The **AWS Command Line Interface (CLI)** is a unified tool that allows you to manage your AWS services using commands in your command-line shell. It is a powerful alternative to using the AWS Management Console, providing automation capabilities, ease of scripting, and faster operations.

1. Why Use AWS CLI Instead of the Console?

- **Automation**: AWS CLI enables you to automate tasks. With scripts, you can create, manage, and delete resources in bulk without manually interacting with the console.
- **Speed**: The CLI is typically faster for making changes than the console, especially when you're working with large numbers of resources.
- **Consistency**: The CLI ensures repeatability and consistency when creating or modifying infrastructure, as you can version control your commands and scripts.
- **Flexibility**: The CLI provides more advanced features than the console, such as creating complex multi-step workflows and integrating with other tools.
- **Access**: You can access AWS services through the CLI from any machine with the AWS CLI installed, whereas the console requires a web browser and is often tied to your AWS user session.

2. How AWS CLI Works (API Calls) [Application Programming Interface]

The CLI interacts with AWS services by making **API calls** to AWS endpoints. Here's a basic flow:

- **Command Input**: You issue a command through the CLI, specifying the AWS service, operation, and parameters.
- **API Request**: The CLI translates your command into an **API request** that is sent to the corresponding AWS service.
- **AWS Service**: The AWS service processes the API request, performs the required operation (e.g., create a VPC, launch an EC2 instance), and returns the response to the CLI.

- **Response**: The CLI displays the response (e.g., resource ID, status, or errors) to the user.

The AWS CLI abstracts the API calls, so you don't need to manually interact with the low-level HTTP API requests.

3. Setting Up AWS CLI (Configuration)

To use the AWS CLI, you need to **configure it** with your AWS credentials (Access Key ID and Secret Access Key) and other settings. The configuration process typically involves the following steps:

Install AWS CLI

Windows: Download the installer from AWS CLI website.

AWS CLI configure commands

VPC

1.create a vpc

```
aws ec2 create-vpc –cidr-block 10.0.0.0/16
```

2.delete a vpc

```
Aws ec2 delete-vpc –vpc-id <id>
```

3.modify or add a name to vpc

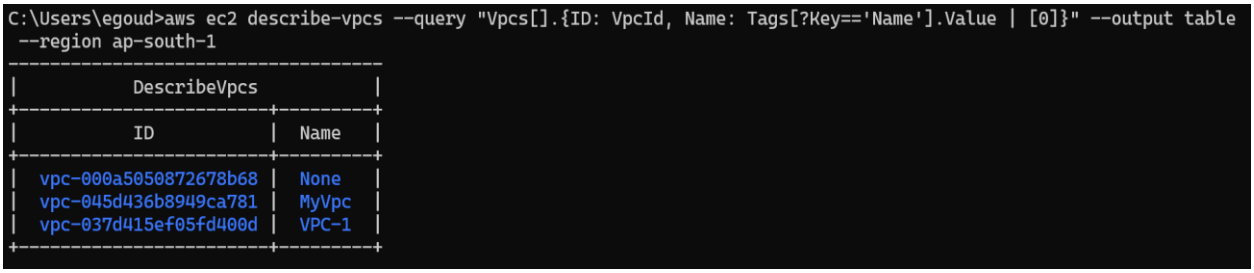
```
Aws ec2 create-tags –resource <id> --tags Key=Name,Value=myVpcName
```

4.to list the vpc’s

```
Aws ec2 describe-vpc’s
```

5.to display vpc’s in table format with only id and name

```
aws ec2 describe-vpcs --query "Vpcs[].{ID: VpcId, Name: Tags[?Key=='Name'].Value | [0]}" --output table --region ap-south-1
```



SUBNETS:-

1.create a subnet

Aws ec2 create-subnet 10.0.0.0/24 –availability-zone-ap-south-1a

Aws ec2 create-subnet 10.0.1.0/24 –availability-zone-ap-south-1b

Aws ec2 create-subnet 10.0.2.0/24 –availability-zone-ap-south-1b

Aws ec2 create-subnet 10.0.3.0/24 –availability-zone-ap-south-1c

2. modify the subnet name

Aws ec2 create-tags –resource <id> --tags Key=Name,Value=subnetname

3.display the subnets with filters

Aws ec2 describes-subnets –filters “Name=vpc-id,Values=<id>”

4.To display the subnet id and names

aws ec2 describe-subnets --query "Subnets[].{ID: SubnetId, Name: Tags[?Key=='Name'].Value | [0]}" --output table --region ap-south-1

```
C:\Users\egoud>aws ec2 describe-subnets --query "Subnets[].{ID: SubnetId, Name: Tags[?Key=='Name'].Value | [0]}" --out
put table --region ap-south-1
-------------------------------------------
|              DescribeSubnets              |
+--------------------------+----------------+
|            ID            |     Name       |
+--------------------------+----------------+
|  subnet-0f3f175d4d4bfc39e |  None          |
|  subnet-02e3bb6eae145efd6 |  prv-2         |
|  subnet-05ddd6ed89ce58cb0 |  publicsub2    |
|  subnet-03c2321d9daa65c90 |  pub-1         |
|  subnet-04dce6eb24c24dc46 |  publicsub1    |
|  subnet-093892633986ba1a0 |  None          |
|  subnet-0c04d02138c01c41f |  None          |
|  subnet-023104549322dfc42 |  privatecsub1  |
|  subnet-022af20df56ae72f9 |  privatecsub2  |
|  subnet-09fb88f4d20317996 |  prv-1         |
|  subnet-0c9c0be268718c966 |  pub-2         |
+--------------------------+----------------+
```

5. to delete the subnets

Aws ec2 delete-subnet –subnet-id <id>

IGW

1.create a IGW

Aws ec2 create-internet-gateway

2.modify the name

Aws ec2 create-tags –resource <ID> --tags Key=Name,Value=<name>

3.attach to the IGW to VPC

Aws ec2 attach-internet-gateway –vpc-id <vpc-id> --internet-gateway <id>

4.To detach use detach

5.to display the IGW's

Aws ec2 describe-internet-gateways

Aws ec2 describe-internet-gateways --filters "Name=attachment.vpc-id,Values=<id>"

aws ec2 describe-internet-gateways --query "InternetGateways[].{ID: InternetGatewayId, Name: Tags[?Key=='Name'].Value | [0], Attachment: Attachments[0].VpcId}" --output table --region ap-south-1

```
C:\Users\egoud>aws ec2 describe-internet-gateways --query "InternetGateways[].{ID: InternetGatewayId, Name: Tags[?Key=
='Name'].Value | [0], Attachment: Attachments[0].VpcId}" --output table --region ap-south-1
-------------------------------------------------------------
|                 DescribeInternetGateways                  |
+-----------------------+---------------------+-------------+
|      Attachment       |          ID         |    Name     |
+-----------------------+---------------------+-------------+
|  vpc-037d415ef05fd400d |  igw-089c5fedafb25464e  |  IGW_1   |
|  vpc-000a5050872678b68 |  igw-0a0df52f215fb18da  |  None    |
|  vpc-045d436b8949ca781 |  igw-0cbfae84787a8df29  |  IGW_CLI |
+-----------------------+---------------------+-------------+
```

6.To delete the IGW

Aws ec2 delete-internet-gateway –internet-gateway <id>

ROUTE-TABLES -Public

1.Create the Public route table

Aws ec2 create-route-table ---vpc-id <id>

2.Add public subnets in route table

Aws ec2 attach-route-table –route-table-id <id> --subnet-id <id>

Aws ec2 attach-route-table –route-table-id <id> --subnet-id <id>

(adding the 2 subnets for public)

3.To detach use detach in place of attach

4. To display the route-tables

Aws ec2 describe-route-tables

Aws ec2 describe-route-tables –filters “Name=vpc-id,Value=<vpcid>”

5.To display specific route id , attached vpc , subnet id’s

aws ec2 describe-route-tables --route-table-ids <your-route-table-id> --query "RouteTables[].{ID: RouteTableId, Name: Tags[?Key=='Name'].Value | [0], VPC: VpcId, SubnetsAttached: length(Associations), Subnets: Associations[].SubnetId}" --output table --region ap-south-1

6. To add name to route table or to modify

Aws ec2 create-tags –resource <id> --tags Key=Name,Value=<route name>

7. To attach the **IGW** to route table

Aws ec2 create-route –route-table-id <id> --destination-cidr-block 0.0.0.0/0 --gateway-id <id>

ROUTE-TABLES -Private

1.create a route table for Private

Aws ec2 create-route-table ---vpc-id <id>

2. To add name to route table or to modify

Aws ec2 create-tags –resource <id> --tags Key=Name,Value=<route name>

3. Add public subnets in route table

Aws ec2 attach-route-table –route-table-id <id> --subnet-id <id>

Aws ec2 attach-route-table –route-table-id <id> --subnet-id <id>

(adding the 2 subnets for Private)

4.To detach use detach in place of attach

5. To display the route-tables

Aws ec2 describe-route-tables

Aws ec2 describe-route-tables –filters “Name=vpc-id,Value=<vpcid>”

6.To display specific route id , attached vpc , subnet id’s

aws ec2 describe-route-tables --route-table-ids <your-route-table-id> --query "RouteTables[].{ID: RouteTableId, Name: Tags[?Key=='Name'].Value | [0], VPC: VpcId, SubnetsAttached: length(Associations), Subnets: Associations[].SubnetId}" --output table --region ap-south-1

```
C:\Users\egoud>aws ec2 describe-route-tables --route-table-ids rtb-04d9d9df21de0799e --query "RouteTables[].{ID: RouteTableId, Name: Tags[?Key=='Name'].Value | [0], VPC: VpcId, SubnetsAtta
ched: length(Associations), Subnets: Associations[].SubnetId}" --output table --region ap-south-1
-----------------------------------------
|           DescribeRouteTables           |
+-----------------+-----------------------+
|  ID             |  rtb-04d9d9df21de0799e  |
|  Name           |  privaterout            |
|  SubnetsAttached|  2                      |
|  VPC            |  vpc-045d436b8949ca781  |
+-----------------+-----------------------+
||               Subnets               ||
|+-------------------------------------+|
||  subnet-023104549322dfc42           ||
||  subnet-022af20df56ae72f9           ||
|+-------------------------------------+|
```

NAT-GW

1.Allocate an elastic-ip

Aws ec2 allocate-address

2.Create a NAT-GW

Aws ec2 create-nat-gateway –subnet-id <id> -allocation-id <id>

3.Attach the NAT-GW to PUBLIC_ROUTE_TABLE

Aws ec2 create-route –route-table-id <id> --destination-cidr-block 0.0.0.0/0

--nat-gateway-id<id>

AMI

1.To display the instances

aws ec2 describe-instances --query "Reservations[].Instances[].{ID: InstanceId, Name: Tags[?Key=='Name']|[0].Value}" --output table

```
C:\Users\egoud>aws ec2 describe-instances --query "Reservations[].Instances[].{ID: InstanceId, Name: Tags[?Key=='Name']|[0].Value}" --output tabl
e
-----------------------------------------------
|              DescribeInstances              |
+----------------------+----------------------+
|          ID          |        Name          |
+----------------------+----------------------+
|  i-0797f46e7c8e1d9a8 |  Fundoo-Database     |
|  i-0cefcfe3b4e9f632d |  Database-instance   |
|  i-0f84c1cde578cfa33 |  Fundoo-Frontend     |
|  i-04d37210f9498163e |  frontend-instance   |
|  i-00ee9de2f513f4e19 |  Fundoo-Backend      |
|  i-0d601582019ceb022 |  Backend-instance    |
+----------------------+----------------------+
```

2.Create a AMI of an existing instances(database,backend,frontend)

Aws ec2 create-image –instance-id <id> --name “<name of ami>” –no-reboot

3.To display the AMI id’s

Aws ec2 describe-images –owners self

aws ec2 describe-images --owners self --query "Images[].{ID: ImageId, Name: Name}" --output table --region <your-region>

```
C:\Users\egoud>aws ec2 describe-images --owners self --query "Images[].{ID: ImageId, Name: Name}" --output table
-----------------------------------------
|            DescribeImages             |
+-----------------------+---------------+
|          ID           |     Name      |
+-----------------------+---------------+
|  ami-0c4869f87115eadd8 |  frontend-AMI  |
|  ami-0ed54cfba4cf96d18 |  Backend-AMI   |
|  ami-0a80a99bcbb3d50d3 |  Database-AMI  |
+-----------------------+---------------+
```

INSTANCE

1.Start the instances in your vpc using the AMI (database,backend,frontend)

Aws ec2 run-instances --image-id <ami-id> --instance-type t3.micro

--subnet-id <id> --key-name <existing key>

(select the AMI , key which is already existed)

2.set a name for instances which you are created

Aws ec2 create-tags –resource <instance-id> --tags Key=Name,Value=<name>

3.To display the instances

Aws ec2 describe-instances

aws ec2 describe-instances --query "Reservations[].Instances[].{ID: InstanceId, Name: Tags[?Key=='Name']|[0].Value}" --output table

```
C:\Users\egoud>aws ec2 describe-instances --query "Reservations[].Instances[].{ID: InstanceId, Name: Tags[?Key=='Name']|[0].Value}" --output tabl
e
-----------------------------------------------
|             DescribeInstances              |
+---------------------+----------------------+
|          ID         |         Name         |
+---------------------+----------------------+
|  i-0797f46e7c8e1d9a8 |  Fundoo-Database     |
|  i-0cefcfe3b4e9f632d |  Database-instance   |
|  i-0f84c1cde578cfa33 |  Fundoo-Frontend     |
|  i-04d37210f9498163e |  frontend-instance   |
|  i-00ee9de2f513f4e19 |  Fundoo-Backend      |
|  i-0d601582019ceb022 |  Backend-instance    |
+---------------------+----------------------+
```

SECURITY GROUPS

5. Create a security group for the instances (database,backend,frontend)4

Aws ec2 create-security-group --group-name <sg-name> --description "----------" --vpc-id <vpc -id> --region <region>

6.To display the groups

Aws ec2 describe-security-groups

Aws ec2 describe-security-groups –group-id <sg-id>

7.To display with filters

C:\Users\egoud>aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-045d436b8949ca781" --query "SecurityGroups[].{ID: GroupId, Name: GroupName}" --output table --region ap-south-1

```
C:\Users\egoud>aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-045d436b8949ca781" --query "SecurityGroups[].{ID: GroupId, Name
: GroupName}" --output table --region ap-south-1
-------------------------------------------
|          DescribeSecurityGroups          |
+----------------------+-------------------+
|          ID          |       Name        |
+----------------------+-------------------+
|  sg-0621e6e2abdd0f463 |  frontend-sg-cli  |
|  sg-0376e9f63060ae3c0 |  Backend-sg-cli   |
|  sg-039beaa0526a1b8a2 |  database-sg-cli  |
|  sg-014e03e2f1fa9dfc5 |  default          |
+----------------------+-------------------+
```

8.create the protocols which you want to allow to instances(data,back,front)

Aws ec2 authorize-security-group-ingress --group-id <sg-id> --protocol tcp --port <port-no> --cidr <ex.0.0.0.0/0>

(create the protocol based on need for instance)

9.Attach the security group to your instances

Aws ec2 modify-instance-attribute --instance-id <id> --groups <sg-id>

(these should be created and attached to all three instances database,backend,frontend)

TARGET -GROUPS

1.create a Traget groups(backend & frontend)

C:\Users\egoud>aws elbv2 create-target-group --name Backend-CLI-TG --protocol HTTP --port 8000 --vpc-id vpc-045d436b8949ca781 --target-type instance --ip-address-type ipv4 --health-check-path /home/

2.To display the target groups

aws elbv2 describe-target-groups

aws elbv2 describe-target-groups --query "TargetGroups[*].[TargetGroupName,TargetGroupArn]" --output table

```
C:\Users\egoud>aws elbv2 describe-target-groups --query "TargetGroups[*].[TargetGroupName,TargetGroupArn]" --output table
-------------------------------------------------------------------------------------------------------------
|                                          DescribeTargetGroups                                             |
+----------------+------------------------------------------------------------------------------------------+
|  Backend-CLI-TG|  arn:aws:elasticloadbalancing:ap-south-1:796973489696:targetgroup/Backend-CLI-TG/17ac7e63902a0818  |
|  Backend-TG    |  arn:aws:elasticloadbalancing:ap-south-1:796973489696:targetgroup/Backend-TG/bfbe15067a372b69      |
|  FRONTEND-TG   |  arn:aws:elasticloadbalancing:ap-south-1:796973489696:targetgroup/FRONTEND-TG/3d26abcc675278d3     |
+----------------+------------------------------------------------------------------------------------------+
```

3.Register the targets in target groups

aws elbv2 register-targets --target-group-arn arn:aws:elasticloadbalancing:ap-south-1:796973489696:targetgroup/Backend-CLI-TG/17ac7e63902a0818 --targets Id=i-0d601582019ceb022

4. health check and instances

aws elbv2 describe-target-health –target-group-arn <id>

aws elbv2 describe-target-health --target-group-arn arn:aws:elasticloadbalancing:ap-south-1:796973489696:targetgroup/Backend-CLI-TG/17ac7e63902a0818 --query "TargetHealthDescriptions[*].[Target.Id, TargetHealth.State]" --output table

```
C:\Users\egoud>aws elbv2 describe-target-health --target-group-arn arn:aws:elasticloadbalancing:ap-south-1:796973489696:targetgroup/Backend-CLI-T
G/17ac7e63902a0818 --query "TargetHealthDescriptions[*].[Target.Id, TargetHealth.State]" --output table
-----------------------------------
|      DescribeTargetHealth       |
+---------------------+-----------+
|  i-0d601582019ceb022 |  unused  |
+---------------------+-----------+
```

5.To derigester the targets

aws elbv2 deregister-targets –target-group-arn <arn> --targets Id=<id>

LAUNCH-TEMPLATE

1.Create a launch template for (backend and frontend)

```
aws ec2 create-launch-template \
   --launch-template-name MyLaunchTemplate \
   --launch-template-data '{
      "ImageId": "ami-xxxxxxxx",
      "InstanceType": "t2.micro",
      "KeyName": "your-key-pair-name",
      "SecurityGroupIds": ["sg-xxxxxxxx"]
   }'
```

```
aws ec2 create-launch-template --launch-template-name Backend-LT --launch-template-data "{\"ImageId\": \"ami-0ed54cfba4cf96d18\", \"InstanceType\": \"t2.micro\", \"KeyName\": \"fundoo-backend-key\", \"SecurityGroupIds\": [\"sg-0376e9f63060ae3c0\"]}"
```

```
aws ec2 create-launch-template --launch-template-name Frontend-LT --launch-template-data "{\"ImageId\": \"ami-0c4869f87115eadd8\", \"InstanceType\": \"t2.micro\", \"KeyName\": \"fundoo-frontend-key\", \"SecurityGroupIds\": [\"sg-0621e6e2abdd0f463\"]}"
```

2.To display the attributes of LT

aws ec2 describe-launch-templates --launch-template-id <id>

aws ec2 describe-launch-templates –query
“LaunchTemplate[*].[LaunchTemplateName,LaunchTemplateId]” --output table

Auto Scaling Group

1. Create an Auto Scaling Group:

aws autoscaling create-auto-scaling-group --auto-scaling-group-name <your-asg-name> --launch-template "LaunchTemplateName=<your-launch-template-name>,Version=<your-template-version>" --min-size 1 --max-size 3 --desired-capacity 2 --vpc-zone-identifier <your-subnet-id> --region <your-region>

2. Create a Scaling Policy:

aws autoscaling put-scaling-policy --auto-scaling-group-name Backend-ASG --policy-name TargetScalingPolicy --policy-type TargetTrackingScaling --target-tracking-configuration "{\"TargetValue\":70.0, \"PredefinedMetricSpecification\": {\"PredefinedMetricType\": \"ASGAverageCPUUtilization\"}, \"DisableScaleIn\": false}"

Verify:-
aws autoscaling describe-policies --auto-scaling-group-name Backend-ASG --query "ScalingPolicies[?PolicyName=='TargetScalingPolicy']" --output table

3. Attach target group to Load Balancer:

   aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name Backend-ASG --target-group-arns arn:aws:elasticloadbalancing:ap-south-1:796973489696:targetgroup/Backend-CLI-TG/17ac7e63902a0818 --region ap-south-1

4. Verify Auto Scaling Group:

aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name <your-asg-name> --region <your-region>