



Persistent

Architecture of docker



Objectives

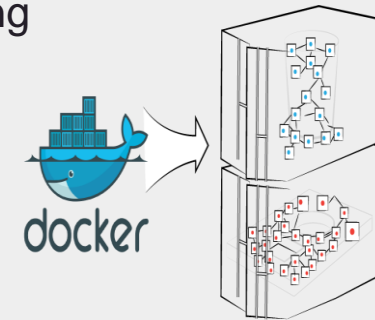
- At the end of this section, you will be able to Learn:
- Docker Architecture
- Important components of Docker

Major Components of Docker

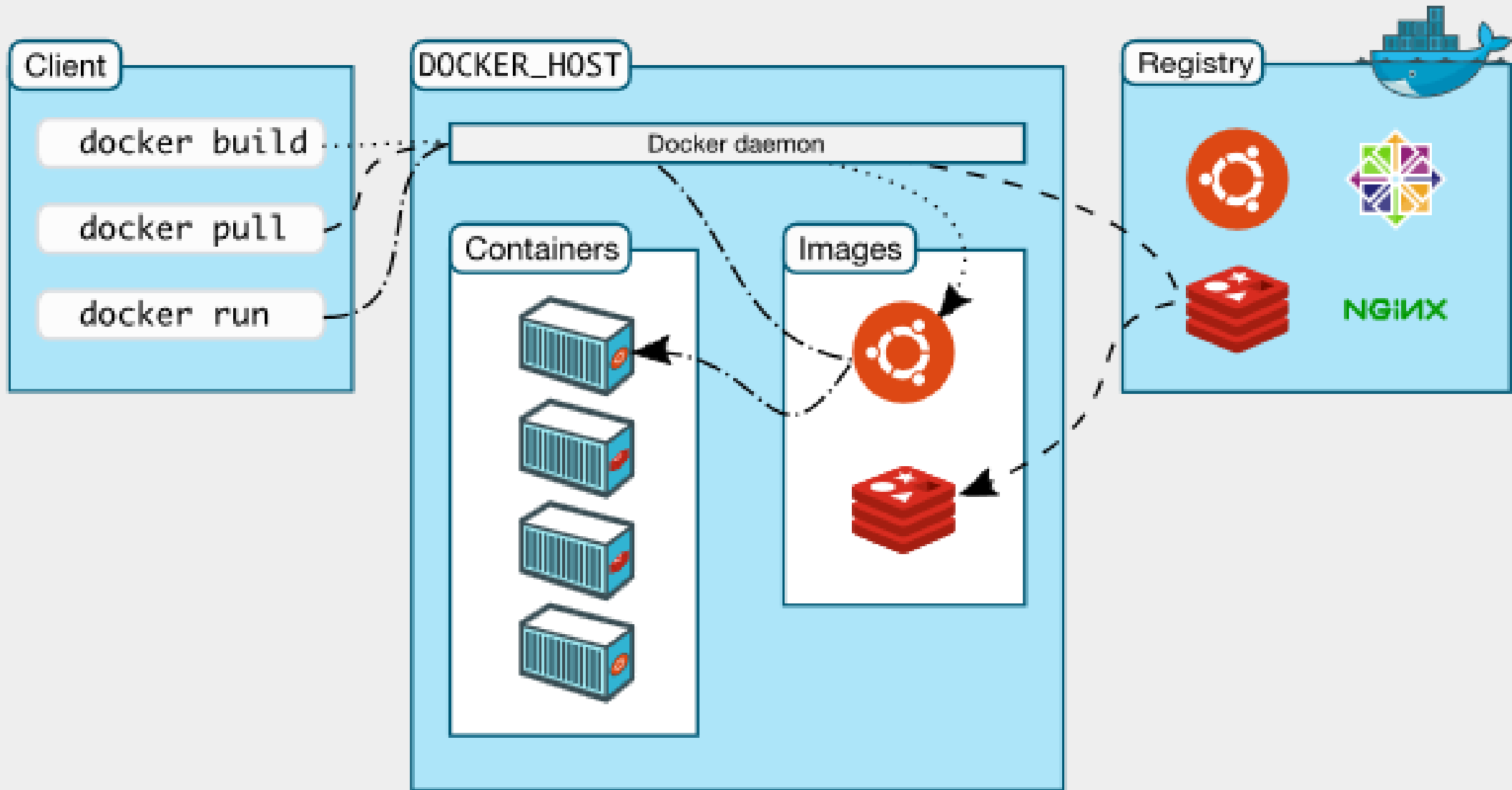


Docker Engine – the open source containerization platform

Docker Hub - Software-as-a-Service platform for sharing and managing Docker containers.



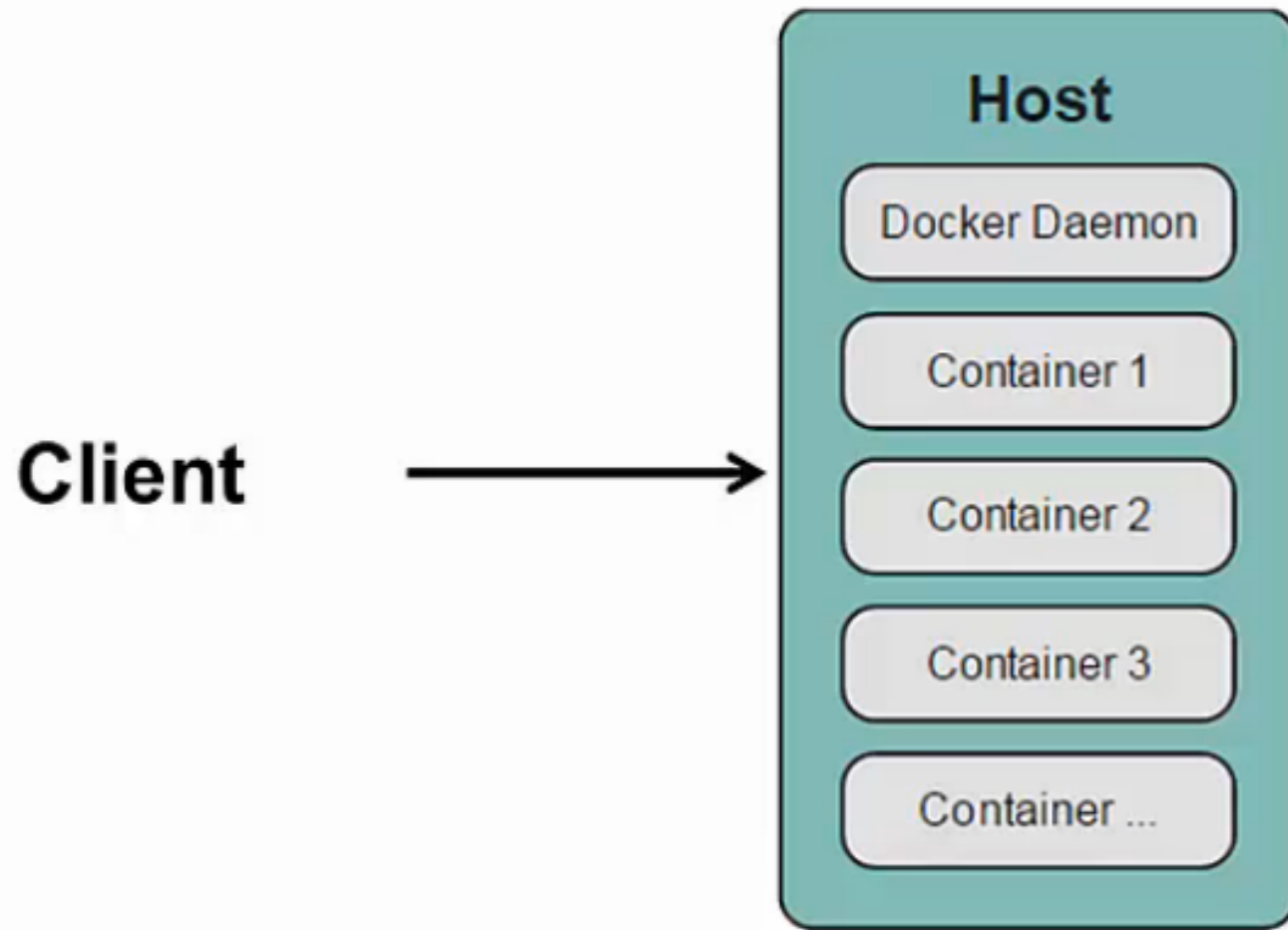
Docker Architecture



Docker Architecture...

- The Docker client talks to the Docker daemon
- Docker Daemon which does the heavy lifting of building, running, and distributing your Docker containers
- Both the Docker client and the daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.
- The Docker client and daemon communicate via sockets or through a RESTful API.

Docker Architecture...



Docker Daemon and client

- Docker Daemon - Docker daemon runs on a host machine. The user does not directly interact with the daemon, but instead through the Docker client.
- Docker Client - is the primary user interface to Docker. It accepts commands from the user and communicates back and forth with a Docker daemon.

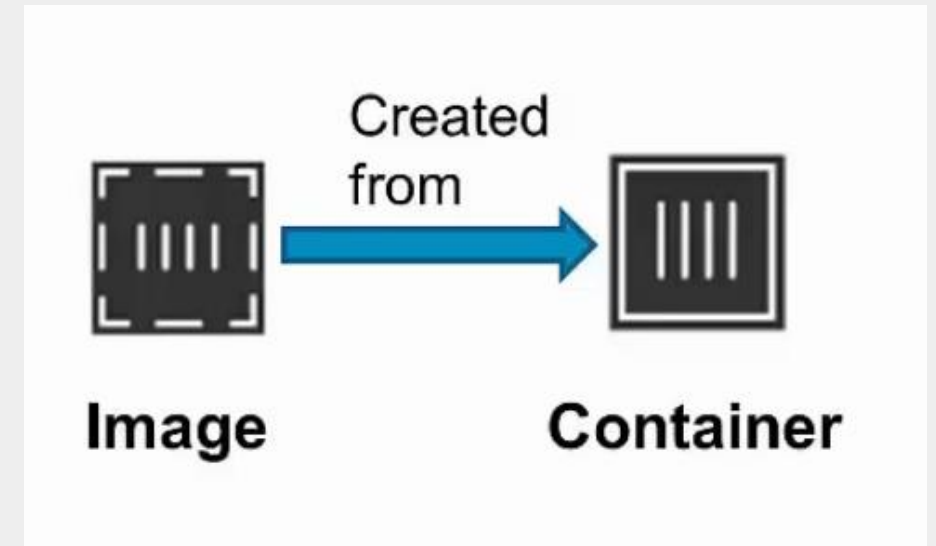
Inside Docker

Below are the main internal components of Docker:

- Docker images
- Docker registries
- Docker containers

Docker Images

- A Docker image is a read-only template
- For example, an image could contain an Ubuntu operating system with Apache and your web application installed
- Images are used to create Docker containers.
- Docker provides a simple way to build new images or update existing images
- You can also download Docker images that other people have already created
- Docker images are the build component of Docker.



Docker Images...

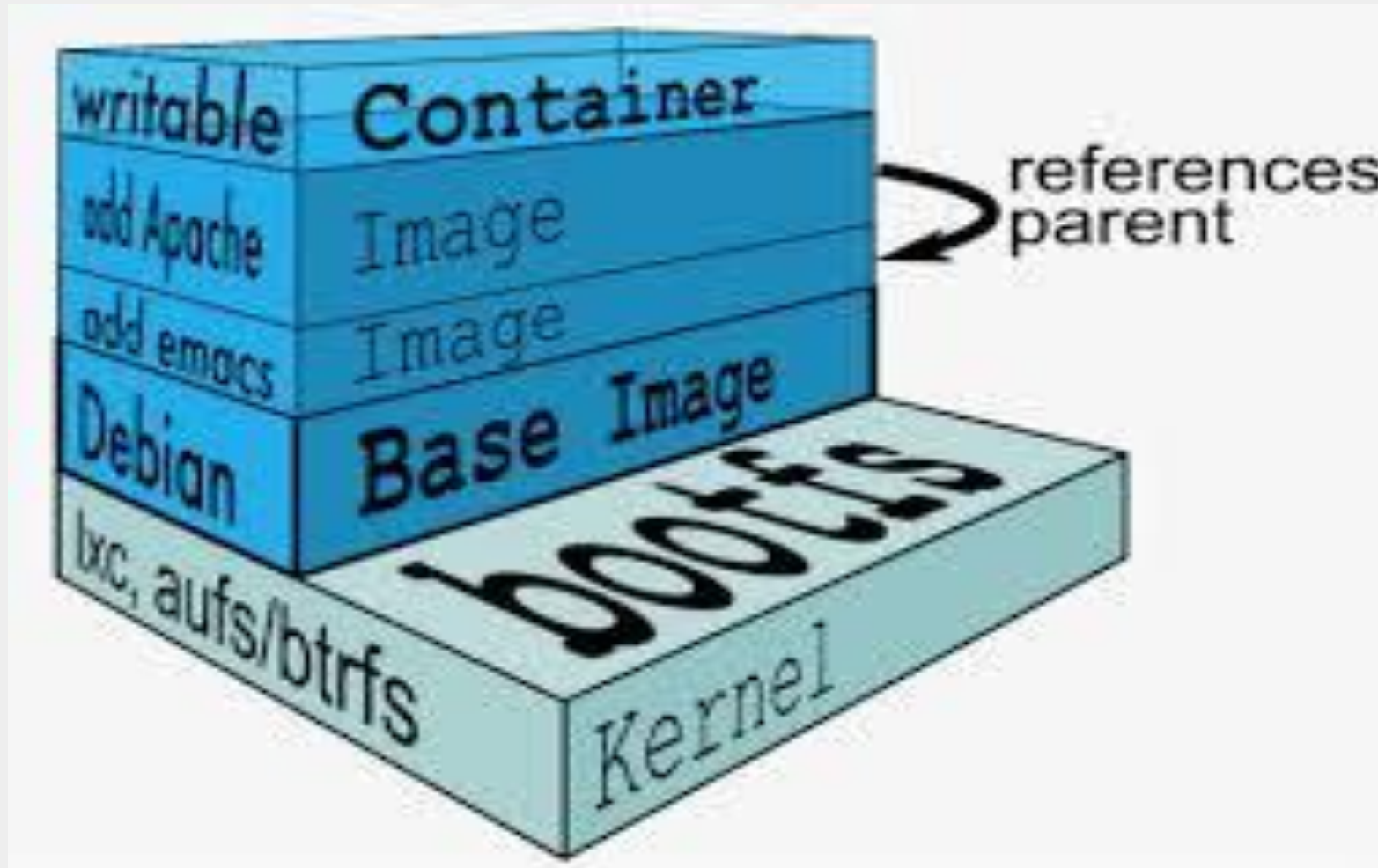
- Each Docker image consists of series of layers.
- Docker is lightweight because of its images, when an application is updated to a newer version, a new layer gets built.
- In case of Virtual machines, the entire image needs to be re-created.
- Every image starts with a base image, e.g. a base Ubuntu image or a base fedora image.
- Docker images are then built from these base images using simple steps called as instructions.
- Instructions are like – run a command, add a file, add environment variable, etc.

Docker Images...

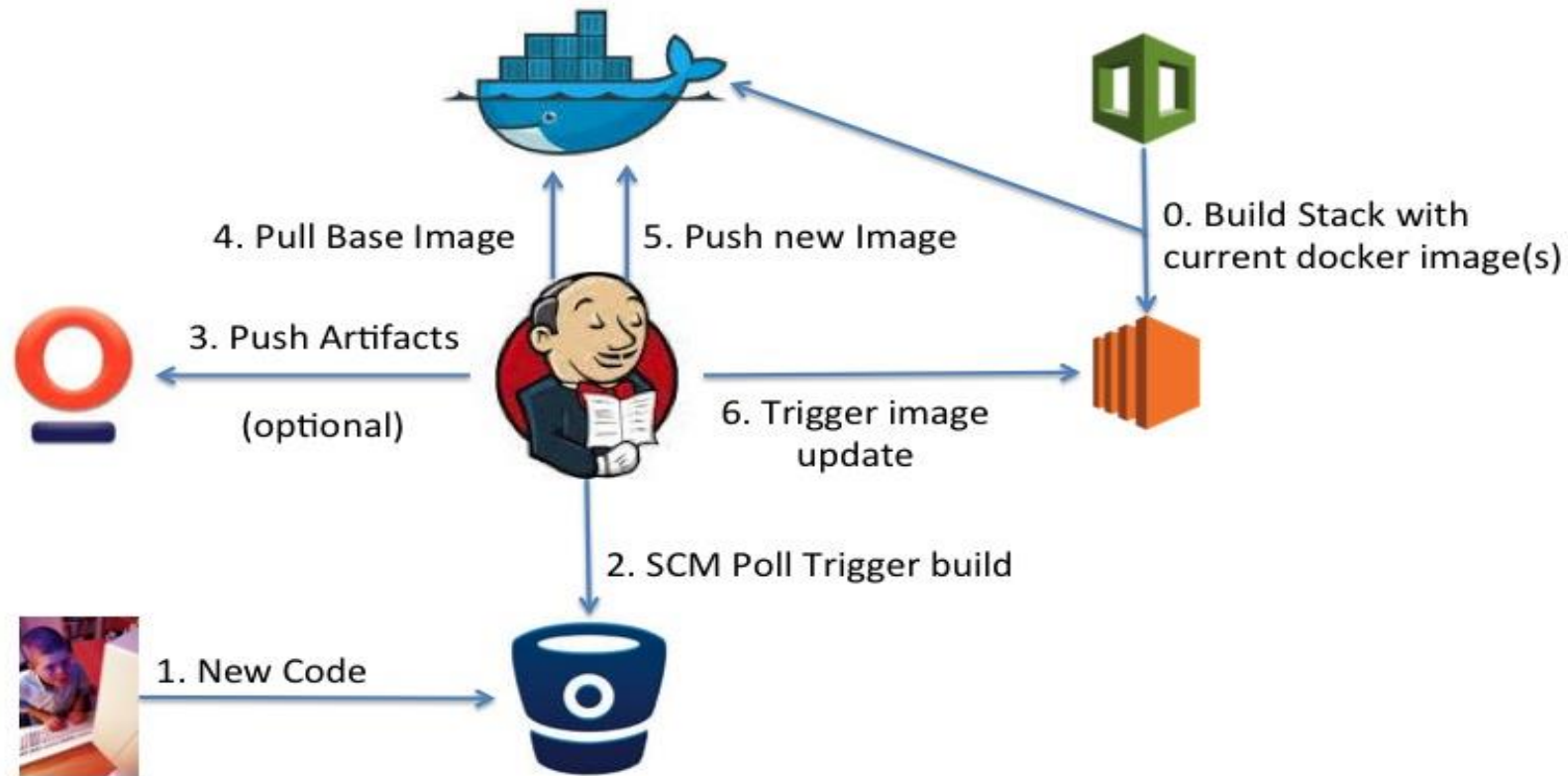
- The instructions to create the Docker container are stored in the **Docker file**.
- Docker reads this Docker file, when you request the build of the image.
- It executes the instructions and returns a final image.



Docker Images - Layering



Docker Continuous Deployment Flow

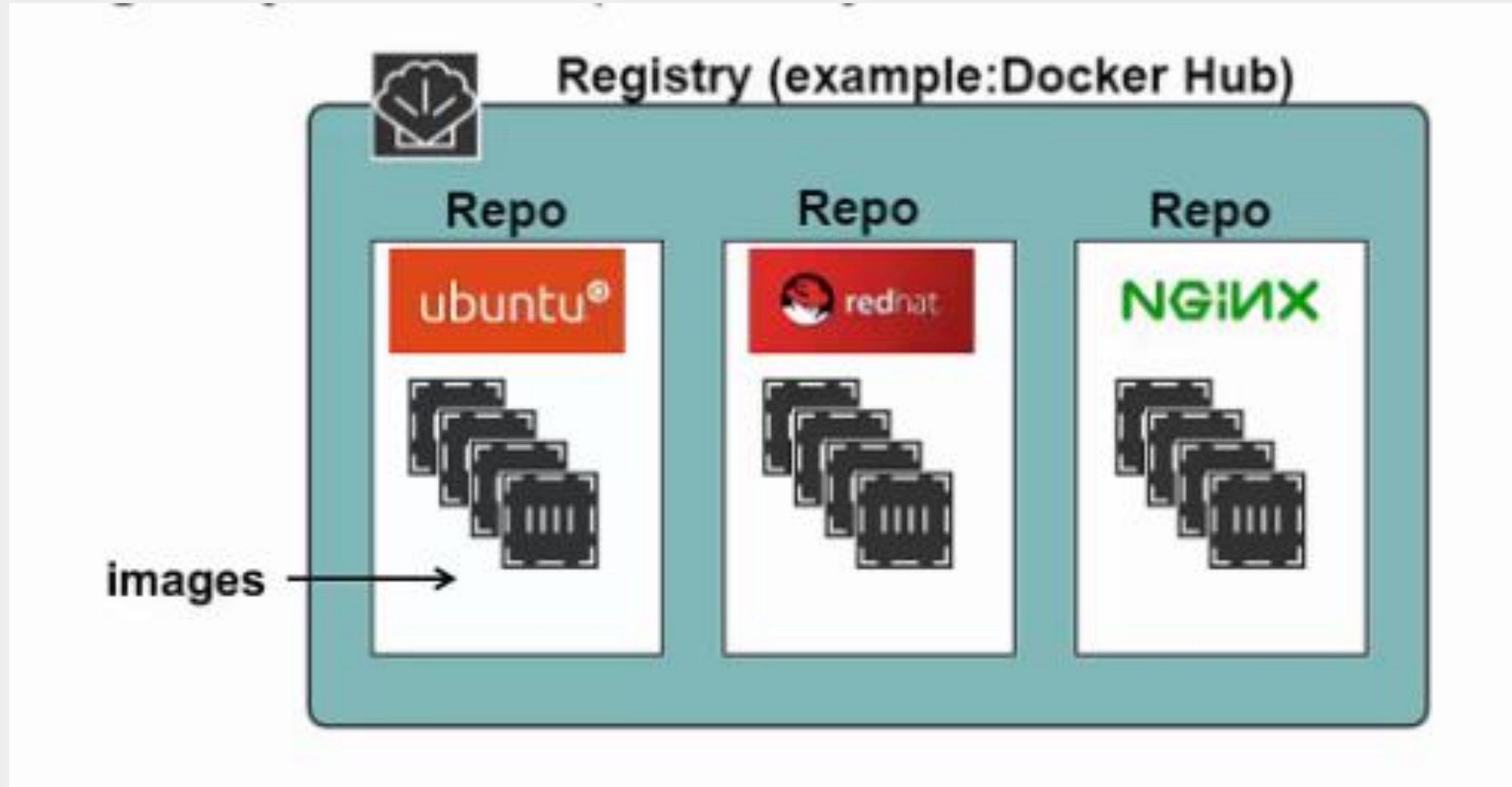


Docker Registries


- Docker registries hold images.
- These are public or private stores from which you upload or download images
- The public Docker registry is provided with the Docker Hub. It serves a huge collection of existing images for your use.
- These can be images you create yourself or you can use images that others have previously created.
- Docker registries are the distribution component of Docker.



Docker Registries and Repositories






Docker Hub

 Explore Help

Search

Sign up Log In

Explore Official Repositories

 nginx official	3.2K STARS	10M+ PULLS	DETAILS
 busybox official	683 STARS	10M+ PULLS	DETAILS
 ubuntu official	4.0K STARS	10M+ PULLS	DETAILS

Docker Registries...

- Once you build a Docker image, you can push it to a public repository like Docker hub or your own private repository.
- Using Docker client, you can search for the already published images and pull them to your host to build containers from them.
- Docker Hub provides both public and private storage.
- Public storage is searchable and can be downloaded by anyone.
- Private storage is excluded from search results and only your user group can pull them to build containers.



Deploying application on Docker host

- The most common way to deploy the application on Docker host
- Create a Dockerfile that specifies what your app needs to run (i.e., define an image).
- Connect to the Docker Host.
- Using the Dockerfile, create an image in the Host.
- Create a new container using this image.
- Start the container. Your app is now “Dockerized”!
- Optionally, you can take a snapshot of this container. This will create a new image which you can then push into a Registry for later use.

Underlying technology for Docker

- Docker is written in Go and makes use of several kernel features.
- Namespaces – provide the isolated workspaces we call the container.
- When you run a container, Docker creates separate set of namespaces for that container.
- This provides level of isolation, each aspect of a container runs in its own namespace.
- Some of the namespaces Docker engine uses are as below –
 - The pid (process ID) namespace for process isolation.
 - The net namespace for managing network interfaces.
 - The ipc (InterProcess Communication) namespace for managing access to IPC resources.
 - The mnt namespace for Managing mount-points.
 - The uts namespace for isolating kernel and version identifiers

Underlying technology for Docker

- Control Groups (cgroups)
 - Allow the Docker to share the available hardware resources to containers.
 - This ensures Docker containers are good multi-tenant citizens on the host.
- Union File Systems
 - File Systems that operate by creating layers.
 - This makes them very lightweight and fast.
 - These serve as the building blocks of containers.

Docker Containers

- A Docker container holds everything that is needed for an application to run.
- Each container is created from a Docker image.
- Docker containers can be run, started, stopped, moved, and deleted.
- Each container is an isolated and secure application platform
- Docker containers are the run component of Docker



Docker Containers...

- A container consists of an operating system, user-added files, and meta-data.
- The Docker image is read-only.
- When Docker runs a container from an image, it adds a read-write layer on top of the image (using a union file system as we saw earlier).
- In this read-write layer, your application can then run.



Reference Material : Websites & Blogs

- <https://www.docker.com/>
- <https://training.docker.com/self-paced-training>
- <https://www.youtube.com/watch?v=Q5POuMHxW-0>

Docker up and Running by Karl Matthias and Sean kane

Key Contacts

Docker Interactive

Dattatray Kulkarni

dattatray_kulkarni@persistent.co.in

Asif Immanad

asif_immanad@persistent.co.in

Vice President

Shubhangi Kelkar

shubhangi_kelkar@persistent.co.in



Persistent

Thank you!

Persistent University

