

DevOps Tools You Should Know: Docker

Background

Interested in **containerization** across any cloud platform? Choose Docker.

Docker is a containerization platform that provides easy way to containerize your applications. If you use Docker, you will build container images, run the images to create containers and also push these containers to container registries such as DockerHub.

In other words, Docker implements containerization.

Common docker commands you should know:

- `docker run`: runs a Docker container.
- `docker build`: builds image from Dockerfile.
- `docker images`: lists docker images on the host machine.

DevOps Project Outline

Your manager asked you to create an Ubuntu EC2 instance and install docker.

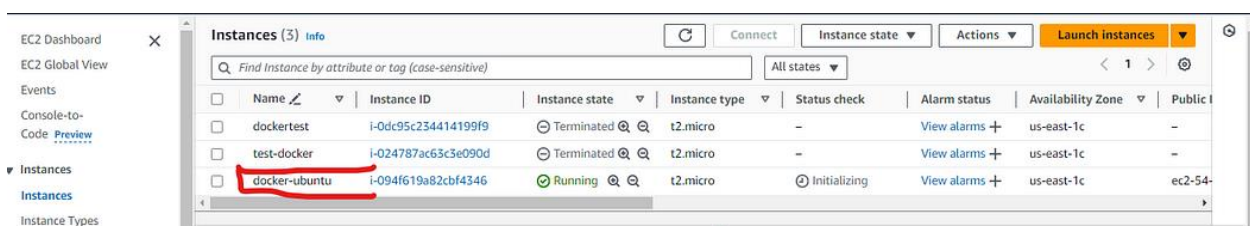
Prerequisite

- Access to AWS account.
- Install docker.
- Access to dockerhub.com

Let's have fun.

1- Create EC2 instance

- Log in to your AWS account
- I have a tutorial on how to create EC2 instance. [Read more](#).
- I created an EC2 instance named *docker-ubuntu*.



The screenshot shows the AWS Management Console 'Instances' page. The table lists three EC2 instances. The instance named 'docker-ubuntu' with ID 'i-094f619a82cbf4346' is highlighted with a red box and is in a 'Running' state. The other two instances, 'dockertest' and 'test-docker', are in a 'Terminated' state.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input type="checkbox"/>	dockertest	i-0dc95c234414199f9	Terminated	t2.micro	-	View alarms	us-east-1c	-
<input type="checkbox"/>	test-docker	i-024787ac63c3e090d	Terminated	t2.micro	-	View alarms	us-east-1c	-
<input type="checkbox"/>	docker-ubuntu	i-094f619a82cbf4346	Running	t2.micro	Initializing	View alarms	us-east-1c	ec2-54-

2- Connect EC2 instance via SSH connection

- I've created a tutorial detailing the steps to SSH connect to EC2 through MobaXterm on your Windows laptop. [Discover more about it](#) .
- Alternatively, you can explore another tutorial explaining how to connect to EC2 instances via SSH using Linux. [Read on for more details](#) .

3- Install docker in your EC2 instance

- After connecting to your EC2, verify if the connection was successful.

```
ubuntu@ip-172-31-25-175:~$ sudo apt update
```

- Now, install docker by typing ***sudo apt install docker.io -y***

```
ubuntu@ip-172-31-25-175:~$ sudo apt install docker.io -y
```

- Verify if docker is running:

```
no VM guests are running validated hypervisor (qemu) binaries on this host
ubuntu@ip-172-31-25-175:~$ sudo systemctl status docker
```

- As you can see below, docker is running.

```

● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sun 2024-05-12 18:19:07 UTC; 27min ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 2497 (dockerd)
      Tasks: 9
     Memory: 32.5M (peak: 33.9M)
        CPU: 435ms
    CGroup: /system.slice/docker.service
            └─2497 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

May 12 18:19:06 ip-172-31-25-175 systemd[1]: Starting docker.service - Docker Application Container Engine...
May 12 18:19:06 ip-172-31-25-175 dockerd[2497]: time="2024-05-12T18:19:06.780126928Z" level=info msg="Starting up"
May 12 18:19:06 ip-172-31-25-175 dockerd[2497]: time="2024-05-12T18:19:06.782553280Z" level=info msg="detected 127.0.0.53 name"
May 12 18:19:06 ip-172-31-25-175 dockerd[2497]: time="2024-05-12T18:19:06.899047586Z" level=info msg="Loading containers: sta"
May 12 18:19:07 ip-172-31-25-175 dockerd[2497]: time="2024-05-12T18:19:07.199783210Z" level=info msg="Loading containers: don"
May 12 18:19:07 ip-172-31-25-175 dockerd[2497]: time="2024-05-12T18:19:07.285811474Z" level=info msg="Docker daemon" commit=25
May 12 18:19:07 ip-172-31-25-175 dockerd[2497]: time="2024-05-12T18:19:07.286298492Z" level=info msg="Daemon has completed in"
May 12 18:19:07 ip-172-31-25-175 dockerd[2497]: time="2024-05-12T18:19:07.341288650Z" level=info msg="API listen on /run/dock"
May 12 18:19:07 ip-172-31-25-175 systemd[1]: Started docker.service - Docker Application Container Engine.

```

4- Add user to docker group

- Now, try to run docker using ***docker run hello-world***.
- If you encounter an error or receive a 'permission denied' message, it's likely because Docker always operates with root privileges.

```

ubuntu@ip-172-31-25-175:~$ docker run hello-world
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/create": dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.

```

- Add ubuntu user to docker group and type the command below.

```

ubuntu@ip-172-31-25-175:~$ sudo usermod -aG docker ubuntu

```

- Log out and log back in to your EC2.
- Now, type ***docker run hello-world***. Docker is running.

```
ubuntu@ip-172-31-25-175:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:a26bfff933ddc26d5cdf7faa98b4ae1e3ec20c4985e6f87ac0973052224d24302
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

ubuntu@ip-172-31-25-175:~$ █
```

Your Docker user profile isn't activated automatically; you'll need to create it manually.

5- Create your first docker file

- Clone this repo from GitHub. Click [here for the url](#).
- Locate the docker file. Open the file.

```
FROM ubuntu:latest
```

```
# Set the working directory in the image
```

```
WORKDIR /app
```

```
COPY . /app
```

```
# Install the necessary packages
```

```
RUN apt-get update && apt-get install -y python3 python3-pip
```

```
# Set environment variables
```

ENV NAME World

```
# Run a command to start the application
```

```
CMD ["python3", "app.py"]
```

2

2

2

25

23

2

2

2

2

2

2

2

25

2

2

2

2

2

2

"Dockerfile" 16L, 362B

Cheat sheet: You can run this docker file or create another one.

- Login to Docker.com using your username (Djcloudguru) and password (K*****). To do so, enter **docker login** in the terminal.

```
ubuntu@ip-172-31-25-175:~/Docker-Zero-to-Hero/examples/first-docker-file$ docker login
```

5- Run your First Docker Image

- Remember we are in the EC2 instance still. Locate docker file and execute this command ***docker build -t djcloudguru/my-first-docker-image:latest .*** (do not forget to add the *.* at the end of the command)

```
ubuntu@ip-172-31-25-175:~/Docker-Zero-to-Hero/examples/first-docker-file$ docker build -t djcloudguru/my-first-docker-image:latest .
```

- The result is as follows.

```
Successfully built 607f547839f0
Successfully tagged djcloudguru/my-first-docker-image:latest
```

Remember the main purpose of this tutorial is to run application.

- Execute the command below. You'll get Hello world as output. Your application is running.

```
ubuntu@ip-172-31-25-175:~/Docker-Zero-to-Hero/examples/first-docker-file$ docker run -it djcloudguru/my-first-docker-image:latest
Hello World
```

6- Run your First Docker Container

- Enter **docker login** in the terminal to verify if you are still logged in.

```
ubuntu@ip-172-31-25-175:~/Docker-Zero-to-Hero/examples/first-docker-file$ docker login
```

- It's a successful login.

```
Authenticating with existing credentials...  
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
Login Succeeded
```

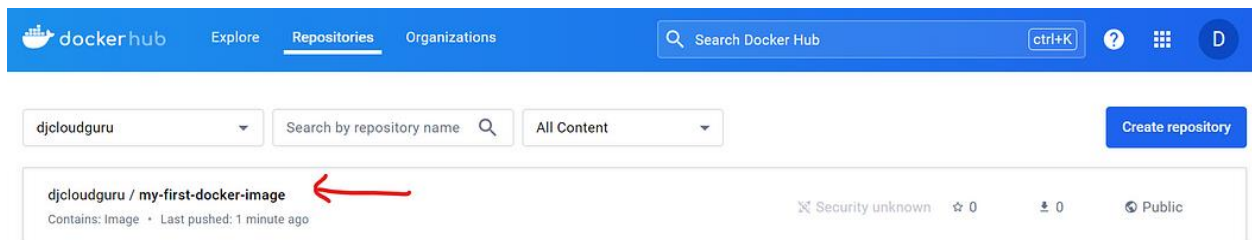
- Now, enter this command and share the image.

```
ubuntu@ip-172-31-25-175:~/Docker-Zero-to-Hero/examples/first-docker-file$ docker push djcloudguru/my-first-docker-image:latest
```

- It's a successful push.

```
The push refers to repository [docker.io/djcloudguru/my-first-docker-image]  
5e777a4ad605: Pushed  
0f19311f3b67: Pushed  
8cf7a8bfe099: Pushed  
80098e3d304c: Pushed  
latest: digest: sha256:73d297fac2dab440a870eecba372742f8c4c7c8e67cbd1e0c077ca78262b59ea size: 1155
```

- After this image is pushed. Login into your docker hub and look it up.



- This docker image can be shared to everyone now.

Voila! You've created a docker file, run a docker image, and push a docker container.