

Docker install on EC2 and usage: Lab guide

Monday, February 15, 2021 12:36 PM

Introduction

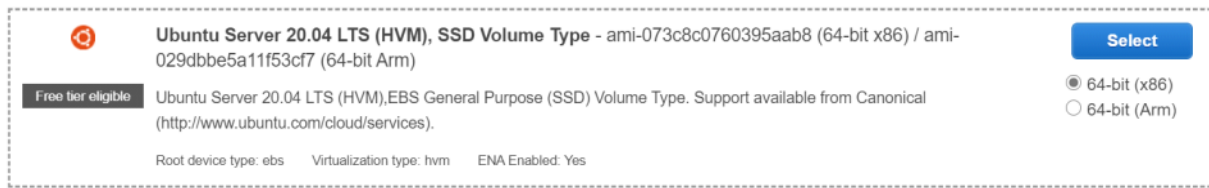
In this guide, we will cover installing Docker on an AWS EC2 instance running Ubuntu 20.04 server and then we will run some useful commands on that. We will also run a simple static website on Docker.

Launching the EC2 instance


This guide assumes that the user is proficient in AWS, so only necessary details have been provided.

Choose your region as us-east-1 -> US East (N. Virginia)

In EC2 service, click on Launch instances and choose Ubuntu server 20.04:



For instance type, choose t2.micro (free tier eligible):

	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
---	----	--------------------------------	---	---	----------	---	-----------------	-----

In configure instance details, choose default VPC, Subnet: no preference. Leave the remaining options as default.

In Add storage, leave the default (8 GB SSD) as root volume.

In Add tags, add the keypair: value -> application:docker.

We should be able to login to the machine from anywhere and we would possibly add a new website to a container.

In security group, add a new security group ssh-login-and-website. A rule for ssh login will be created for you, but it is open to the world. To restrict it, click on Custom under source and choose My IP. The caveat of this method is, every time your laptop is assigned a new dynamic IP, this rule should be updated to choose your current IP.

Add a new rule for website access: Make type as HTTP, default port 80 and other information will be populated automatically.

In the next screen, if you have already a key pair generated before, click on choose an existing key pair and provide its name. Else click on "Create a new key pair" and provide a name (E.g., docker-keypair). Don't forget to download the key pair (this will have the extension .pem file). Please safeguard this file.

In a few minutes, the instance will be running. If you are using putty on Windows to connect to the instance, you need to first convert the .pem file to .ppk file by using puttygen program. Now, use the DNS name or IP of this instance in Putty, along with the .ppk file to connect to the instance (user: ubuntu).

Install docker pre-requisites

Run the commands:

```
sudo apt update
sudo apt install curl apt-transport-https ca-certificates software-properties-
common
```

Install Docker

Import the GPG key to your system to verify packages signature before installing them. Then add the Docker repository to your Ubuntu system.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add
```

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
```

Update apt index and then install docker:

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io
```

Now let us verify that docker is installed, by running a simple docker image:

```
sudo docker run hello-world
```

which will output:
Hello from Docker.
This messages shows that your installation appears to be working correctly.

To know the status of docker command, run:

```
sudo systemctl status docker
```

To stop, start and restart docker engine in future, you would run:

```
sudo systemctl stop docker
sudo systemctl start docker
sudo systemctl restart docker
```

At this, we have run docker successful on AWS EC2 instance.

Running some docker commands

The user ubuntu does not have permission to run docker commands. Therefore we have to add sudo before every docker command to run. There are ways to add this user to sudoers to avoid mentioning sudo, but for simplicity, in this lab we will simply login to root and run docker commands. (Note: Don't do this in a Prod environment, as running as root is not a good practice).

Temporarily login to root by running the command

```
sudo su (then the prompt will change to #)
```

Let us play with a busybox container. Let us pull the image from Docker Registry to local first:

```
docker pull busybox
```

Then run

[docker images](#)

which will show all local images.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
busybox	latest	22667f53682a	2 weeks ago	1.23MB
hello-world	latest	bf756fb1ae65	13 months ago	13.3kB

Now let us run

[docker run busybox](#)

Though it ran successfully, it produced no output, as we didn't provide a command to run as part of this container, so it ran an empty command and exited.

Now let us try this one:

[docker run busybox echo "Welcome to Container Capability!"](#)

It outputs the string after the echo command and then exits.

Now let us run a command to find out what containers are running.

[docker ps](#)

As expected, no containers are running (they are all exited).

To include also the ones that exited, run this command:

[docker ps -a](#)

And you get the output:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
112e9fc3836f	busybox	"echo 'Welcome to Co..."	2 minutes ago	Exited (0) 2 minutes ago
74e7f655a9a3	busybox	"sh"	4 minutes ago	Exited (0) 4 minutes ago
5e0064e2a6ad	hello-world	"/hello"	23 hours ago	Exited (0) 23 hours ago

root@ip-172-31-63-144:/home/ubuntu#

How can we run more than one command in a container? The best way to do that will be to open a shell inside a container using this command

[docker run -it busybox sh](#)

Which produces a prompt `/`, Now we are inside the container, so we can run commands to our heart's content.

We can come out this container by an `exit` command.

To remove a stopped container, we can use a `docker rm` command:

[docker rm 112e](#) (Note that, most of the time it is enough if we just provide the first 4 digits of the container ID. See the `docker ps -a` output above). We just removed one of the containers in the list.

Running a Static Website on Docker

Let us try running a static website example hosted in Docker repository:
(More details at: [A Docker Tutorial for Beginners \(docker-curriculum.com\)](https://docker-curriculum.com))

```
docker run -d -p 80:80 --name static-site prakhar1989/static-site
```

-d will detach this long running static site container from the terminal (so we can continue to run commands)

--name is the name of the container (show that you can provide your own name)

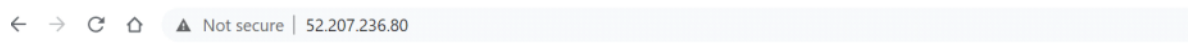
prakhar1989/static-site is the image in the Docker repository.

-p maps the container port (first number) to the host port (second number)

Since we have allowed port 80 as incoming HTTP port in our security group, go to a browser and try

<http://<PublicIP-of-the-EC2-instance>>

It will show this webpage:

A screenshot of a web browser's address bar. It shows navigation icons (back, forward, refresh, home) on the left, followed by a warning icon and the text 'Not secure | 52.207.236.80'.

Hello Docker!

This is being served from a **docker** container running Nginx.

Conclusion

In this lab, we learned how to install Docker on an AWS EC2 instance, then ran a few Docker commands and then ran a simple docker static website and exposed it to public.

Edited by ramesh.In on 16 February, 2021