Docker Containers Tutorial

Seri Lee[*]

Seoul National University
sally20921@snu.ac.kr

January 11, 2021

Containers are a packetized bundle of software that encapsulates everything that is required to run, including all dependencies. It only requires a compatible OS kernel to run autonomously.

Docker is the lead actor in containers for web applications. The best way to describe a container is to think of a process that's surrounded by its own filesystem. This makes containers extremely portable, as they are detached from the underlying hardware and the platform that runs them.

I. Docker Run

To build a container with Docker, we need a definition of its content. The filesystem is created by applying layer after layer.

In this section, we are going to run a "'SimCLR" container on our system. To get started, run the following in the terminal:

```
    docker pull zimmerrol/simclr-pt
```

The `pull` command fetches the zimmerrol/simclr-pt image from the Docker registry and saves it to our system. You can use the `docker images` commnad to see a list of all images on your system.

Let's run a Docker container based on this image.

```
docker run zimmerrol/simclr-pt
```

When you call `run`. the Docker client finds the image, loads up the container and then runs a command in that container. We didn't provide a command, so the container booted up, ran an empty command and then exited.

The `docker ps` command shows you all containers you are cuurently running.

Running the `run` command with the `-it` flag attaches us to an interative tty in the container. Now we can run as many commands in the container as we want. You can exit the container (`exit` and press Enter). Since Docker creates a new container every time, everything should start working again.

Running `docker run` multiple times and leaving stray containers will eat up disk space. Hence, as a rule of thumb, clean up containers once you are done with them. To do that, run `docker rm` command. Just copy the container IDs from `docker ps -a` and paste them alongside the command.

```
docker rm $(docker ps -a -q -f status=exited)
```

This command deletes all containers that have a status of `exited`. The `-q` flag only returns the numeric IDs and `-f` filters output based on conditions provided.

`--rm` flag can be passed to `docker run` which automatically deletes the container once it's exited from.

In later versions, the `docker container prune` command can be used to achieve the same effect. Lastly, you can also delete images that you no longer need by running `docker rmi`.

i. Terminology

- **Images** The blueprints of our applications which form the basis of containers.
- **Containers** Created from Docker images and run the actual application. We create a container using `docker run`. A list of

[*]A thank you or further information

running containers can be seen using the `docker ps` command.
- **Docker Daemon** The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system which client talks to.
- **Docker Client** The Command line tool that allows the users to interact with the daemon.
- **Docker Hub** A registry of Docker images.

II. Deploying with Docker