Table of Contents

[LAB 01: DOCKER INSTALLATION 2](#_Toc102385484)

[LAB 02: BASIC DOCKER COMMANDS 2](#_Toc102385485)

[LAB 3: BUILD A DOCKE IMAGE 3](#_Toc102385486)

[LAB 4: BUILDING A CUSTOMER CISCO LEARNING LABS CONTAINER 4](#_Toc102385487)

# LAB 01: DOCKER INSTALLATION

Install docker on ubuntu server by following the below guide:

<https://docs.docker.com/engine/install/ubuntu/>

# LAB 02: BASIC DOCKER COMMANDS

Step 1: What is the version of Docker Server Engine running on the Host?

Step 2: How many containers are running on this host?

Step 3: How many images are available on this host?

Step 4: Run a container using the redis image in background

Step 5: Stop the container you just created

Step 6: How many containers are RUNNING on this host now?

Step 7: Create some containers.

Step 8: How many containers are RUNNING on this host now? How many containers are PRESENT on the host now? (Including both Running and Not Running ones)

Step 9: For each container, identify image name, container name, container id

Step 10: Delete all containers from the Docker Host

Step 11: Delete the ubuntu Image

Step 12: You are required to pull a docker image which will be used to run a container later. Pull the image nginx:1.14-alpine

Only pull the image, do not create a container.

Step 13: Run a container with the nginx:1.14-alpine image and name it webapp

Hint: Run the command docker run --name webapp nginx:1.14-

Step 14: Run an instance of redis in background and map port 8080 on the container to 38282 on the host.

* Image: redis
* Container port 8080
* Host Port: 38282

**Step 15: Cleanup:** Delete all images on the host

Remove containers as necessary

(docker image prune -a)

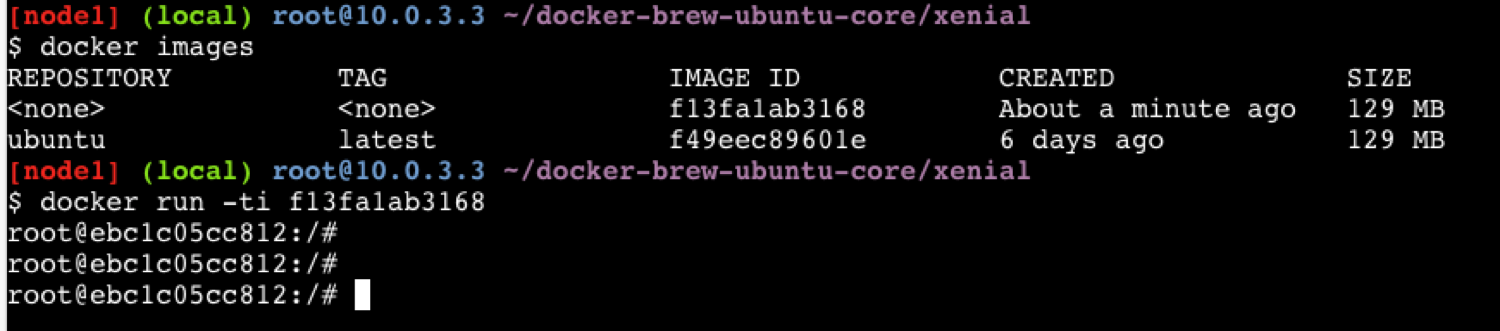
# LAB 3: BUILD A DOCKE IMAGE

We'll build our own Ubuntu Docker image instead of using the one from Docker Hub.

1. Download the Ubuntu Dockerfile and its dependencies.
2. git clone --single-**branch** --**branch** dist-amd64 https://github.com/tianon/docker-**brew-ubuntu-core.git**
3. Change directory to the cloned repository and ensure that the Dockerfile is present.
4. **cd** docker-brew-ubuntu-core/xenial
5. **ls** .
6. Build a new image using Docker and the Dockerfile.
7. docker **build** .

Docker prints progress messages as it builds the Ubuntu image. Once it finishes building the image, Docker assigns it a randomly-chosen name.

1. Run the command docker images.  
   Docker lists all the images on your system, including both the newly-built one and any that you previously downloaded. You can tell which image is the one you just created by examining the list; it's the only image that isn't associated with a repository:



1. Run the newly created Docker image by giving the randomly-chosen ID to Docker:
2. docker run -ti <your image ID>

The newly-created image behaves exactly the same way as the Ubuntu image from Docker Hub, because it is built from the same Dockerfile. You now have a bash root prompt.

1. Run the following command to confirm standard Ubuntu directories are installed:
2. **ls**

# LAB 4: BUILDING A CUSTOMER CISCO LEARNING LABS CONTAINER

To modify the image that we just built, all we have to do is change the Dockerfile and ensure that all required resources are available in the repository. Let's make an image that's more specific to our application's needs.

Let's change the image to display the following message:

Hello **from** DevNet!

To make the image display the above message we will:

1. Create a new Python script.
2. Edit the Dockerfile to include that script in the container build.
3. Change the Dockerfile to install Python in the container (remember, all dependencies must be present in the container).
4. Build and test the new container.

To update the image, follow the below instructions:

1. Clone the Github repository that contains the Dockerfile for this section using the following command:
2. git clone https://github.**com**/CiscoDevNet/container-**intro**-devnet.git
3. **cd** container-**intro**-devnet

The cloned repository contains these files:

* + hellodevnet.py
  + Dockerfile

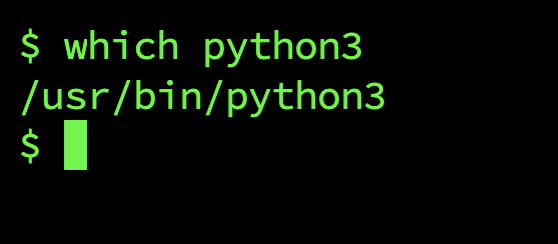
The "hellodevnet.py" file is our custom application. It contains the following Python code:

#!/usr/bin/python3

print("Hello from DevNet!")

To determine the path to the python interpreter (#!/usr/bin/python3), at the prompt type:

which python



The Dockerfile contains the instructions required to build the custom image:

**FROM** ubuntu

**RUN** apt-get update

**RUN** apt-get -y install python3.10

**COPY** hellodevnet.py /hellodevnet.py

**RUN** ["chmod", "+x", "/hellodevnet.py"]

**CMD** ["/hellodevnet.py"]

This Dockerfile says:

* + FROM ubuntu  
    Extend the existing ubuntu public Docker image. Our previous examples built an image from scratch. In this case we begin with a previously-built container image and extend with our own customisations.
  + RUN apt-get update  
    Ensure the package-management tools in the base Ubuntu container are updated to use the latest software.
  + RUN apt-get -y install python  
    Use apt-get to install Python and all its dependencies in the container. The reason to build this container on an existing Ubuntu image is that we could use apt-get to install needed software.
  + COPY hellodevnet.py /hellodevnet.py  
    Copy the Python program from the local directory into the container as /hellodevnet.py.
  + RUN ["chmod", "+x", "/hellodevnet.py"]  
    Grant permission to execute the /hellodevnet.py file
  + CMD ["/hellodevnet.py"]  
    Run the Python program when the container starts up.

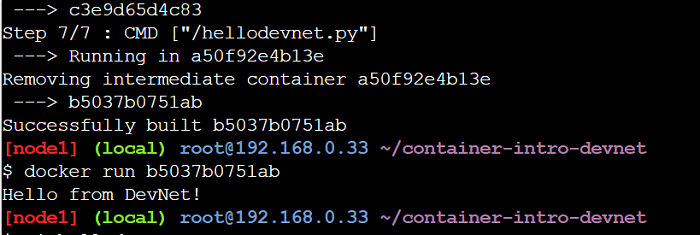
**Note:** Instead of cloning this GitHub repository, you can also add the above mentioned Python code in the hellodevnet.py file and Dockerfile commands in the Dockerfile using the vi command in the terminal and continue with step 2.

1. Build the Docker image using the following command:
2. docker **build** .

Docker prints progress messages as it builds the Ubuntu image. Once it finishes building the image, Docker displays Succesfully built <CONTAINER ID> message.  
Make note of the newly built <CONTAINER ID> so that you can use it in the next step.

1. Run the new container using the following command.  
   Use the <CONTAINER ID> that you recieved from the docker build . command output.
2. docker run <CONTAINER ID>

The "Hello from DevNet!" message is displayed in the container terminal.



Congratulations, you've just built and run a custom Docker image!