**Installation of Docker in Ubuntu Server**

**What is Docker:**

Docker is an application that makes it simple and easy to run application processes in a container, which are like virtual machines, only more portable, more resource-friendly, and more dependent on the host operating system.

**Docker Images:**

In Docker, everything is based on Images. An image is a combination of a file system and parameters. Let’s take an example of the following command in Docker.

Docker containers are based on Docker images. A Docker image is a binary that includes all of the requirements for running a single Docker container, as well as metadata describing its needs and capabilities. You can think of it as a packaging technology. Docker containers only have access to resources defined in the image, unless you give the container additional access when creating it. By deploying the same image in multiple containers across multiple hosts and load balancing between them.

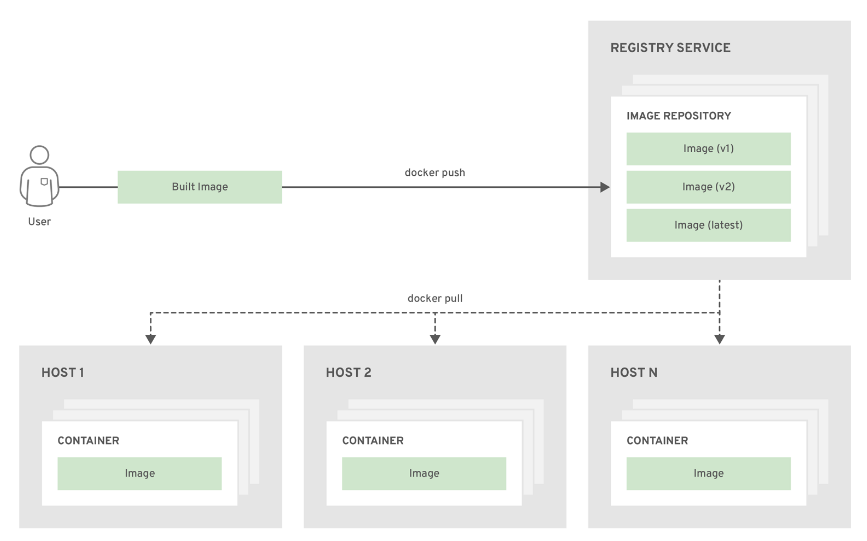
**Docker Containers:**

[Docker container](https://www.sdxcentral.com/listings/docker-inc/) is an [open source](https://www.sdxcentral.com/directory/nfv-sdn/open-source-projects/) software development platform. Its main benefit is to package [applications](https://www.sdxcentral.com/term/application-2/) in “containers,” allowing them to be portable among any system running the Linux operating system (OS). [Container technology](https://www.sdxcentral.com/cloud/containers/definitions/what-are-containers-like-docker-linux-containers/) has been around for a while, but momentum and hype around Docker’s approach to containers have pushed this approach to the forefront. It is one form of container technology.

**Docker Registries:**

A Docker registry is a service for storing and retrieving Docker images. A registry contains a collection of one or more Docker image repositories. Each image repository contains one or more tagged images. Docker provides its own registry, the [Docker Hub](https://registry.hub.docker.com/), but you may also use private or third-party registries.

Here we represent the relationship between Docker Containers Images and registries in Diagrammatical Manner.



Now let’s see how Docker Installation will be in Ubuntu Servers

**Installation Steps of Docker in Ubuntu**

Step 1 — Installing Docker

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

Add the Docker repository to APT sources

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**

Next, update the package database with the Docker packages from the newly added repo

**sudo apt-get update**

Make sure you are about to install from the Docker repo instead of the default Ubuntu 16.04 repo:

**apt-cache policy docker-ce**

Notice that docker-ce is not installed, but the candidate for installation is from the Docker repository for Ubuntu 16.04. The docker-ce version number might be different

**sudo apt-get install -y docker-ce**

**Docker Commands:**

To know the running containers followed by the command

**Sudo Docker ps**

To list out the containers which are running or which or stopped

**Sudo docker ps –a**

To enter in to the docker container

**Sudo docker-enter container name (wish to enter name of the container welcome)**

To search images online repo

**Sudo docker search images**

If you want to search official images

**Sudo docker search Jenkins or Ubuntu python**

These are all some official images its wish to pull

To pull the docker images

**Sudo docker pull images**

Then the images will be downloaded

To stop the docker Container

**Sudo docker stop container name**

If we want to restart the docker container followed by this command

**Sudo docker restart container name**

To create a Docker container followed by the command

**sudo Docker run -it docker\_image\_name - to create with default name**

Command to delete a container (Containers needs to be turned off manually before deletion)

**sudo docker rm container name/container\_id**

Here we are using the docker in our organization standard will be described below

Initially we used to create the mount points for the purpose of docker.

The mount points named as

To create the mount points

Mkdir odoo\_addons

Mkdir odoo\_logs

Mkdir odoo\_config

Mkdir odoo\_filestore

Here we are configured the addons path in docker to deploy the code manually.

Here we configured the logs to tail the logs of the application

Here we configured the config to maintain the configuration file of the docker.

Here we configured the filestore to store the database in docker.

This is how we configured these 4 mount points in docker container.

Customized command for odoo\_db\_update

**sudo docker exec -t container \_name /bin/bash ""./opt/odoo/odoo.sh dbname internalP""**

**Docker run command to create the container with mount points and it’s locations mentioned below :**

**sudo docker run -v /pappaya/narayanatraining/filestore:/var/lib/odoo -v /pappaya/narayanatraining/config:/etc/odoo -v /pappaya/narayanatraining/addons:/opt/odoo/extra-addons -v /pappaya/narayanatraining/logs:/var/log/odoo -p 8093:8069 --name narayanatraining -t narayana >>/dev/null 2>>/dev/null &**

**if we want to enter in to docker container there is a command to be passed and it’s mentioned below :**

**sudo docker jpnz meter**

**here the docker port no is very mandatory since we are using the odoo frame work in our organization port no starts from initially 8069.**

**Once we start the deployment manually by using the docker**

**We used to connect to the mount points named as addons where we configured with the docker**

**Will pass the repository url to connect to the code**

**wget** [**http://172.28.166.171:8081/repository/DIS/DISAdd\_17.tar.gz**](http://172.28.166.171:8081/repository/DIS/DISAdd_17.tar.gz)

**next is we need to tar this file**

**tar -xvzf DISAdd\_17.tar.gz**

**rm -rf DISAdd\_17.tar.gz**

**next step Is we need to restart the Docker container .**

**sudo docker restart container name**