**Lab 7: Scheduling Docker Workloads with Bash and Cron**

Objective:*Use Bash and Cron to automate Docker workloads.*

Tasks:

1. Write a script to run periodic backups using a Dockerized backup tool.

2. Schedule the script using Cron.

Documentation:

- Introduction to cron jobs.

- Automating Docker operations using Bash and cron.

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 the automation of Docker workloads using Bash scripts and Cron. Docker workloads often require routine maintenance tasks like backups. We will create a Bash script that performs periodic backups using a Dockerized backup tool and then schedule this script to run at specified intervals using Cron.

**Let's create a very simple backup tool using the cp command. In this example, we'll create a Docker image that copies a directory from one location to another.**

**Step 1: Create Dockerfile**

Create a file named Dockerfile (without any file extension) with the following content:

| # Use an official base image FROM ubuntu:latest  # Create a directory to serve as the source of our backup RUN mkdir /source WORKDIR /source  # Create some test files RUN echo "This is file 1." > file1.txt && echo "This is file 2." > file2.txt  # Create a directory to serve as the backup destination RUN mkdir /backup  # Copy the files from source to backup CMD cp -r /source /backup |
| --- |

**Step 2: Build the Docker Image**

Run the following command in the directory containing your Dockerfile:

*docker build -t backup\_tool:latest .*

**2. Script Implementation**

To run the script, follow these steps:

In terminal goto / directory e.g., **cd /**

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

Here's the Bash script that accomplishes the tasks:

| #!/bin/bash  # Define variables backup\_directory="/backup\_data" docker\_container\_name="my\_backup\_container" backup\_tool\_image="backup\_tool:latest"  # Function to perform a backup using Dockerized backup tool perform\_backup() {  # Create a backup directory if it doesn't exist  mkdir -p "$backup\_directory"   # Run the Docker container to perform the backup  docker run --rm -v "$backup\_directory:/backup" "$backup\_tool\_image" }  # Main script workflow main() {  echo "Starting periodic backup..."    # Perform the backup  perform\_backup   echo "Backup completed." }  # Call the main function to start the script main |
| --- |

Make the script executable by running **docker\_workload.sh.**

**Explanation of the Script**

We begin by defining variables for the backup directory, the name of the Docker container running the backup tool, and the image and tag of the backup tool.

The perform\_backup function ensures that the backup directory exists and then runs a Docker container based on the specified image. It mounts the backup directory as a volume to store the backup data.

In the main function, we start the periodic backup process. We then call the perform\_backup function to execute the backup.

**3. Scheduling with Cron**

To automate this script, we can use Cron, a time-based job scheduler in Unix-like operating systems. To schedule the script to run at specific intervals, follow these steps:

Open the Cron configuration file by running crontab -e.

Press 1

Add the following line to schedule the script to run, for example, every day at midnight:

*0 0 \* \* \* /docker\_workload.sh*

Save the file and exit the text editor. Cron will now execute your script at the specified interval.

**4. Conclusion**

In this lab, we have demonstrated how to automate Docker workloads using Bash scripts and Cron. By scheduling the script, you can ensure that periodic backups using a Dockerized tool are performed automatically. This approach saves time and ensures the regular maintenance of your Docker environment.