Docker Installation

Raghuram PSS | QShore

Docker Installation on Ubuntu 16.04 **(**The below set of instructions stand good for Ubuntu 16.04**)**

apt-get update		
Add the GPG key for the official Docker repository to the system:	curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -	You should see "OK"
Add the Docker repository to APT sources:	sudo add-apt-repository \ "deb [arch=amd64] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) \ stable"	We are trying to add the docker path for the current OS version "xenial" Alternately try the below if it doesn't work.. add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable" apt-cache search docker-ce
Update the package manager database again	apt-get update	
Just to ensure that we are about to install Docker repo	apt-cache policy docker-ce	You should see something like this… docker-ce: Installed: (none) Candidate: 17.03.1~ce-0~ubuntu-xenial Version table: 17.03.1~ce-0~ubuntu-xenial 500 500 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages 17.03.0~ce-0~ubuntu-xenial 500 500 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages
Install Docker community edition	apt-get install -y docker-ce	
Check the Docker service	service docker status	You should see status as Active (Running)
See the installed Docker version	docker --version	Docker version 17.03.1-ce, build c6d412e

Docker Intro

Docker is a containerization platform which packages your application and all its dependencies together in the form of Containers so as to ensure that your application works seamlessly in any environment be it Dev. or test or production.

Image is the definition of environment. Container is a running instance of an image.
Docker engine has the ability to run Docker containers; It is an API.

- Docker is an open source container based technology. Container allows to package up application and all its parts and dependencies.
- You can easily configure environments and share them on any platform. Each environment can contain many containers

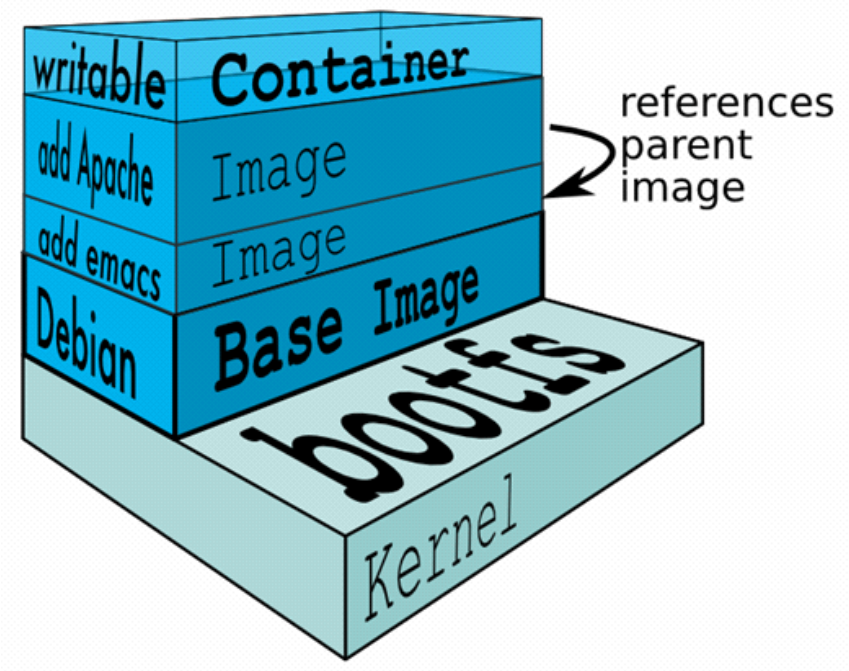
Docker Image: Is the definition of the environment
Docker container: Is a running instance of an image
Docker File: For building a Docker image. It contains our project code, installation of apache tomcat, nodeJS etc. It is a complete application wrapped up in our machine. This image sits on top of our machine. From that image we can run as many containers as we need depending on the processing power of our machine. Only thing it requires to run the image is, Docker installed in the machine.

Docker Daemon: Runs on host machine and creates and manages the Docker objects…

- Images
- Containers
- Networks
- Volumes
- Data

There is no hardware emulation with container technologies. The Guest OS gets installed into the directory of the host OS. Containers use the same kernel of the host OS. So, containers can't have a separate OS. e.g. You can have windows container on Windows server only.
Since the Kernel is the same for all the Linux OS distributions, we can have Cent OS, SLES or Ubuntu container on top of Linux Redhat host OS.

Docker Layers:

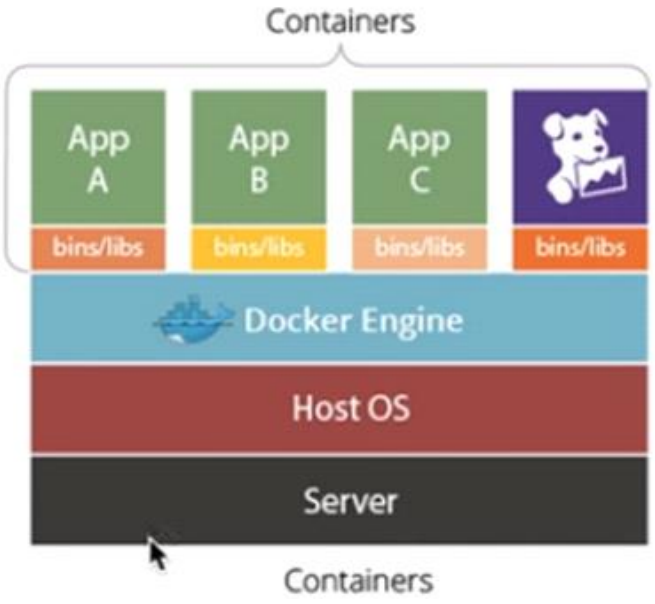


In a traditional Linux boot, the kernel first mounts the root File System as read-only, checks its integrity, and then switches the whole rootfs volume to read-write mode.

Layer
When Docker mounts the rootfs, it starts read-only, as in a traditional Linux boot, but then, instead of changing the file system to read-write mode, it takes advantage of a union mount to add a read-write file system over the read-only file system. In fact there may be multiple read-only file systems stacked on top of each other. We think of each one of these file systems as a layer.

At first, the top read-write layer has nothing in it, but any time a process creates a file, this happens in the top layer. And if something needs to update an existing file in a lower layer, then the file gets copied to the upper layer and changes go into the copy. The version of the file on the lower layer cannot be seen by the applications anymore, but it is there, unchanged.

We call the union of the read-write layer and all the read-only layers a union file system.



Linux Containers:
Lightweight virtualization provided by libraries inside the Linux kernal.

cgroups	cpparmor	namespaces	netlink
capabilities	netfileter	selinux	

Docker is a utility that improves the usability of Linux containers by providing:
A layered file system (aufs) - Docker images
A cross platform execution engine (Docker Engine)
A social space to share common libraries (Docker Hub)

Common Use Cases:

Shared Hosting	
Microservices	For highly scalable
Lightweight testing	

Advantages of container-based architecture…

Development Environment

- Just install Docker, the application and the configuration resides in the container
- Isolated Environment
- Identical to staging/QA/production
    - Solves "It works on my machine"
- Only one virtual machine required

Production Environment

Immutable (Disposable Infrastructure)	The servers and containers are immutable, the servers are just dumb servers having just Docker. We can easily create/dispose them.
Portable infrastructure	Everything in an application to run is defined within the container, and we can run the container anywhere you have a Docker environment. It could be AWS, GCP, Digital Ocean etc.,
Simple & rapid scaling	If we need more web apps, we can have more web app. Containers running.

- Applications can be decoupled from individual servers
- Cost saving

Testing/QA Environments

- Test production containers before deploying
- Easy to create identical environments
    - The exact same image can be in any server cluster
    - New environments can be quickly created and destroyed
- No complicated provisioning scripts (the way Vagrant does)
- Containers are dynamically configurable

Docker Commands - Part 1

docker run <image>	Creates a new container from an image; if your run this command for 4 times, 4 new containers get started from that image docker run -p 8080:80 <image> mapping the container's port 80 to web server port 8080 Docker run -d --name web1 -p 8081:80 <image> runs dockers command as daemon docker --rm to remove the container once we exit from the container docker image:tag /bin/sh works like docker image:tag /bin/bash
docker start <name|id>	Starts an existing container
docker stop <name|id>	Stops the existing container docker stop $(docker ps -aq)
docker ps [-a include stopped containers]	List of all running containers
docker rm <name|id>	Removes a container docker rm $(docker ps -aq)
docker login	To loginto to docker repository, dockerhub
docker images	Lists all the images docker images -q (only IDs)
docker ps -aq	List all containers (Only IDs)
Docker rmi <image name>	Removes the image docker rmi $(docker images -q) Remove all images
docker history <Image Id>	Shows the history of the image
docker logs <ContainerId>	To see the logs of the container
docker attach <Conainer>	Attach local standard input, output, and error streams to a running container
docker pause CONTAINER [CONTAINER...]	Pause all processes within one or more containers