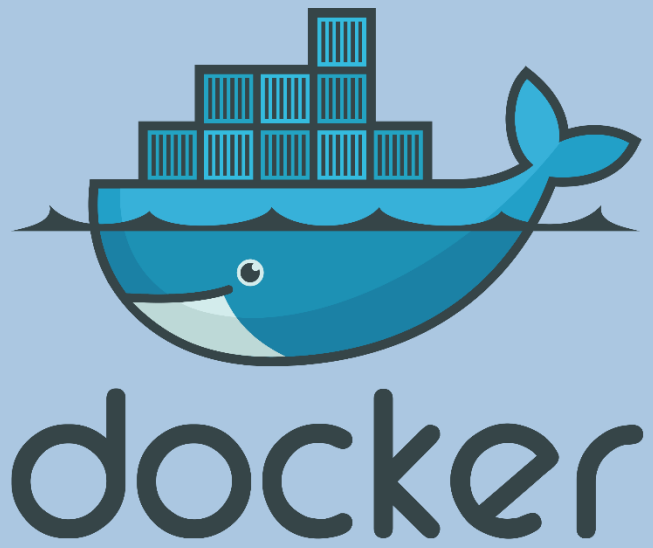

Docker Tutorials

Orchestration using Docker Compose

Introduction

- When working with multiple containers, it can be difficult to manage the starting along with the configuration of variables and links. To solve this problem, Docker has a tool called *Docker Compose* to manage the orchestration, of launching, of containers.

Installation

- This environment has the latest Docker Compose installed. Visit https://docs.docker.com/compose/install/ for instructions on how to install this into your local environment.

Step 1 - Defining First Container❶

- Docker Compose is based on a *docker-compose.yml* file. This file defines all of the containers and settings you need to launch your set of clusters. The properties map onto how you use the docker run commands, however, are now stored in source control and shared along with your code.
- The format of the file is based on YAML (Yet Another Markup Language).
  - *container_name:*
    property: value
    - or options

Step 1 - Defining First Container❷

Task: Defining First Container

- In this scenario, we have a Node.js application which requires connecting to Redis. To start, we need to define our docker-compose.yml file to launch the Node.js application.
  - *web:*

    build: .

Step 2 - Defining Settings

- To link two containers together to specify a links property and list required connections. For example, the following would link to the redis source container defined in the same file and assign the same name to the alias.

```
links:
    - redis
ports:
    - "3000"
    - "8000"
```

Step 3 - Defining Second Container

```
redis:
  image: redis:alpine
  volumes:
    - /var/redis/data:/data
```

Step 4 - Docker Up

- With the created *docker-compose.yml* file in place, you can launch all the applications with a single command of up. If you wanted to bring up a single container, then you can use *up <name>*.

- The *-d* argument states to run the containers in the background, similar to when used with docker run.
- Launch your application using
  - *docker-compose up -d*
  - ```
    Creating tutorial_redis_1
    Creating tutorial_web_1
    ```
    ```

Step 5 - Docker Management

- Not only can Docker Compose manage starting containers but it also provides a way manage all the containers using a single command.

```
$ docker-compose ps
       Name                     Command               State                    Ports
-------------------------------------------------------------------------------------------
tutorial_redis_1        docker-entrypoint.sh    Up                       6379/tcp
                        redis ...
tutorial_web_1          npm start               Up                       0.0.0.0:32769->3000/t
                                                                         cp, 0.0.0.0:32768->80
                                                                         00/tcp
```

typing

- *docker-compose*

Step 6 - Docker Scale

- The scale option allows you to specify the service and then the number of instances you want. If the number is greater than the instances already running then, it will launch additional containers. If the number is less, then it will stop the unrequired containers.
- Scale the number of web containers you're running using the command
  - *docker-compose scale web=3*
- You can scale it back down using
  - *docker-compose scale web=1*

Step 7 - Docker Stop

- As when we launched the application, to *stop a set of containers* you can use the command
  - *docker-compose stop*

- To *remove all the containers* use the command
  - *docker-compose rm*

Reference

- https://www.katacoda.com/courses/docker/11