**Lab 8: Bash Scripting with AWS CLI and Docker**

Objective:*Integrate AWS CLI operations with Docker in a Bash script.*

Tasks:

1. Write a Bash script to launch an EC2 instance.

2. Once launched, SSH into the instance and run a Docker container.

Documentation:

- AWS CLI basics.

- Combining AWS operations with Docker using Bash.

Prerequisites:

1- An AWS account with administrative access.

2- Docker Deep Dive Course

3- Bash Script Deep Dive Course

4- Complete Previous labs

Implementation Documentation:

**1. Introduction**

In this lab, we will explore how to integrate AWS CLI operations with Docker in a Bash script. We will create a script that launches an EC2 instance using the AWS CLI, and once the instance is running, we will SSH into it and run a Docker container. This demonstrates the power of combining AWS and Docker operations within a single Bash script.

**2. Script Implementation**

Here's the Bash script that accomplishes the tasks:

| #!/bin/bash  # Define AWS EC2 instance details aws\_region="your\_aws\_region" instance\_type="t2.micro" ami\_id="your\_ami\_id" key\_name="your\_key\_name" security\_group\_ids="sg-xxxxxxxxxxxxxxxxx" # Specify your security group ID subnet\_id="subnet-xxxxxxxxxxxxxxxxx" # Specify your subnet ID  # Define Docker container details docker\_image="ubuntu:latest" docker\_command="docker run -it --rm $docker\_image /bin/bash"  # Function to launch an EC2 instance launch\_instance() {  echo "Launching EC2 instance..."    # Use AWS CLI to launch the EC2 instance  instance\_id=$(aws ec2 run-instances \  --region "$aws\_region" \  --instance-type "$instance\_type" \  --image-id "$ami\_id" \  --key-name "$key\_name" \  --security-group-ids "$security\_group\_ids" \  --subnet-id "$subnet\_id" \  --query "Instances[0].InstanceId" \  --output text)    # Wait for the instance to become running  aws ec2 wait instance-running --instance-ids "$instance\_id" --region "$aws\_region"    # Get the public DNS  public\_dns=$(aws ec2 describe-instances \  --region "$aws\_region" \  --instance-ids "$instance\_id" \  --query "Reservations[0].Instances[0].PublicDnsName" \  --output text)    echo "Instance launched with ID: $instance\_id"  echo "Public DNS: $public\_dns" }  # Function to SSH into the EC2 instance and run Docker container ssh\_and\_run\_docker() {  # Wait for the EC2 instance to become available (modify sleep time as needed)  sleep 60    # Get the public DNS of the EC2 instance  public\_dns=$(aws ec2 describe-instances \  --region "$aws\_region" \  --filters "Name=instance-state-name,Values=running" "Name=key-name,Values=$key\_name" \  --query "Reservations[0].Instances[0].PublicDnsName" \  --output text)    echo "SSH-ing into EC2 instance..."    # SSH into the instance and run the Docker container  ssh -i "your\_key.pem" ec2-user@"$public\_dns" "$docker\_command" }  # Main script workflow main() {  # Launch the EC2 instance  launch\_instance   # SSH into the EC2 instance and run the Docker container  ssh\_and\_run\_docker }  # Call the main function to start the script Main |
| --- |

**Explanation of the Script**

We begin by defining variables for AWS EC2 instance details such as the AWS region, instance type, AMI ID, key name, security group IDs, and subnet ID.

Similarly, we define variables for Docker container details, including the Docker image and command to run.

The launch\_instance function uses the AWS CLI to launch an EC2 instance with the specified configuration.

The ssh\_and\_run\_docker function waits for the EC2 instance to become available, retrieves its public DNS name, and then SSHes into the instance to run the specified Docker container.

In the main function, we call the launch\_instance function to create the EC2 instance and then call the ssh\_and\_run\_docker function to SSH into the instance and run the Docker container.

**3. Running the Script**

To run the script:

Save the script in a .sh file, e.g., **aws\_docker\_integration.sh.**

Make the script executable by running **chmod +x aws\_docker\_integration.sh.**

Execute the script by running **./aws\_docker\_integration.sh.**

The script will launch an EC2 instance, SSH into it, and run the specified Docker container.

**4. Conclusion**

In this lab, we have demonstrated how to integrate AWS CLI operations with Docker using a Bash script. This allows for seamless automation of AWS resource provisioning and Docker container execution within a single script, providing flexibility and efficiency in managing cloud-based Docker workloads.