Orchestration using Docker Compose

Step 1: Defining First Container

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 2: Defining Settings

Docker Compose supports all of the properties which can be defined using docker run.

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.

Additional documentation on the options can be found at <https://docs.docker.com/compose/compose-file/>

TEST

Update our web container to expose the port 3001 and create a link to our Redis container.

Step 3: Defining Second Container:

In the previous step, we used the Dockerfile in the current directory as the base for our container. In this step, we want to use an existing image from Docker Hub as a second container.

To find the second container you simply use the same format as before on a new line. The YAML format is flexible enough to define multiple containers within the same file.

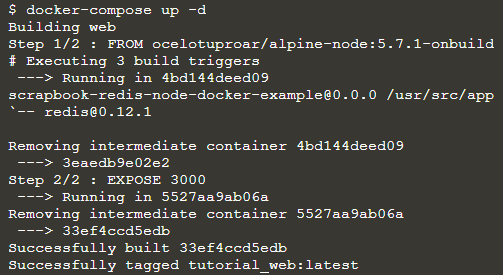
Step 4: DOCKER UP

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.

TASK

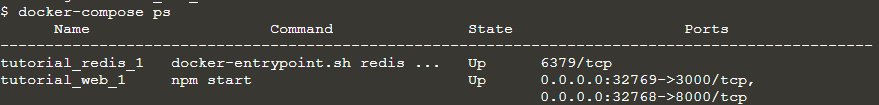
Launch your application using docker-compose up -d



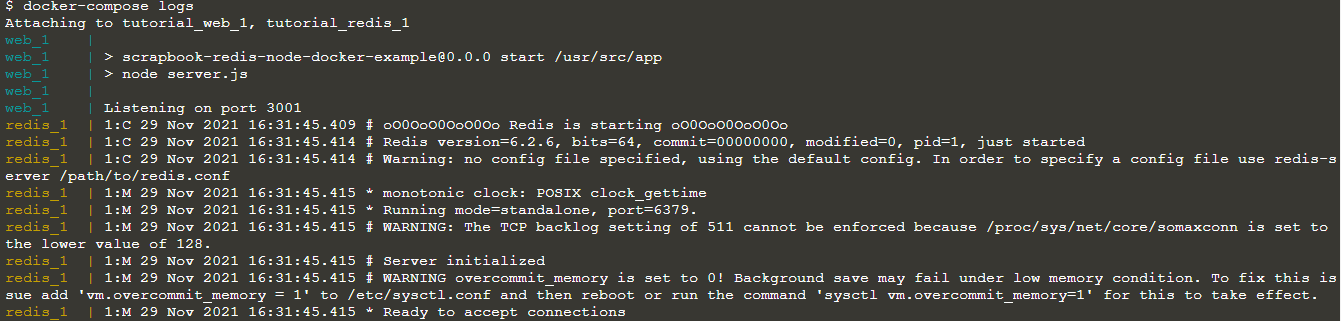
Step 5: Docker Management

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

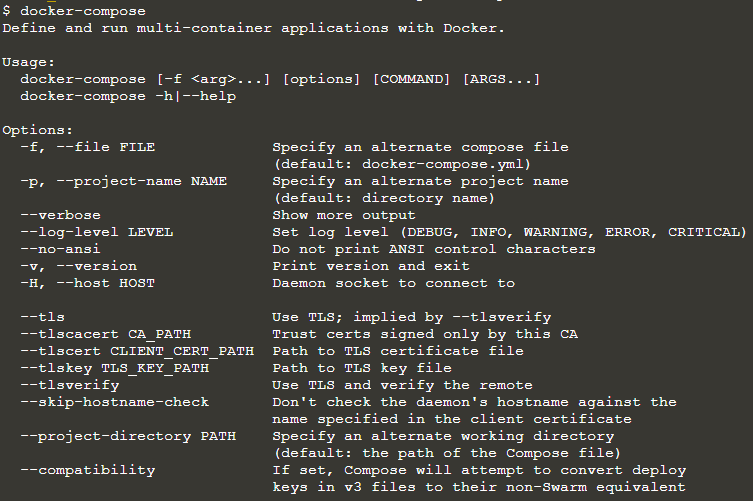
For example, to see the details of the launched containers you can use docker-compose ps



To access all the logs via a single stream you use docker-compose logs



Other commands follow the same pattern. Discover them by typing docker-compose



Step 6: Docker Scale

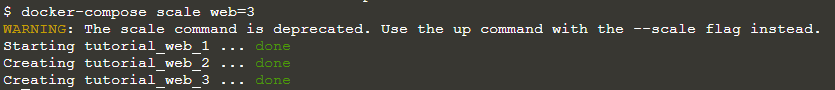
As Docker Compose understands how to launch your application containers, it can also be used to scale the number of containers running.

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.

TASK

Scale the number of web containers you're running using the command

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.

