Lab 9: AWS CLI Automation Using CloudShell

Lab Title:

Automating AWS Resource Management with AWS CLI in CloudShell

Objective:

Learn how to use AWS Command Line Interface (CLI) via CloudShell to automate the creation, management, and deletion of AWS resources such as S3, EC2, IAM, and more.

Duration:

2 hours

Pre-requisites:

- AWS Free Tier account
- AWS CloudShell access
- Basic knowledge of Bash and AWS services

Part A: Introduction to AWS CLI & CloudShell (10 mins)

What is AWS CLI?

- A unified tool to manage AWS services via terminal commands.
- Useful for scripting and automation.

What is AWS CloudShell?

- A browser-based shell pre-authenticated with your AWS account.
- Pre-installed with AWS CLI and other tools.

Part B: Verify Environment (5 mins)

1. Open CloudShell

- Console: https://console.aws.amazon.com/cloudshell

2. Check AWS CLI Version

```
aws --version
```

3. View Current IAM Identity

```
aws sts get-caller-identity
```

Part C: Automate S3 Operations (20 mins)

1. Create an S3 Bucket

```
bucket_name=my-cli-lab-bucket-$(date +%s)
aws s3 mb s3://$bucket_name
```

2. Upload a File

```
echo "Welcome to AWS CLI Lab" > index.html
aws s3 cp index.html s3://$bucket_name
```

3. Enable Static Website Hosting

```
aws s3 website s3://$bucket_name/ --index-document index.html
```

3.1 Allow Public Access

```
aws s3api put-public-access-block --bucket $bucket_name –public-access-block-
configuration
BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPub
licBuckets=false
```

4. Make File Public (Bucket Policy)

```
cat <<EOF > bucket-policy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::$bucket_name/*"
    }
  ]
}
EOF
```

4.1 Apply bucket policy

```
aws s3api put-bucket-policy --bucket $bucket_name --policy file://bucket-
policy.json
```

5. Test Website URL

```
echo "http://$bucket_name.s3-website.$(aws configure get region).amazonaws.com"
```

Part D: Automate EC2 Instance Launch (30 mins)

1. Get Latest Amazon Linux 2 AMI

```
aws ssm get-parameters-by-path --path "/aws/service/ami-amazon-linux-latest" --
query "Parameters[?Name=='/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2'].Value" --output text
```

2. Create a Key Pair

```
aws ec2 create-key-pair --key-name cli-key --query 'KeyMaterial' --output text >
cli-key.pem

chmod 400 cli-key.pem
```

3. Create Security Group

vpc_id=$(aws ec2 describe-vpcs --query "Vpcs[0].VpcId" --output text)

```
group_id=$(aws ec2 create-security-group --group-name cli-sg --description "CLI
SG" --vpc-id $vpc_id --query 'GroupId' --output text)

aws ec2 authorize-security-group-ingress --group-id $group_id --protocol tcp --
port 22 --cidr 0.0.0.0/0
```

4. Launch Instance

```
ami_id=$(aws ssm get-parameters-by-path --path "/aws/service/ami-amazon-linux-
latest" --query "Parameters[?Name=='/aws/service/ami-amazon-linux-latest/amzn2-
ami-hvm-x86_64-gp2'].Value" --output text)
```

4.1 Get subnetID

```
subnet_id=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --
query "Subnets[0].SubnetId" --output text)
```

4.2 launch EC2 instance

```
aws ec2 run-instances --image-id $ami_id --instance-type t2.micro --key-name
cli-key --security-group-ids $group_id --subnet-id $subnet_id   --tag-
specifications 'ResourceType=instance,Tags=[{Key=Name,Value=cli-ec2}]' --count 1
```

Part E: Automation Script Example (20 mins) - optional

Create a reusable script file:

```
cat <<EOF > launch-ec2.sh
#!/bin/bash

set -e
ami_id=
$(aws ssm get-parameters-by-path --path "/aws/service/ami-amazon-linux-latest" \
--query "Parameters[?Name=='/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2'].Value" \
--output text)

aws ec2 run-instances \
  --image-id $ami_id \
  --count 1 \
  --instance-type t2.micro \
  --key-name cli-key \
  --security-group-ids $group_id \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=script-
ec2}]'
EOF


chmod +x launch-ec2.sh
```

```
./launch-ec2.sh
```

Part F: Cleanup Resources (10 mins)

```
# Terminate all instances
aws ec2 describe-instances --query "Reservations[*].Instances[*].InstanceId" --
output text | xargs aws ec2 terminate-instances --instance-ids

# Delete security group
aws ec2 delete-security-group --group-id $group_id


# Delete key pair
aws ec2 delete-key-pair --key-name cli-key

rm cli-key.pem

# Delete S3 bucket
aws s3 rb s3://$bucket_name --force
```