Install AWS CLI :

Download and run the AWS CLI MSI installer for Windows (64-bit):

<https://awscli.amazonaws.com/AWSCLIV2.msi>

To confirm installation:

Open Command Prompt (cmd) and type :

aws –version

To access CLI:

<https://docs.aws.amazon.com/cli/latest/userguide/cli-authentication-user.html>

SET 1

**Q1: Create two S3 buckets and transfer data from one S3 bucket to another using AWS CLI.**

**Solution:**

1. **Create Two S3 Buckets:**

aws s3 mb s3://source-bucket-name

aws s3 mb s3://destination-bucket-name

Replace source-bucket-name and destination-bucket-name with your unique bucket names.

1. **Upload a Sample File to the Source Bucket:**

echo "This is a sample file" > sample.txt

aws s3 cp sample.txt s3://source-bucket-name/

1. **Copy Data from Source Bucket to Destination Bucket:**

aws s3 cp s3://source-bucket-name/sample.txt s3://destination-bucket-name/

1. **Verify the Transfer:**

aws s3 ls s3://destination-bucket-name/

**Q2: Identify and list all unused Elastic IPs in your AWS account using the AWS CLI.**

**Solution:**

1. **List All Elastic IPs:**

aws ec2 describe-addresses --query 'Addresses[\*].[PublicIp,AssociationId]' --output table

1. **Filter Unused Elastic IPs (IPs Without an Association ID):**

aws ec2 describe-addresses --query 'Addresses[?AssociationId==null].[PublicIp]' --output table

This command retrieves and displays Elastic IPs that are not currently associated with any instance.

SET 2

**Q3: Identify and list all EC2 instances that are running but have no associated Elastic IP.**

**Solution:**

1. **List Running EC2 Instances:**

aws ec2 describe-instances --filters Name=instance-state-name,Values=running --query 'Reservations[\*].Instances[\*].[InstanceId, PublicIpAddress]' --output table

1. **Filter Instances Without an Elastic IP:**

aws ec2 describe-instances --filters Name=instance-state-name,Values=running --query 'Reservations[\*].Instances[\*].[InstanceId]' --output text | while read id; do

eip=$(aws ec2 describe-addresses --filters "Name=instance-id,Values=$id" --query 'Addresses[\*].PublicIp' --output text)

if [[ -z "$eip" ]]; then echo "Instance without Elastic IP: $id"; fi

done

**Q4: Find all the publicly accessible S3 buckets with the keyword "s3bucket".**

**Solution:**

1. **List All S3 Buckets:**

aws s3 ls | grep "s3bucket"

1. **Check Public Permissions for Each Bucket:**

for bucket in $(aws s3api list-buckets --query "Buckets[?contains(Name, 's3bucket')].Name" --output text); do

echo "Checking bucket: $bucket"

aws s3api get-bucket-acl --bucket $bucket

done

1. **Alternatively, Find Public Buckets Using Policy Check:**

aws s3api list-buckets --query "Buckets[\*].Name" --output text | while read bucket; do

public=$(aws s3api get-bucket-policy-status --bucket $bucket --query 'PolicyStatus.IsPublic' --output text)

if [[ "$public" == "True" ]]; then echo "Public S3 Bucket Found: $bucket"; fi

done

SET 3

**Q5: Deploy a Lambda function in the Mumbai region that gets triggered when an object is uploaded to an S3 bucket.**

**Solution:**

To deploy an AWS Lambda function triggered by an S3 upload event in the Mumbai region:

1. **Create an IAM Role with Required Permissions:**
2. aws iam create-role --role-name S3LambdaExecutionRole --assume-role-policy-document file://trust-policy.json

**trust-policy.json:**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": { "Service": "lambda.amazonaws.com" },

"Action": "sts:AssumeRole"

}

]

}

Attach required permissions:

aws iam attach-role-policy --role-name S3LambdaExecutionRole --policy-arn arn:aws:iam::aws:policy/AWSLambdaBasicExecutionRole

aws iam attach-role-policy --role-name S3LambdaExecutionRole --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

1. **Create the Lambda Function:**
   * Write a simple Python function (lambda\_function.py):
   * import json
   * def lambda\_handler(event, context):
   * print("Event: ", json.dumps(event, indent=2))
   * return {"statusCode": 200, "body": "File uploaded!"}
   * Zip the function:
   * zip function.zip lambda\_function.py
   * Deploy the function in the **Mumbai region (ap-south-1)**:
   * aws lambda create-function --function-name S3UploadTrigger \
   * --runtime python3.8 --role arn:aws:iam::<AWS\_ACCOUNT\_ID>:role/S3LambdaExecutionRole \
   * --handler lambda\_function.lambda\_handler --zip-file fileb://function.zip \
   * --region ap-south-1
2. **Set Up S3 Event Notification for Lambda Trigger:**
3. aws s3api put-bucket-notification-configuration --bucket my-bucket-name --notification-configuration '
4. {
5. "LambdaFunctionConfigurations": [
6. {
7. "LambdaFunctionArn": "arn:aws:lambda:ap-south-1:<AWS\_ACCOUNT\_ID>:function:S3UploadTrigger",
8. "Events": ["s3:ObjectCreated:\*"]
9. }
10. ]
11. }'

**Q6: Create two S3 buckets and transfer data from one S3 bucket to another using AWS CLI.**

**Solution:**

1. **Create Two S3 Buckets (Change Bucket Names):**
2. aws s3 mb s3://source-bucket-mumbai
3. aws s3 mb s3://destination-bucket-mumbai
4. **Upload a File to the Source Bucket:**
5. aws s3 cp myfile.txt s3://source-bucket-mumbai/
6. **Copy the File from Source to Destination Bucket:**
7. aws s3 cp s3://source-bucket-mumbai/myfile.txt s3://destination-bucket-mumbai/
8. **Verify the File Transfer:**
9. aws s3 ls s3://destination-bucket-mumbai/

SET 4

**Q7: Identify and list all unused Elastic IPs in your AWS account using the AWS CLI.**

**Solution:**

To find unused Elastic IPs (EIPs) in your AWS account, follow these steps:

1. **List All Elastic IPs:**
2. aws ec2 describe-addresses --query "Addresses[\*].[PublicIp, AllocationId, AssociationId]" --output table

This command lists all EIPs along with their allocation and association IDs.

1. **Filter Unused Elastic IPs:**
2. aws ec2 describe-addresses --query "Addresses[?AssociationId==null].[PublicIp, AllocationId]" --output table
   * The AssociationId==null filter ensures only **unassociated (unused) Elastic IPs** are displayed.
   * The output will show **Public IPs** and **Allocation IDs** of unused EIPs.
3. **Release an Unused Elastic IP (Optional):**  
   If you want to release an unused EIP, use:
4. aws ec2 release-address --allocation-id <ALLOCATION\_ID>

Replace <ALLOCATION\_ID> with the actual allocation ID of the unused Elastic IP.

**Q8: Enumerate IAM Permissions against the exposed credentials of an IAM role.**

**Solution:**

If you have access to **exposed AWS credentials** (e.g., via AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_ACCESS\_KEY), you can enumerate IAM permissions using AWS CLI:

1. **Check Caller Identity:**
2. aws sts get-caller-identity

This returns the **IAM role or user details** associated with the exposed credentials.

1. **List Attached Policies (For IAM Role/User):**
2. aws iam list-attached-user-policies --user-name <USERNAME>

or for IAM roles:

aws iam list-attached-role-policies --role-name <ROLE\_NAME>

1. **Enumerate All IAM Permissions:**
2. aws iam get-user-policy --user-name <USERNAME> --policy-name <POLICY\_NAME>

or

aws iam get-role-policy --role-name <ROLE\_NAME> --policy-name <POLICY\_NAME>

1. **Check Effective Permissions:**  
   To get all permissions available to the IAM entity:
2. aws iam simulate-principal-policy --policy-source-arn arn:aws:iam::<ACCOUNT\_ID>:user/<USERNAME> --action-names "\*"

Replace <ACCOUNT\_ID> and <USERNAME> with actual values.

These steps help determine the **privileges of exposed credentials** and assess potential security risks.

SET 5

**Q9: Create an S3 bucket and give versioning permission to show the last version of objects.**

**Solution:**

To create an S3 bucket and enable versioning, follow these steps using AWS CLI:

1. **Create an S3 Bucket:**
2. aws s3api create-bucket --bucket my-versioned-bucket --region us-east-1

Replace my-versioned-bucket with your preferred bucket name.

1. **Enable Versioning on the Bucket:**
2. aws s3api put-bucket-versioning --bucket my-versioned-bucket --versioning-configuration Status=Enabled
3. **Verify Versioning Status:**
4. aws s3api get-bucket-versioning --bucket my-versioned-bucket

This should return "Status": "Enabled" if versioning is correctly applied.

1. **List Object Versions (Optional):**
2. aws s3api list-object-versions --bucket my-versioned-bucket

This command displays all versions of objects stored in the bucket.

**Q10: Create an EC2 t2.micro instance in the Mumbai region, host a website that can only be accessed with your current public IP.**

**Solution:**

1. **Launch a t2.micro EC2 Instance in Mumbai Region:**
2. aws ec2 run-instances --image-id ami-0abcdef1234567890 --instance-type t2.micro --key-name my-key --security-group-ids sg-12345678 --subnet-id subnet-12345678 --region ap-south-1
   * Replace ami-0abcdef1234567890 with the latest **Amazon Linux AMI ID** in **Mumbai (ap-south-1)**.
   * Replace my-key with your **SSH key name**.
   * Replace sg-12345678 with your **Security Group ID**.
   * Replace subnet-12345678 with a **valid subnet ID**.
3. **Install and Configure Web Server:**  
   Connect to the instance via SSH:
4. ssh -i my-key.pem ec2-user@<EC2-PUBLIC-IP>

Then, install and start a basic web server:

sudo yum update -y

sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd

echo "Hello, this is my secured website!" | sudo tee /var/www/html/index.html

1. **Restrict Access to Your Public IP:**  
   Modify the **security group** to allow only your **current public IP**:
2. aws ec2 authorize-security-group-ingress --group-id sg-12345678 --protocol tcp --port 80 --cidr <YOUR\_PUBLIC\_IP>/32 --region ap-south-1

Replace <YOUR\_PUBLIC\_IP> with your **actual public IP**.

Now, you can **access the website** from your browser using:

http://<EC2-PUBLIC-IP>