**Lab 9: Advanced Bash Error Handling with Docker Operations**

Objective:*Implement robust error handling in Bash scripts for Docker tasks.*

Tasks:

1. Write a script that tries to run a non-existent Docker container.

2. Implement error handling to catch the error and notify the user.

Documentation:

- Bash error handling mechanisms.

- Ensuring resilient Docker operations through 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 advanced error handling in Bash scripts for Docker operations. We will create a script that attempts to run a Docker container that does not exist and then implement error handling to capture this error and provide informative feedback to the user. Robust error handling ensures that scripts gracefully handle unexpected situations and provide meaningful responses.

**2. Script Implementation**

Here's the Bash script that accomplishes the tasks:

| #!/bin/bash  # Function to run a Docker container (simulating an error) run\_container() {  local container\_name="$1"  echo "Attempting to run Docker container: $container\_name"    # Try to run the Docker container (intentionally providing a non-existent image)  docker run -it --rm "$container\_name"    # Check the exit status of the previous command  if [ $? -ne 0 ]; then  echo "Error: Docker container '$container\_name' does not exist."  fi }  # Main script workflow main() {  container\_name="non\_existent\_container:tag"    # Run the Docker container and handle errors  run\_container "$container\_name" }  # Call the main function to start the script main |
| --- |

**Explanation of the Script**

We define a function run\_container that takes the name of a Docker container as an argument. This function attempts to run the specified Docker container. However, we intentionally provide a non-existent image to simulate an error.

After running the Docker container, we check the exit status of the previous command using $?. A non-zero exit status indicates an error.

If an error is detected (exit status is not 0), we notify the user that the Docker container does not exist.

In the main function, we set the container\_name variable to a non-existent container name and call the run\_container function to simulate running a non-existent container.

**3. Running the Script**

To run the script:

Save the script in a .sh file, e.g., **docker\_error\_handling.sh**.

Make the script executable by running **chmod +x docker\_error\_handling.sh**.

Execute the script by running **./docker\_error\_handling.sh**.

The script will attempt to run a non-existent Docker container and then correctly handle the error, notifying the user that the container does not exist.

**4. Conclusion**

In this lab, we have demonstrated advanced error handling in Bash scripts for Docker operations. Proper error handling is crucial for making scripts more robust and user-friendly. By implementing error handling mechanisms like checking exit statuses, scripts can gracefully respond to unexpected situations and provide informative feedback to users, improving overall reliability and usability.