Fatima Jinnah
Women University

Lab 9

Lab title: Codespaces + AWS: GH CLI (Codespaces), AWS CLI, EC2, IAM, Security Groups, Filters & Queries

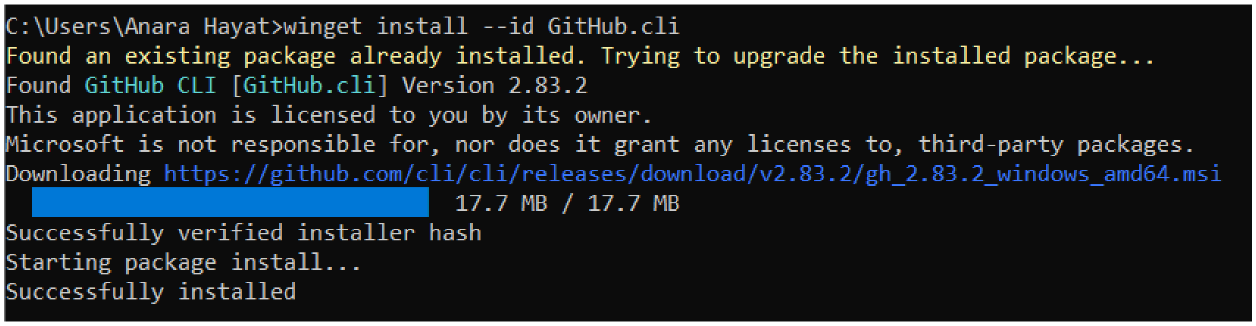
Submitted to: Engr. Mohammad Shoaib

Submitted by: Anara Hayat

Reg#No: 2023-BSE-008

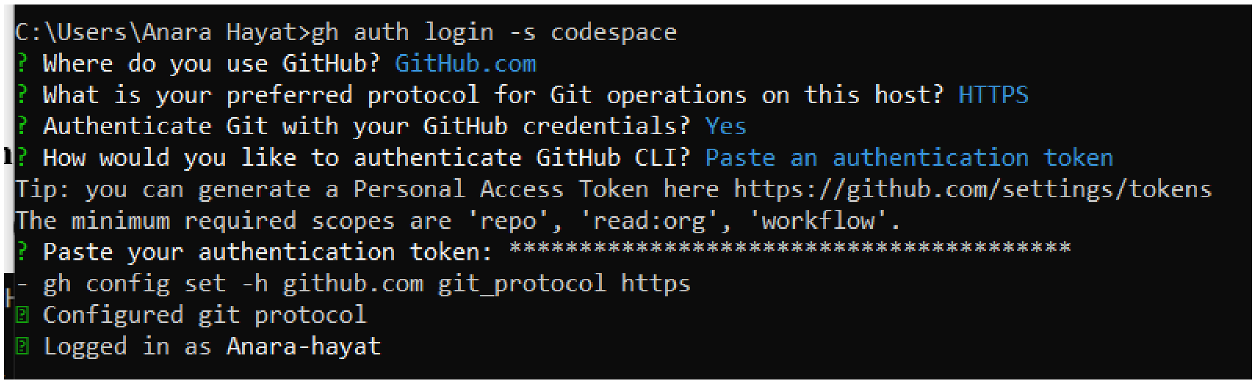
Task 1 — GitHub CLI, Codespace setup and authentication

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



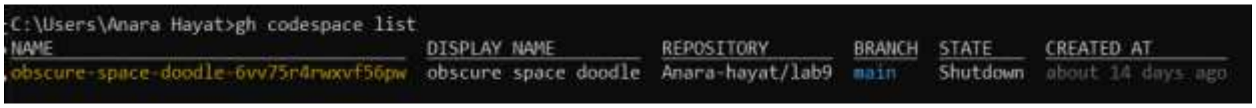
```
C:\Users\Anara Hayat>winget install --id GitHub.cli
Found an existing package already installed. Trying to upgrade the installed package...
Found GitHub CLI [GitHub.cli] Version 2.83.2
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses 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
```

2. **(Local) Authenticate GH CLI for Codespaces:**



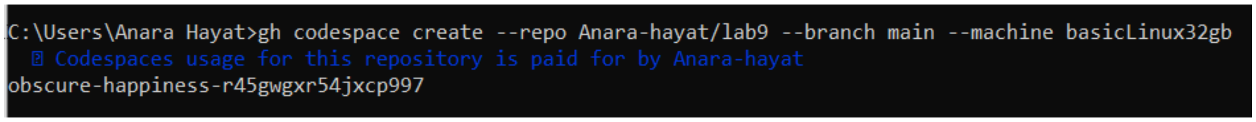
```
C:\Users\Anara Hayat>gh auth login -s codespace
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you 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 Anara-hayat
```

3. **(Local) List available Codespaces (optional verification):**

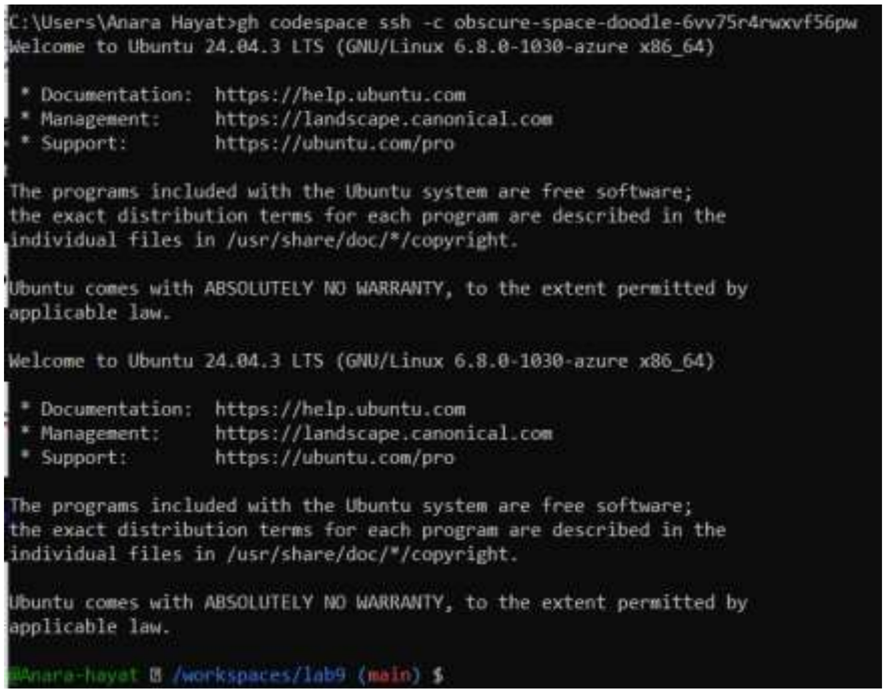


```
C:\Users\Anara Hayat>gh codespace list
NAME                                  DISPLAY NAME          REPOSITORY         BRANCH  STATE     CREATED AT
obscure-space-doodle-6vv75r4rwxvf56pw obscure space doodle  Anara-hayat/lab9   main    Shutdown  about 14 days ago
```

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



```
C:\Users\Anara Hayat>gh codespace create --repo Anara-hayat/lab9 --branch main --machine basicLinux32gb
  ✓ Codespaces usage for this repository is paid for by Anara-hayat
obscure-happiness-r45gwgxr54jxcp997
```

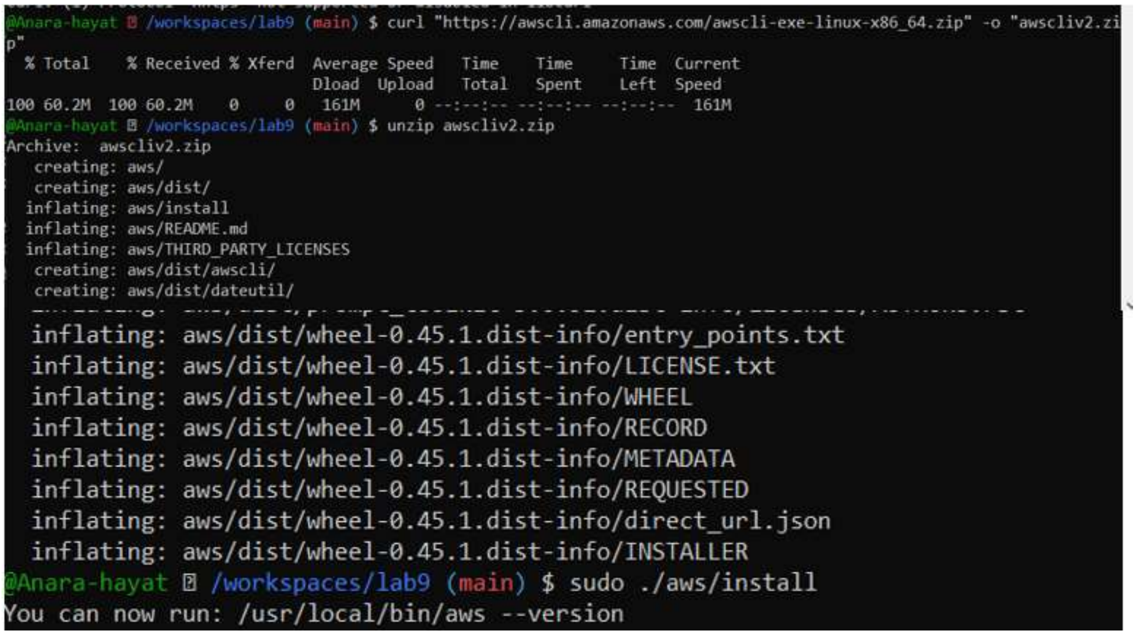


```
C:\Users\Anara Hayat>gh codespace ssh -c obscure-space-doodle-6vv75r4rwxvf56pw
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-1030-azure x86_64)

 * 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.

@Anara-hayat □ /workspaces/lab9 (main) $
```

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

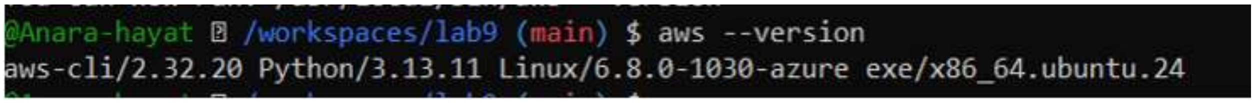
Inside the Codespace shell run:

1. **Download and install AWS CLI:**



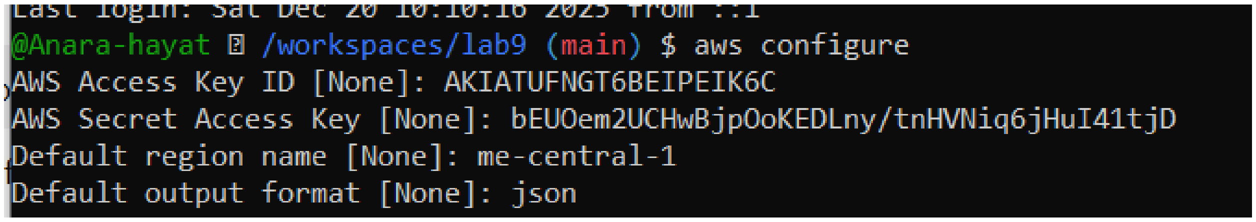
```
@Anara-hayat □ /workspaces/lab9 (main) $ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zi
p"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 60.2M  100 60.2M    0     0   161M      0 --:--:-- --:--:-- --:--:--  161M
@Anara-hayat □ /workspaces/lab9 (main) $ unzip awscliv2.zip
Archive:  awscliv2.zip
   creating: aws/
   creating: aws/dist/
  inflating: aws/install
  inflating: aws/README.md
  inflating: aws/THIRD_PARTY_LICENSES
   creating: aws/dist/awscli/
   creating: aws/dist/dateutil/
  inflating: aws/dist/wheel-0.45.1.dist-info/entry_points.txt
  inflating: aws/dist/wheel-0.45.1.dist-info/LICENSE.txt
  inflating: aws/dist/wheel-0.45.1.dist-info/WHEEL
  inflating: aws/dist/wheel-0.45.1.dist-info/RECORD
  inflating: aws/dist/wheel-0.45.1.dist-info/METADATA
  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/INSTALLER
@Anara-hayat □ /workspaces/lab9 (main) $ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

2. **Verify installation:**



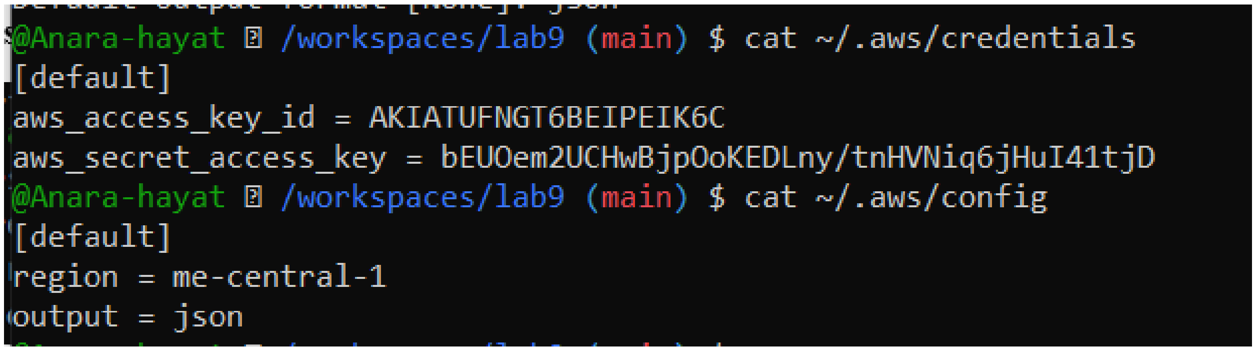
```
@Anara-hayat □ /workspaces/lab9 (main) $ aws --version
aws-cli/2.32.20 Python/3.13.11 Linux/6.8.0-1030-azure exe/x86_64.ubuntu.24
```

3. **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):**



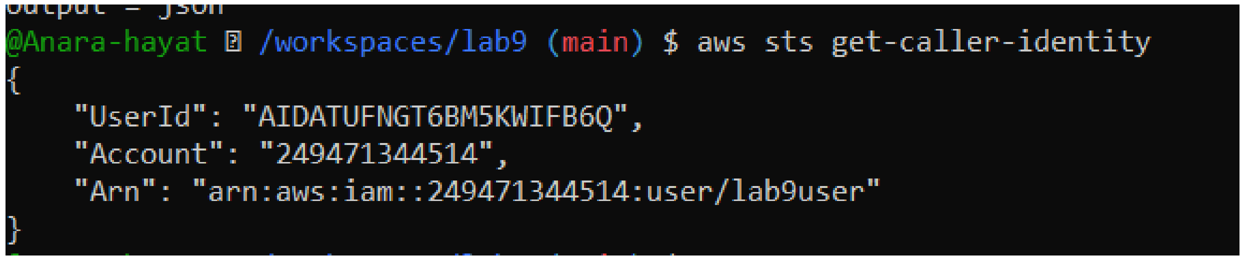
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws configure
AWS Access Key ID [None]: AKIATUFNGT6BEIPEIK6C
AWS Secret Access Key [None]: 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
Default region name [None]: me-central-1
Default output format [None]: json
```

4. **Verify credentials/config files:**



```
@Anara-hayat ➜ /workspaces/lab9 (main) $ cat ~/.aws/credentials
[default]
aws_access_key_id = AKIATUFNGT6BEIPEIK6C
aws_secret_access_key = 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
@Anara-hayat ➜ /workspaces/lab9 (main) $ cat ~/.aws/config
[default]
region = me-central-1
output = json
```

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

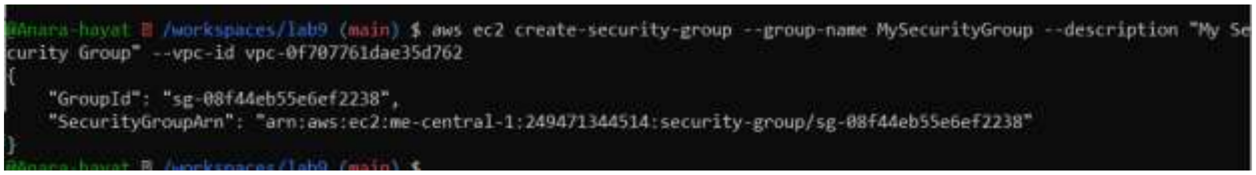


```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws sts get-caller-identity
{
    "UserId": "AIDATUFNGT6BM5KWIFB6Q",
    "Account": "249471344514",
    "Arn": "arn:aws:iam::249471344514:user/lab9user"
}
```

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

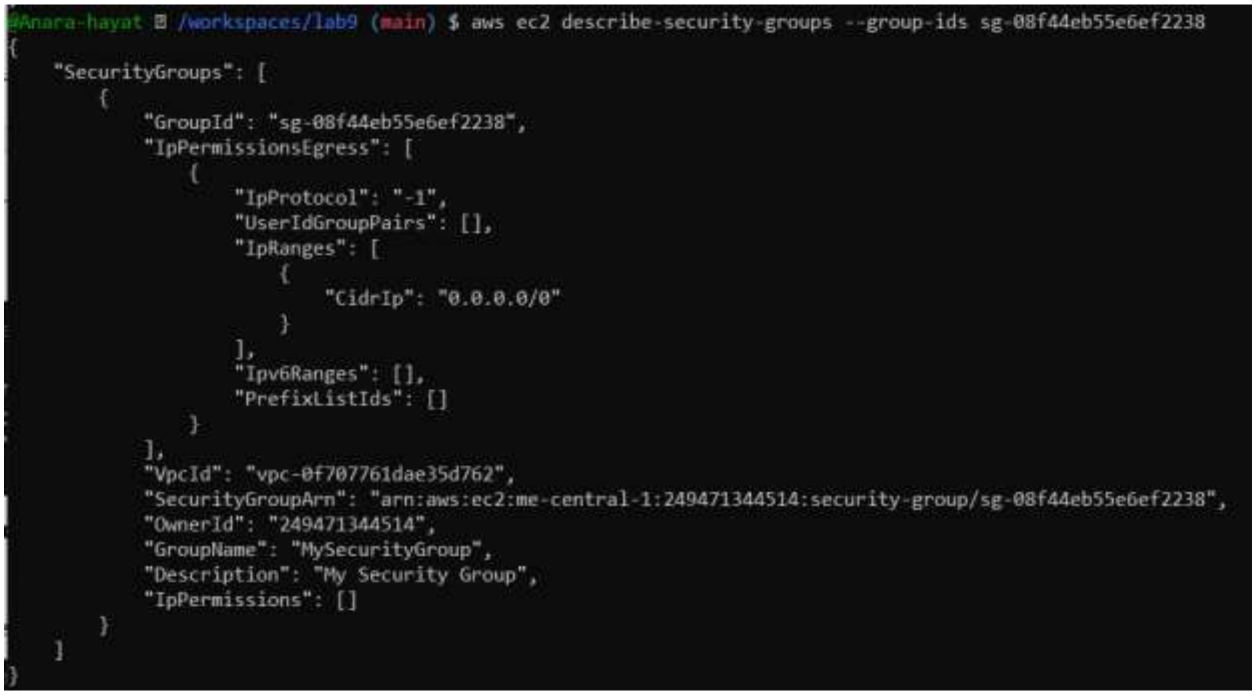
Steps (run in the Codespace shell):

1. **Create a security group (substitute your VPC id):**



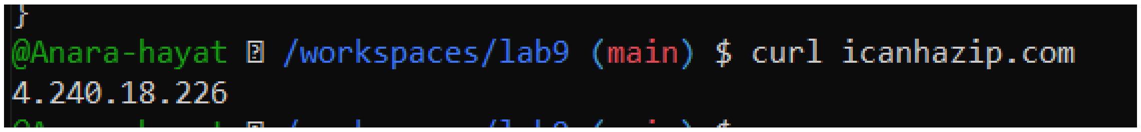
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 create-security-group --group-name MySecurityGroup --description "My Se
curity Group" --vpc-id vpc-0f707761dae35d762
{
    "GroupId": "sg-08f44eb55e6ef2238",
    "SecurityGroupArn": "arn:aws:ec2:me-central-1:249471344514:security-group/sg-08f44eb55e6ef2238"
}
```

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



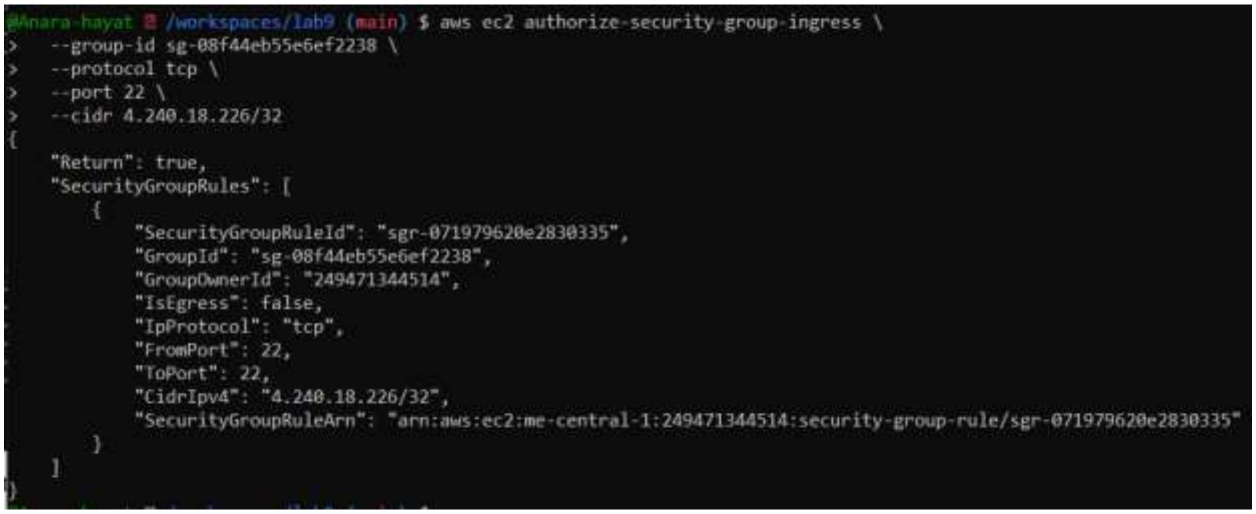
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-security-groups --group-ids sg-08f44eb55e6ef2238
{
    "SecurityGroups": [
        {
            "GroupId": "sg-08f44eb55e6ef2238",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-0f707761dae35d762",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:249471344514:security-group/sg-08f44eb55e6ef2238",
            "OwnerId": "249471344514",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
            "IpPermissions": []
        }
    ]
}
```

3. **Get your Codespace public IP (from inside the Codespace):**



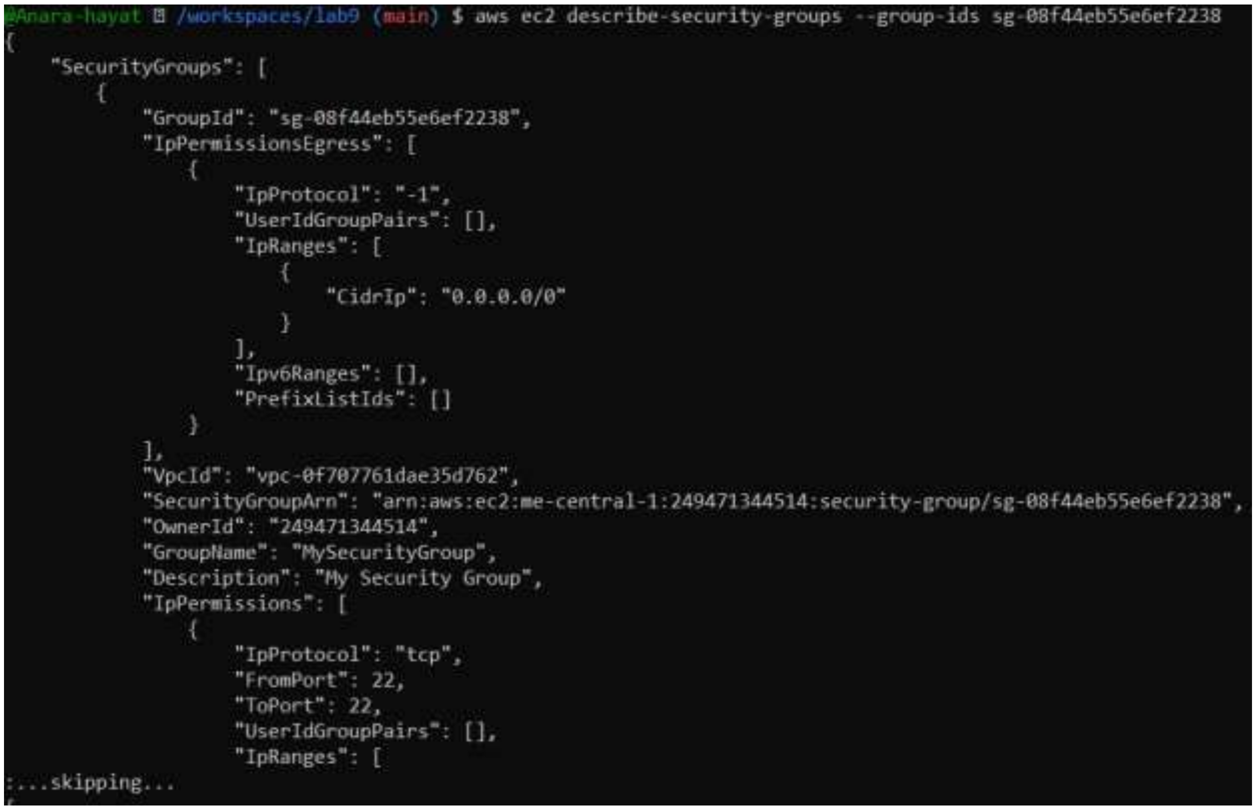
```
}
@Anara-hayat ➜ /workspaces/lab9 (main) $ curl icanhazip.com
4.240.18.226
```

4. **Authorize SSH inbound on port 22 from your Codespace IP:**

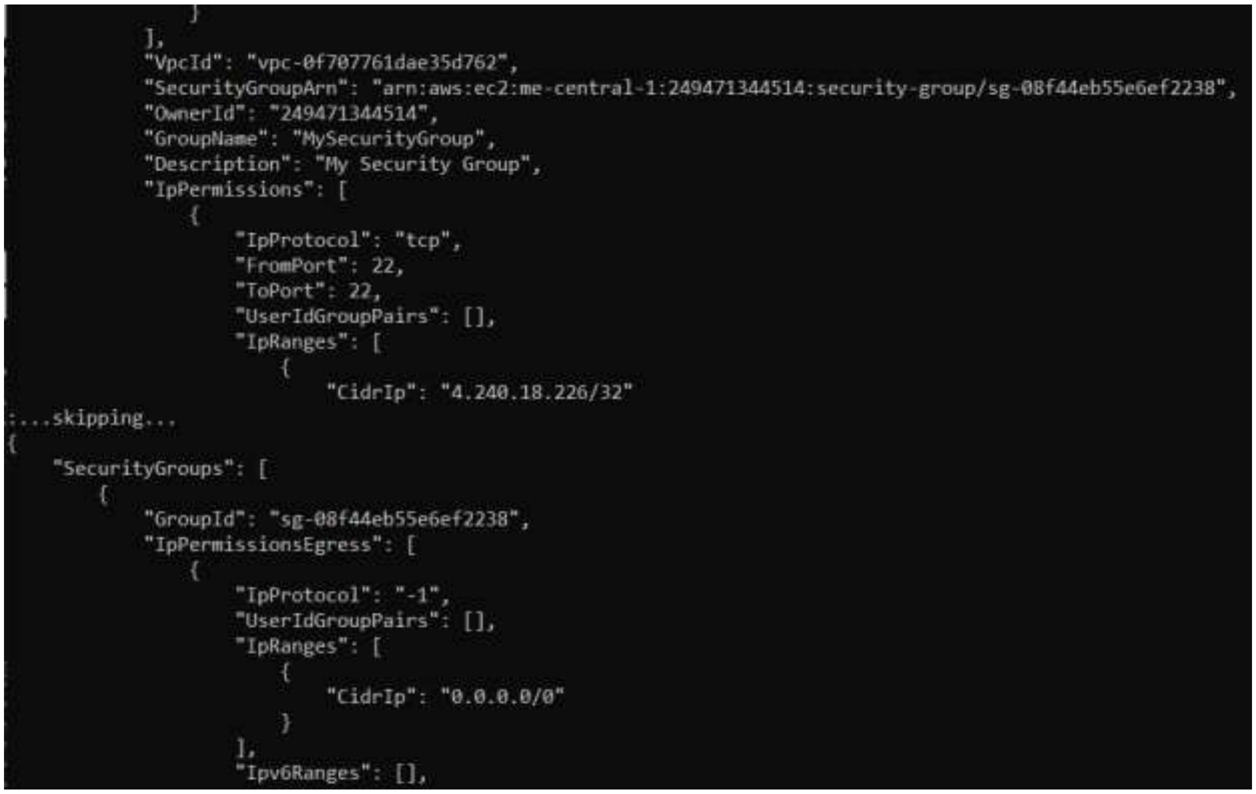


```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 authorize-security-group-ingress \
>   --group-id sg-08f44eb55e6ef2238 \
>   --protocol tcp \
>   --port 22 \
>   --cidr 4.240.18.226/32
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-071979620e2830335",
            "GroupId": "sg-08f44eb55e6ef2238",
            "GroupOwnerId": "249471344514",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "4.240.18.226/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:me-central-1:249471344514:security-group-rule/sgr-071979620e2830335"
        }
    ]
}
```

5. **Verify ingress rule appears:**

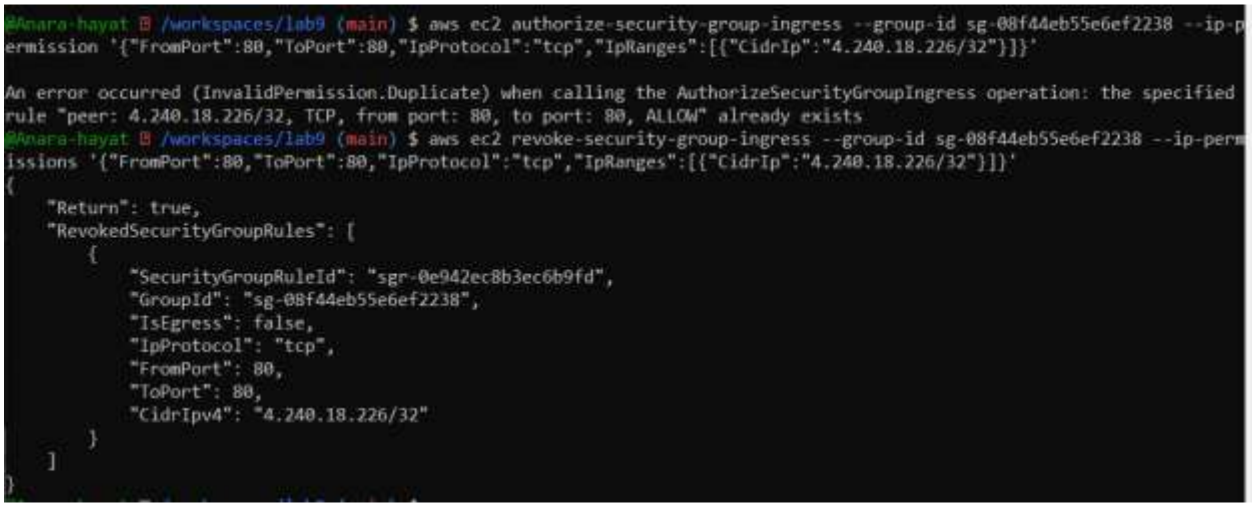


```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-security-groups --group-ids sg-08f44eb55e6ef2238
{
    "SecurityGroups": [
        {
            "GroupId": "sg-08f44eb55e6ef2238",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-0f707761dae35d762",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:249471344514:security-group/sg-08f44eb55e6ef2238",
            "OwnerId": "249471344514",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
            "IpPermissions": [
                {
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "UserIdGroupPairs": [],
                    "IpRanges": [
:...skipping...
```



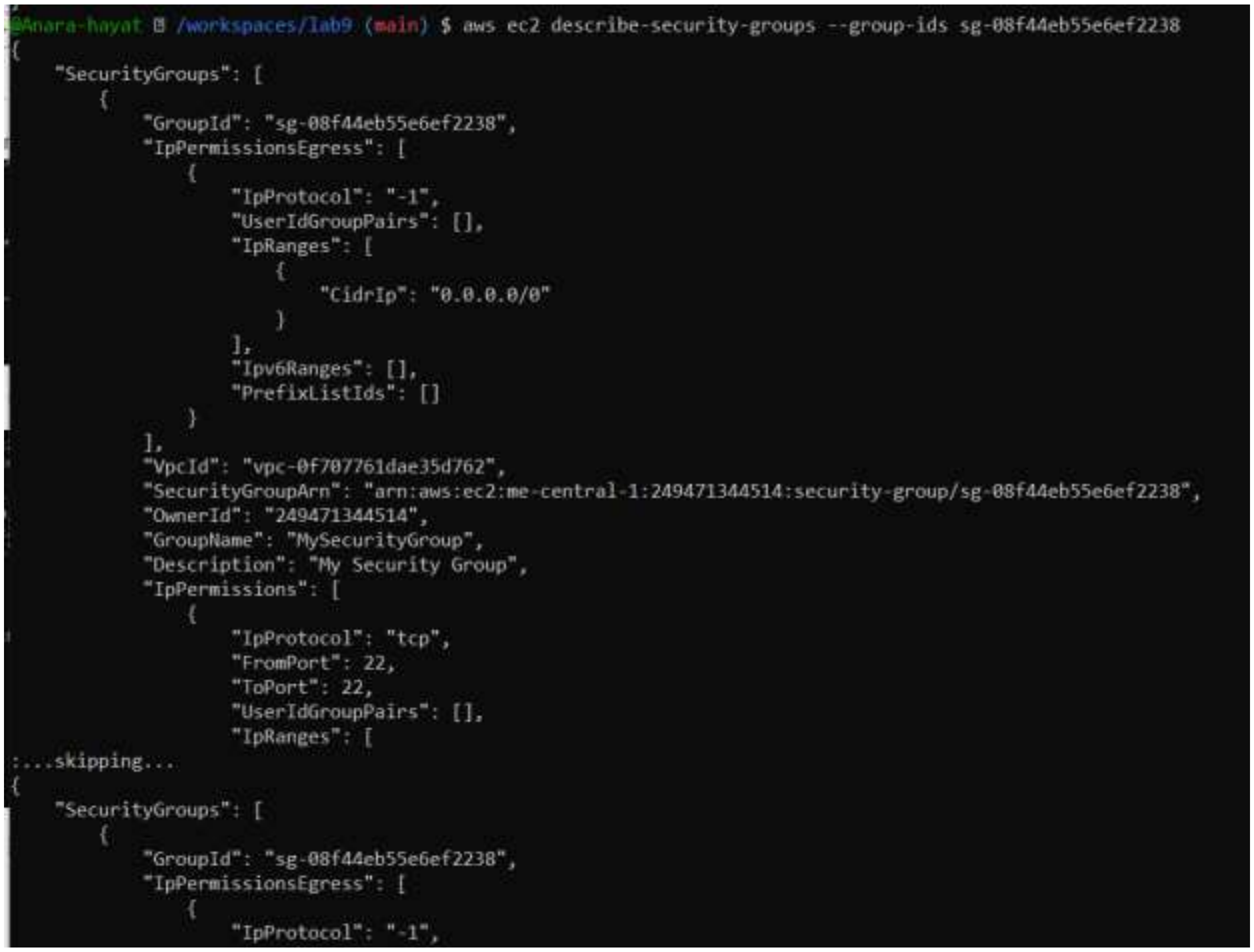
```
                }
            ],
            "VpcId": "vpc-0f707761dae35d762",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:249471344514:security-group/sg-08f44eb55e6ef2238",
            "OwnerId": "249471344514",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
            "IpPermissions": [
                {
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "4.240.18.226/32"
:...skipping...
{
    "SecurityGroups": [
        {
            "GroupId": "sg-08f44eb55e6ef2238",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
```

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



```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 authorize-security-group-ingress --group-id sg-08f44eb55e6ef2238 --ip-p
ermission '{"FromPort":80,"ToPort":80,"IpProtocol":"tcp","IpRanges":[{"CidrIp":"4.240.18.226/32"}]}'

An error occurred (InvalidPermission.Duplicate) when calling the AuthorizeSecurityGroupIngress operation: the specified
rule "peer: 4.240.18.226/32, TCP, from port: 80, to port: 80, ALLOW" already exists
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 revoke-security-group-ingress --group-id sg-08f44eb55e6ef2238 --ip-perm
issions '{"FromPort":80,"ToPort":80,"IpProtocol":"tcp","IpRanges":[{"CidrIp":"4.240.18.226/32"}]}'
{
    "Return": true,
    "RevokedSecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0e942ec8b3ec6b9fd",
            "GroupId": "sg-08f44eb55e6ef2238",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "4.240.18.226/32"
        }
    ]
}
```

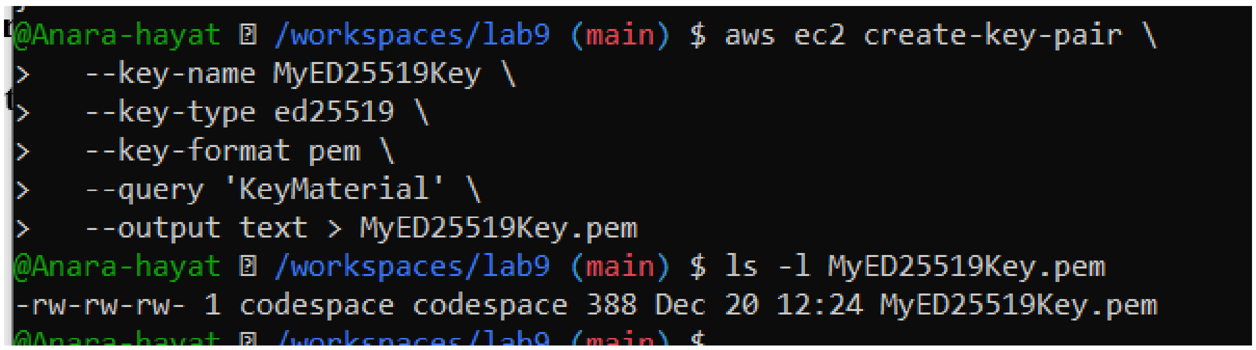
7. **Verify both ingress rules are present:**



```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-security-groups --group-ids sg-08f44eb55e6ef2238
{
    "SecurityGroups": [
        {
            "GroupId": "sg-08f44eb55e6ef2238",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-0f707761dae35d762",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:249471344514:security-group/sg-08f44eb55e6ef2238",
            "OwnerId": "249471344514",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
            "IpPermissions": [
                {
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "UserIdGroupPairs": [],
                    "IpRanges": [
:...skipping...
{
    "SecurityGroups": [
        {
            "GroupId": "sg-08f44eb55e6ef2238",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
```

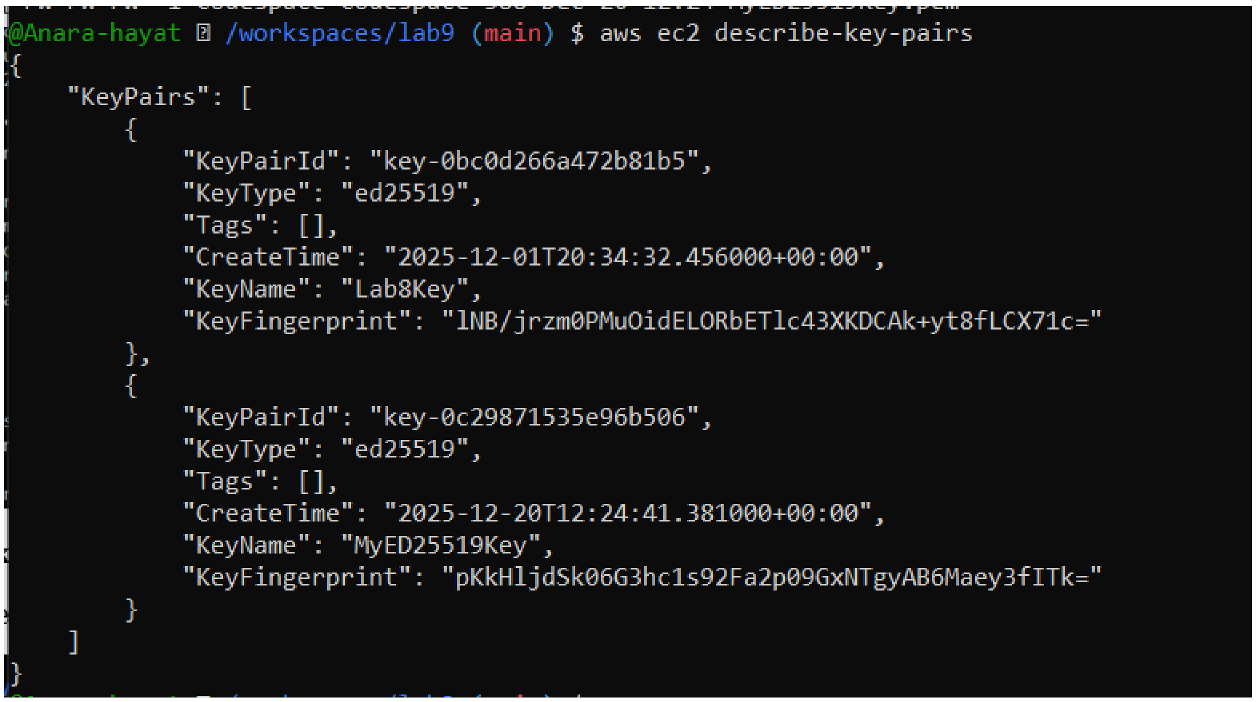
Task 4 — Create a key pair, describe key pairs, and launch EC2 instance

1. **Create the key pair and save the PEM file into the Codespace workspace**



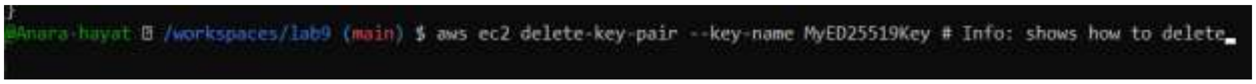
```
@Anara-hayat □ /workspaces/lab9 (main) $ aws ec2 create-key-pair \
>   --key-name MyED25519Key \
>   --key-type ed25519 \
>   --key-format pem \
>   --query 'KeyMaterial' \
>   --output text > MyED25519Key.pem
@Anara-hayat □ /workspaces/lab9 (main) $ ls -l MyED25519Key.pem
-rw-rw-rw- 1 codespace codespace 388 Dec 20 12:24 MyED25519Key.pem
@Anara-hayat □ /workspaces/lab9 (main) $
```

2. **View created key pairs:**



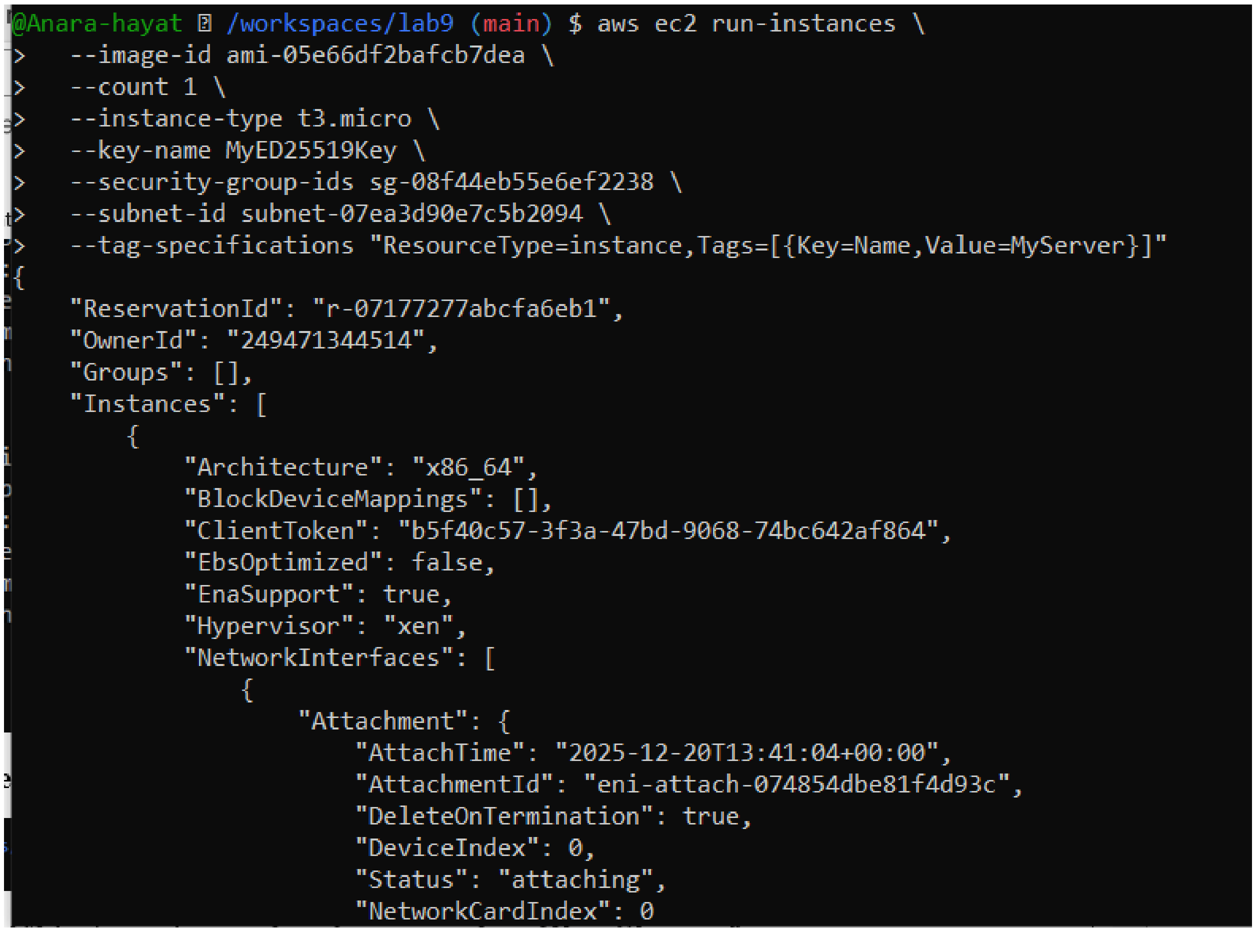
```
-rw-rw-rw- 1 codespace codespace 388 Dec 20 12:24 MyED25519Key.pem
@Anara-hayat □ /workspaces/lab9 (main) $ aws ec2 describe-key-pairs
{
    "KeyPairs": [
        {
            "KeyPairId": "key-0bc0d266a472b81b5",
            "KeyType": "ed25519",
            "Tags": [],
            "CreateTime": "2025-12-01T20:34:32.456000+00:00",
            "KeyName": "Lab8Key",
            "KeyFingerprint": "1NB/jrzm0PMuOidELORbETlc43XKDCAk+yt8fLCX71c="
        },
        {
            "KeyPairId": "key-0c29871535e96b506",
            "KeyType": "ed25519",
            "Tags": [],
            "CreateTime": "2025-12-20T12:24:41.381000+00:00",
            "KeyName": "MyED25519Key",
            "KeyFingerprint": "pKkHljdSk06G3hc1s92Fa2p09GxNTgyAB6Maey3fITk="
        }
    ]
}
```

3. **(Do not) Delete key pair:**



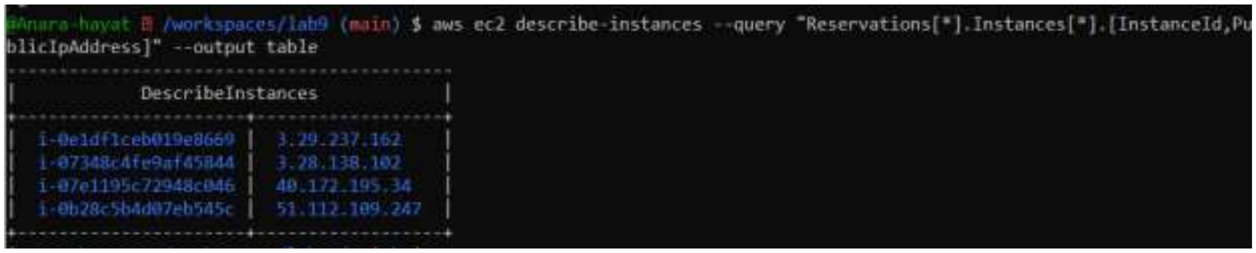
```
@Anara-hayat □ /workspaces/lab9 (main) $ aws ec2 delete-key-pair --key-name MyED25519Key # Info: shows how to delete
```

4. **Launch an EC2 instance (example values — replace IDs with ones from your account/region):**



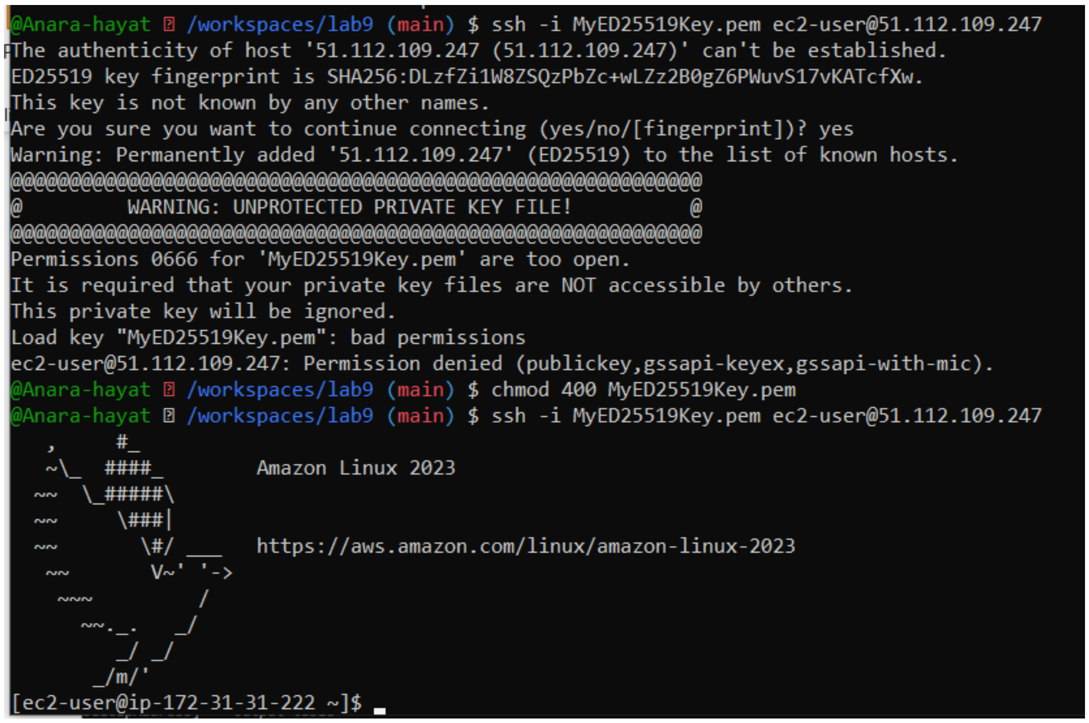
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 run-instances \
>   --image-id ami-05e66df2bafcb7dea \
>   --count 1 \
>   --instance-type t3.micro \
>   --key-name MyED25519Key \
>   --security-group-ids sg-08f44eb55e6ef2238 \
>   --subnet-id subnet-07ea3d90e7c5b2094 \
>   --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=MyServer}]"
{
    "ReservationId": "r-07177277abcfa6eb1",
    "OwnerId": "249471344514",
    "Groups": [],
    "Instances": [
        {
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "b5f40c57-3f3a-47bd-9068-74bc642af864",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2025-12-20T13:41:04+00:00",
                        "AttachmentId": "eni-attach-074854dbe81f4d93c",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "Status": "attaching",
                        "NetworkCardIndex": 0
```

5. **Get the public IP address of your instance:**



```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances --query "Reservations[*].Instances[*].[InstanceId,Pu
blicIpAddress]" --output table
---------------------------------------------
|             DescribeInstances             |
+----------------------+--------------------+
|  i-0e1df1ceb019e8669 |  3.29.237.162      |
|  i-07348c4fe9af45844 |  3.28.138.102      |
|  i-07e1195c72948c046 |  40.172.195.34     |
|  i-0b28c5b4d07eb545c |  51.112.109.247    |
+----------------------+--------------------+
```

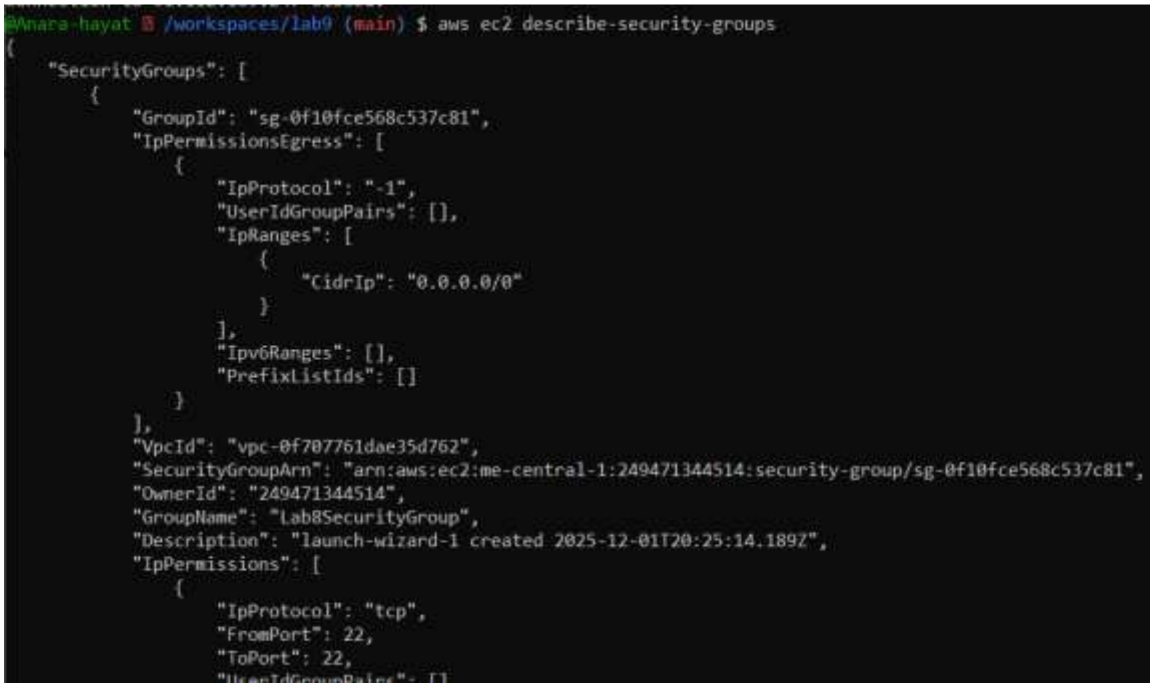
6. **Attempt SSH into the instance from the Codespace or from a machine whose IP is allowed in the security group:**



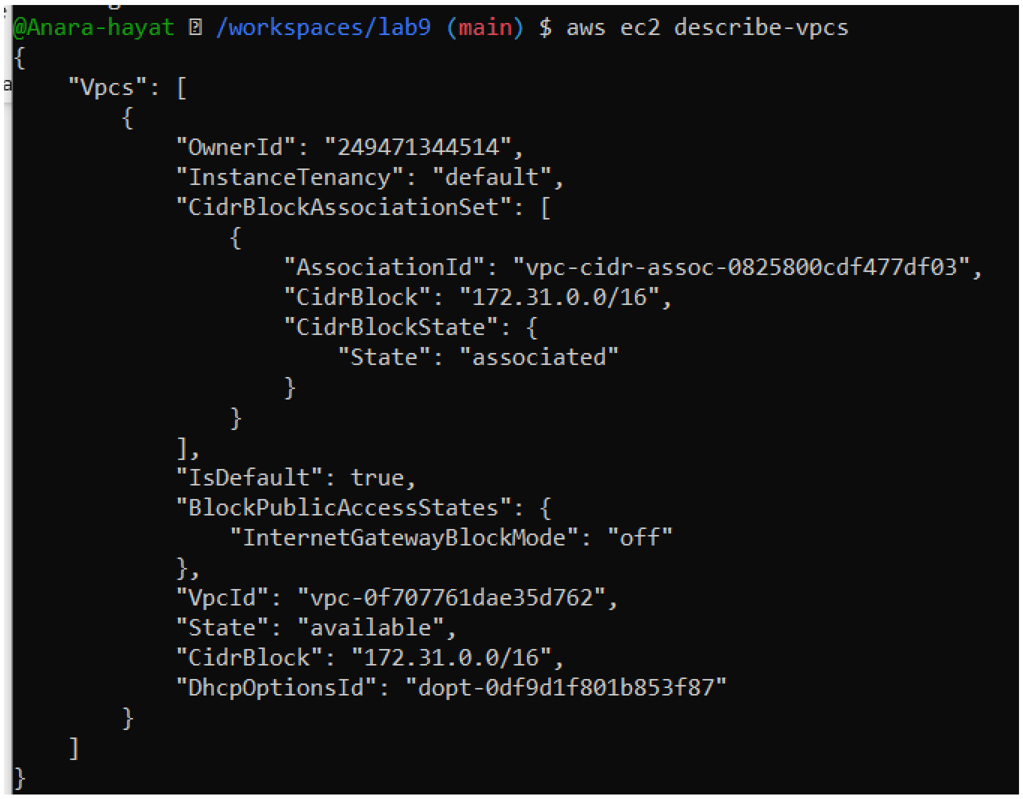
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ ssh -i MyED25519Key.pem ec2-user@51.112.109.247
The authenticity of host '51.112.109.247 (51.112.109.247)' can't be established.
ED25519 key fingerprint is SHA256:DLzfZi1W8ZSQzPbZc+wLZz2B0gZ6PWuvS17vKATcfXw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '51.112.109.247' (ED25519) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0666 for 'MyED25519Key.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "MyED25519Key.pem": bad permissions
ec2-user@51.112.109.247: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
@Anara-hayat ➜ /workspaces/lab9 (main) $ chmod 400 MyED25519Key.pem
@Anara-hayat ➜ /workspaces/lab9 (main) $ ssh -i MyED25519Key.pem ec2-user@51.112.109.247
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-31-222 ~]$
```

Task 5 — Understand AWS describe-* commands

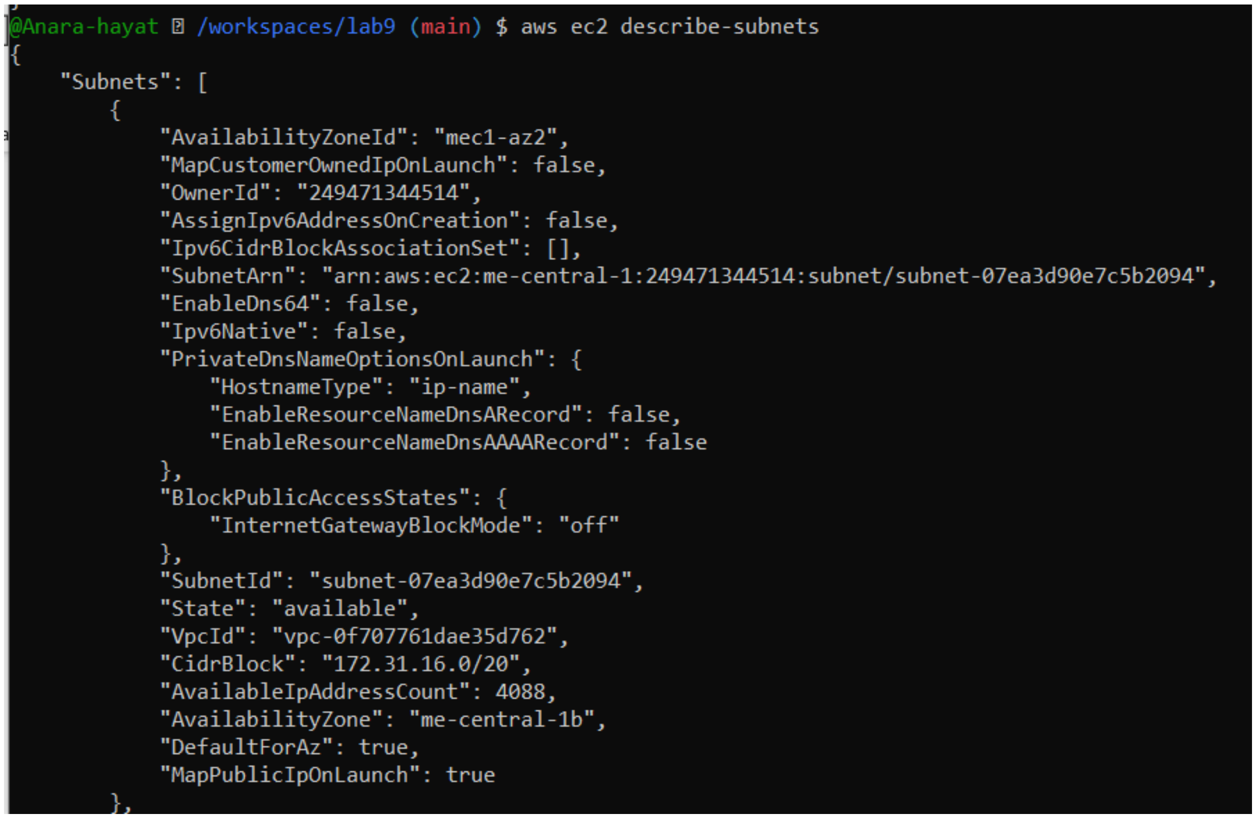
1. **Run and understand these commands (run each, then capture screenshot immediately after)**



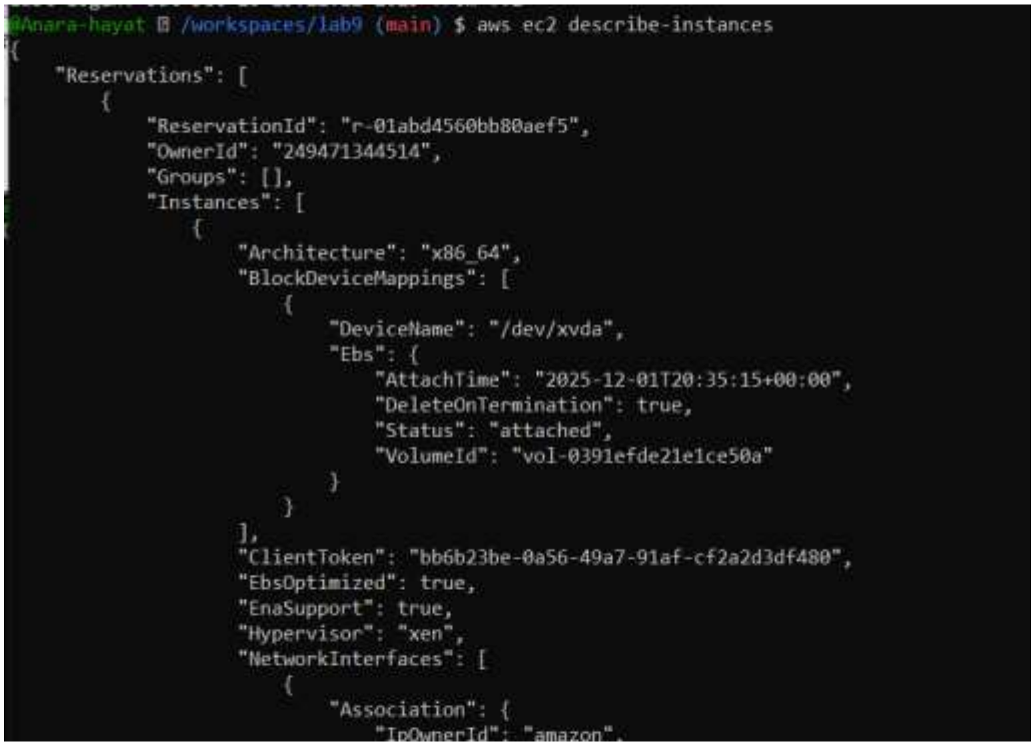
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-security-groups
{
    "SecurityGroups": [
        {
            "GroupId": "sg-0f10fce568c537c81",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-0f707761dae35d762",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:249471344514:security-group/sg-0f10fce568c537c81",
            "OwnerId": "249471344514",
            "GroupName": "Lab8SecurityGroup",
            "Description": "launch-wizard-1 created 2025-12-01T20:25:14.189Z",
            "IpPermissions": [
                {
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "UserIdGroupPairs": []
```



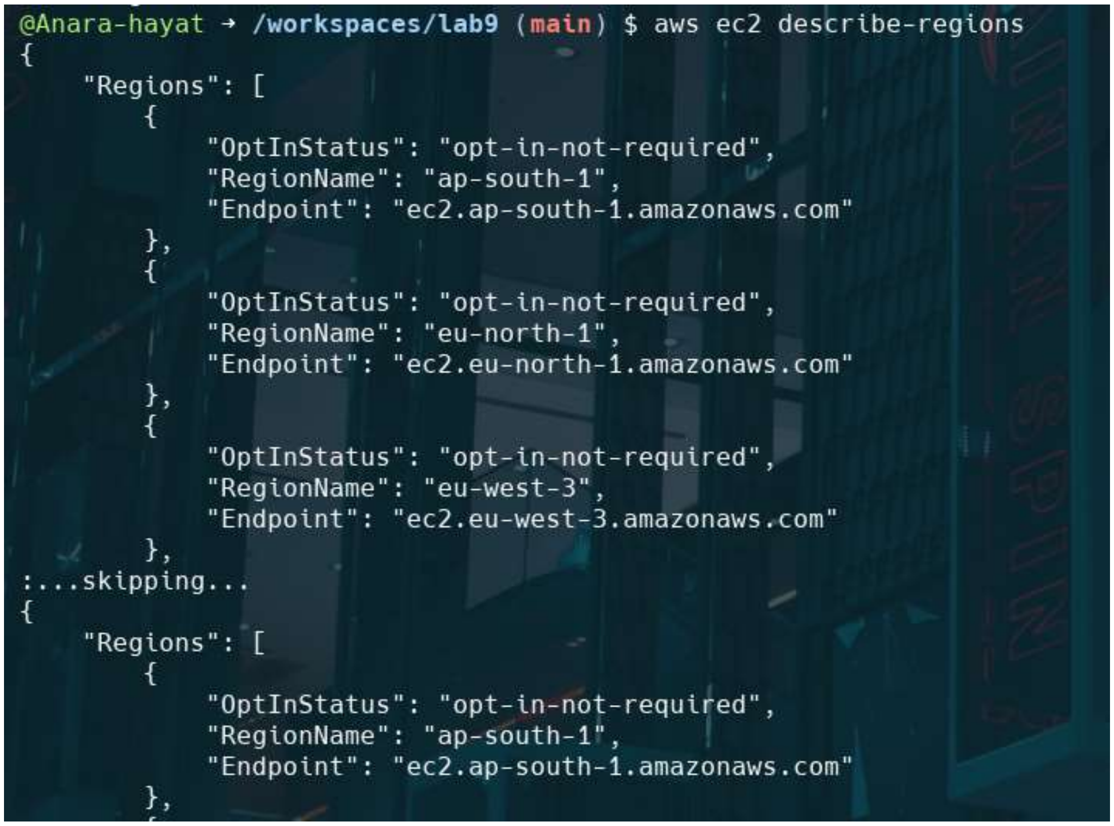
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-vpcs
{
    "Vpcs": [
        {
            "OwnerId": "249471344514",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-0825800cdf477df03",
                    "CidrBlock": "172.31.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": true,
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "VpcId": "vpc-0f707761dae35d762",
            "State": "available",
            "CidrBlock": "172.31.0.0/16",
            "DhcpOptionsId": "dopt-0df9d1f801b853f87"
        }
    ]
}
```



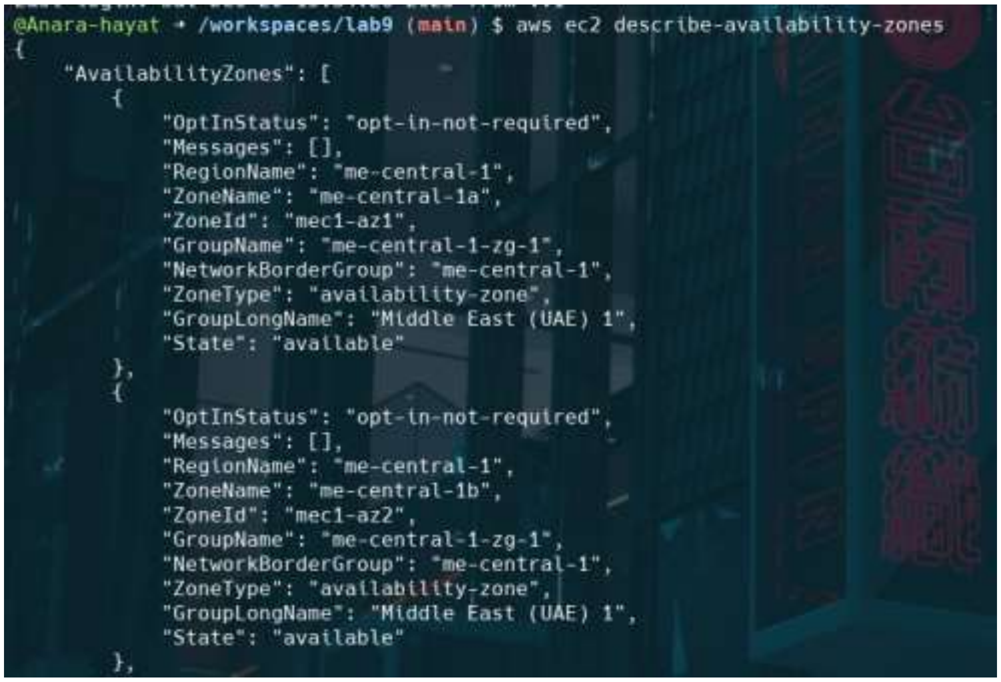
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-subnets
{
    "Subnets": [
        {
            "AvailabilityZoneId": "mec1-az2",
            "MapCustomerOwnedIpOnLaunch": false,
            "OwnerId": "249471344514",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn": "arn:aws:ec2:me-central-1:249471344514:subnet/subnet-07ea3d90e7c5b2094",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            },
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "SubnetId": "subnet-07ea3d90e7c5b2094",
            "State": "available",
            "VpcId": "vpc-0f707761dae35d762",
            "CidrBlock": "172.31.16.0/20",
            "AvailableIpAddressCount": 4088,
            "AvailabilityZone": "me-central-1b",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true
        },
```



```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances
{
    "Reservations": [
        {
            "ReservationId": "r-01abd4560bb80aef5",
            "OwnerId": "249471344514",
            "Groups": [],
            "Instances": [
                {
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/xvda",
                            "Ebs": {
                                "AttachTime": "2025-12-01T20:35:15+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-0391efde21e1ce50a"
                            }
                        }
                    ],
                    "ClientToken": "bb6b23be-0a56-49a7-91af-cf2a2d3df480",
                    "EbsOptimized": true,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "NetworkInterfaces": [
                        {
                            "Association": {
                                "IpOwnerId": "amazon",
```



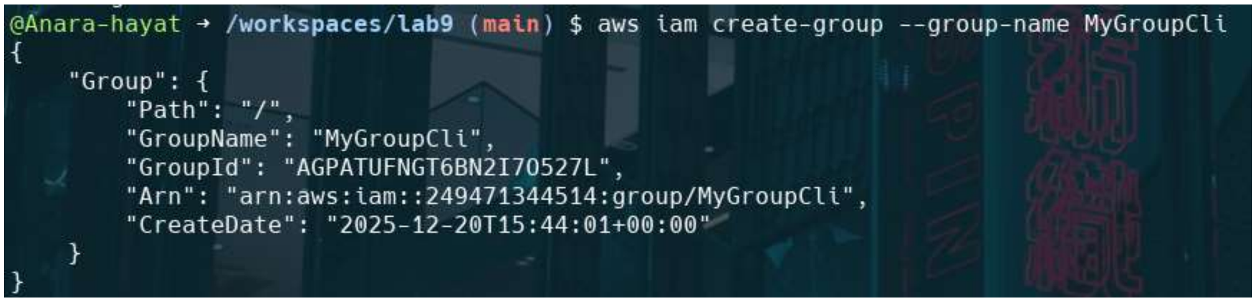
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-regions
{
    "Regions": [
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-south-1",
            "Endpoint": "ec2.ap-south-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "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"
        },
:...skipping...
{
    "Regions": [
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-south-1",
            "Endpoint": "ec2.ap-south-1.amazonaws.com"
        },
```



```
@Anara-hayat ➜ /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"
        },
```

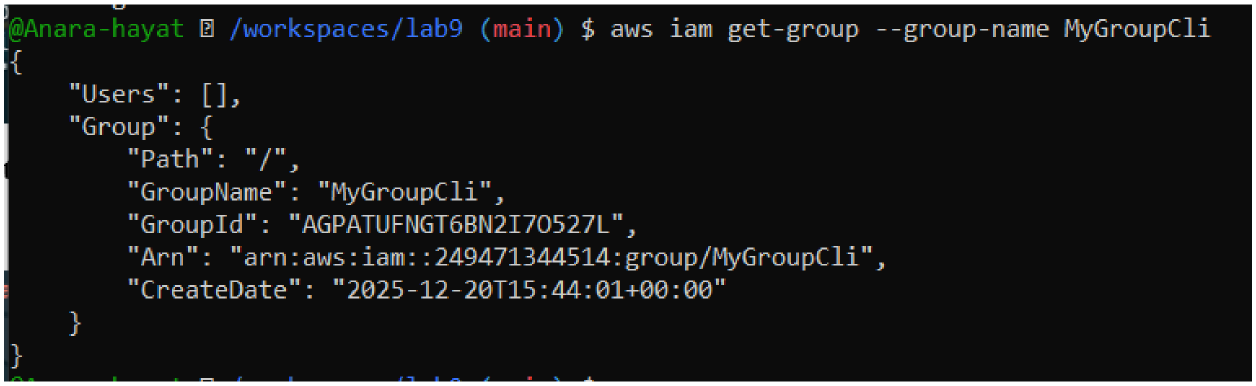
Task 6 — IAM: create group, user, attach policies, create console login & keys

1. **Create group:**



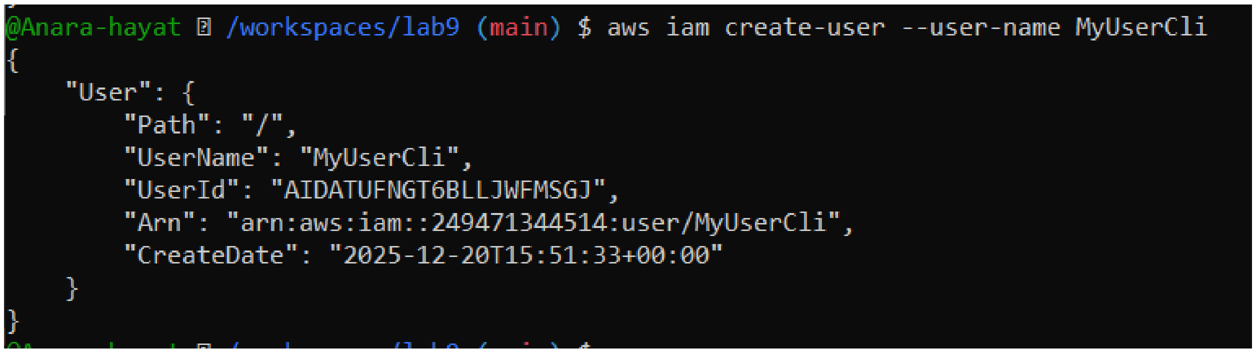
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam create-group --group-name MyGroupCli
{
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPATUFNGT6BN2I70527L",
        "Arn": "arn:aws:iam::249471344514:group/MyGroupCli",
        "CreateDate": "2025-12-20T15:44:01+00:00"
    }
}
```

2. **Get group details:**



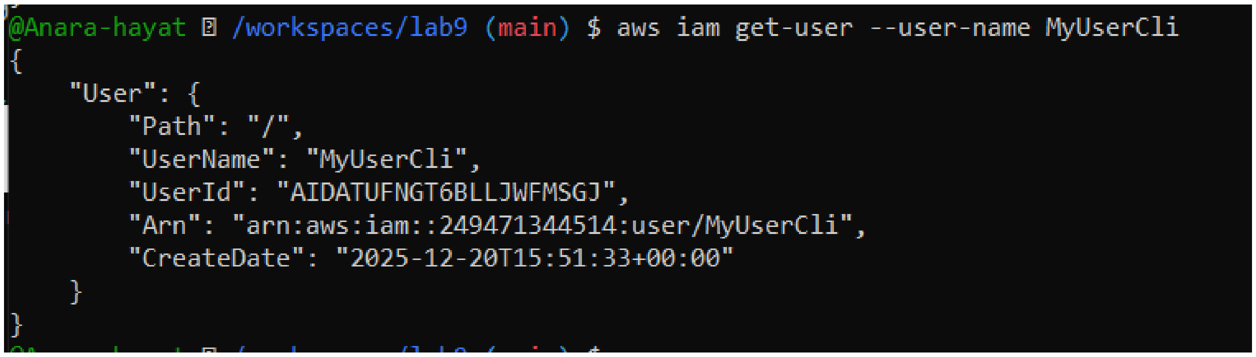
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam get-group --group-name MyGroupCli
{
    "Users": [],
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPATUFNGT6BN2I70527L",
        "Arn": "arn:aws:iam::249471344514:group/MyGroupCli",
        "CreateDate": "2025-12-20T15:44:01+00:00"
    }
}
```

3. **Create user**



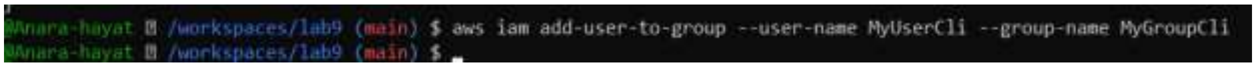
```
@Anara-hayat □ /workspaces/lab9 (main) $ aws iam create-user --user-name MyUserCli
{
    "User": {
        "Path": "/",
        "UserName": "MyUserCli",
        "UserId": "AIDATUFNGT6BLLJWFMSGJ",
        "Arn": "arn:aws:iam::249471344514:user/MyUserCli",
        "CreateDate": "2025-12-20T15:51:33+00:00"
    }
}
```

4. **Get user details:**



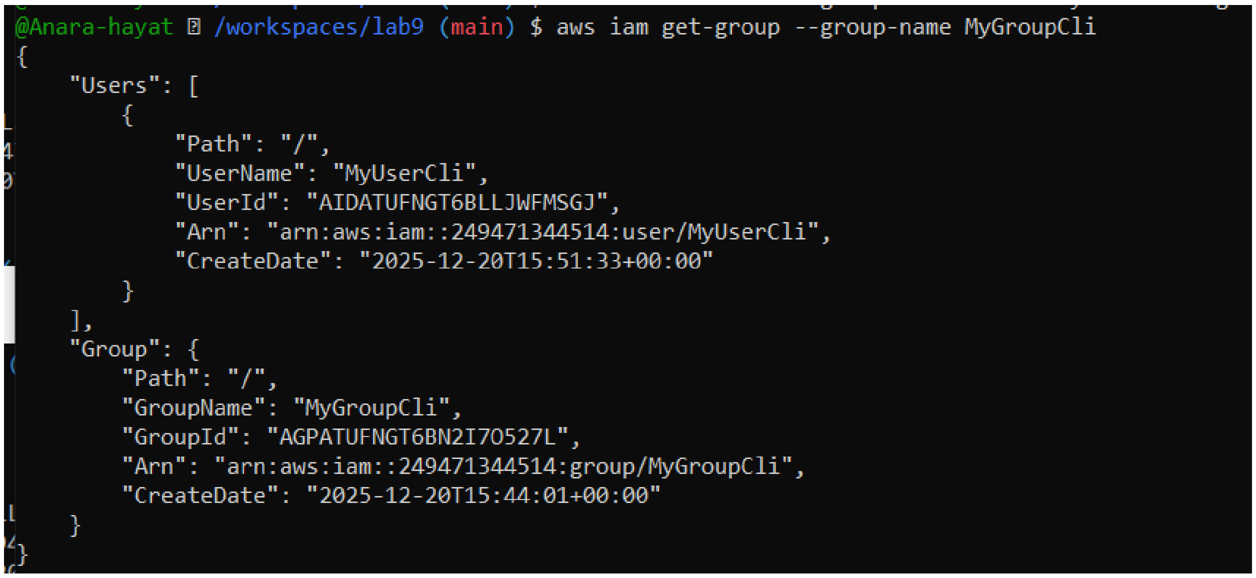
```
@Anara-hayat □ /workspaces/lab9 (main) $ aws iam get-user --user-name MyUserCli
{
    "User": {
        "Path": "/",
        "UserName": "MyUserCli",
        "UserId": "AIDATUFNGT6BLLJWFMSGJ",
        "Arn": "arn:aws:iam::249471344514:user/MyUserCli",
        "CreateDate": "2025-12-20T15:51:33+00:00"
    }
}
```

5. **Add user to group:**



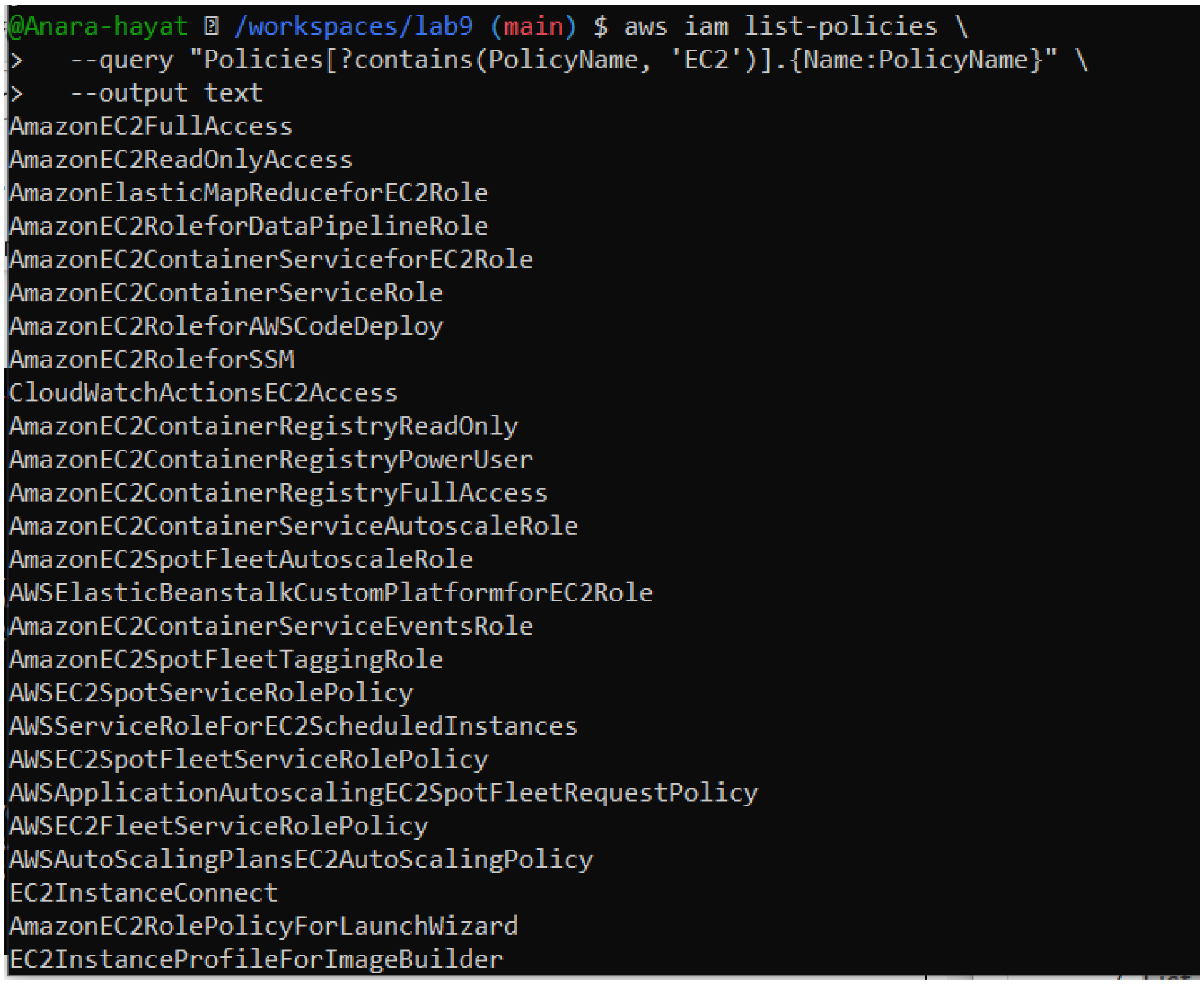
```
@Anara-hayat □ /workspaces/lab9 (main) $ aws iam add-user-to-group --user-name MyUserCli --group-name MyGroupCli
@Anara-hayat □ /workspaces/lab9 (main) $
```

6. **See group again:**



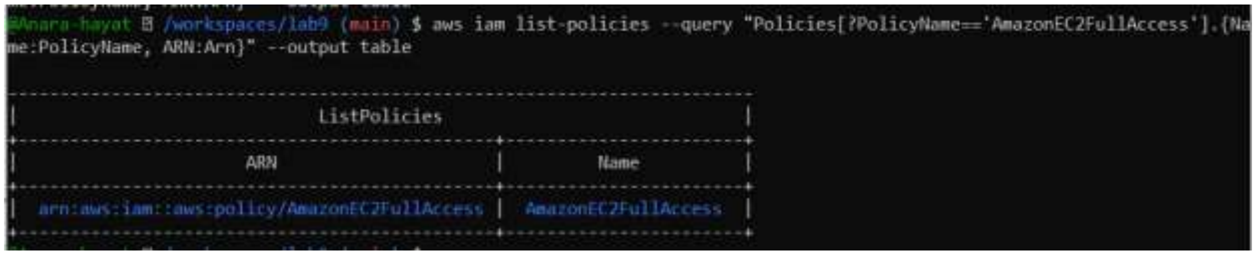
```
@Anara-hayat □ /workspaces/lab9 (main) $ aws iam get-group --group-name MyGroupCli
{
    "Users": [
        {
            "Path": "/",
            "UserName": "MyUserCli",
            "UserId": "AIDATUFNGT6BLLJWFMSGJ",
            "Arn": "arn:aws:iam::249471344514:user/MyUserCli",
            "CreateDate": "2025-12-20T15:51:33+00:00"
        }
    ],
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPATUFNGT6BN2I70527L",
        "Arn": "arn:aws:iam::249471344514:group/MyGroupCli",
        "CreateDate": "2025-12-20T15:44:01+00:00"
    }
}
```

7. **List policies that mention EC2:**



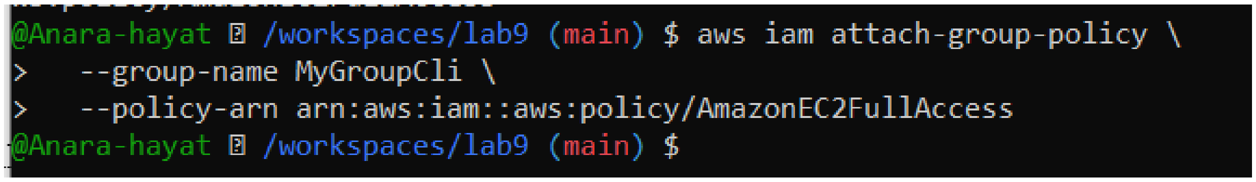
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam list-policies \
>   --query "Policies[?contains(PolicyName, 'EC2')].{Name:PolicyName}" \
>   --output text
AmazonEC2FullAccess
AmazonEC2ReadOnlyAccess
AmazonElasticMapReduceforEC2Role
AmazonEC2RoleforDataPipelineRole
AmazonEC2ContainerServiceforEC2Role
AmazonEC2ContainerServiceRole
AmazonEC2RoleforAWSCodeDeploy
AmazonEC2RoleforSSM
CloudWatchActionsEC2Access
AmazonEC2ContainerRegistryReadOnly
AmazonEC2ContainerRegistryPowerUser
AmazonEC2ContainerRegistryFullAccess
AmazonEC2ContainerServiceAutoscaleRole
AmazonEC2SpotFleetAutoscaleRole
AWSElasticBeanstalkCustomPlatformforEC2Role
AmazonEC2ContainerServiceEventsRole
AmazonEC2SpotFleetTaggingRole
AWSEC2SpotServiceRolePolicy
AWSServiceRoleForEC2ScheduledInstances
AWSEC2SpotFleetServiceRolePolicy
AWSApplicationAutoscalingEC2SpotFleetRequestPolicy
AWSEC2FleetServiceRolePolicy
AWSAutoScalingPlansEC2AutoScalingPolicy
EC2InstanceConnect
AmazonEC2RolePolicyForLaunchWizard
EC2InstanceProfileForImageBuilder
```

8. **Get ARN for AmazonEC2FullAccess (example query):**



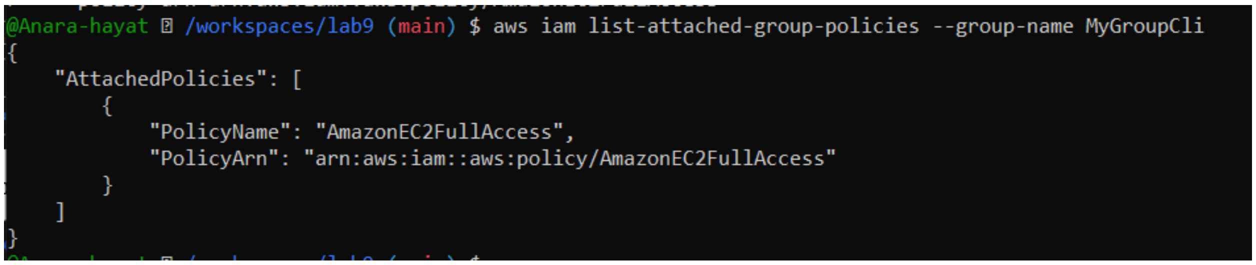
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam list-policies --query "Policies[?PolicyName=='AmazonEC2FullAccess'].{Na
me:PolicyName, ARN:Arn}" --output table

-------------------------------------------------------------------------
|                             ListPolicies                              |
+---------------------------------------------+-------------------------+
|                     ARN                     |          Name           |
+---------------------------------------------+-------------------------+
|  arn:aws:iam::aws:policy/AmazonEC2FullAccess |  AmazonEC2FullAccess   |
+---------------------------------------------+-------------------------+
```

9. **Attach policy to group (use the ARN you retrieved):**



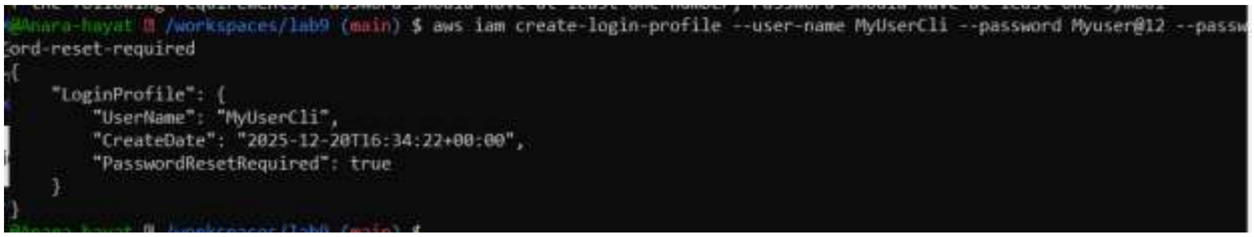
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam attach-group-policy \
>   --group-name MyGroupCli \
>   --policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess
@Anara-hayat ➜ /workspaces/lab9 (main) $
```

10. **List attached policies for the group:**



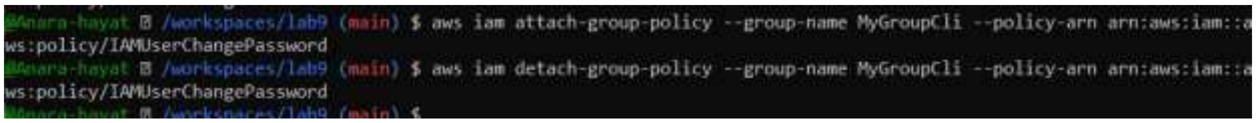
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam list-attached-group-policies --group-name MyGroupCli
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEC2FullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
        }
    ]
}
```

11. Create a console login profile for the user:



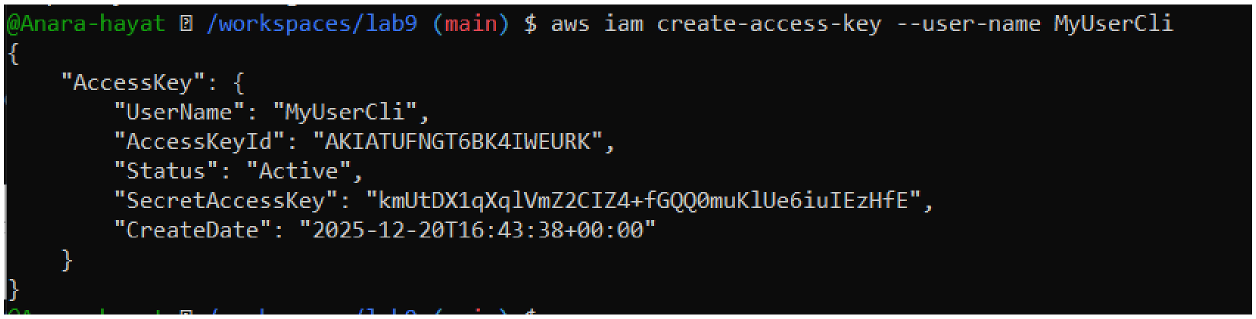
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam create-login-profile --user-name MyUserCli --password Myuser@12 --passw
ord-reset-required
{
    "LoginProfile": {
        "UserName": "MyUserCli",
        "CreateDate": "2025-12-20T16:34:22+00:00",
        "PasswordResetRequired": true
    }
}
```

12. If the user cannot change password, attach IAMUserChangePassword to the group temporarily:



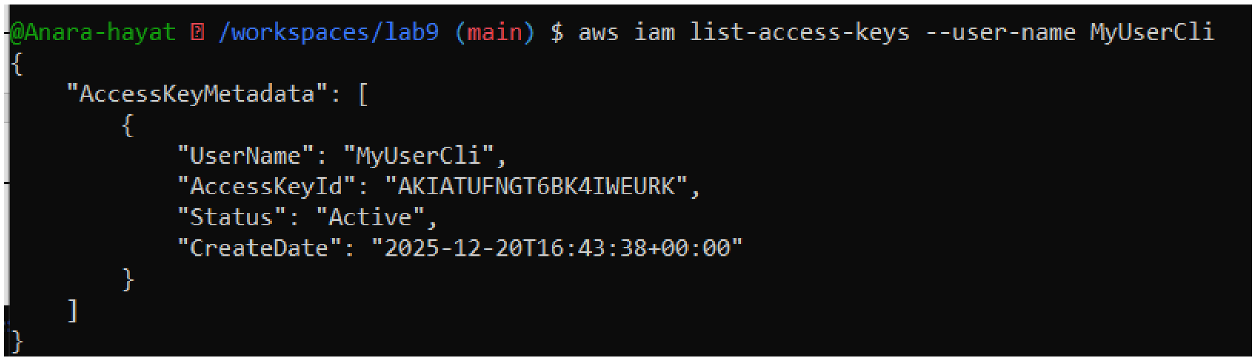
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam attach-group-policy --group-name MyGroupCli --policy-arn arn:aws:iam::a
ws:policy/IAMUserChangePassword
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam detach-group-policy --group-name MyGroupCli --policy-arn arn:aws:iam::a
ws:policy/IAMUserChangePassword
@Anara-hayat ➜ /workspaces/lab9 (main) $
```

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



```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam create-access-key --user-name MyUserCli
{
    "AccessKey": {
        "UserName": "MyUserCli",
        "AccessKeyId": "AKIATUFNGT6BK4IWEURK",
        "Status": "Active",
        "SecretAccessKey": "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",
        "CreateDate": "2025-12-20T16:43:38+00:00"
    }
}
```

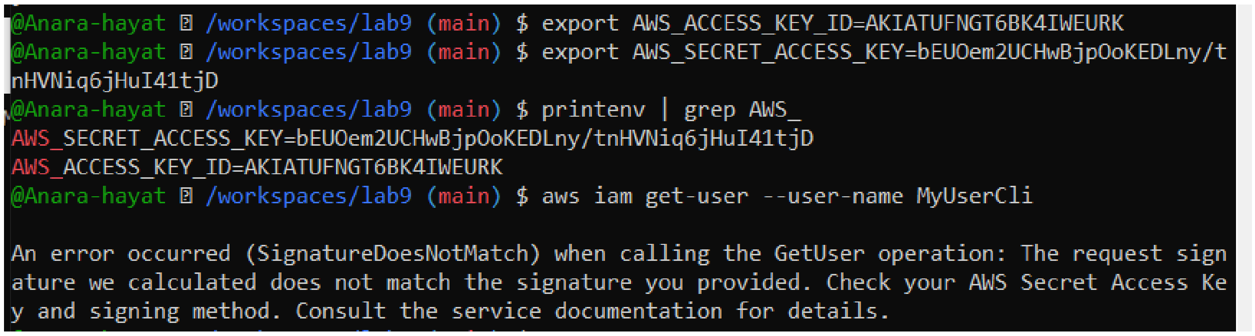
14. List access keys:



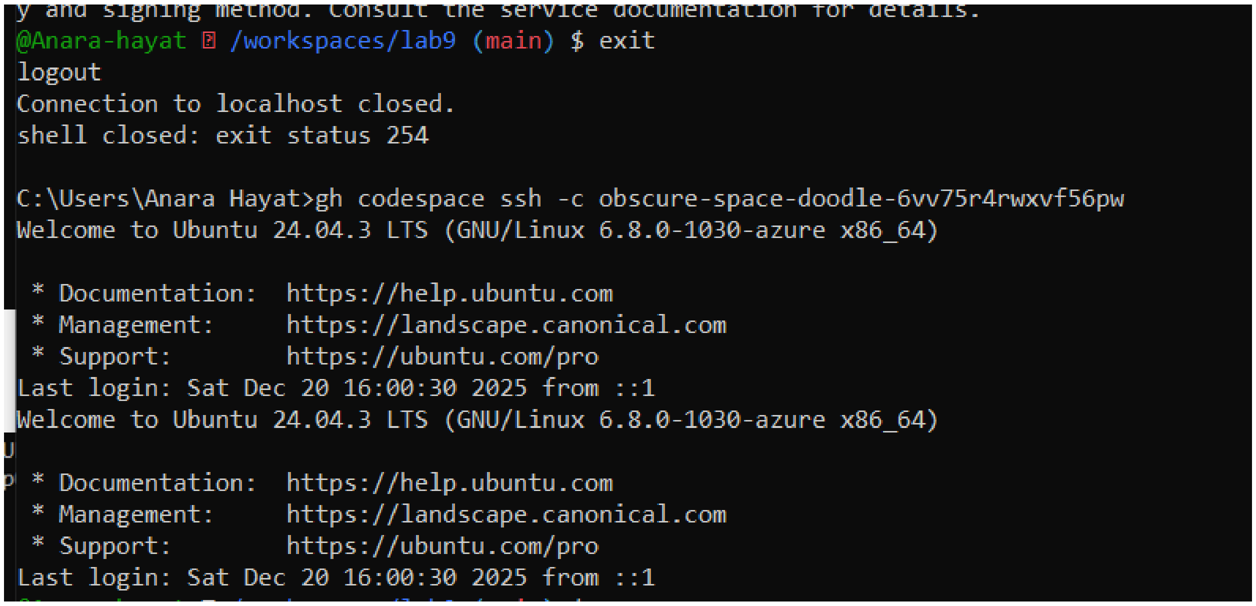
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws iam list-access-keys --user-name MyUserCli
{
    "AccessKeyMetadata": [
        {
            "UserName": "MyUserCli",
            "AccessKeyId": "AKIATUFNGT6BK4IWEURK",
            "Status": "Active",
            "CreateDate": "2025-12-20T16:43:38+00:00"
        }
    ]
}
```

15. Not deleted any key

16. Use environment variables to authenticate as that user in the Codespace:



```
@Anara-hayat ▯ /workspaces/lab9 (main) $ export AWS_ACCESS_KEY_ID=AKIATUFNGT6BK4IWEURK
@Anara-hayat ▯ /workspaces/lab9 (main) $ export AWS_SECRET_ACCESS_KEY=bEUOem2UCHwBjpOoKEDLny/t
nHVNiq6jHuI41tjD
@Anara-hayat ▯ /workspaces/lab9 (main) $ printenv | grep AWS_
AWS_SECRET_ACCESS_KEY=bEUOem2UCHwBjpOoKEDLny/tnHVNiq6jHuI41tjD
AWS_ACCESS_KEY_ID=AKIATUFNGT6BK4IWEURK
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam get-user --user-name MyUserCli

An error occurred (SignatureDoesNotMatch) when calling the GetUser operation: The request sign
ature we calculated does not match the signature you provided. Check your AWS Secret Access Ke
y and signing method. Consult the service documentation for details.
```

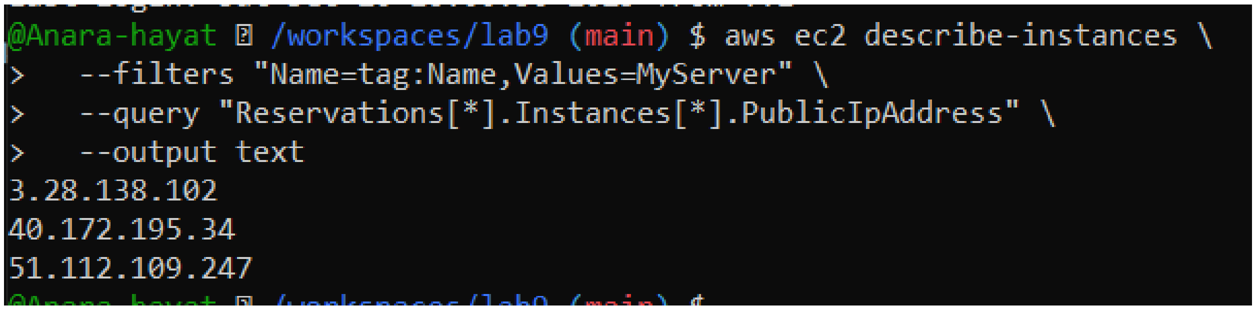


```
y and signing method. Consult the service documentation for details.
@Anara-hayat ▯ /workspaces/lab9 (main) $ exit
logout
Connection to localhost closed.
shell closed: exit status 254

C:\Users\Anara Hayat>gh codespace ssh -c obscure-space-doodle-6vv75r4rwxvf56pw
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
Last login: Sat Dec 20 16:00:30 2025 from ::1
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
Last login: Sat Dec 20 16:00:30 2025 from ::1
```

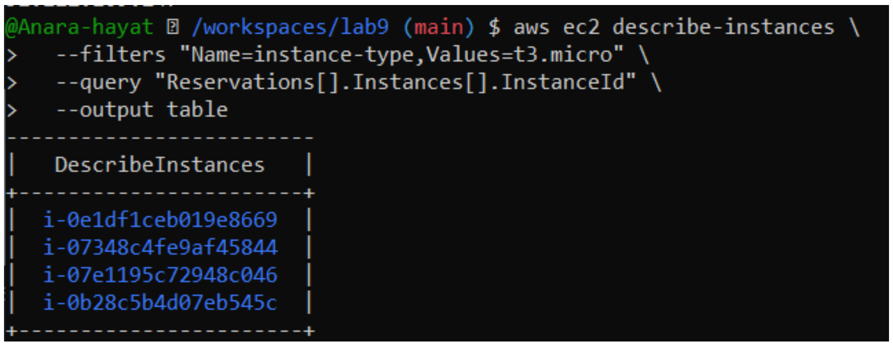
Task 7 — Filters: query with filters to find instances and their attributes

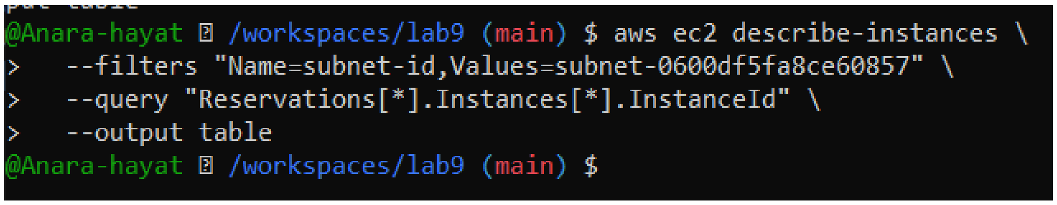
Goal: Use filters and queries to list specific instances and attributes.

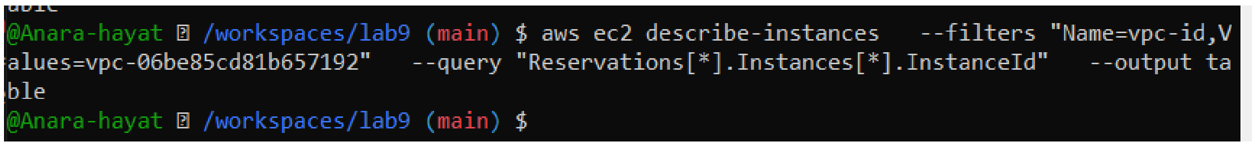
Examples (run each and take a screenshot immediately after):



```
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=tag:Name,Values=MyServer" \
>   --query "Reservations[*].Instances[*].PublicIpAddress" \
>   --output text
3.28.138.102
40.172.195.34
51.112.109.247
```

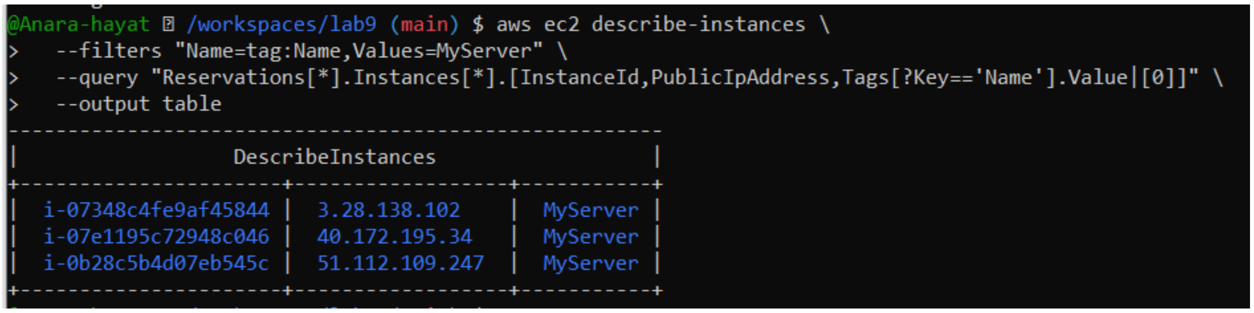

```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=instance-type,Values=t3.micro" \
>   --query "Reservations[].Instances[].InstanceId" \
>   --output table
-------------------------
|   DescribeInstances   |
+-----------------------+
|  i-0e1df1ceb019e8669  |
|  i-07348c4fe9af45844  |
|  i-07e1195c72948c046  |
|  i-0b28c5b4d07eb545c  |
+-----------------------+
```

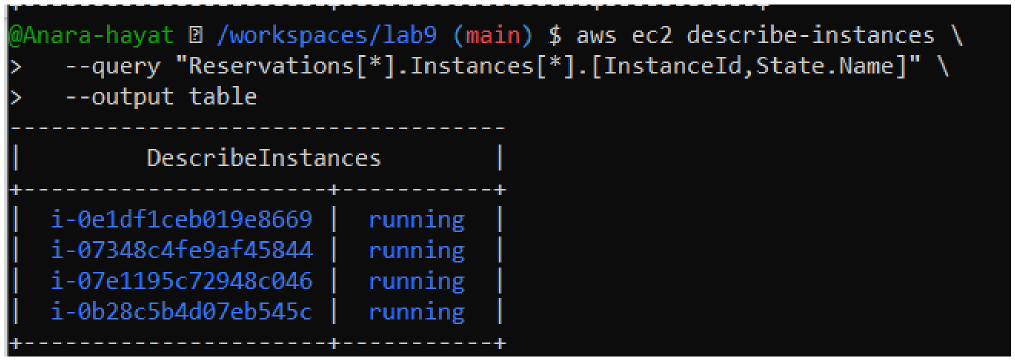

```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --filters "Name=subnet-id,Values=subnet-0600df5fa8ce60857" \
>   --query "Reservations[*].Instances[*].InstanceId" \
>   --output table
@Anara-hayat ➜ /workspaces/lab9 (main) $
```

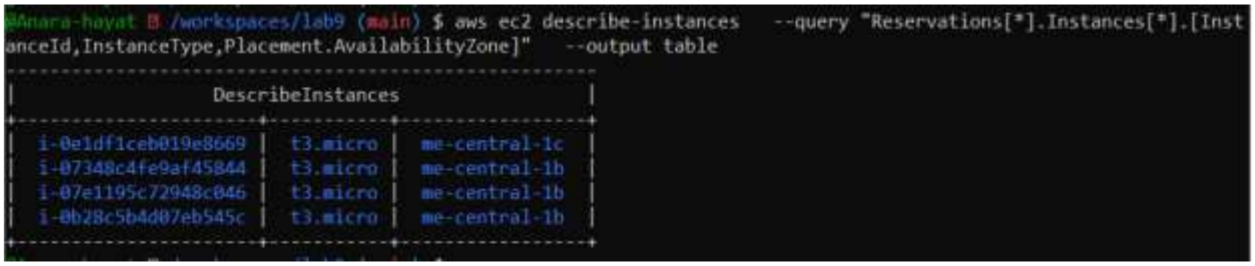

```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances   --filters "Name=vpc-id,V
alues=vpc-06be85cd81b657192"   --query "Reservations[*].Instances[*].InstanceId"   --output ta
ble
@Anara-hayat ➜ /workspaces/lab9 (main) $
```

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

Examples (run each and take a screenshot immediately after):


```
@Anara-hayat ➜ /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-07348c4fe9af45844 |  3.28.138.102    |  MyServer |
|  i-07e1195c72948c046 |  40.172.195.34   |  MyServer |
|  i-0b28c5b4d07eb545c |  51.112.109.247  |  MyServer |
+---------------------+------------------+----------+
```

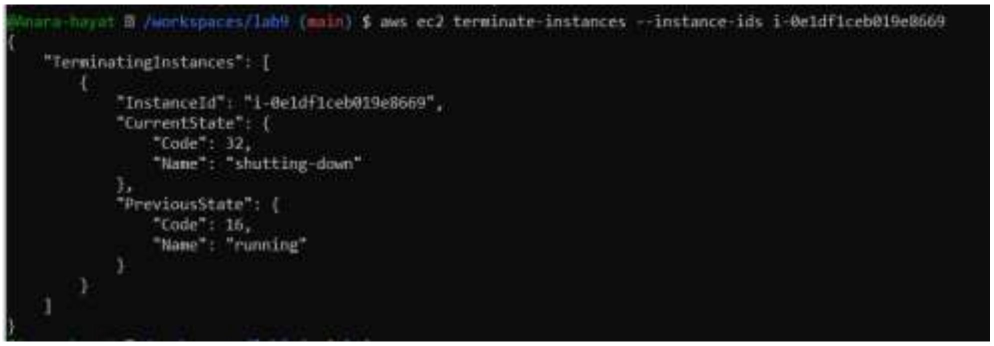

```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --query "Reservations[*].Instances[*].[InstanceId,State.Name]" \
>   --output table
-----------------------------------
|        DescribeInstances        |
+---------------------+-----------+
|  i-0e1df1ceb019e8669 |  running  |
|  i-07348c4fe9af45844 |  running  |
|  i-07e1195c72948c046 |  running  |
|  i-0b28c5b4d07eb545c |  running  |
+---------------------+-----------+
```



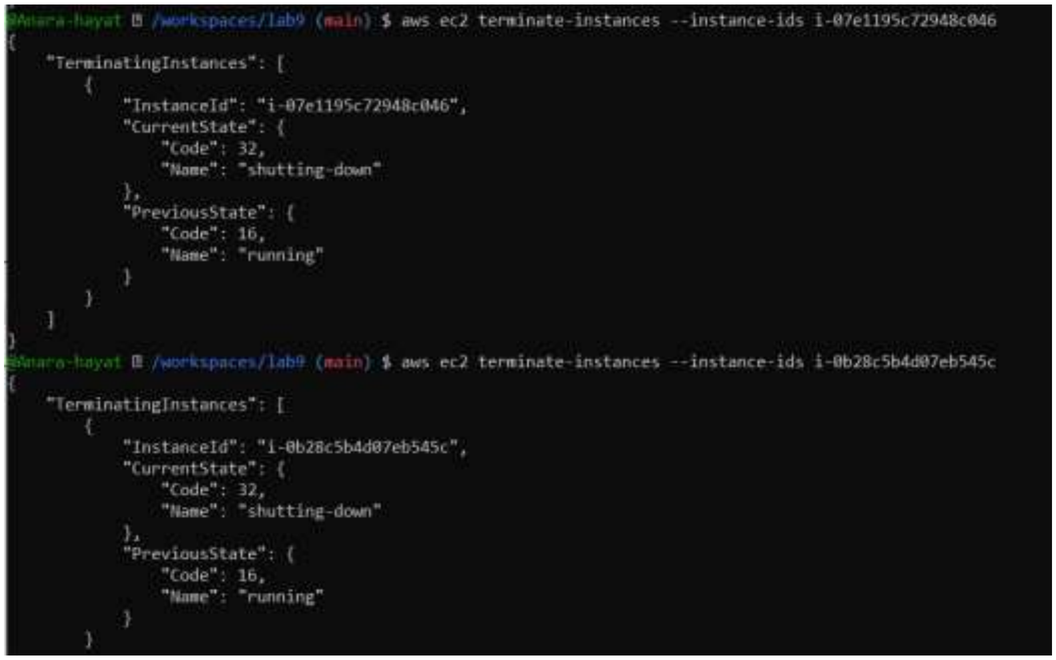
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-instances   --query "Reservations[*].Instances[*].[Inst
anceId,InstanceType,Placement.AvailabilityZone]"   --output table
-------------------------------------------------------
|                  DescribeInstances                  |
+----------------------+------------+-----------------+
|  i-0e1df1ceb019e8669 |  t3.micro  |  me-central-1c  |
|  i-07348c4fe9af45844 |  t3.micro  |  me-central-1b  |
|  i-07e1195c72948c046 |  t3.micro  |  me-central-1b  |
|  i-0b28c5b4d07eb545c |  t3.micro  |  me-central-1b  |
+----------------------+------------+-----------------+
```

Cleanup — Remove resources to avoid charges

1. **Terminate instances:**

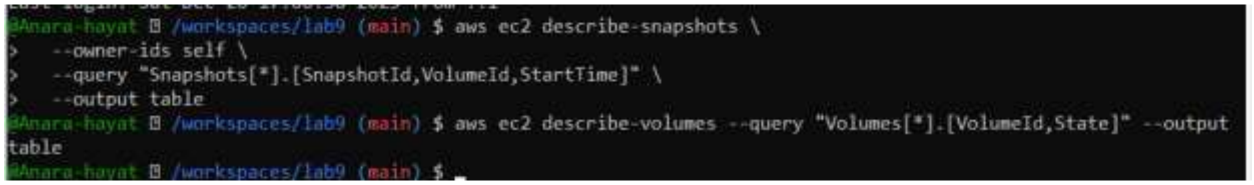


```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 terminate-instances --instance-ids i-0e1df1ceb019e8669
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-0e1df1ceb019e8669",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```



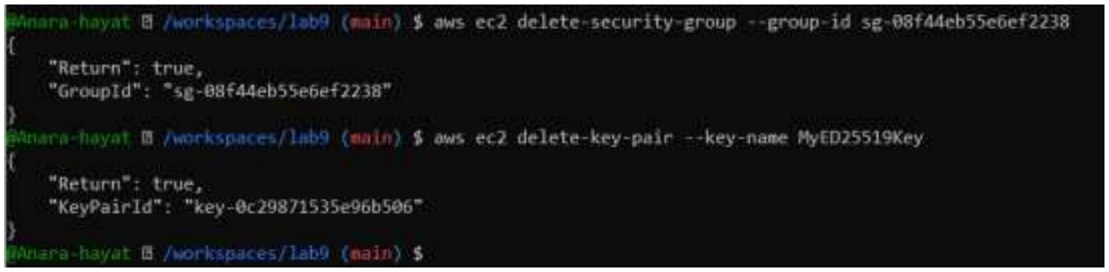
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 terminate-instances --instance-ids i-07e1195c72948c046
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-07e1195c72948c046",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 terminate-instances --instance-ids i-0b28c5b4d07eb545c
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-0b28c5b4d07eb545c",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
```

2. **Delete EBS volumes & snapshots (if any):**



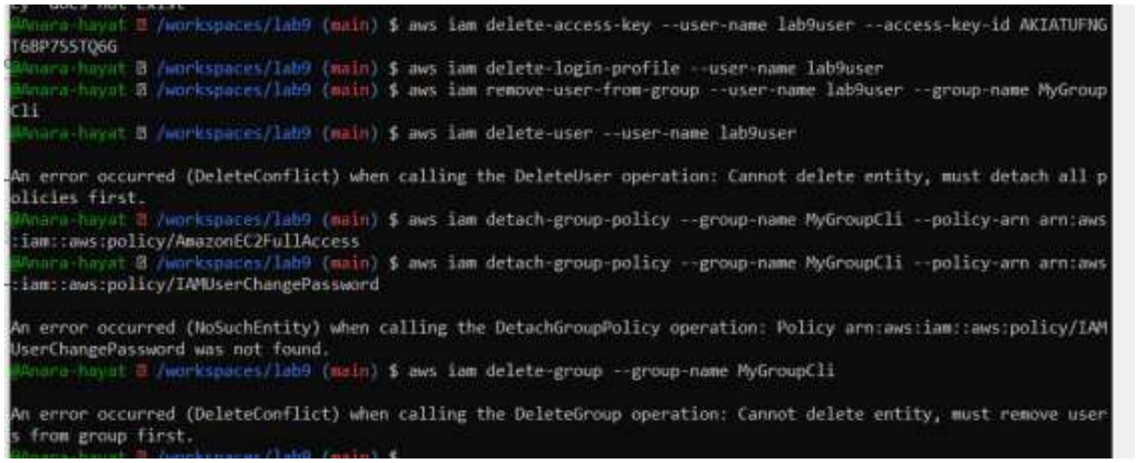
```
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-snapshots \
>   --owner-ids self \
>   --query "Snapshots[*].[SnapshotId,VolumeId,StartTime]" \
>   --output table
@Anara-hayat ➜ /workspaces/lab9 (main) $ aws ec2 describe-volumes --query "Volumes[*].[VolumeId,State]" --output
table
@Anara-hayat ➜ /workspaces/lab9 (main) $ _
```

3. **Delete security group and key pair:**



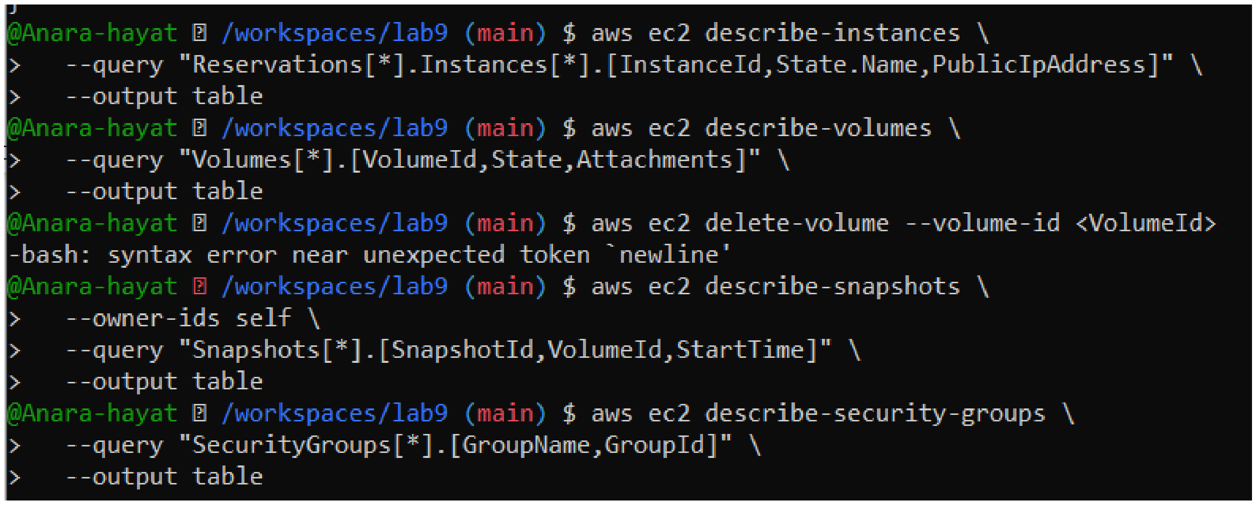
```
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 delete-security-group --group-id sg-08f44eb55e6ef2238
{
    "Return": true,
    "GroupId": "sg-08f44eb55e6ef2238"
}
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 delete-key-pair --key-name MyED25519Key
{
    "Return": true,
    "KeyPairId": "key-0c29871535e96b506"
}
@Anara-hayat ▯ /workspaces/lab9 (main) $
```

4. **Remove IAM users, access keys, groups:**

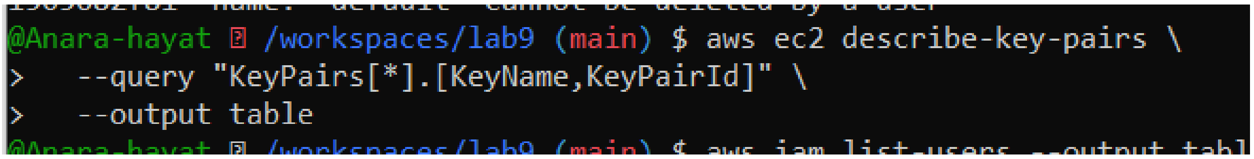


```
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam delete-access-key --user-name lab9user --access-key-id AKIATUFNG
T6BP75STQ6G
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam delete-login-profile --user-name lab9user
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam remove-user-from-group --user-name lab9user --group-name MyGroup
Cli
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam delete-user --user-name lab9user

An error occurred (DeleteConflict) when calling the DeleteUser operation: Cannot delete entity, must detach all p
olicies first.
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam detach-group-policy --group-name MyGroupCli --policy-arn arn:aws
:iam::aws:policy/AmazonEC2FullAccess
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam detach-group-policy --group-name MyGroupCli --policy-arn arn:aws
:iam::aws:policy/IAMUserChangePassword

An error occurred (NoSuchEntity) when calling the DetachGroupPolicy operation: Policy arn:aws:iam::aws:policy/IAM
UserChangePassword was not found.
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam delete-group --group-name MyGroupCli

An error occurred (DeleteConflict) when calling the DeleteGroup operation: Cannot delete entity, must remove user
s from group first.
```

5. **Final verification (billing/resource groups):**



```
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 describe-instances \
>   --query "Reservations[*].Instances[*].[InstanceId,State.Name,PublicIpAddress]" \
>   --output table
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 describe-volumes \
>   --query "Volumes[*].[VolumeId,State,Attachments]" \
>   --output table
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 delete-volume --volume-id <VolumeId>
-bash: syntax error near unexpected token `newline'
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 describe-snapshots \
>   --owner-ids self \
>   --query "Snapshots[*].[SnapshotId,VolumeId,StartTime]" \
>   --output table
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 describe-security-groups \
>   --query "SecurityGroups[*].[GroupName,GroupId]" \
>   --output table
```



```
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws ec2 describe-key-pairs \
>   --query "KeyPairs[*].[KeyName,KeyPairId]" \
>   --output table
@Anara-hayat ▯ /workspaces/lab9 (main) $ aws iam list-users --output tabl
```