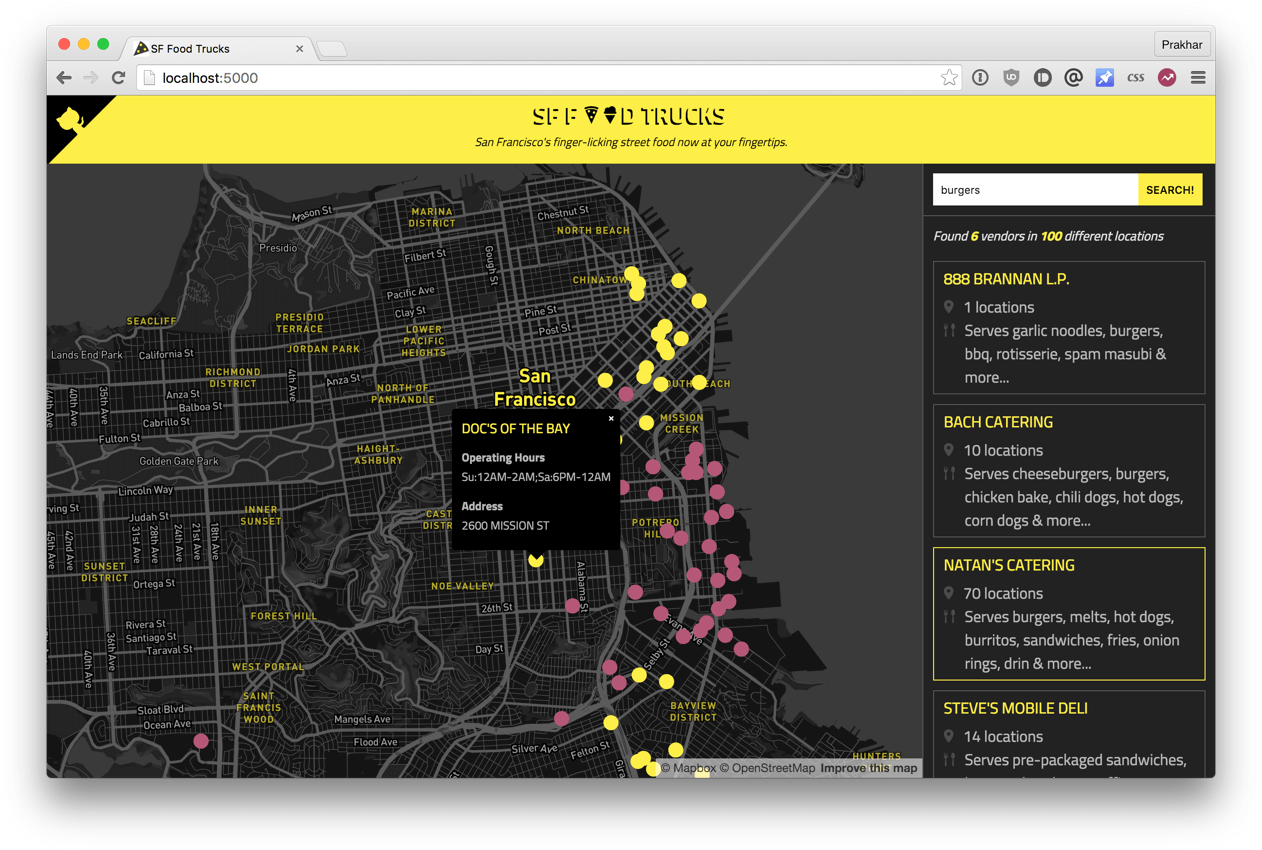
## Module: EGT307 AI Application Development

Practical 3d: Docker Compose

Compose is a tool that is used for defining and running multi-container Docker apps in an easy way. It provides a configuration file called compose.yaml that can be used to bring up an application and the suite of services it depends on with just one command. Compose works in all environments: production, staging, development, testing, as well as CI workflows, although Compose is ideal for development and testing environments.

In the last tutorial, we have built and deployed a multi-container application called SF Food Trucks.



In this practical, let's create a compose.yaml file for our SF-Foodtrucks app.

Preparation

1. Before we start, we need to make sure the ports and names are free. So if you have the Flask and ES containers running, lets turn them off.

A black background with white text

Description automatically generated

1. Download compose.yaml from BrightSpace and paste into “FoodTrucks” directory (you downloaded in last practical session)

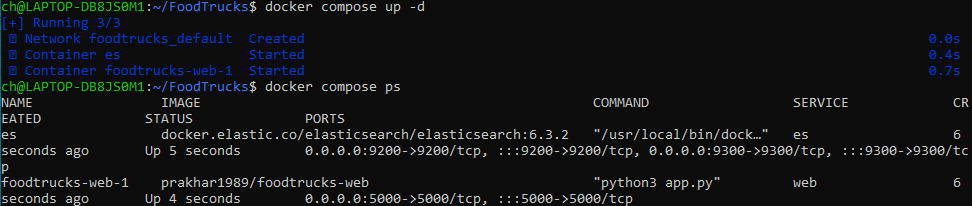
Activity 1: Docker Compose file

Edit the compose.yaml accordingly.

We add volumes for the *es* container so that the data we load persists between restarts. We also specify *depends\_on*, which tells docker to start the *es* container before *web*.

Now we can run docker-compose up in detached mode.

Make sure you are in FoodTrucks folder before you run the following command.



Unsurprisingly, we can see both the containers running successfully. Where do the names come from? Those were created automatically by Compose.

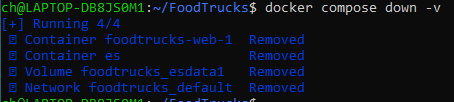
Head over to the IP to see your app live.

Activity 2: Network automatically created by Compose

Remember that we need to manually create the network bridge in last practical?

Why don’t we need to do it this time round?

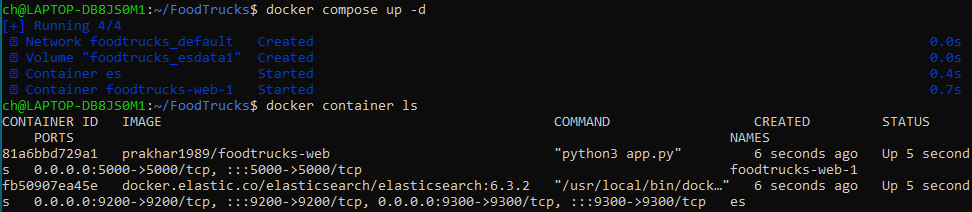
First off, let us stop the services from running. We can always bring them back up in just one command. To destroy the cluster and the data volumes, just type docker compose down -v.



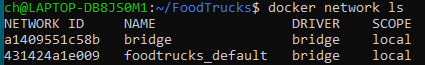
While we're are at it, we'll also remove the foodtrucks network that we created last time.



Great! Now that we have a clean slate, let's re-run our services and see if Compose does its magic.



So far, so good. Time to see if any networks were created.

  
You can see that compose went ahead and created a new network called *foodtrucks\_default* and attached both the new services in that network so that each of these are discoverable to the other.

A screen shot of a computer code

Description automatically generated

Each container for a service joins the default network and is both reachable by other containers on that network, and discoverable by them at a hostname identical to the container name.