Cloud Computing

2nd laboratory

Tasks

1.

Read and test the node-bulletin-board example https://docs.docker.com/get-started/part2/.

2.

Create an image that uses the Linux curl command to get a webpage from the internet (e.g., https://google.com). The contents of the webpage must be saved in a directory from the host system. See https://docs.docker.com/storage/bind-mounts/.

3.

Create a producer Docker image and a consumer Docker image. The producer image will contain a program that generates files every second and the consumer image will contain a program that outputs the list of files generated by the consumer.

The producer and the consumer programs may be written as a bash script, a cpp program, a java program, or a python script, but they must use different languages.

Test with multiple producers. What are the possible ways to view the consumer results?

Read the Storage overview from Docker – https://docs.docker.com/storage/.

Tips & Ticks

Docker commands

docker run = (**docker create**) + (**docker start -a**)

docker create – creates a container from the named image and outputs the created container id

docker start -a – is used to start the container with that id; the -a option causes the terminal to attach so that the container runs in the foreground

Example of an image running a cpp program

```
# Get the GCC preinstalled image from Docker Hub
FROM gcc:4.9

# Copy the current folder which contains C++ source code to the Docker
# image under /usr/src
COPY . /usr/src/dockercpptest

# Specify the working directory
WORKDIR /usr/src/dockercpptest

# Use GCC to compile the Test.cpp source file
RUN g++ -o Test Test.cpp

# Run the program output from the previous step
CMD ["./Test"]
```

Example of an image executing a script copied from the host system

Dockerfile

```
FROM alpine

RUN apk add --no-cache bash

#COPY src dest

COPY . /app

RUN chmod +x /app/hello.sh

CMD /app/hello.sh
```

hello.sh

```
#!/usr/bin/env bash
echo "Hello world"
```

Commands

Create an image named myimage:1.0 from a Dockerfile

```
docker build -t myimage:1.0 -f Dockerfile .
```

View the existing images

```
docker image ls
```

Launch a container named mycontainer1 from image myimage:1.0

* the -d command line argument launched the image in detached mode

```
docker run -it --name mycontainer1 myimage:1.0
```

View all containers

```
docker ps --all
```

Launch a container named mycontainer2 from image myimage:1.0 and execute the bash command

```
docker run -it --name mycontainer2 myimage:1.0 bash
```