Computer Science & Information Systems

**DevOps for Cloud - Lab Sheet 3 – Module 6**

**(M6: Docker and Kubernetes)**

This lab sheet needs to be administered along with Module 6: Docker and Kubernetes

**Notation used in the document**

* ‘>’ represents the terminal, where we type the commands.
* The text mentioned within ‘[‘ and ‘]’ brackets provides additional documentation for the step.

1. Objectives:
2. To demonstrate the steps to install and run Docker
3. To build custom docker image in Docker Desktop
4. To dockerize a RESTful Microservice application
5. To push the docker image from Docker Desktop to Docker Hub
6. To push the docker image from Docker Desktop to Amazon ECR (Elastic Container Registry)
7. Pre-requisite:

* Basic knowledge of docker, docker architecture, and docker hub
* Valid credentials to access Docker Hub and Amazon ECR
* IDE such as Visual Studio Code

1. Lab Exercise:

**Task 1: Install Docker Desktop**

* Purpose: Install Docker Desktop on any OS
* URLs to download Docker Desktop on various OS are provided below
  + Windows - https://www.docker.com/products/docker-desktop/
  + MAC - https://docs.docker.com/desktop/install/mac-install/
  + LINUX - https://docs.docker.com/desktop/install/linux-install/
* In Windows -> Run the Docker Desktop Installer and follow the Instructions
* Configuration -> Check both ‘WSL2’ and ‘Add Shortcut to Desktop’
* In Windows -> Start -> Docker Desktop -> Open
* Sign-In to Docker (Optional step)

[Docker Desktop lets you to sign in to Docker Hub using two-factor authentication]

**Task 2: Run “Hello World” Image**

* Purpose: Demonstrate the connection between Docker Client, Docker Desktop Daemon and Docker Hub
* Prerequisite: Docker Desktop must be running

**Steps**

* Visit the website <https://hub.docker.com/> and search for official image of “hello-world”
* Open any terminal
* > docker pull hello-world
* [We are running this command on the docker client. This will check if the image is present in docker desktop daemon (local computer). Since there is no such image, docker desktop will pull this image from docker hub. This step will only create the image; still container is not yet created]
* > docker run hello-world
* [Creates the container and exits]
* > docker run --name my-hello-world hello-world
* [Syntax: docker run --name <container-name> <image-name>]

**Task 3: Interactive Terminal**

* Purpose: Create a Ubuntu container and open in interactive mode
* Prerequisite: Docker Desktop must be running

**Steps**

* Visit the website <https://hub.docker.com/> and search for official image of “hello-world”
* Open any terminal
* > docker pull Ubuntu
* > docker run -it ubuntu bash
* [Run the container interactively. “it” flag denotes interactive terminal]
* > apt update
* > apt install vim

[Sample commands to run inside the container]

**Sub Task: Commands to verify a running container, stop a container, and remove a container**

* 1. Command to verify a running container
* > docker ps
* [Lists all the “running” containers in Docker Desktop]
  1. Command to stop the container [Only stops the running container, but does not remove]
* > docker stop <container-name>
* [Ex: if the container name for ‘ubuntu’ is “Ubuntu-container”, then the command will be: > docker stop Ubuntu-container ]
  1. Command to remove the container [Removes the container from the list of containers in Docker Desktop]
* > docker rm <container-name>
* [Ex: if the container name for ‘ubuntu’ is “Ubuntu-container”, then the command will be: > docker rm Ubuntu-container ]

**Task 4: Create Custom Docker Image in Docker Desktop**

* Go to any directory of your choice and create a new folder called “custom-docker”
* Open Visual Studio Code IDE
* Open the folder “custom-docker” from Visual Studio Code
* Create a new file called “dockerfile” [It is a text file]
* Go to “Extensions” -> search for “Docker” -> click Install
* Copy the below ‘dockerfile’ contents into the file

# First Layer - from which Docker Image to start with

FROM ubuntu:latest

# Run indicates execute the commands inside the container

RUN mkdir /vim

RUN apt update

# -y is used to avoid prompt to install or not

RUN apt install vim -y

# WORKDIR informs which container working directory

WORKDIR /vim

# Which command to run in container - any executable as start point

ENTRYPOINT ["/bin/bash"]

* Open Terminal inside VS Code (View -> Terminal)
* > docker build –t myubuntu .
* [build custom image with tag called ‘myubuntu’. Dot indicates current location]
* > docker images [to view the myubuntu image]
* > docker run -it myubuntu
* [Starts the container and places in /vim directory]
* > pwd [/vim directory]
* > vim
* [check vim is installed; press escape-colon-q to exit]
* > exit [to exit the interactive mode]

**Task 5: Dockerize a RESTful Microservice Application**

* Purpose: Create a Custom docker image in Docker Desktop
* Prerequisite: Docker Desktop must be running

**Steps to download the application and test locally**

* Download or Clone the RESTful Application from GitHub repo -  [https://github.com/shreyassureshrao/RestApp/](%20https://github.com/shreyassureshrao/RestApp/)
* [> git clone https://github.com/shreyassureshrao/RestApp.git]
* Open the application in Visual Studio Code
* Open Terminal inside the VS Code [Directory -> “RestApp”]
* > pip install -r requirements.txt
* Run the Uvicorn server in command prompt and see the application output
* > cd app
* > uvicorn main:app --reload [It runs the uvicorn server at port 8000]
* In browser type http://localhost:8000/docs (FastAPI Swagger)

**Steps to build and run the Docker Image - into a container**

* In VS Code Terminal, run the following commands
* [Make sure that the terminal is in “RestApp” directory, not “app”]
* Command: docker build -t restimage .
* Command: docker run -d --name fastapicontainer -p 8000:8000 restimage
* Test the Container running on port 8000
* In the browser -> http://localhost:8000/docs [Opens Swagger API]
* Verify the container in 'Docker Desktop' - "fastapicontainer" should be running

**Task 6: Push the image to DockerHub**

* [Steps to push the docker image ‘restimage’ from Docker Desktop to Docker Hub]
* Prerequisite – Create a Free account in ‘Docker Hub’
* [URL - https://hub.docker.com/]

1. Open any terminal
2. Check login to DockerHub

> docker login -u “username” -p “pwd” docker.io

[docker login -u “6945” -p “pwd” docker.io]

[Provide your username and password used to create DockerHub account]

1. Create a Tag for the Custom Image

> docker tag restimage:latest 6945/restimage:latest

1. Push the image from local Docker Desktop to DockerHub

> docker push 6945/restimage:latest

1. Login to DockerHub and verify that the image is visible

[Go to Repository – where you can find the ‘restimage’ image]

1. Pull the image [Optional]

> docker pull 6945/ restimage:latest

**Task 7: Push the image to Amazon ECR (Elastic Container Registry)**

**Pre-requisites**

1. AWS CLI [Steps to install - https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html]

2. Docker Desktop

3. AWS Free Tier Account

**Steps**

[Ref URL - https://www.freecodecamp.org/news/build-and-push-docker-images-to-aws-ecr/]

1. Open any Terminal (Ex: command prompt)
2. > aws ecr create-repository --repository-name restrepo

[Create a repository in ECR through AWS CLI]

3. Need to connect the AWS ECR with Docker and setup permissions

> aws ecr get-login-password --region ap-northeast-1 | docker login --username AWS --password-stdin 729716288173.dkr.ecr.ap-northeast-1.amazonaws.com

[My region - “ap-northeast-1”; AWS Account Id – “729716288173”]

[Syntax is -> aws ecr get-login-password –region <region-name> | docker login –username AWS –password AWS –password-stdin <AWS Account Id>.dkr.ecr.<region name>.amazonaws.com]

4. Tag the local Docker image with the ECR Repo.

> docker tag restimage 729716288173.dkr.ecr.ap-northeast-1.amazonaws.com/restrepo

[Syntax : docker tag <source\_image\_tag> <target\_ecr\_repo\_uri>]

5. Push the Docker Image to ECR

> docker push 729716288173.dkr.ecr.ap-northeast-1.amazonaws.com/restrepo

[Syntax: docker push <ecr-repo-uri>]

[Pushes the image to ECR]

1. Go to the ECR Repo within the AWS Portal and verify whether the image is migrated

4. Outputs/Results:

Students are expected to perform the tasks provided in the lab capsule, and thereby gain a practical understanding of the container based architectures, i.e. docker. Additionally, the students will be able to create a custom docker image, dockerize a restful application, push docker images to external container registries such as Docker Hub and Amazon ECR.

1. Observations:

* None

References:

* <https://hub.docker.com/>
* <https://aws.amazon.com/ecr/>