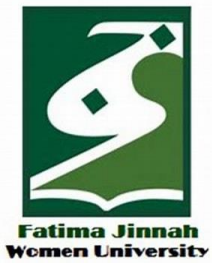
CLOUD COMPUTING

LAB 09



Submitted To:

Engr. Shoaib

Submitted By:

Komal Kashif

BSE V-A

2023-BSE-031

Task 1 — GitHub CLI, Codespace setup and authentication

1. (Local desktop) Install GitHub CLI: **`winget install --id GitHub.cli`**

```
C:\Users\Dell>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 prope
rly (ex. "US").

Do you agree to all the source agreements terms?
[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
```

2. Authenticate GH CLI for Codespaces: **`gh auth login -s codespace`**

```
C:\Users\Dell>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 KomalKashif
```

3. List available Codespaces: **`gh codespace list`**

```
C:\Users\Dell>gh codespace list
NAME                              DISPLAY NAME          REPOSITORY                BRANCH  STATE     CREATED AT
fantastic-space-lamp-r45gx77wv...  fantastic space lamp  KomalKashif/CC-KomalKas...  main    Shutdown  about 7 hours ago
```

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

1. Download and install AWS CLI:

```
  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/licenses/AUTHORS.rst
  inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/LICENSE
  inflating: aws/dist/wheel-0.45.1.dist-info/METADATA
  inflating: aws/dist/wheel-0.45.1.dist-info/WHEEL
  inflating: aws/dist/wheel-0.45.1.dist-info/LICENSE.txt
  inflating: aws/dist/wheel-0.45.1.dist-info/entry_points.txt
  inflating: aws/dist/wheel-0.45.1.dist-info/REQUESTED
  inflating: aws/dist/wheel-0.45.1.dist-info/RECORD
  inflating: aws/dist/wheel-0.45.1.dist-info/direct_url.json
  inflating: aws/dist/wheel-0.45.1.dist-info/INSTALLER
You can now run: /usr/local/bin/aws --version
```

2. Verify installation:

```
● @KomalKashif ➜/workspaces/Lab09 (main) $ aws --version
  aws-cli/2.33.6 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):

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ aws configure
AWS Access Key ID [****************AIUY]: AKIA3TFVF2NR3Z7GAIUY
AWS Secret Access Key [None]: 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
Default region name [None]: me-central-1
Default output format [me-central-1]: json
```

4. Verify credentials/config files:

```
@KomalKashif ➜ /workspaces/Lab09 (main) $  cat ~/.aws/credentials
cat ~/.aws/config
[default]
aws_access_key_id = AKIA3TFVF2NR3Z7GAIUY
aws_secret_access_key = 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
[default]
output = json
region = me-central-1
```

5. Verify connectivity

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ aws sts get-caller-identity
{
    "UserId": "AIDA3TFVF2NR5PGSDGMJQ",
    "Account": "797096399715",
    "Arn": "arn:aws:iam::797096399715:user/KomalKashif"
}
```

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

1. Create a security group

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ aws ec2 create-security-group \
  --group-name MySecurityGroup \
  --description "My Security Group" \
  --vpc-id vpc-06d4de56607216514
{
    "GroupId": "sg-0eeb09431d70b795b",
    "SecurityGroupArn": "arn:aws:ec2:me-central-1:797096399715:security-group/sg-
0eeb09431d70b795b"
}
```

2. Inspect the security group

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ aws ec2 describe-security-groups \
  --group-ids sg-0eeb09431d70b795b
{
    "SecurityGroups": [
        {
            "GroupId": "sg-0eeb09431d70b795b",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-06d4de56607216514",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:797096399715:security-g
roup/sg-0eeb09431d70b795b",
            "OwnerId": "797096399715",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
            "IpPermissions": []
        }
    ]
```

3. Get your Codespace public IP

```
@KomalKashif ➜/workspaces/Lab09 (main) $ curl icanhazip.com
4.240.18.229
```

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

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 authorize-security-group-ingres
s \
  --group-id sg-0eeb09431d70b795b \
  --protocol tcp \
  --port 22 \
  --cidr 4.240.18.229/32
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0375d84b0369190d1",
            "GroupId": "sg-0eeb09431d70b795b",
            "GroupOwnerId": "797096399715",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "4.240.18.229/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:me-central-1:797096399715:securi
ty-group-rule/sgr-0375d84b0369190d1"
        }
    ]
}
```

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

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 authorize-security-group-ingres
s \
  --group-id 'sg-0eeb09431d70b795b' \
  --ip-permissions '{"FromPort":80,"ToPort":80,"IpProtocol":"tcp","IpRanges":[{"C
idrIp":"4.240.18.229/32"}]}'
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0abc69cda5dd856f9",
            "GroupId": "sg-0eeb09431d70b795b",
            "GroupOwnerId": "797096399715",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "4.240.18.229/32",
            "SecurityGroupRuleArn": "arn:aws:ec2:me-central-1:797096399715:securi
ty-group-rule/sgr-0abc69cda5dd856f9"
        }
    ]
}
```

6. Verify both ingress rules are present:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 describe-security-groups --grou
p-ids sg-0eeb09431d70b795b
{
    "SecurityGroups": [
        {
            "GroupId": "sg-0eeb09431d70b795b",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "UserIdGroupPairs": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": []
                }
            ],
            "VpcId": "vpc-06d4de56607216514",
            "SecurityGroupArn": "arn:aws:ec2:me-central-1:797096399715:security-g
roup/sg-0eeb09431d70b795b",
            "OwnerId": "797096399715",
            "GroupName": "MySecurityGroup",
            "Description": "My Security Group",
            "IpPermissions": [
                {
                  "IpProtocol": "tcp",
                  "FromPort": 80,
                  "ToPort": 80,
```

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:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 create-key-pair \
  --key-name MyED25519Key \
  --key-type ed25519 \
  --key-format pem \
  --query 'KeyMaterial' \
  --output text > MyED25519Key.pem
@KomalKashif ➜/workspaces/Lab09 (main) $ ls -l MyED25519Key.pem
-rw-rw-rw- 1 codespace codespace 388 Jan 30 17:55 MyED25519Key.pem
```

2. View created key pairs:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 describe-key-pairs
{
    "KeyPairs": [
        {
            "KeyPairId": "key-027ed0723b570c01b",
            "KeyType": "ed25519",
            "Tags": [],
            "CreateTime": "2026-01-30T17:55:49.265000+00:00",
            "KeyName": "MyED25519Key",
            "KeyFingerprint": "BlfQWNIUjonERh4TLIT8yduZuq/MIJPAO27gAh+wmgU="
        },
        {
            "KeyPairId": "key-01864027c44776d7f",
            "KeyType": "ed25519",
            "Tags": [],
            "CreateTime": "2026-01-07T10:07:27.478000+00:00",
            "KeyName": "Lab8Key",
            "KeyFingerprint": "bV3mdZ3WQf0qY0L6p4QOXEED0d4eGCGVtoIxOfJ6eRU="
        }
    ]
}
```

3. Delete key pair:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 delete-key-pair --key-name MyED
25519Key
{
    "Return": true,
    "KeyPairId": "key-027ed0723b570c01b"
}
```

4. Launch an EC2 instance

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 run-instances \
  --image-id ami-05e66df2bafcb7dea \
  --count 1 \
  --instance-type t3.micro \
  --key-name MyED25519Key \
  --security-group-ids sg-0d71081b3fc2ef953 \
  --subnet-id subnet-00700425219acc229 \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=MyServer}]"
{
    "ReservationId": "r-03ddad9dfdee8e054",
    "OwnerId": "797096399715",
    "Groups": [],
    "Instances": [
        {
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "db91927f-a427-4984-bb78-2070bcac7ba0",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2026-01-30T18:21:50+00:00",
                        "AttachmentId": "eni-attach-060e954bb8ffe1400",
```

5. Get the public IP address of your instance:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 describe-instances \
  --query "Reservations[*].Instances[*].[InstanceId,PublicIpAddress]" \
  --output table
-------------------------------------
|          DescribeInstances          |
+---------------------+-------------+
|  i-0e362e092e9a63d03 |  3.28.56.89  |
|  i-05308635db300636a |  None        |
|  i-0ecaade95445b1294 |  None        |
|  i-054588957c7e65052 |  None        |
+---------------------+-------------+
```

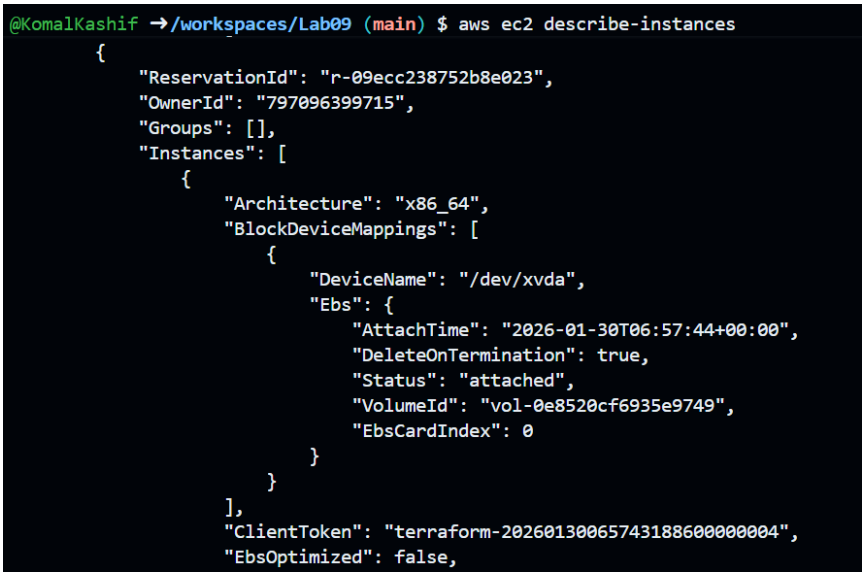
Task 5 — Understand AWS describe-* commands

1. aws ec2 describe-security-groups

2. aws ec2 describe-vpcs

3. aws ec2 describe-subnets

4. aws ec2 describe-instances



5. aws ec2 describe-regions

6. aws ec2 describe-availability-zones

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

1. Create group:

2. Get group details:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam get-group --group-name MyGroupCli
{
    "Users": [],
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPA3TFVF2NR2JBG4IKBB",
        "Arn": "arn:aws:iam::797096399715:group/MyGroupCli",
        "CreateDate": "2026-01-30T18:39:35+00:00"
    }
}
```

3. Create user:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam create-user --user-name MyUserCli
{
    "User": {
        "Path": "/",
        "UserName": "MyUserCli",
        "UserId": "AIDA3TFVF2NRTAFEDXE7P",
        "Arn": "arn:aws:iam::797096399715:user/MyUserCli",
        "CreateDate": "2026-01-30T18:41:28+00:00"
    }
}
```

4. Get user details:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam get-user --user-name MyUserCli
{
    "User": {
        "Path": "/",
        "UserName": "MyUserCli",
        "UserId": "AIDA3TFVF2NRTAFEDXE7P",
        "Arn": "arn:aws:iam::797096399715:user/MyUserCli",
        "CreateDate": "2026-01-30T18:41:28+00:00"
    }
}
```

5. Add user to group:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam add-user-to-group --user-name MyUserCli --group-name MyGroupCli
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam get-group --group-name MyGroupCli
{
    "Users": [
        {
            "Path": "/",
            "UserName": "MyUserCli",
            "UserId": "AIDA3TFVF2NRTAFEDXE7P",
            "Arn": "arn:aws:iam::797096399715:user/MyUserCli",
            "CreateDate": "2026-01-30T18:41:28+00:00"
        }
    ],
    "Group": {
        "Path": "/",
        "GroupName": "MyGroupCli",
        "GroupId": "AGPA3TFVF2NR2JBG4IKBB",
        "Arn": "arn:aws:iam::797096399715:group/MyGroupCli",
        "CreateDate": "2026-01-30T18:39:35+00:00"
    }
}
```

6. List policies that mention EC2:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam list-policies \
  --query "Policies[?contains(PolicyName, 'EC2')].{Name:PolicyName}" \
  --output text
AmazonEC2ContainerServiceAutoscaleRole
AmazonEC2SpotFleetAutoscaleRole
AWSElasticBeanstalkCustomPlatformforEC2Role
AmazonEC2ContainerServiceEventsRole
AmazonEC2SpotFleetTaggingRole
AWSEC2SpotServiceRolePolicy
AWSServiceRoleForEC2ScheduledInstances
AWSEC2SpotFleetServiceRolePolicy
AWSApplicationAutoscalingEC2SpotFleetRequestPolicy
AWSEC2FleetServiceRolePolicy
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam list-policies --query 'Policies[?PolicyName==`AmazonEC2FullAccess`]
.{Name:PolicyName, ARN:Arn}' --output table
-------------------------------------------------------------------------
|                           ListPolicies                                |
+-----------------------------------------------+-----------------------+
|                      ARN                      |         Name          |
+-----------------------------------------------+-----------------------+
|  arn:aws:iam::aws:policy/AmazonEC2FullAccess  |  AmazonEC2FullAccess  |
+-----------------------------------------------+-----------------------+
```

7. Create a console login profile for the user:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam create-login-profile \
   --user-name MyUserCli \
   --password "MySecureP@ssw0rd123" \
   --password-reset-required
{
    "LoginProfile": {
        "UserName": "MyUserCli",
        "CreateDate": "2026-01-30T18:48:09+00:00",
        "PasswordResetRequired": true
    }
}
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam attach-group-policy --group-name MyGroupCli --policy-arn arn:aws:ia
m::aws:policy/IAMUserChangePassword
```

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

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam create-access-key --user-name MyUserCli
{
    "AccessKey": {
        "UserName": "MyUserCli",
        "AccessKeyId": "AKIA3TFVF2NR52DRWKXA",
        "Status": "Active",
        "SecretAccessKey": "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",
        "CreateDate": "2026-01-30T18:51:07+00:00"
    }
}
@KomalKashif ➜/workspaces/Lab09 (main) $ aws iam list-access-keys --user-name MyUserCli
{
    "AccessKeyMetadata": [
        {
            "UserName": "MyUserCli",
            "AccessKeyId": "AKIA3TFVF2NR52DRWKXA",
            "Status": "Active",
            "CreateDate": "2026-01-30T18:51:07+00:00"
        }
    ]
}
```

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

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ export AWS_ACCESS_KEY_ID=AKIA3TFVF2NR52DRWKXA
export AWS_SECRET_ACCESS_KEY=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
printenv | grep AWS_
aws iam get-user --user-name MyUserCli
AWS_SECRET_ACCESS_KEY=XAANyzNYiZUu7u217vGd8y/rJ3YnyTdjzhQb7XmV
AWS_ACCESS_KEY_ID=AKIA3TFVF2NR52DRWKXA

An error occurred (AccessDenied) when calling the GetUser operation: User: arn:aws:iam::797096399715:user/MyUserCli i
s not authorized to perform: iam:GetUser on resource: user MyUserCli because no identity-based policy allows the iam:
GetUser action
```

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

1. Filter by instance

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ aws ec2 describe-instances \
  --filters "Name=instance-type,Values=t3.micro" \
  --query "Reservations[].Instances[].InstanceId" \
  --output table
-------------------------
|   DescribeInstances   |
+-----------------------+
|  i-0e362e092e9a63d03  |
|  i-05308635db300636a  |
|  i-0ecaade95445b1294  |
|  i-054588957c7e65052  |
+-----------------------+
```

2. Filter by subnet

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ aws ec2 describe-instances \
  --filters "Name=subnet-id,Values=subnet-00700425219acc229" \
  --query "Reservations[*].Instances[*].[InstanceId,State.Name,PublicIpAddress]" \
  --output table
|                DescribeInstances                |
+---------------------+---------+-----------------+
|  i-05308635db300636a |  running |  158.252.137.60  |
|  i-0ecaade95445b1294 |  running |  None            |
|  i-054588957c7e65052 |  running |  None            |
+---------------------+---------+-----------------+
```

3. Filter by vpc

```
@KomalKashif ➜ /workspaces/Lab09 (main) $ aws ec2 describe-instances \
  --filters "Name=vpc-id,Values=vpc-0381f9009778d1432" \
  --query "Reservations[*].Instances[*].[InstanceId,State.Name,PublicIpAddress]" \
  --output table
-------------------------------------------------------
|                DescribeInstances                    |
+---------------------+---------+-------------------+
|  i-0e362e092e9a63d03 |  running |  3.28.56.89       |
|  i-05308635db300636a |  running |  158.252.137.60   |
|  i-0ecaade95445b1294 |  running |  None             |
|  i-054588957c7e65052 |  running |  None             |
+---------------------+---------+-------------------+
```

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

1. Query table instances name

```
@KomalKashif ➜/workspaces/Lab09 (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-054588957c7e65052 |  None |  MyServer    |
+----------------------+-------+--------------+
```

2. Query table instances state

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 describe-instances \
  --query "Reservations[*].Instances[*].[InstanceId,State.Name]" \
  --output table
-------------------------------------
|          DescribeInstances        |
+----------------------+------------+
|  i-0e362e092e9a63d03 |  running   |
|  i-05308635db300636a |  running   |
|  i-0ecaade95445b1294 |  running   |
|  i-054588957c7e65052 |  running   |
+----------------------+------------+
```

3. Query table instances type

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 describe-instances \
  --query "Reservations[*].Instances[*].[InstanceId,InstanceType,Placement.AvailabilityZone]" \
  --output table
-------------------------------------------------------
|                  DescribeInstances                  |
+----------------------+------------+-----------------+
|  i-0e362e092e9a63d03 |  t3.micro  |  me-central-1c  |
|  i-05308635db300636a |  t3.micro  |  me-central-1c  |
|  i-0ecaade95445b1294 |  t3.micro  |  me-central-1c  |
|  i-054588957c7e65052 |  t3.micro  |  me-central-1c  |
+----------------------+------------+-----------------+
```

Clean-up:

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 terminate-instances --instance-ids i-054588957c7e65052
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-054588957c7e65052",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

```
@KomalKashif ➜/workspaces/Lab09 (main) $ aws ec2 delete-security-group --group-id sg-0eeb09431d70b795b
aws ec2 delete-key-pair --key-name MyED25519Key
{
    "Return": true,
    "GroupId": "sg-0eeb09431d70b795b"
}
{
    "Return": true
}
```