# Objective

* Create a container with Python image and send the source code for Flask application – this is to run the Flask application with gunicorn
* Create another container with nginx image
* Connect the two containers to have nginx work as the reverse proxy for the Flask app

## Part 1: Python Container with Flask and Gunicorn

After saving the url-shortener source code locally (i.e. /Users/cadenhong/Desktop/url-shortener-app/), run the following commands:

**docker run -v /Users/cadenhong/Desktop/url-shortener-app:/url-shortener-app --name pythonApp -ti python bash**

>>> Creates a pythonApp container with a Python image and mounts the url-shortener-app directory inside the container

**pip install -r requirements.txt**

>>> Installs all dependencies listed in requirements.txt

**pip install gunicorn**

**python3 -m gunicorn -w 4 application:app -b 0.0.0.0**

>>> Run the application using gunicorn locally

**apt update && apt install net-tools**

>>> To have access to the ifconfig command (to get IP address of the container for next part)

**ifconfig**

>>> 172.17.0.2

*Or, you can run this command outside the container:*

**docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' pythonApp**

>>> 172.17.0.2

## Part 2: Nginx Container

**docker run -ti --name web001 -p 8111:80 nginx bash**

*HOST PORT:CONTAINER PORT*

*Any port:default nginx port*

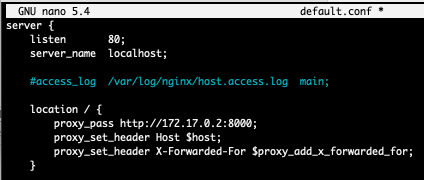
>>> Creates the nginx container – configures port 8111 of local machine to port 80 of nginx container

**apt update && apt install nano**

**cd /etc/nginx/conf.d/**

**nano default.conf**

>>> Setting the configuration to listen on port 80 and forward to gunicorn container’s port 8000



Server listen & server\_name is both for nginx container

Location IP address is for Python container

**service nginx stop**

**service nginx start**

>>> Restarting nginx to apply the new configuration

## Part 3: Confirm Configurations

Navigating to <http://localhost:8111> or <http://0.0.0.0:8111> will route you to the gunicorn container as configured:

Graphical user interface, application

Description automatically generated

To see how ports are connected, run the commands in the containers:

**apt install net-tools**

**netstat -tupln**

Graphical user interface, website

Description automatically generated

Local computer -> nginx container -> Python container

*Pushing Docker image on Docker Hub:*

**docker commit pythonApp pyappdockerized:v1.0**

>>>> Created an image of the container that had url-shortener-app folder

Text

Description automatically generated

## Diagram of Activity

