DevOps

Docker: Containers

Collabera®
Value. Accelerated.

Collabera®
TACT 360° Training

Trainer: Abhijith V G – AWS, eCommerce, Mobile & DevOps Architect

Magento Certified DEVELOPER PLUS

amazon web services | Certified

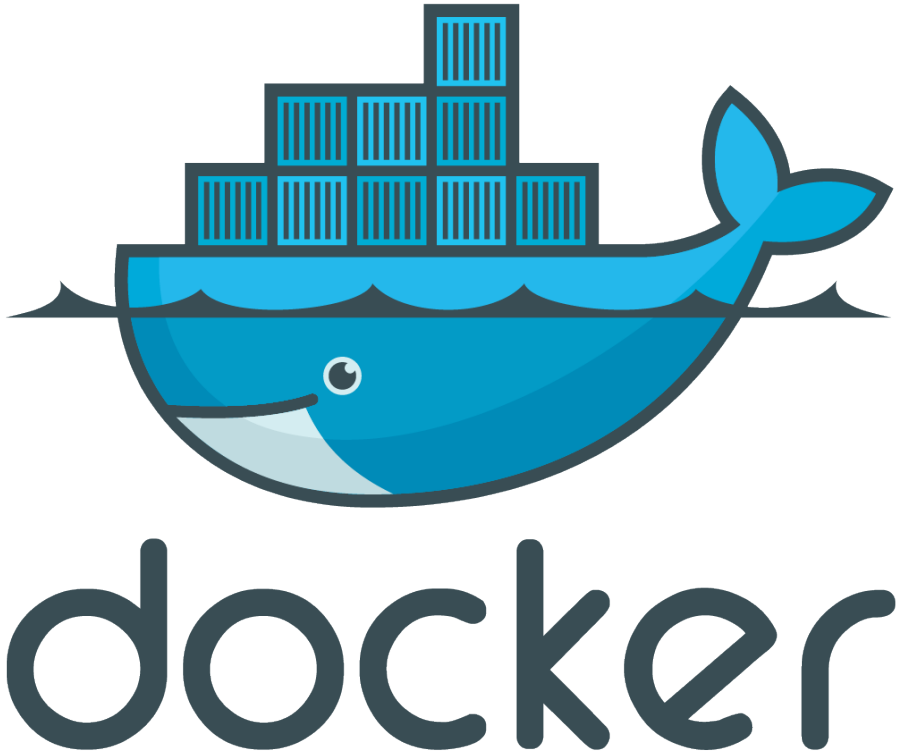
Developer - Associate

Docker: Introduction - What is a Container

A container is a way of packaging a software into standardized units for development, shipment and deployment.

Docker: Introduction - What is a Docker

Docker is currently the #1 container platform.



Docker: Introduction - Use cases of Docker

Portability, Security & Costs: Package existing apps into containers immediately improves security, reduce costs, and gain cloud portability with no changes to the app code.

Micro-services: Containers streamline development and deployment of apps designed with the micro-services architecture pattern.

DevOps (CI/CD): Accelerate and automate development pipelines with rapid feedback loops while eliminating app conflicts and increasing developer productivity.

Infrastructure Optimization: Containerize apps and improve workload density by running them side-by-side on the same servers. Docker helps reduce costs by consolidating infrastructure, improving utilization, and accelerating cloud migration.

Hybrid Cloud: From private datacenters to public cloud infrastructure, Docker allows apps to be fully portable from one infrastructure to another without rewriting code. Accelerate migration to cloud and enable a hybrid or multi cloud environment.

Docker: Introduction - For Developers

Docker automates the repetitive tasks of setting up and configuring development environments so that developers can focus on what matters: building great software.

Docker: Introduction - For Ops

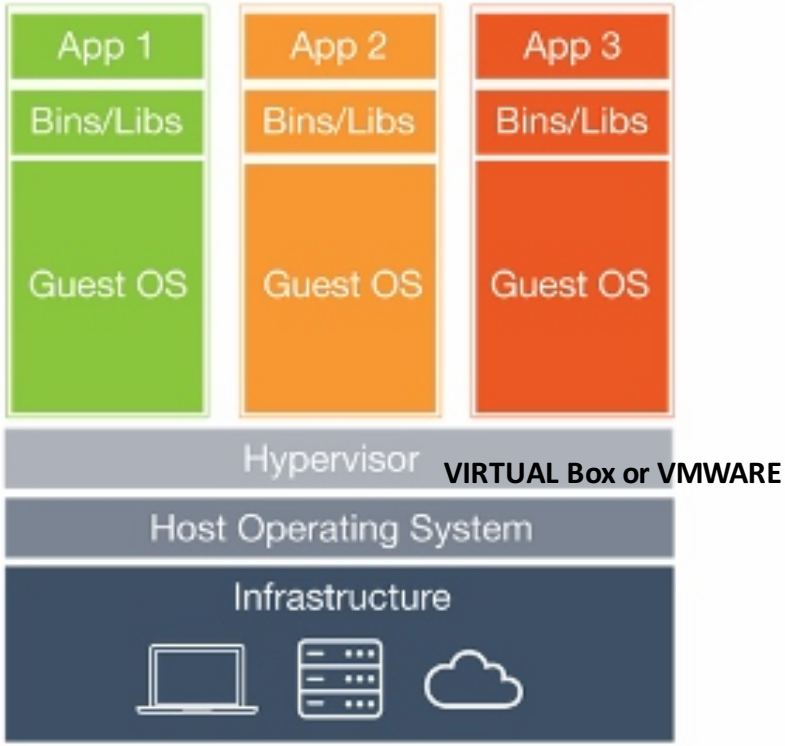
Docker streamlines software delivery. Develop and deploy bug fixes and new features without roadblocks. Scale applications in real time.

Docker: Introduction - Platforms for Docker

Docker can be integrated into various infrastructure tools, including
Amazon Web Services, Google Cloud Platform, Microsoft Azure
Ansible, Chef, Puppet, Salt, Jenkins
Vagrant, and VMware
Plus….IBM Bluemix, , HPE Helion Stackato, , Jelastic, Kubernetes,
OpenStack Nova, OpenSVC, Oracle Container Cloud Service etc.

Docker: Introduction - Dockers vs Virtualization

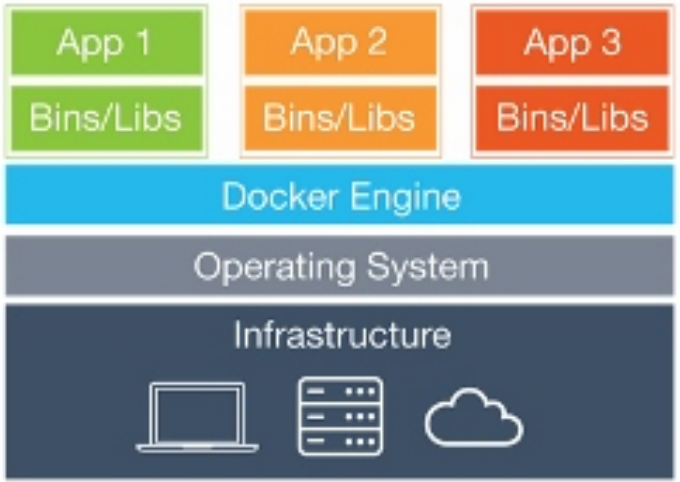
Virtual Machines (Vagrant)



VIRTUAL Box or VMWARE

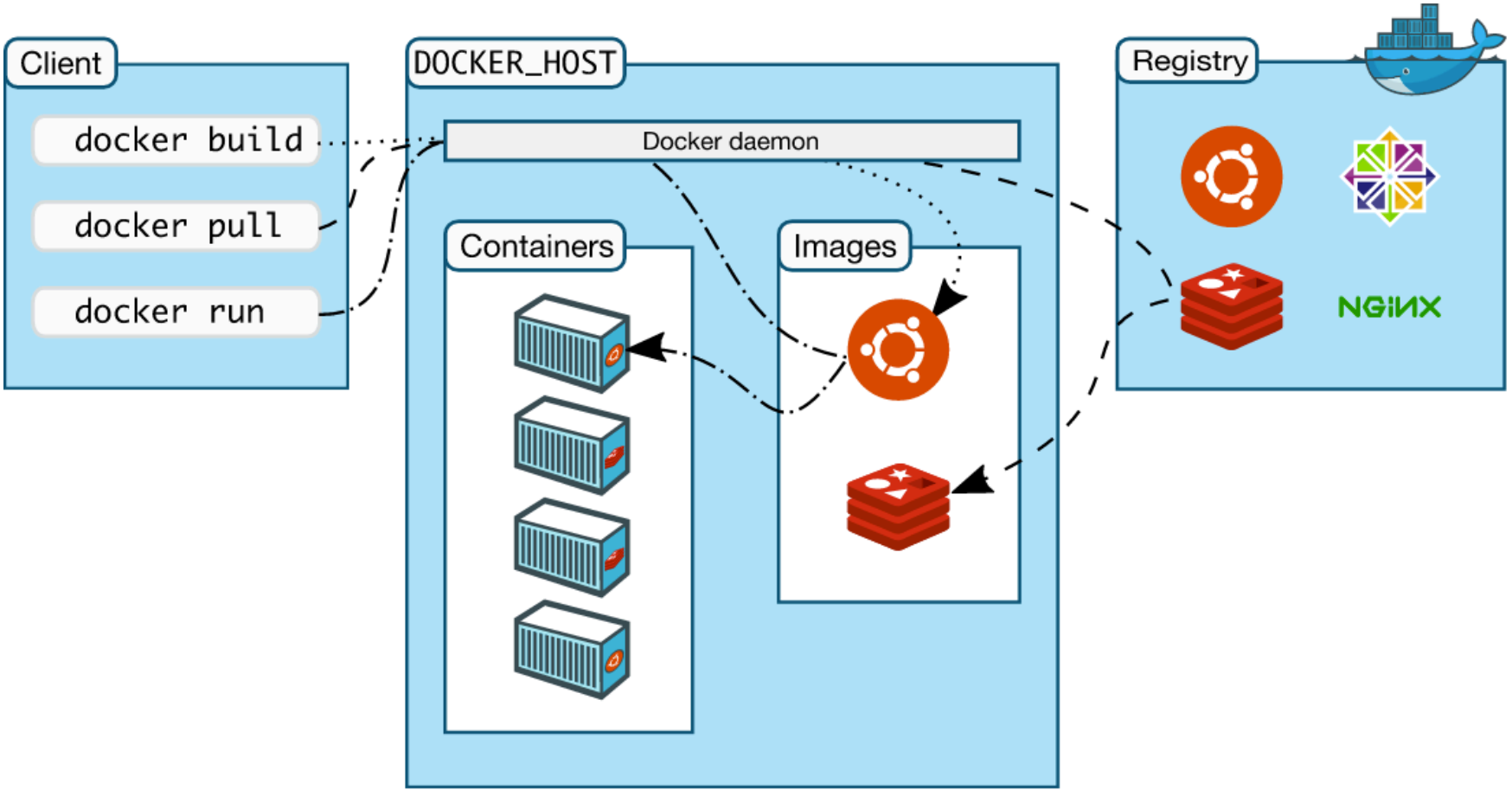
Virtual Machines

Containers (Docker)



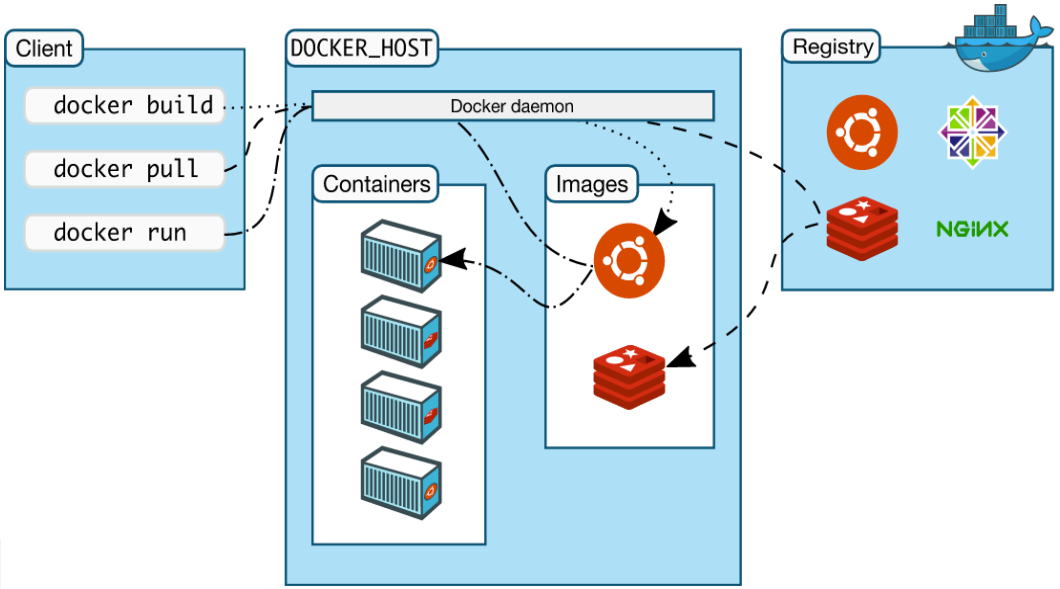
Containers

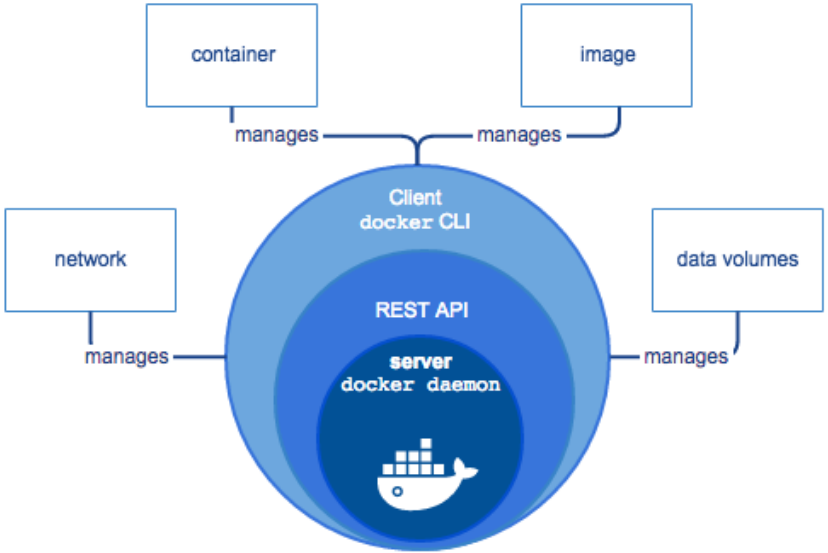
Docker: Architecture - Docker Architecture



Docker: Architecture - Important Docker components

1. The Docker Daemon
2. The Docker Client
3. Docker Registries

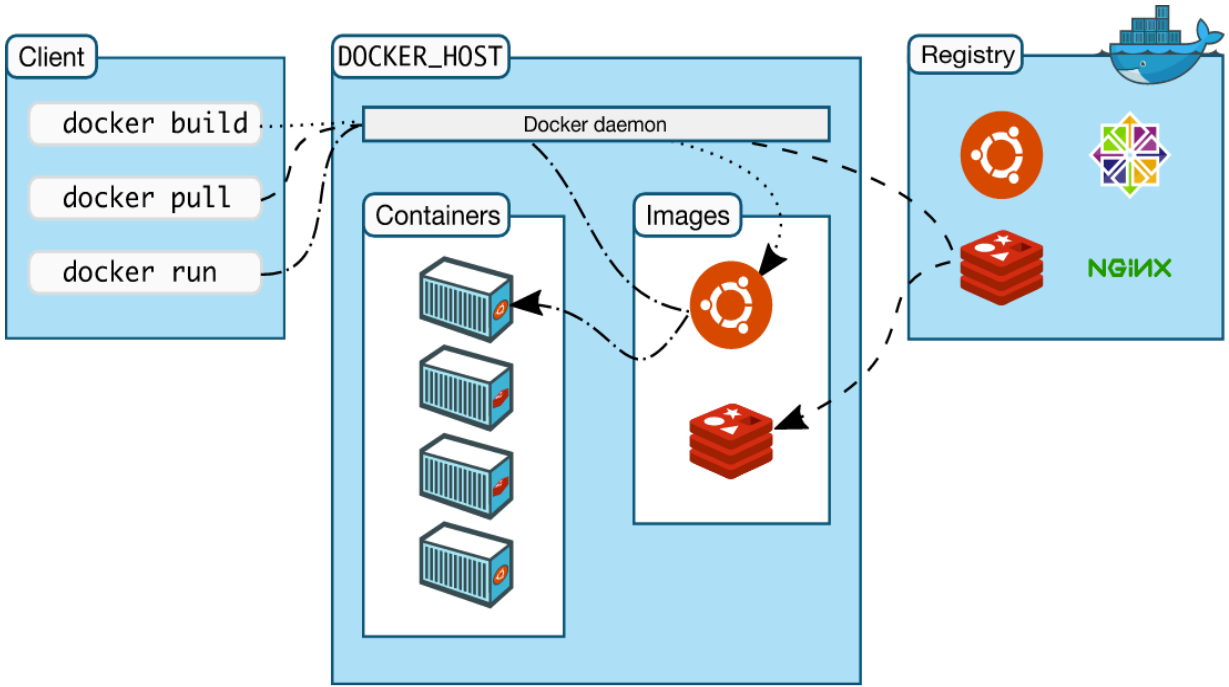




Docker: Architecture - Understanding the Docker components

The Docker daemon

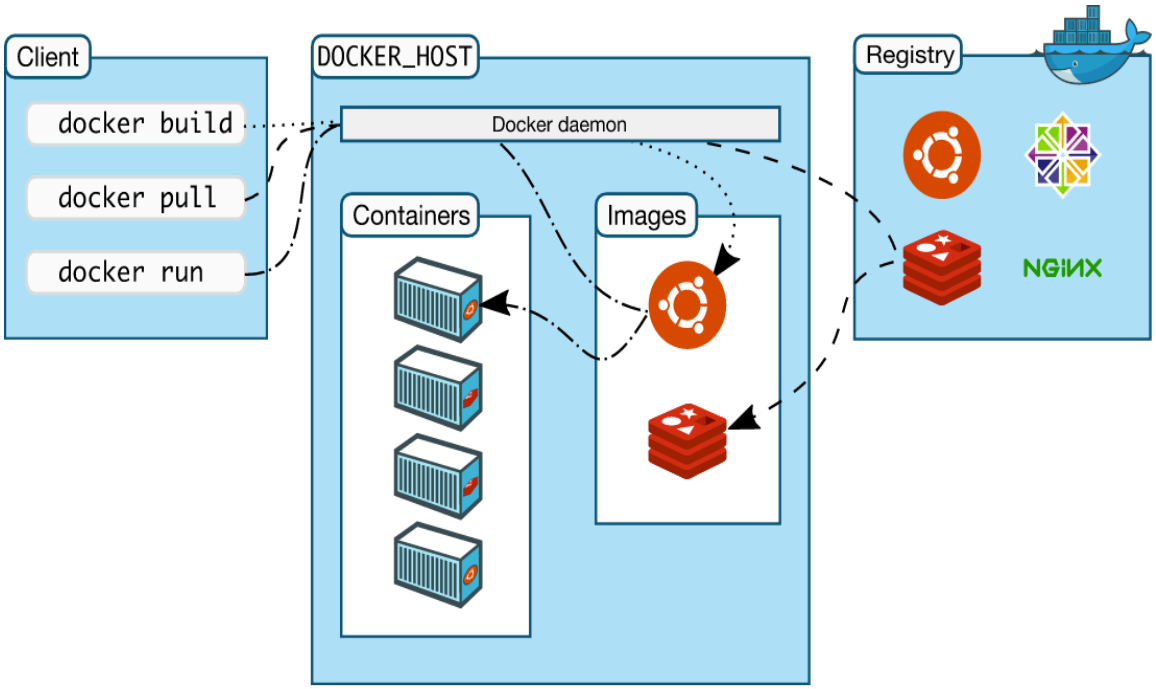
The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.



Docker: Architecture - Understanding the Docker components

The Docker client

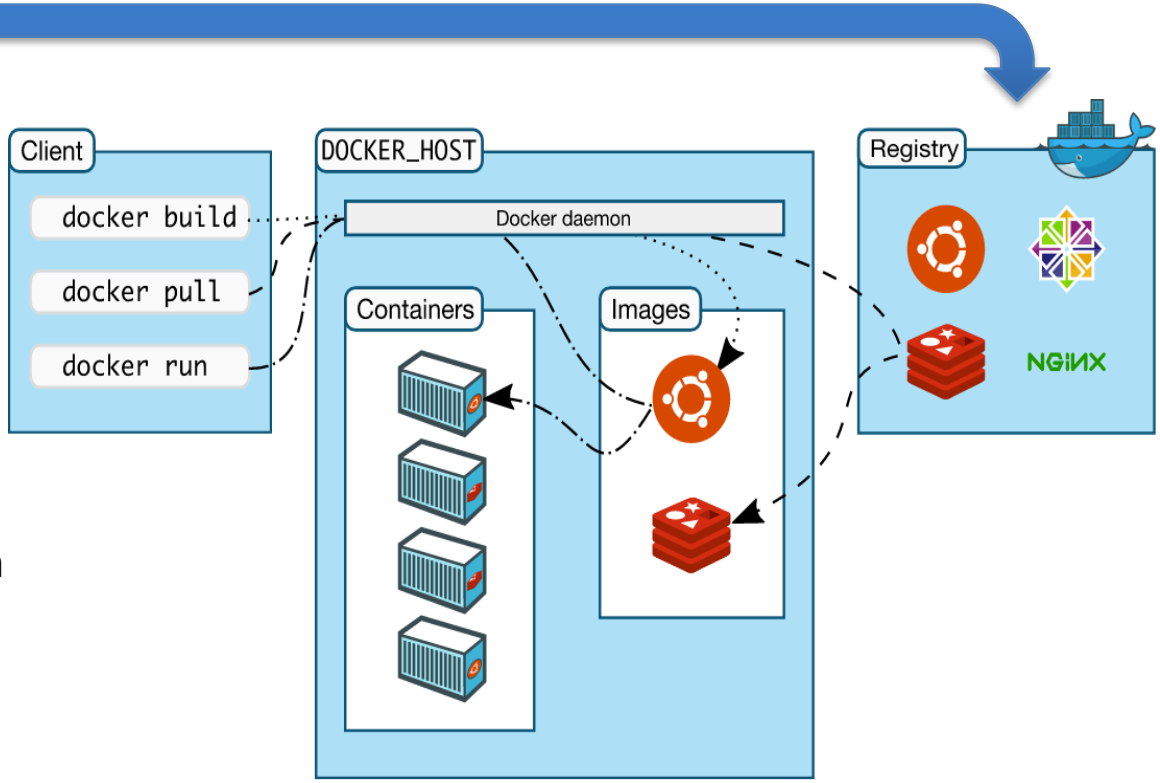
The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.



Docker: Architecture - Understanding the Docker components

Docker registries

A Docker registry stores Docker images. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.



Docker: Installation - Installing Docker on Linux - Ubuntu

Url: https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#install-docker-ce

1. Update the apt package index.

   sudo apt-get update

2. Install the latest version of Docker CE, or go to the next step to install a specific version. Any existing installation of Docker is replaced.

   sudo apt-get install docker-ce

3. On production systems, you should install a specific version of Docker CE instead of always using the latest.

   sudo apt-get install docker-ce=<VERSION>

The Docker daemon starts automatically.

4. Verify that Docker CE is installed correctly by running the hello-world image.

   sudo docker run hello-world

Docker: Installation - Installing Docker on Windows

Url: https://docs.docker.com/docker-for-windows/install/#install-docker-for-windows

1. Download the **docker for windows** stable version.

2. Double-click InstallDocker.msi to run the installer.

3. Follow the install wizard to accept the license, authorize the installer, and proceed with the install.

4. Click Finish on the setup complete dialog to launch Docker.

5. Verify that Docker CE is installed correctly by running
   docker run hello-world

Docker: Installation - Installing Docker on Mac

Url: https://docs.docker.com/docker-for-mac/install/#install-and-run-docker-for-mac

1. Download the **docker for mac** stable version.

2. Double-click Docker.dmg to open the installer, then drag Moby the whale to the Applications folder.

3. Double-click Docker.app in the Applications folder to start Docker and authorize.

4. Verify that Docker CE is installed correctly by running the hello-world image in the terminal.
   docker run hello-world

Docker: Installation – Testing Docker Installation

```
Sree>docker run hello-world

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.
 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://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

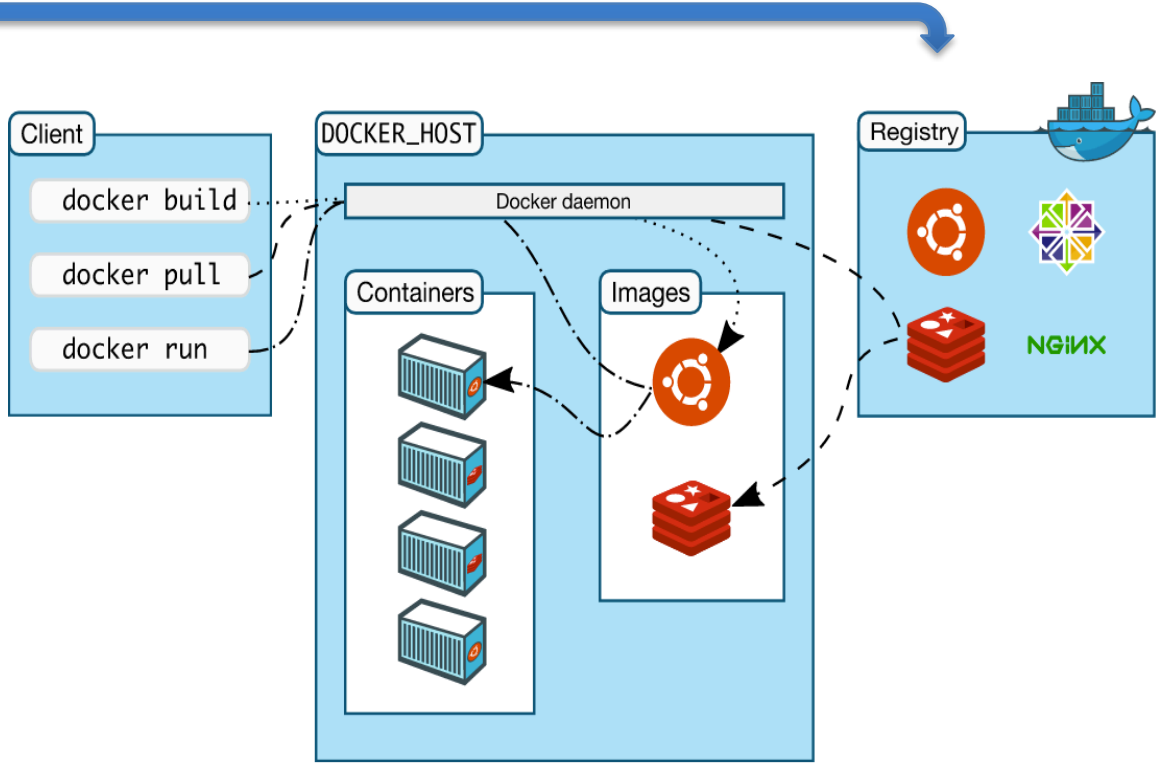
Docker: Installation - Some Docker commands

docker info	Display system-wide information
docker version	Show the Docker version information
docker run	Run a command in a new container
docker ps	List containers
docker rm	Remove one or more containers
docker pull	Pull an image or a repository from a registry
docker images	List images
docker rmi	Remove one or more images
docker exec	Run a command in a running container
docker logs	Fetch the logs of a container
docker pause	Pause all processes within one or more containers
docker start	Start one or more stopped containers
docker stop	Stop one or more running containers
docker unpause	Unpause all processes within one or more containers
docker volume	Manage volumes

Docker: Provisioning - Docker Hub

Url: https://hub.docker.com/

Docker Hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.



Docker: Searching Repository

docker search nginx

```
Sree>docker search nginx
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
nginx                               Official build of Nginx.                        6653      [OK]
jwilder/nginx-proxy                 Automated Nginx reverse proxy for docker c...   1091                 [OK]
richarvey/nginx-php-fpm             Container running Nginx + PHP-FPM capable ...   424                  [OK]
jrcs/letsencrypt-nginx-proxy-companion   LetsEncrypt container to use with nginx as...   209                  [OK]
kong                                Open-source Microservice & API Management ...   100       [OK]
webdevops/php-nginx                 Nginx with PHP-FPM                              88                   [OK]
bitnami/nginx                       Bitnami nginx Docker Image                      32                   [OK]
linuxserver/nginx                   An Nginx container, brought to you by Linu...   24
tutum/nginx                         Base Docker Image to run Nginx server           10
webdevops/nginx                     Nginx container                                 8                    [OK]
blacklabelops/nginx                 Dockerized Nginx Reverse Proxy Server.          6                    [OK]
nginxdemos/nginx-ingress            NGINX Ingress Controller for Kubernetes         5
1science/nginx                      Nginx Docker images that include Consul Te...   4                    [OK]
```

Docker: Provisioning - Docker Hub - Features

- **Image Repositories:** Find and pull images from community and official libraries, and manage, push to, and pull from private image libraries to which you have access.
- **Automated Builds:** Automatically create new images when you make changes to a source code repository.
- **Webhooks:** A feature of Automated Builds, Webhooks let you trigger actions after a successful push to a repository.
- **Organizations:** Create work groups to manage access to image repositories.
- **GitHub and Bitbucket Integration:** Add the Hub and your Docker Images to your current workflows.

Docker: Provisioning - Downloading Docker images

Command: ***docker pull***

Description: Pull an image or a repository from a registry

Usage:
docker pull [OPTIONS] NAME[:TAG|@DIGEST]

eg: docker pull ubuntu

Docker: Provisioning - Running Docker images

Command: ***docker run***

Description: When we docker run, the container loads the specified image and starts one or more processes within it.

General Form:
docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]

```
Sree>docker run -ti ubuntu
root@2f25eda2022a:/# pwd
/
root@2f25eda2022a:/#
```

Docker: Provisioning - Running Docker images

Ways of seeing and accessing running Docker Containers

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
Sree>docker run -tid ubuntu
9fcddfa8d146854274652ab978c9435cbac572fb5768c056896c65c490d4430c
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
9fcddfa8d146        ubuntu              "/bin/bash"         2 seconds ago       Up 1 second                             musing_sinoussi
Sree>docker exec -ti musing_sinoussi bash
root@9fcddfa8d146:/# exit
exit
Sree>docker attach  musing_sinoussi
root@9fcddfa8d146:/#
root@9fcddfa8d146:/#
root@9fcddfa8d146:/#
root@9fcddfa8d146:/# read escape sequence
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
9fcddfa8d146        ubuntu              "/bin/bash"         About a minute ago  Up About a minute                       musing_sinoussi
Sree>
```

Remember Ctrl+P+Q to exit from a containers PID 1 prompt without terminating it

Docker: Provisioning - Running multiple containers

We can run as many containers our host can accommodate using docker run –d option.
If we ever need to access the containers prompt remember to add –ti as well. But in practice
you will never need to access the container's shell.
Starting two more containers.

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
9fcddfa8d146        ubuntu              "/bin/bash"         7 minutes ago       Up 7 minutes                            musing_sinoussi
Sree>docker run -tid ubuntu
b175cf804e8c193f90a60fbea700764cf9cdd11ebbc68c1103c09b90a8da6f12
Sree>docker run -tid ubuntu
77f675279c3e8371da56ba571d3c760f542020fee22d32682b6701f3d19d4c37
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
77f675279c3e        ubuntu              "/bin/bash"         3 seconds ago       Up 1 second                             dazzling_mayer
b175cf804e8c        ubuntu              "/bin/bash"         4 seconds ago       Up 3 seconds                            jovial_boyd
9fcddfa8d146        ubuntu              "/bin/bash"         8 minutes ago       Up 8 minutes                            musing_sinoussi
Sree>
```

Docker: Ports

Opening ports:
If you want containers to accept incoming connections, you will need to provide special options when invoking docker run. There are two approaches.

First, you can supply -P or --publish-all=true|false to docker run which is a blanket operation that identifies every port with an EXPOSE line in the image's Dockerfile

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
Sree>ls -l
Sree>echo Test123 > sree.txt
Sree>ls -l
total 8
-rw-r--r--  1 sree  staff  8 Aug 19 14:45 sree.txt
Sree>docker run -tid -v $PWD:/usr/share/nginx/html -P nginx
d89d38e7fa28e81c3f77648ed8547d68d45176903be8fd0da2e66a1ae38f8a1a
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
d89d38e7fa28        nginx               "nginx -g 'daemon ..."   4 seconds ago       Up 3 seconds        0.0.0.0:32779->80/tcp   cranky_engelbart
Sree>curl http://localhost:32779/sree.txt
Test123
```

Docker: Provisioning - Running commands in container

3 ways of running commands in container:

1. Using "docker run" command for creating containers.

Example: ***docker run ubuntu:trusty ls –l***

Note: Here the linux command "ls -l" will be executed as soon as the container is created and ran.

2. Using "-i" & "-t" options of "docker run" command. (-d option starts it in the background)

Example: ***docker run -ti ubuntu:trusty***

Note: Here the container will start in terminal interactive mode, where you will be able to execute your linux commands.

3. Using "docker exec" command for existing containers.

Example: ***docker exec <CONTAINER NAME/ID> ls -l***

Note: First execute "docker ps" command to get the name/id of running container. After that, use this command to execute the linux command "ls –l" on the running container.

Docker: Mapping a host folder

To map a host folder to the container use the –v flag of the docker run command:
docker run –v <full path to host folder>:<full path inside the container> <image name>

Example:

docker run –d –P –v $PWD/team1:/usr/share/nginx/html nginx

This command will start an nginx container whose document root is mapped to the team1 folder in our current working directory. Now, any change that you make in the team1 folder will be accessible inside the container and vice versa.

Docker: Logs

Viewing logs

docker logs <CONTAINER>
docker logs -f <CONTAINER>

```
Sree>docker logs d89d38e7fa28
172.17.0.1 - - [19/Aug/2017:09:15:43 +0000] "GET /sree.txt HTTP/1.1" 200 8 "-" "curl/7.45.0" "-"
Sree>docker logs -f d89d38e7fa28
172.17.0.1 - - [19/Aug/2017:09:15:43 +0000] "GET /sree.txt HTTP/1.1" 200 8 "-" "curl/7.45.0" "-"
172.17.0.1 - - [19/Aug/2017:09:19:52 +0000] "GET /sree.txt HTTP/1.1" 200 8 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHT
ML, like Gecko) Chrome/60.0.3112.78 Safari/537.36 OPR/47.0.2631.55" "-"
2017/08/19 09:19:52 [error] 6#6: *2 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, server: local
host, request: "GET /favicon.ico HTTP/1.1", host: "localhost:32779", referrer: "http://localhost:32779/sree.txt"
172.17.0.1 - - [19/Aug/2017:09:19:52 +0000] "GET /favicon.ico HTTP/1.1" 404 571 "http://localhost:32779/sree.txt" "Mozilla/5.0 (Macintosh; Intel Mac OS
 X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.78 Safari/537.36 OPR/47.0.2631.55" "-"
```

Docker: Inspect Images or Containers

```
docker inspect d89d38e7fa28|less

docker inspect --format='{{.Mounts}}' d89d38e7fa28
[{bind  /Users/sree/vms/docker/nginx /usr/share/nginx/html   true rprivate}]

docker inspect --format='{{.NetworkSettings.Ports}}' d89d38e7fa28
map[80/tcp:[{0.0.0.0 32779}]]

docker inspect --format='{{.Config.Cmd}}' d89d38e7fa28
[nginx -g daemon off;]

docker inspect --format='{{.Config.Entrypoint}}' d89d38e7fa28
[]
```

Docker: HW-1 (To help you understand the concepts so far)

1. Install Docker-CE for Windows
2. Test the installation with docker run hello-world
3. Start the ubuntu container and get inside it
4. Start the ubuntu container in the background and access the shell prompt in 2 ways
5. Launch NGINX container and serve a fully working website from the host directory and be able to make live edits.

For technical support:

Call : +91 75730 27611

E-mail : tactsupport@collabera.com

Visit us at : http://www.collaberatact.com

Collabera