HARVARD School of Engineering and Applied Sciences

CS205: Computing Foundations for Computational Science, Spring 2021

IACS INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE AT HARVARD UNIVERSITY

Guide: Docker on AWS

Ignacio M. Llorente, Simon Warchol

v3.0 - 14 February 2020

Abstract

This is a guideline document to show the necessary actions to set-up Docker and run Jupyter Notebook on Ubuntu (18.04).

Requirements

- **First you should have followed the Guide "First Access to AWS"**. It is assumed you already have an AWS account and a key pair, and you are familiar with the AWS EC2 environment.

Acknowledgments

The author is grateful for constructive comments and suggestions from David Sondak, Charles Liu, Matthew Holman, Keshavamurthy Indireshkumar, Kar Tong Tan, Zudi Lin, Nick Stern, Dylan Randle.

1. Spin up EC2 instance

- You should create a new security group (docker) to open access to Jupyter and Docker default ports at https://console.aws.amazon.com/ec2/v2/home?#SecurityGroups:
    - SSH: Port Range: 22, Source: Anywhere
    - HTTP: Port Range: 80, Source: Anywhere
    - HTTPS: Port Range: 443, Source: Anywhere
    - Custom TCP Rule: Port Range: 2376, Source: Anywhere
    - Custom TCP Rule: Port Range: 8888, Source: Anywhere
- The free tier `t2.micro` is sufficient for many applications. When selecting an Amazon Machine

Image (AMI) select **Ubuntu 18.04, SSD Volume Type**. Continue with default options until the Configure Security Group section and select your created security group (if you did not create the security group 'docker' before, you can also create it now).

- Connect to your newly created instance with SSH (PuTTY on Windows)

2. Install Docker

- Update the installed packages and package cache on your instance

```
$ sudo apt-get update
```

- Install the most recent Docker Community Edition package from the Ubuntu repo (see that the Docker installation package available in the official Ubuntu 16.04 repository may not be the latest version)

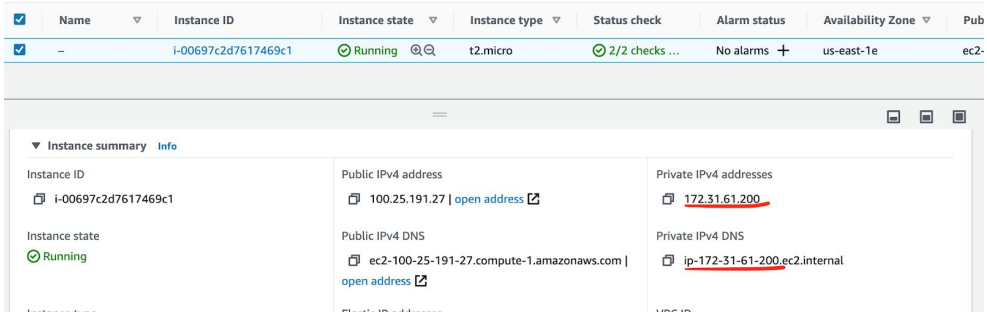
```
$ sudo apt-get install -y docker.io
```

- If you are using an AWS VM, you should include the internal hostname and IP to `/etc/hosts`.

Update `/etc/hosts` and add your hostname and IP below localhost in the following format

```
172.30.4.210 ip-172-30-4-210
```

Both are found on the far right of the EC2 instance details page. Note that you'll omit the `ec2.internal` from your hostname



You can edit `/etc/hosts` with vim/emacs/nano, but you'll want to run it with `sudo` to have the permissions to edit the file. In my case I did

```
$ sudo vim /etc/hosts
```

To confirm you were successful, type the following and confirm that your hostname is in the output

```
$ cat /etc/hosts
127.0.0.1 localhost
172.30.4.210 ip-172-30-4-210
```

- Start the Docker service

```
$ sudo service docker start
```

- Add the `ubuntu` user to the `docker` group so you can execute Docker commands without using `sudo`

```
$ sudo usermod -a -G docker ubuntu
```

- Log out and log back in again to pick up the new docker group permissions. Verify that the `ubuntu` user can run Docker commands without sudo

```
$ docker info
```

- Verify Docker installation

```
$ docker -v
```

3. Working with Docker

- To view all available subcommands, type

```
$ docker
```

- Docker containers are run from Docker images. By default, it pulls these images from Docker Hub, a Docker registry managed by Docker, the company behind the Docker project. Anybody can build and host their Docker images on Docker Hub, so most applications and Linux distributions you'll need to run Docker containers have images that are hosted on Docker Hub. To check whether you can access and download images from Docker Hub, type

```
$ docker run hello-world
```

- You can search for images available on Docker Hub by using the docker command with the `search` subcommand. For example, to search for the jupyter image, type

```
$ docker search jupyter
```

4. Run the Jupyter Image

- Pull the Docker image with the jupyter/scipy-notebook

```
$ docker pull jupyter/scipy-notebook
```

- Once you have pulled the image, it is now present in your docker images cache. Run the jupyter notebook server linking port 8888 on the host machine (VM on EC2) to the port 8888 on which the jupyter notebook server is running in the Docker container

```
$ docker run -p 8888:8888 jupyter/scipy-notebook

    Executing the command: jupyter notebook

    ...
```

```
The Jupyter Notebook is running at:

http://86da73c7b195:8888/?token=69fd02a3fd08c7ef96a4bbc079841c71b490ad4e36f979

or

http://127.0.0.1:8888/?token=69fd02a3fd08c7ef96a4bbc079841c71b490ad4e36f97961
```

- The output provides you with an application token that you should use to access the notebook server through a browser. **Replace localhost (127.0.0.1) with the public IP (which can be found on the EC2 instances page)** of the VM

- Try the notebook

Stop your instances when are done for the day to avoid incurring charges
Terminate them when you are sure you are done with your instance