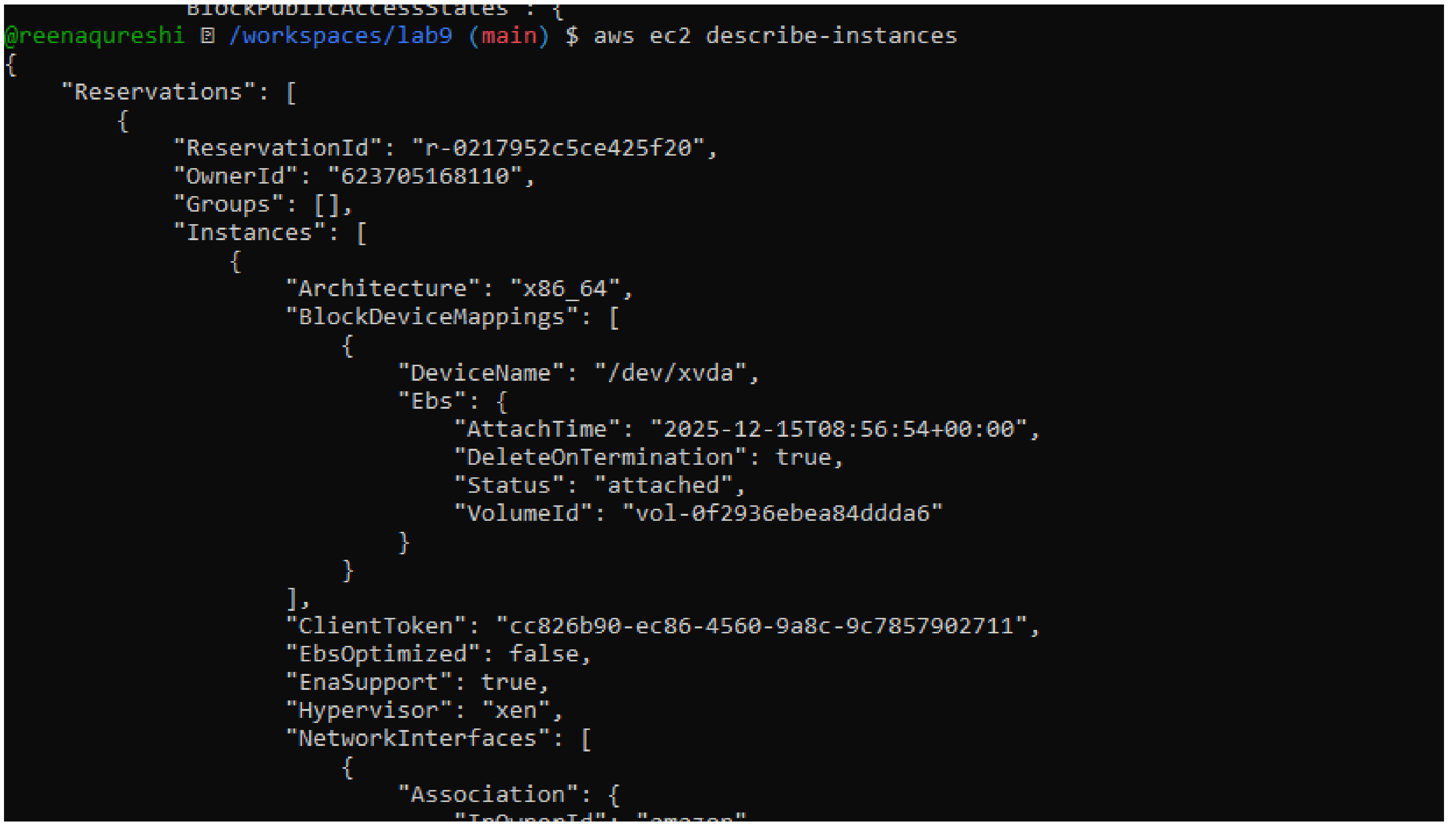
```
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-subnets
{
    "Subnets": [
        {
            "AvailabilityZoneId": "mec1-az2",
            "MapCustomerOwnedIpOnLaunch": false,
            "OwnerId": "623705168110",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn": "arn:aws:ec2:me-central-1:623705168110:subnet/subnet-036b24d524a4a8b41",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
```



```
                                    "GroupId": "sg-00d11b47f7ecf2551",
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-regions
{
    "Regions": [
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-south-1",
            "Endpoint": "ec2.ap-south-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-north-1",
            "Endpoint": "ec2.eu-north-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-west-3",
            "Endpoint": "ec2.eu-west-3.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-west-2",
            "Endpoint": "ec2.eu-west-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-west-1",
            "Endpoint": "ec2.eu-west-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-northeast-3",
            "Endpoint": "ec2.ap-northeast-3.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-northeast-2",
            "Endpoint": "ec2.ap-northeast-2.amazonaws.com"
        },
```

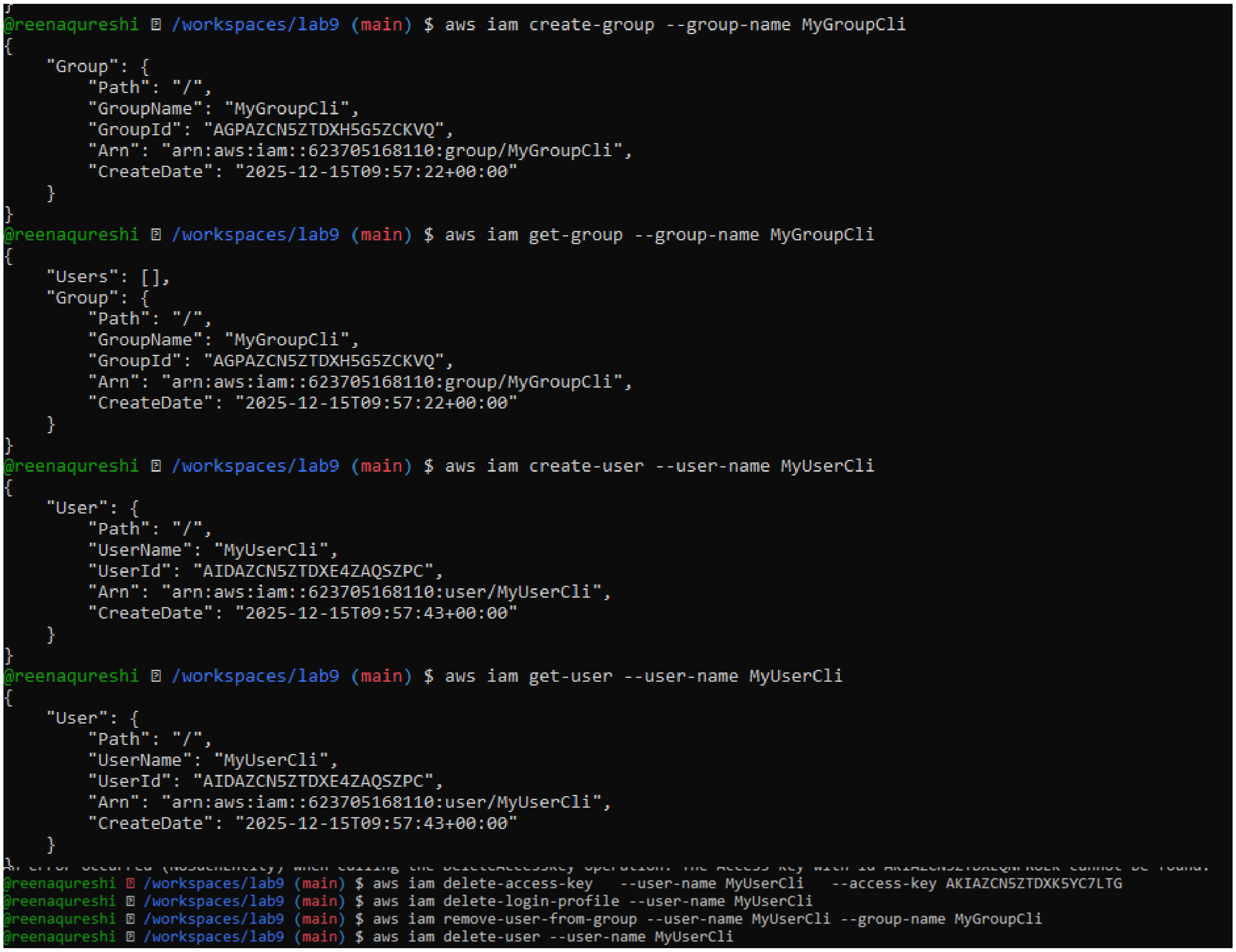


```
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-availability-zones
{
    "AvailabilityZones": [
        {
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "me-central-1",
            "ZoneName": "me-central-1a",
            "ZoneId": "mec1-az1",
            "GroupName": "me-central-1-zg-1",
            "NetworkBorderGroup": "me-central-1",
            "ZoneType": "availability-zone",
            "GroupLongName": "Middle East (UAE) 1",
            "State": "available"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "me-central-1",
            "ZoneName": "me-central-1b",
            "ZoneId": "mec1-az2",
            "GroupName": "me-central-1-zg-1",
            "NetworkBorderGroup": "me-central-1",
            "ZoneType": "availability-zone",
            "GroupLongName": "Middle East (UAE) 1",
            "State": "available"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "me-central-1",
            "ZoneName": "me-central-1c",
            "ZoneId": "mec1-az3",
            "GroupName": "me-central-1-zg-1",
            "NetworkBorderGroup": "me-central-1",
```

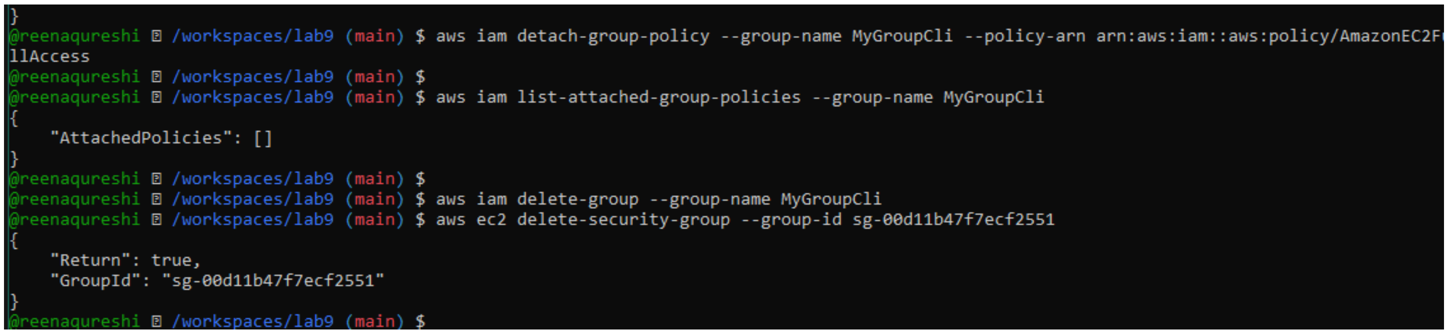


```
            "BlockPublicAccessStates": {
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws ec2 describe-instances
{
    "Reservations": [
        {
            "ReservationId": "r-0217952c5ce425f20",
            "OwnerId": "623705168110",
            "Groups": [],
            "Instances": [
                {
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/xvda",
                            "Ebs": {
                                "AttachTime": "2025-12-15T08:56:54+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-0f2936ebea84ddda6"
                            }
                        }
                    ],
                    "ClientToken": "cc826b90-ec86-4560-9a8c-9c7857902711",
                    "EbsOptimized": false,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "NetworkInterfaces": [
                        {
                            "Association": {
```

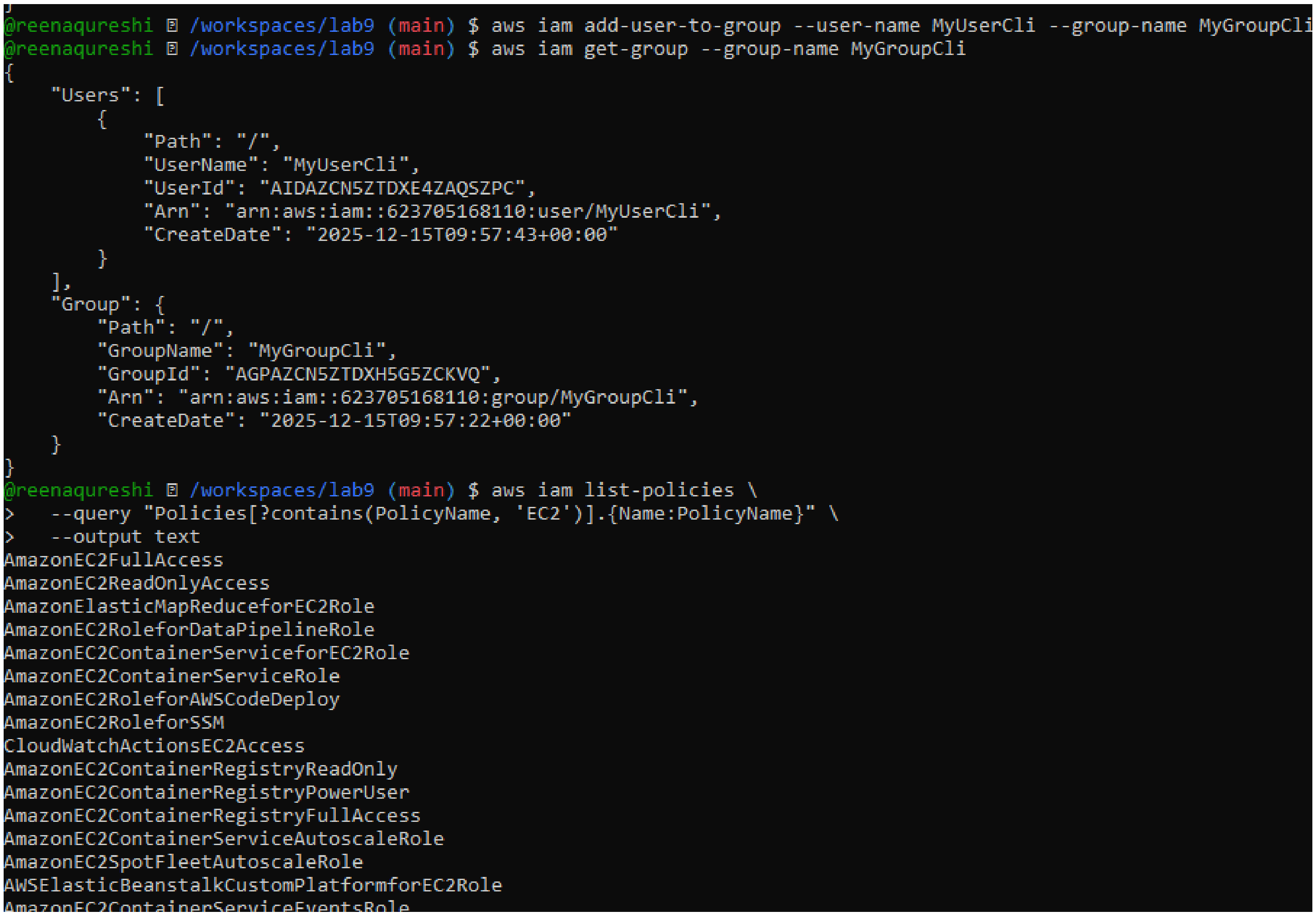
Task 6 — IAM: create group, user, attach policies, create console login & keys



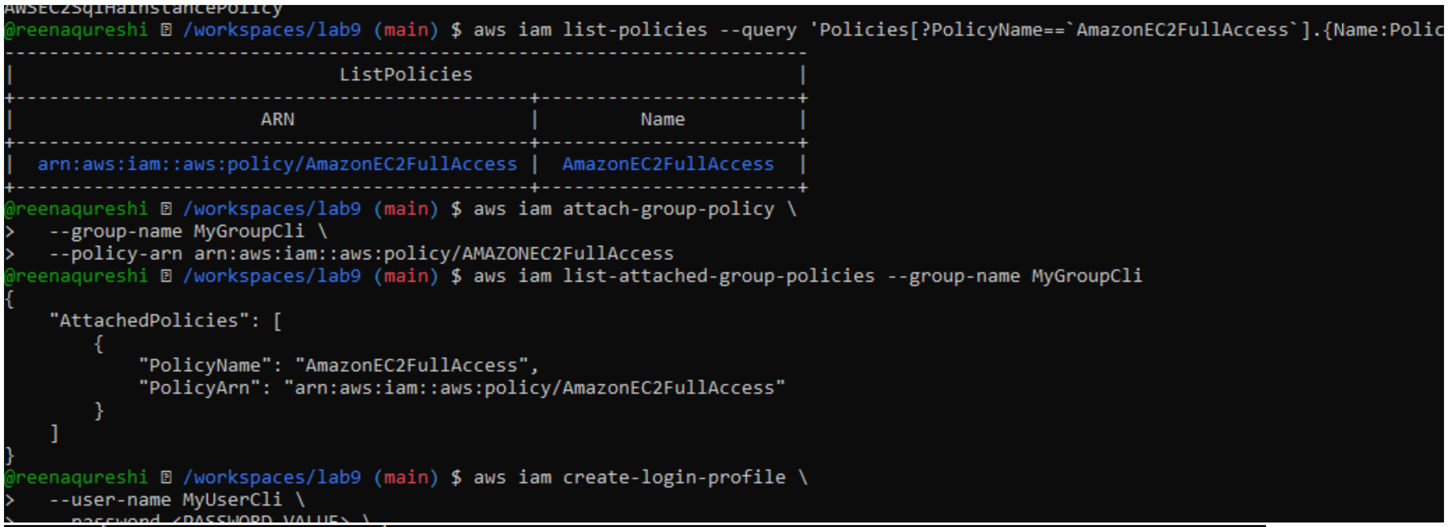
```
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam create-group --group-name MyGroupCli
{
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPAZCN5ZTDXH5G5ZCKVQ",
        "Arn": "arn:aws:iam::623705168110:group/MyGroupCli",
        "CreateDate": "2025-12-15T09:57:22+00:00"
    }
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam get-group --group-name MyGroupCli
{
    "Users": [],
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPAZCN5ZTDXH5G5ZCKVQ",
        "Arn": "arn:aws:iam::623705168110:group/MyGroupCli",
        "CreateDate": "2025-12-15T09:57:22+00:00"
    }
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam create-user --user-name MyUserCli
{
    "User": {
        "Path": "/",
        "UserName": "MyUserCli",
        "UserId": "AIDAZCN5ZTDXE4ZAQSZPC",
        "Arn": "arn:aws:iam::623705168110:user/MyUserCli",
        "CreateDate": "2025-12-15T09:57:43+00:00"
    }
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam get-user --user-name MyUserCli
{
    "User": {
        "Path": "/",
        "UserName": "MyUserCli",
        "UserId": "AIDAZCN5ZTDXE4ZAQSZPC",
        "Arn": "arn:aws:iam::623705168110:user/MyUserCli",
        "CreateDate": "2025-12-15T09:57:43+00:00"
    }
}
An error occurred (NoSuchEntity) when calling the DeleteAccessKey operation: The Access key with id [illegible] cannot be found.
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam delete-access-key   --user-name MyUserCli   --access-key AKIAZCN5ZTDXK5YC7LTG
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam delete-login-profile --user-name MyUserCli
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam remove-user-from-group --user-name MyUserCli --group-name MyGroupCli
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam delete-user --user-name MyUserCli
```



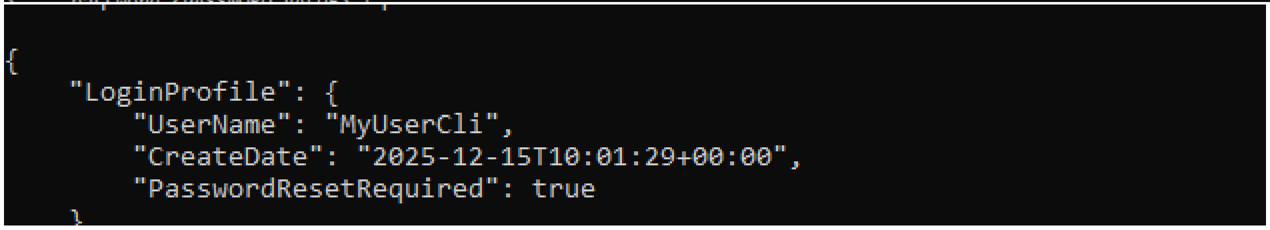
```
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam detach-group-policy --group-name MyGroupCli --policy-arn arn:aws:iam::aws:policy/AmazonEC2F
llAccess
@reenaqureshi ▯ /workspaces/lab9 (main) $
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam list-attached-group-policies --group-name MyGroupCli
{
    "AttachedPolicies": []
}
@reenaqureshi ▯ /workspaces/lab9 (main) $
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam delete-group --group-name MyGroupCli
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws ec2 delete-security-group --group-id sg-00d11b47f7ecf2551
{
    "Return": true,
    "GroupId": "sg-00d11b47f7ecf2551"
}
@reenaqureshi ▯ /workspaces/lab9 (main) $
```



```
]
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam add-user-to-group --user-name MyUserCli --group-name MyGroupCli
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam get-group --group-name MyGroupCli
{
    "Users": [
        {
            "Path": "/",
            "UserName": "MyUserCli",
            "UserId": "AIDAZCN5ZTDXE4ZAQSZPC",
            "Arn": "arn:aws:iam::623705168110:user/MyUserCli",
            "CreateDate": "2025-12-15T09:57:43+00:00"
        }
    ],
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPAZCN5ZTDXH5G5ZCKVQ",
        "Arn": "arn:aws:iam::623705168110:group/MyGroupCli",
        "CreateDate": "2025-12-15T09:57:22+00:00"
    }
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam list-policies \
>   --query "Policies[?contains(PolicyName, 'EC2')].{Name:PolicyName}" \
>   --output text
AmazonEC2FullAccess
AmazonEC2ReadOnlyAccess
AmazonElasticMapReduceforEC2Role
AmazonEC2RoleforDataPipelineRole
AmazonEC2ContainerServiceforEC2Role
AmazonEC2ContainerServiceRole
AmazonEC2RoleforAWSCodeDeploy
AmazonEC2RoleforSSM
CloudWatchActionsEC2Access
AmazonEC2ContainerRegistryReadOnly
AmazonEC2ContainerRegistryPowerUser
AmazonEC2ContainerRegistryFullAccess
AmazonEC2ContainerServiceAutoscaleRole
AmazonEC2SpotFleetAutoscaleRole
AWSElasticBeanstalkCustomPlatformforEC2Role
AmazonEC2ContainerServiceEventsRole
```



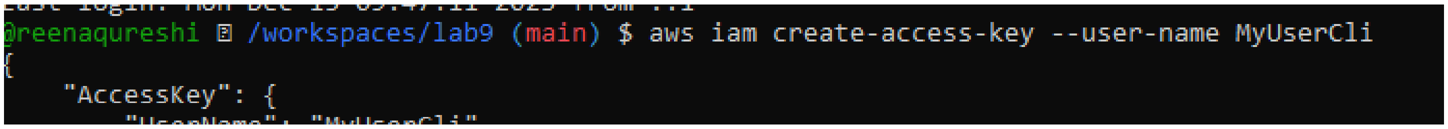
```
AWSEC2SqlHaInstancePolicy
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam list-policies --query 'Policies[?PolicyName==`AmazonEC2FullAccess`].{Name:Polic
-----------------------------------------------------------------------
|                            ListPolicies                             |
+---------------------------------------------+-----------------------+
|                    ARN                      |         Name          |
+---------------------------------------------+-----------------------+
|  arn:aws:iam::aws:policy/AmazonEC2FullAccess |  AmazonEC2FullAccess  |
+---------------------------------------------+-----------------------+
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam attach-group-policy \
>   --group-name MyGroupCli \
>   --policy-arn arn:aws:iam::aws:policy/AMAZONEC2FullAccess
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam list-attached-group-policies --group-name MyGroupCli
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEC2FullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
        }
    ]
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam create-login-profile \
>   --user-name MyUserCli \
>   --password <PASSWORD_VALUE> \
```



```
{
    "LoginProfile": {
        "UserName": "MyUserCli",
        "CreateDate": "2025-12-15T10:01:29+00:00",
        "PasswordResetRequired": true
    }
```

Create access keys for the user (save AccessKeyId and SecretAccessKey securely):

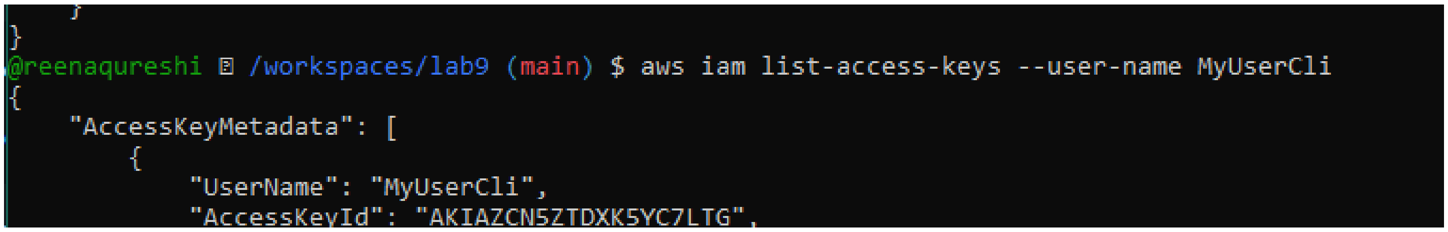
aws iam create-access-key --user-name MyUserCli



```
Last login: Mon Dec 15 09:47:11 2025 from ::1
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam create-access-key --user-name MyUserCli
{
    "AccessKey": {
        "UserName": "MyUserCli",
```

List access keys:

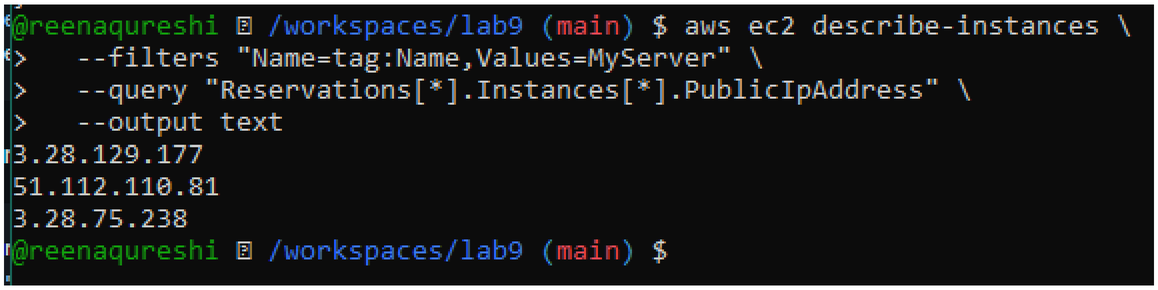
aws iam list-access-keys --user-name MyUserCli



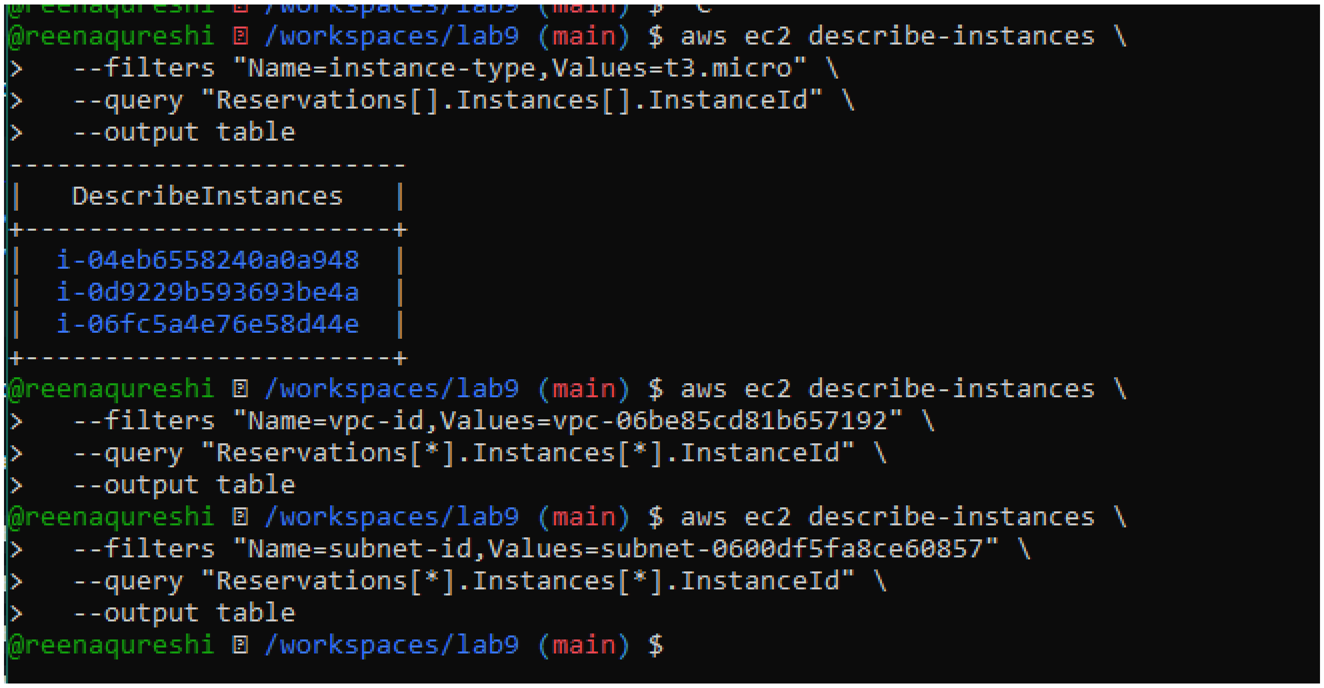
```
    }
}
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws iam list-access-keys --user-name MyUserCli
{
    "AccessKeyMetadata": [
        {
            "UserName": "MyUserCli",
            "AccessKeyId": "AKIAZCN5ZTDXK5YC7LTG",
```

Task 7 — Filters: query with filters to find instances and their attributes

Examples (run each and take a screenshot immediately after)



```
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=tag:Name,Values=MyServer" \
>   --query "Reservations[*].Instances[*].PublicIpAddress" \
>   --output text
3.28.129.177
51.112.110.81
3.28.75.238
@reenaqureshi ▯ /workspaces/lab9 (main) $
```

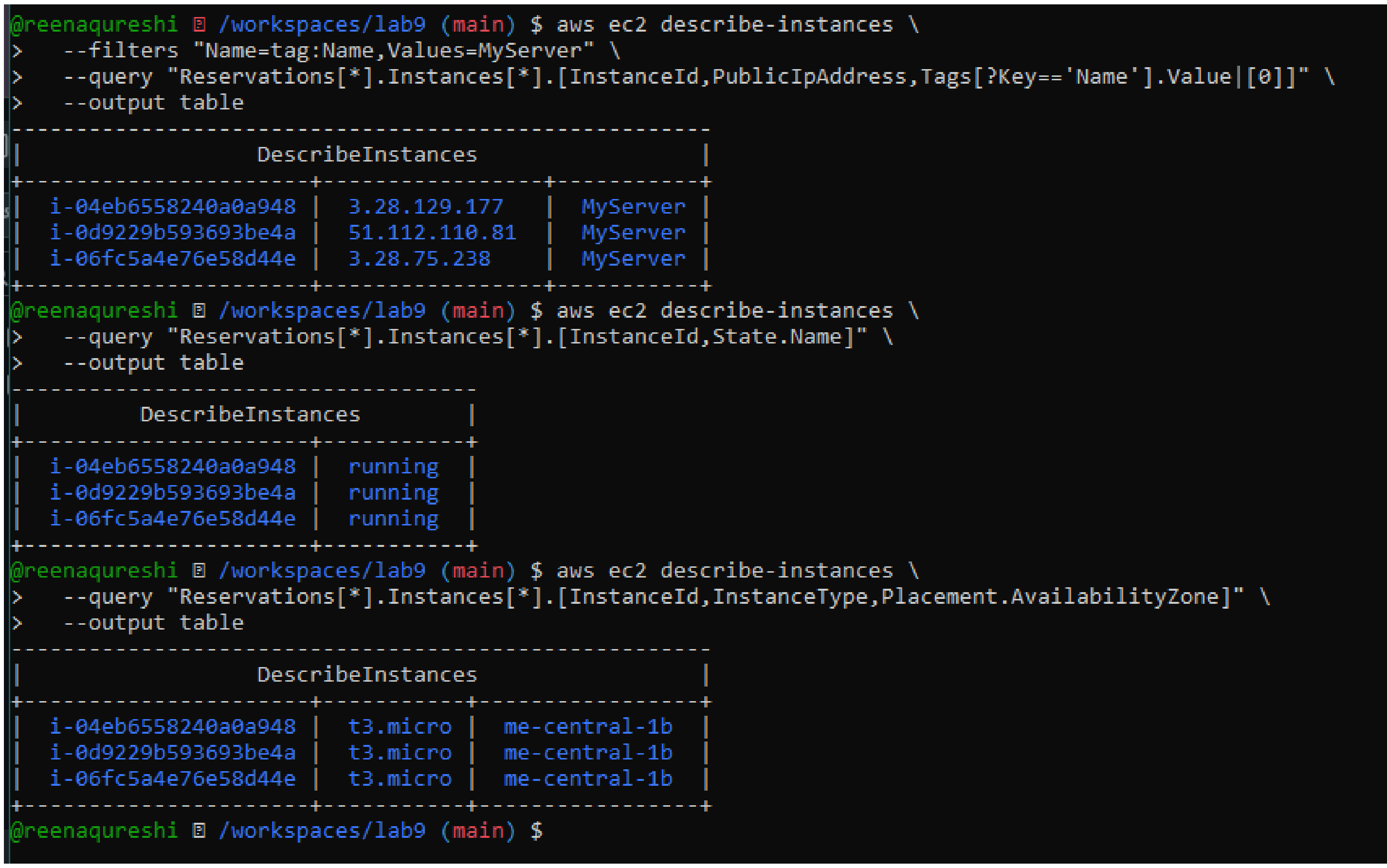


```
@reenaqureshi ▯ /workspaces/lab9 (main) $ ^C
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=instance-type,Values=t3.micro" \
>   --query "Reservations[].Instances[].InstanceId" \
>   --output table
------------------------
|   DescribeInstances   |
+----------------------+
|  i-04eb6558240a0a948  |
|  i-0d9229b593693be4a  |
|  i-06fc5a4e76e58d44e  |
+----------------------+
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=vpc-id,Values=vpc-06be85cd81b657192" \
>   --query "Reservations[*].Instances[*].InstanceId" \
>   --output table
@reenaqureshi ▯ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=subnet-id,Values=subnet-0600df5fa8ce60857" \
>   --query "Reservations[*].Instances[*].InstanceId" \
>   --output table
@reenaqureshi ▯ /workspaces/lab9 (main) $
```

Task 8 — Use --query to format outputs for reporting

Goal: Extract useful fields in table format for reporting.

Examples (run each and take a screenshot immediately after)



```
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=tag:Name,Values=MyServer" \
>   --query "Reservations[*].Instances[*].[InstanceId,PublicIpAddress,Tags[?Key=='Name'].Value|[0]]" \
>   --output table
------------------------------------------------------
|                  DescribeInstances                 |
+----------------------+-----------------+-----------+
|  i-04eb6558240a0a948 |  3.28.129.177   |  MyServer |
|  i-0d9229b593693be4a |  51.112.110.81  |  MyServer |
|  i-06fc5a4e76e58d44e |  3.28.75.238    |  MyServer |
+----------------------+-----------------+-----------+
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --query "Reservations[*].Instances[*].[InstanceId,State.Name]" \
>   --output table
------------------------------------
|         DescribeInstances        |
+----------------------+-----------+
|  i-04eb6558240a0a948 |  running  |
|  i-0d9229b593693be4a |  running  |
|  i-06fc5a4e76e58d44e |  running  |
+----------------------+-----------+
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --query "Reservations[*].Instances[*].[InstanceId,InstanceType,Placement.AvailabilityZone]" \
>   --output table
------------------------------------------------------
|                  DescribeInstances                 |
+----------------------+----------+------------------+
|  i-04eb6558240a0a948 |  t3.micro |  me-central-1b  |
|  i-0d9229b593693be4a |  t3.micro |  me-central-1b  |
|  i-06fc5a4e76e58d44e |  t3.micro |  me-central-1b  |
+----------------------+----------+------------------+
@reenaqureshi ➜ /workspaces/lab9 (main) $
```

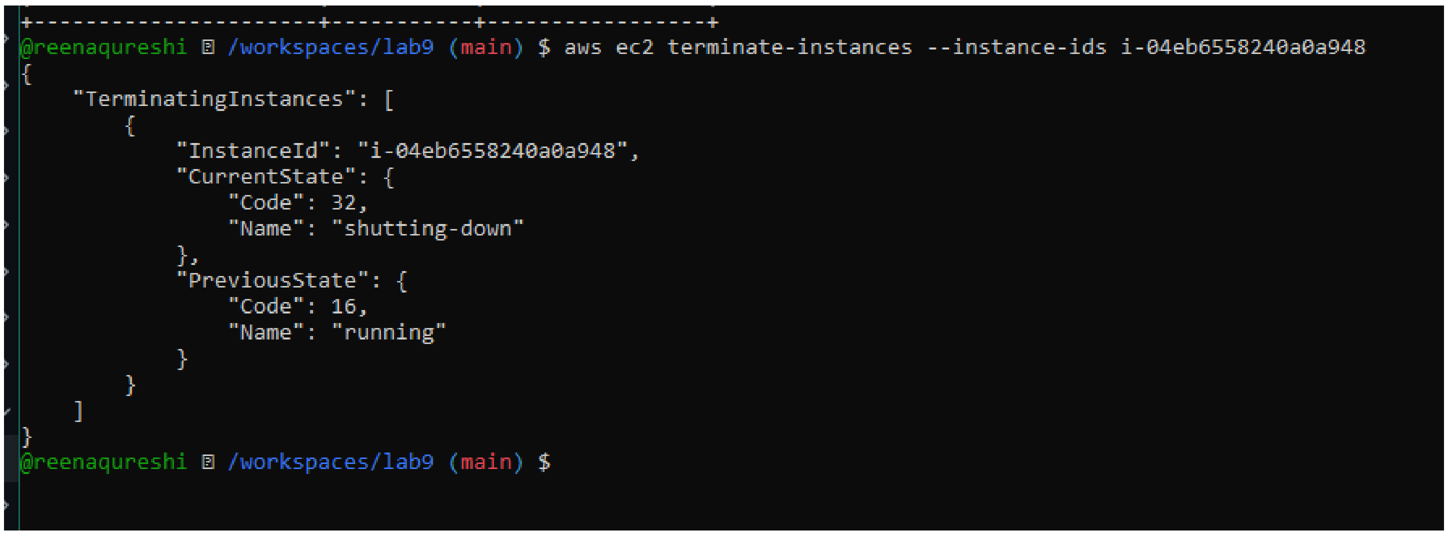
Cleanup — Remove resources to avoid charges

After verification, terminate and delete everything you created in AWS. Capture screenshots for each step.

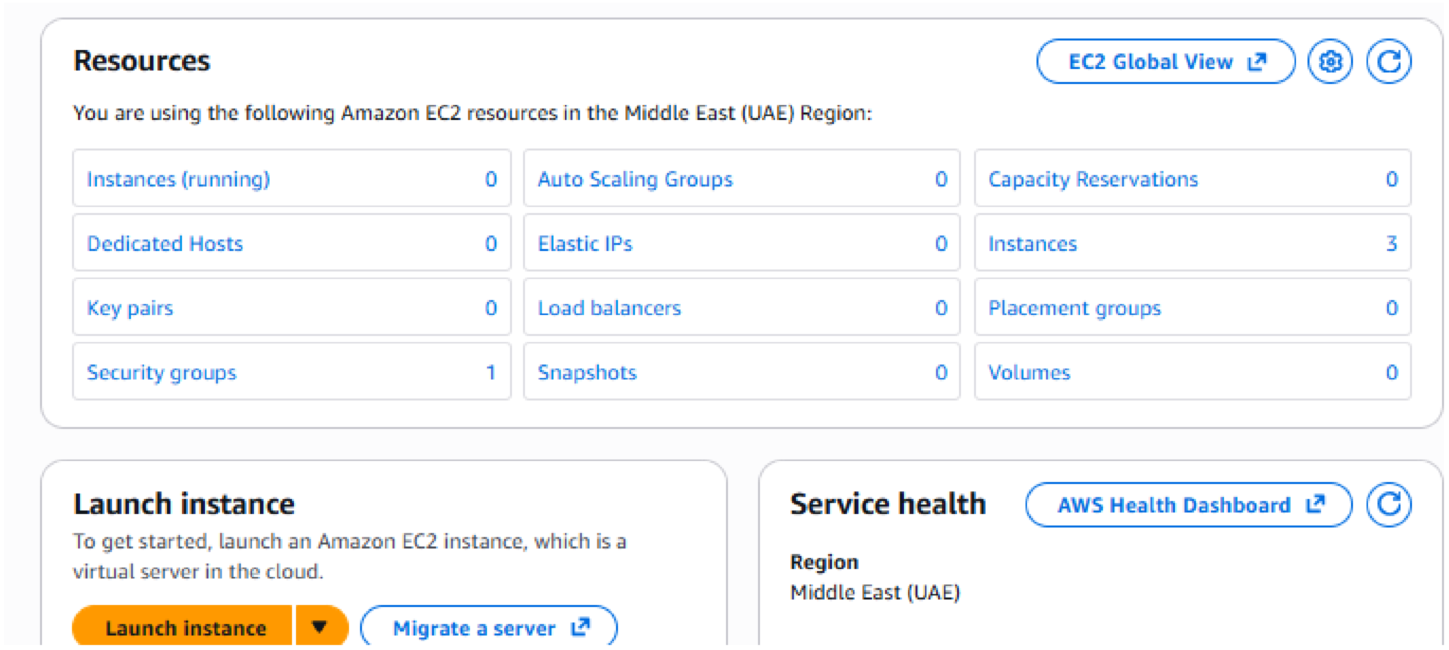
Cleanup steps and required screenshots (take each screenshot immediately after running the command):

Terminate instances:

aws ec2 terminate-instances --instance-ids i-EXAMPLE1234567890



```
+---------------------+----------+----------------+
@reenaqureshi ➜ /workspaces/lab9 (main) $ aws ec2 terminate-instances --instance-ids i-04eb6558240a0a948
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-04eb6558240a0a948",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
@reenaqureshi ➜ /workspaces/lab9 (main) $
```



Resources

You are using the following Amazon EC2 resources in the Middle East (UAE) Region:

Instances (running)	0	Auto Scaling Groups	0	Capacity Reservations	0
Dedicated Hosts	0	Elastic IPs	0	Instances	3
Key pairs	0	Load balancers	0	Placement groups	0
Security groups	1	Snapshots	0	Volumes	0

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Service health

Region
Middle East (UAE)