# Lab-13-1: Docker Networking

# Objectives

* Configuring networking inside docker
* Creating a Flask application on Docker
* Deploying a nginx reverse proxy in front of Flask application container

--------------------------------------------------------------------------------------------------

**Today:**

Login to your ubuntu installation (locally, through Azure/AWS instance) where you have installed docker before.

**Task1: Create Docker network**

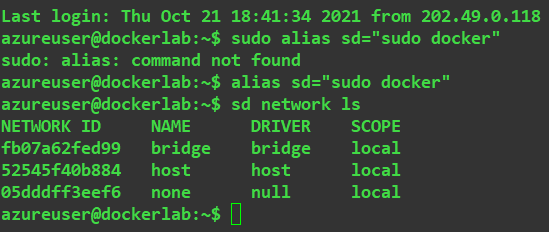
1. Examine Docker network
2. Create bridge network
3. Launch containers to connect to different networks

Examine Docker Container

---------------------------------------

* Type the following command to know about networking
  + Docker network ls

**Q1.** What are the different types of networks available on Docker? (print screen shot from above command)



Overlay, and macvlan

**Q2**. Use docker documentation to find out what each type of network does.

Ans:

g. **Bridge networks** are usually used when your applications run in standalone containers that need to communicate. **Host**: For standalone containers, remove network isolation between the container and the Docker host, and use the host’s networking directly.  **None**: disable all networking. Usually used in conjunction with a custom network driver.

**Q3**. Which one is the default network?

Ans: bridge network

**Q4**. What are the commands to launch two containers named lab13a and lab13b

Ans: docker run -dit --name lab13a ubuntu /bin/bash

docker run -dit --name lab13b ubuntu /bin/bash

[hint: use docker run command with -dit option]

* Use the following command to check which networks the new containers are connected
  + Docker network inspect <networkname>

Use docker to generate and register a ssh key and register with your account

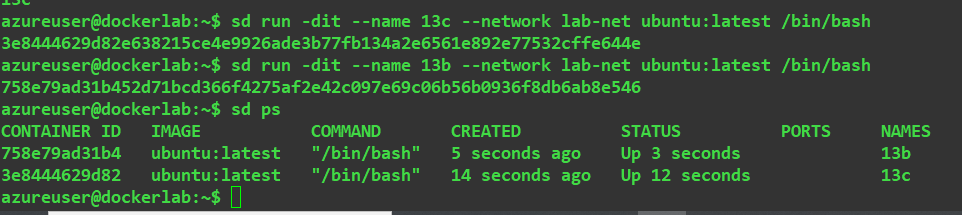
* Create an additional bridged network named ‘lab-net’
  + Docker network create –-driver bridge lab-net

**Q5.** What happens if you omit the ‘–-driver bridge’ from the above command

Ans: the command still runs

**Q6.** Launch two containers named 13b and 13c and connect to ‘lab-net’

Ans:

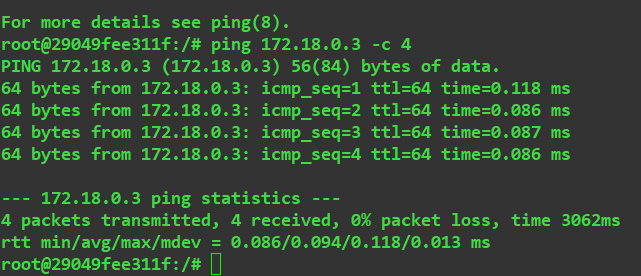


**Q7.** Add screen shot to verify that 13b and 13c connected to ‘lab-net’

Ans:



* Attach to lab13a container and do the following:
  + apt update && apt install iputils-ping
  + ping <ipaddress of lab13b>
  + ping <ipaddress of lab13c>
  + ping lab13b

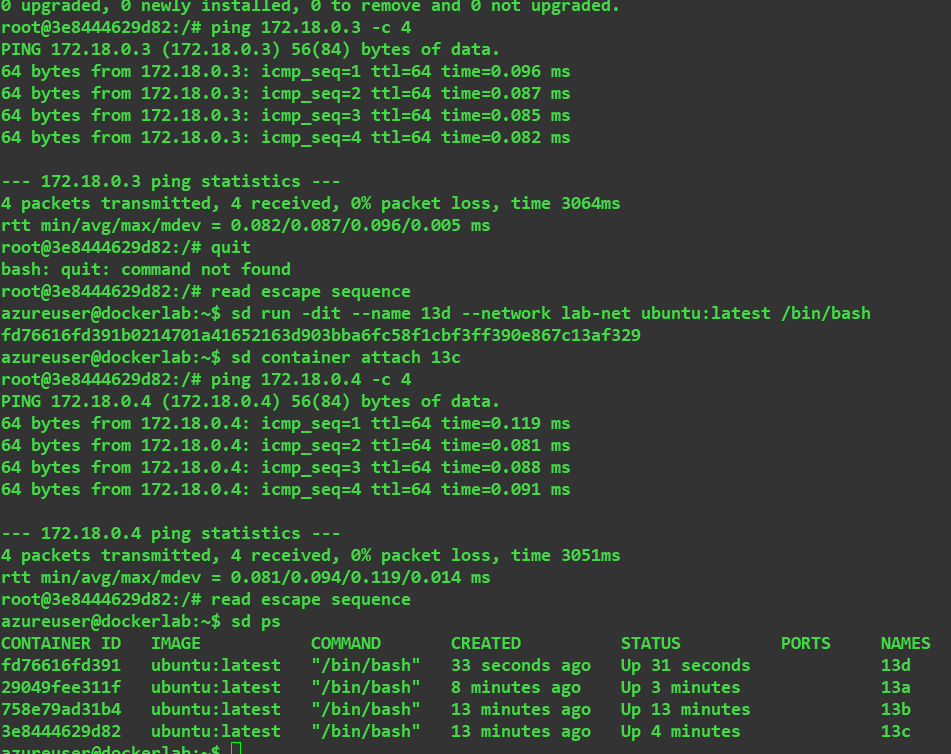


* Detach (Ctrl-p Ctrl-q)
* Attach to lab13c and do the following
  + ping <ipaddress of lab13b>
  + ping <ipaddress of lab13d>
  + ping lab13d

**Q8**. What are your observation from running pings for the containers connected

to different networks?

Ans: all pings successful



**Task 2: Create a Flask Application Container**

1. Build a Flask container using dockerfile
2. Access the webapp through browser

Build a Flask App container and access through browser

--------------------------------------------------------------------------------

* Make a directory flaskapp to serve as a build context for Python/Flask application image
* Copy the lab-4-app folder in the build context from https://github.com/tclark/lab-4-app.git
* Create a docker file in the build context to have the following:

FROM ubuntu:18.04

LABEL updated\_on="2021-10-27 09:00"

RUN apt-get update

RUN apt-get -y upgrade

RUN apt-get -y install python3 python3-setuptools python3-pip gunicorn3

RUN update-alternatives --install /usr/bin/python python /usr/bin/python3 10

COPY lab-4-app /flaskapp

WORKDIR /flaskapp

RUN pip3 install -r requirements.txt

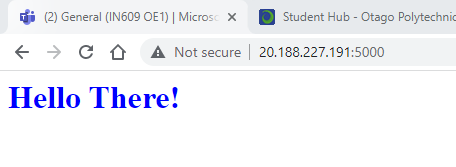
EXPOSE 5000

ENTRYPOINT "./startup.sh"

**Q9:** What does this docker file do?

Ans: defines the instructions for building a image.

* Build your container image with tag user/flaskapp
* Test it by running:
  + docker run -d –-rm –-name flaskapp -p 5000:5000 user/flaskapp
* Verify that it works by checking it with a browser



**Task 3: Add a reverse proxy for the Flask app**

1. Create a nginx container
2. Create a docker network connecting nginx and the Flask app containers

Create a nginx container:

-------------------------------------------------------------------------------------------

* Set up a build context:

1. place a dockerfile in it:

FROM nginx:1.13

COPY flaskapp.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

1. create the flaskapp.conf file in our build context:

resolver 127.0.0.11 valid=1s;

server {

set $alias "flaskapp";

location / {

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_pass http://$alias:5000;

}

listen 80;

}

* Build this image with tag user/nginx

**Q10;** What does the from the above dockerfile do?

Ans:

Copies the flaskapp configuration file to the nginx default.conf and exposes port 80. The conf file maps port 80 to the flask app port 5000

* Run the flaskapp and the nginx containers

**Q11**. What do **you** see if you run the containers?

Ans: the site is being served on port 8080.

Stop the containers

* Create a new bridged network named ‘app’
* Run the above containers to use the new network
  + docker run -d --rm --name flaskapp --network app user/flaskapp
  + docker run -d --rm --name nginx --network app -p 8080:80 user/nginx

**Q12:** Submit the screen shot that verifies that you can connect to your flask app through nginx?

Ans:

