

*System Admin*

*Training Assignments*

|  |  |
| --- | --- |
| **Program Code** |  |
| **Issue/Revision** | **x/y** |
| **Effective date** | **04/Aug /2023** |

**Assignment Day 14: Docker Container**

Part 1: Docker Containers

1. Install Docker: If you haven't already, install Docker on your system following the official Docker documentation.
2. Create a Dockerfile: Write a Dockerfile to build a simple web application using your choice of programming language and framework (e.g., Node.js, Python Flask). The application should display "Hello, Docker!" when accessed through a web browser.
3. Build and Run a Container: Build an image from your Dockerfile and run a container using that image. Ensure that you can access the web application in your browser by navigating to http://localhost:80.
4. Inspect and Publish: Use the docker inspect command to gather information about your running container. Then, publish your Docker image to Docker Hub.

Part 2: Docker Compose

1. Install Docker Compose: If you haven't already, install Docker Compose on your system following the official Docker documentation.
2. Create a Docker Compose File: Write a Docker Compose YAML file that includes your web application container and a separate database container (e.g., MySQL or PostgreSQL). Configure the web application to connect to the database.
3. Start the Application Stack: Use docker-compose to start your application stack. Ensure both containers are running and that your web application can interact with the database.

Part 3: Docker Swarm

1. Initialize a Docker Swarm: Initialize a Docker Swarm on your local machine.
2. Deploy a Service: Use Docker Swarm to deploy your web application as a service. Ensure it runs on at least two replicas.
3. Scale the Service: Scale the web application service to four replicas.
4. Visualize the Stack: Use docker service and docker node commands to visualize the services and nodes in your Docker Swarm.

**Solution**

**Part 1: Docker Containers**

1. **Install Docker**

*# Check OS version, at lease Ubuntu 18.04 (LTS) or above*

lsb\_release –a

*# First Update*

sudo apt-get update

*# Allow to install security packages*

sudo apt-get -y install ca-certificates curl gnupg lsb-release apt-transport-https

*# Allow trust application that will be registed by Docker*

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

*# Configure to download Docker*

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

*# Second Update after configure Docker download link*

sudo apt-get update

*# Install Docker and docker compose*

sudo apt-get -y install docker-ce docker-ce-cli containerd.io docker-compose-plugin

*# Checking*

sudo docker run hello-world

*# Add user to group docker so we can run docker CLI without sudo pemission.* Remember to logout then login again to apply.

sudo usermod –aG docker [your user name]

1. **Create a Dockerfile:** Create a file named **Dockerfile** with the following content (assuming you're using Node.js):

# Use an official Node.js runtime as a parent image

FROM node:14

# Set the working directory to /app

WORKDIR /app

# Copy package.json and package-lock.json to /app

COPY package\*.json ./

# Install app dependencies

RUN npm install

# Copy the current directory contents to /app

COPY . .

# Expose port 80

EXPOSE 80

# Define environment variable

ENV NAME World

# Run "node app.js" when the container launches

CMD [ "node", "app.js" ]

1. **Build and Run a Container:** In the directory containing your **Dockerfile**, run these commands:

*# Build the Docker image*

docker build -t my-node-app .

*# Run the Docker container*

docker run -p 80:80 -d my-node-app

1. **Inspect and Publish:** Use the following commands to inspect your running container and publish your Docker image to Docker Hub:

*# Get container ID*

docker ps

*# Inspect the container (replace CONTAINER\_ID with your actual container ID)*

docker inspect CONTAINER\_ID

*# Log in to Docker Hub*

docker login

*# Tag your image (replace YOUR\_DOCKERHUB\_USERNAME with your Docker Hub username)*

docker tag my-node-app YOUR\_DOCKERHUB\_USERNAME/my-node-app *# Push the image to Docker Hub docker push* YOUR\_DOCKERHUB\_USERNAME/my-node-app

**Part 2: Docker Compose**

1. **Install Docker Compose:** Already install with Docker in Part 1.
2. **Create a Docker Compose File:** Create a file named **docker-compose.yml** with the following content:

version: '3'

services:

web:

image: my-node-app

ports:

- "80:80"

db:

image: mysql:5.7

environment:

MYSQL\_ROOT\_PASSWORD: [Your strong password]

1. **Start the Application Stack:** Run the following command in the same directory as your **docker-compose.yml** file:

docker-compose up -d

**Part 3: Docker Swarm**

1. **Initialize a Docker Swarm:** Run the following command to initialize a Docker Swarm:

docker swarm init

1. **Deploy a Service:** Deploy your web application service using this command (make sure to replace **my-node-app** with your Docker Hub image if different):

docker service create --replicas 2 --name my-node-app -p 80:80 my-node-app

1. **Scale the Service:** Scale the service to four replicas using the following command:

docker service scale my-node-app=4

1. **Visualize the Stack:** You can visualize the stack using the following commands:

# List services

docker service ls

# List nodes

docker node ls