ADDA

DOCKER

1.)

git clone https://github.com/dockersamples/linux\_tweet\_app

docker container run alpine hostname

docker container ls --all

-Run an interactive Ubuntu container:

docker container run --interactive --tty --rm ubuntu bash

ls /

ps aux

cat /etc/issue

[ls / will list the contents of the root directory in the container, ps aux will show running processes in the container, cat /etc/issue will show which Linux distro the container is running, in this case Ubuntu 20.04.3 LTS]

exit

-Run a background MySQL container

docker container run --detach --name mydb -e MYSQL\_ROOT\_PASSWORD=my-secret-pw mysql:latest

[List the running containers.]

docker container ls

docker container logs mydb

docker container top mydb

docker exec -it mydb mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version

docker exec -it mydb sh

mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version

exit

task 2: Package and run a custom app using Docker

-Build a simple website image

**cd ~/linux\_tweet\_app**

**cat Dockerfile**

**echo $DOCKERID**

**[to build a docker image]**

**docker image build --tag $DOCKERID/linux\_tweet\_app:1.0 .**

**[Use the docker container run command to start a new container from the image you created.]**

**docker container run --detach --publish 80:80 --name linux\_tweet\_app $DOCKERID/linux\_tweet\_app:1.0**

**[remove]**

**docker container rm --force linux\_tweet\_app**

**Task 3: Modify a running website**

**[Start our web app with a bind mount]**

**docker container run --detach --publish 80:80 --name linux\_tweet\_app --mount type=bind,source="$(pwd)",target=/usr/share/nginx/html $DOCKERID/linux\_tweet\_app:1.0**

[Modify the running website]

cp index-new.html index.html

[Stop and remove the currently running container.]

docker rm --force linux\_tweet\_app

[Rerun the current version without a bind mount.]

docker container run --detach --publish 80:80 --name linux\_tweet\_app $DOCKERID/linux\_tweet\_app:1.0

[Notice the [website](https://training.play-with-docker.com/) is back to the original version.]

[Stop and remove the current container]

docker rm --force linux\_tweet\_app

Update the image

docker image build --tag $DOCKERID/linux\_tweet\_app:2.0 .

docker image ls

Test the new version

docker container run --detach --publish 80:80 --name linux\_tweet\_app $DOCKERID/linux\_tweet\_app:2.0

docker container run --detach --publish 80:80 --name linux\_tweet\_app $DOCKERID/linux\_tweet\_app:1.0

Push your images to Docker Hub

docker image ls -f reference="$DOCKERID/\*"

docker login

docker image push $DOCKERID/linux\_tweet\_app:1.0

docker image push $DOCKERID/linux\_tweet\_app:2.0

2.)Docker

git clone <https://github.com/docker/getting-started-app.git>

-[Build the app's image](https://docs.docker.com/get-started/workshop/02_our_app/#build-the-apps-image)

# syntax=docker/dockerfile:1

FROM node:lts-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000

Build the image using the following commands:

cd /path/to/getting-started-app

build the image:

docker build -t getting-started .

[Start an app container](https://docs.docker.com/get-started/workshop/02_our_app/#start-an-app-container)

docker run -d -p 127.0.0.1:3000:3000 getting-started

docker ps

**Lab Scenario 3: GitHub Actions – Run Tests**

**🔹Question:**

*Create a GitHub Actions workflow that installs dependencies and runs tests for a Node.js app on every push.*

1. Create .github/workflows/test.yml

name: Run Node Tests

on: [push]

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v3

- name: Set up Node

uses: actions/setup-node@v3

with:

node-version: '18'

- name: Install Dependencies

run: npm install

- name: Run Tests

run: npm test

**Lab Scenario 4: GitHub Actions – Docker Build & Push**

**🔹Question:**

*Set up a CI/CD workflow that builds your Docker image and pushes it to Docker Hub on push to main.*

**1. Add secrets in GitHub Repo:**

* DOCKER\_USERNAME
* DOCKER\_PASSWORD

1. .github/workflows/docker-publish.yml

name: Docker CI/CD

on:

push:

branches:

- main

jobs:

build-and-push:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v3

- name: Log in to Docker Hub

run: echo "${{ secrets.DOCKER\_PASSWORD }}" | docker login -u "${{ secrets.DOCKER\_USERNAME }}" --password-stdin

- name: Build the Docker image

run: docker build -t ${{ secrets.DOCKER\_USERNAME }}/flask-app:latest .

- name: Push the Docker image

run: docker push ${{ secrets.DOCKER\_USERNAME }}/flask-app:latest