# Lab-13-2: Docker Compose

# Objectives

* Deploying a nginx reverse proxy in front of Flask application container using docker compose

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

**Today:**

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

In the previous lab, you have built a a nginx container that serves as a reverse proxy to a webapp. You then access the webapp using your web browser. The nginx container had a port that was mapped on a port inside the application container.

In this lab, you will deploy an application that has nginx reverse proxy as a front end of your app which connects to redis database. You will be using docker-compose file to deploy the application.

**Task 1: Build the application container**

1. Copy the build context of lab13.1 to a new folder.
2. clone this repo this <https://github.com/hasansfop/lab132.git>
3. replace the myproject.py with app.py
4. replace the requirements.txt with the requirements.txt you downloaded today
5. Modify it to have ENTRYPOINT python app.py
6. Build the container

**Task 2: Build the nginx container**

You can build the same nginx container as you did for lab13.1, but rename it as lab13-nginx

**Task 3: Creating Compose file**:

Docker compose is a docker tool that is used to deploy containers using code.

Finally., create the docker-compose.yaml file. In it, you will need directives to:

- Build and run the flaskapp container, using the app network.

- Build and run the nginx container, using the app network and mapping its port 80 to a port on the host.

- Run a redis-svr container, also attached to the app network. It’s not necessary to set up a special image for this. You can just use the standard redis:latest image from Dockerhub.

- Create the app network.

[hint: consult https://docs.docker.com/compose/]

Once this is done, you can build any images you need with the docker-compose build command. You will also need to run this any time you make changes in your build contexts to update your images.

Start the whole thing running with the docker-compose up command, and shut it down when you’re done with docker-compose down.