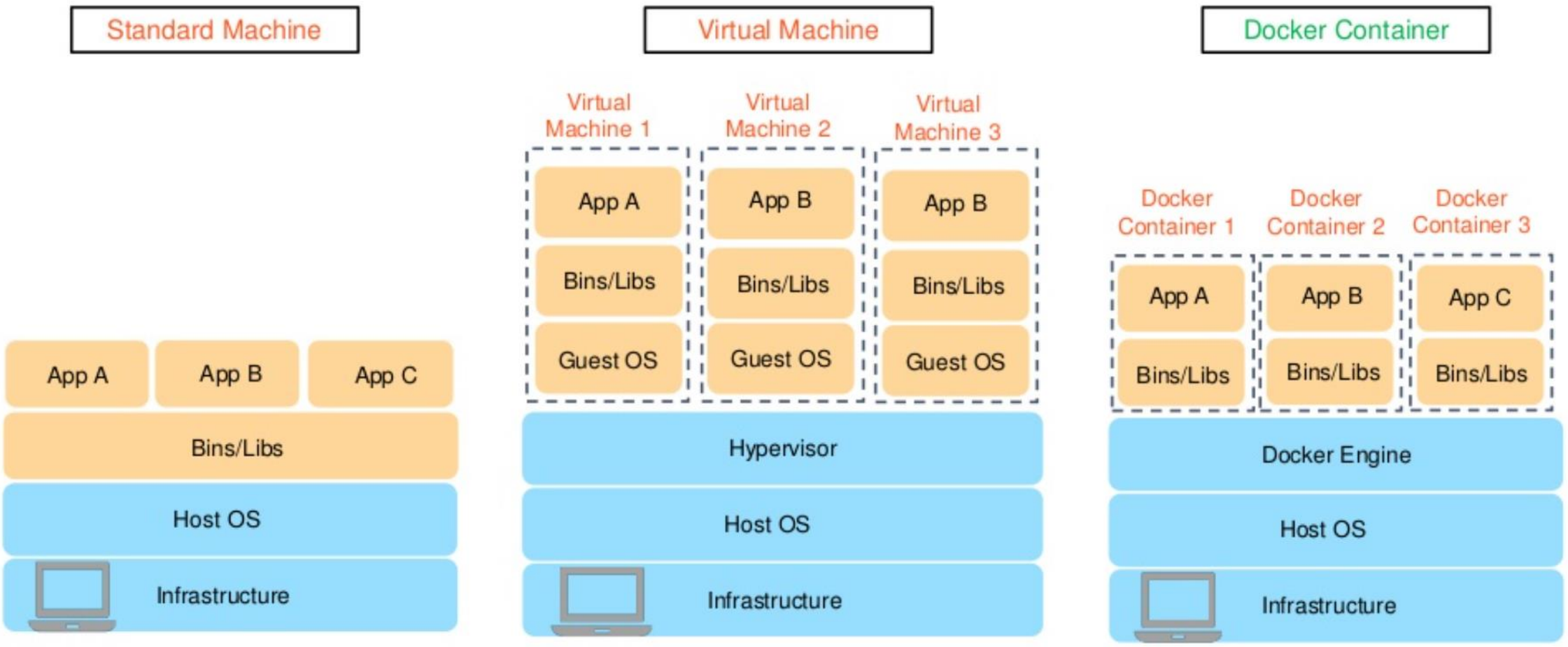
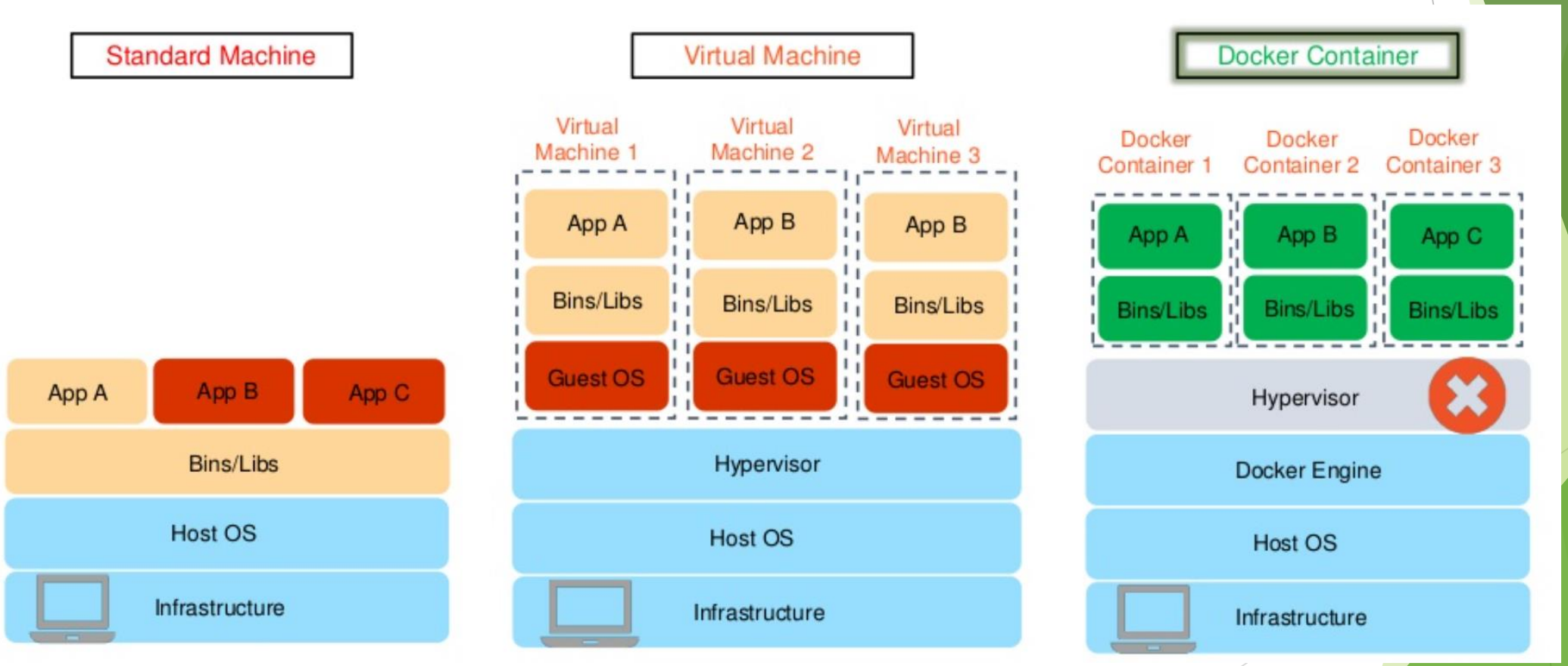
Docker Basic

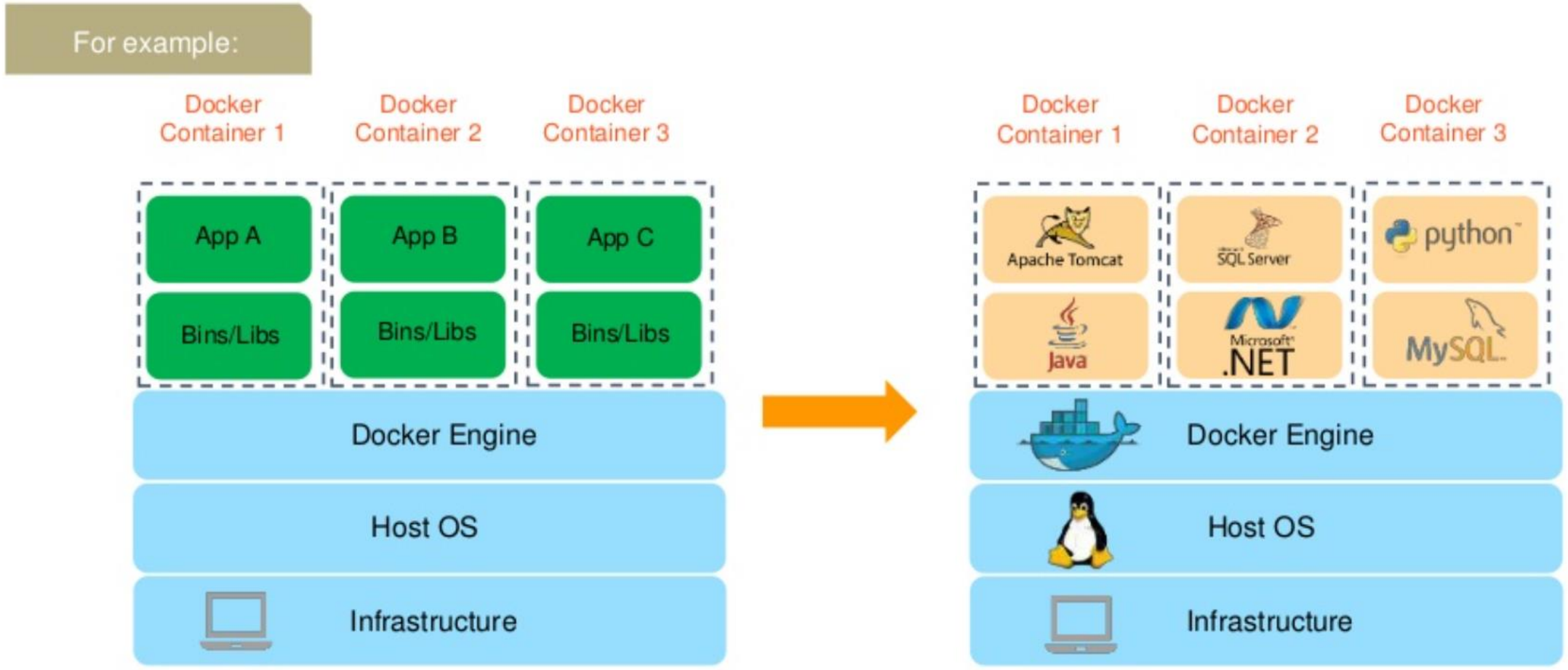
Why Docker Container?



Why Docker Container?



Why Docker Container?



Why Docker Container?

An example where a company develops a Java Application

A developer will setup a JBoss software on his system

After the application is developed, it is examined by the testing team

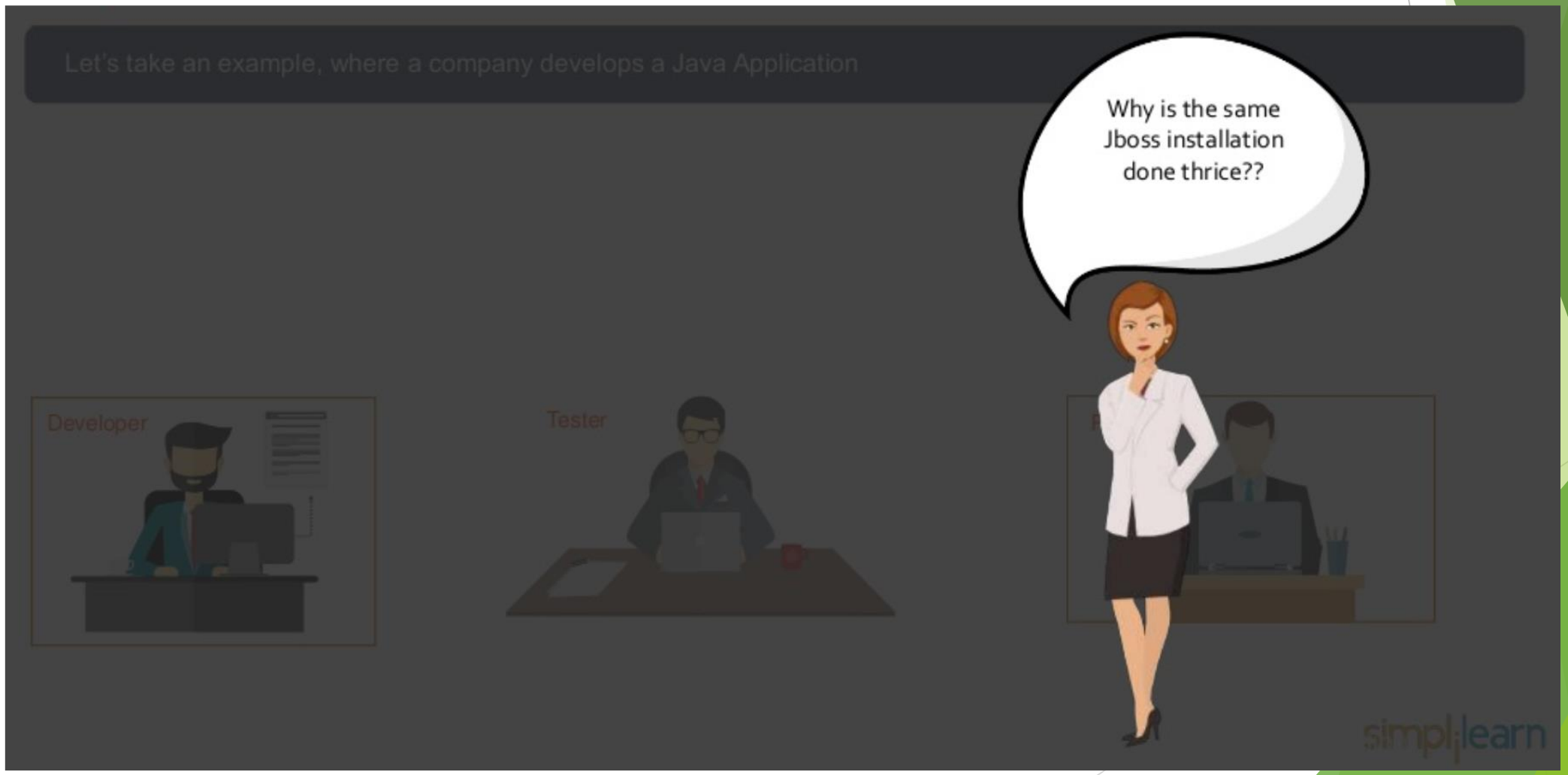
Here, the tester repeats the installation process of JBoss

Once the application is tested, it will be deployed by the production team

To host the Java application, the system admin also has to install JBoss on his system

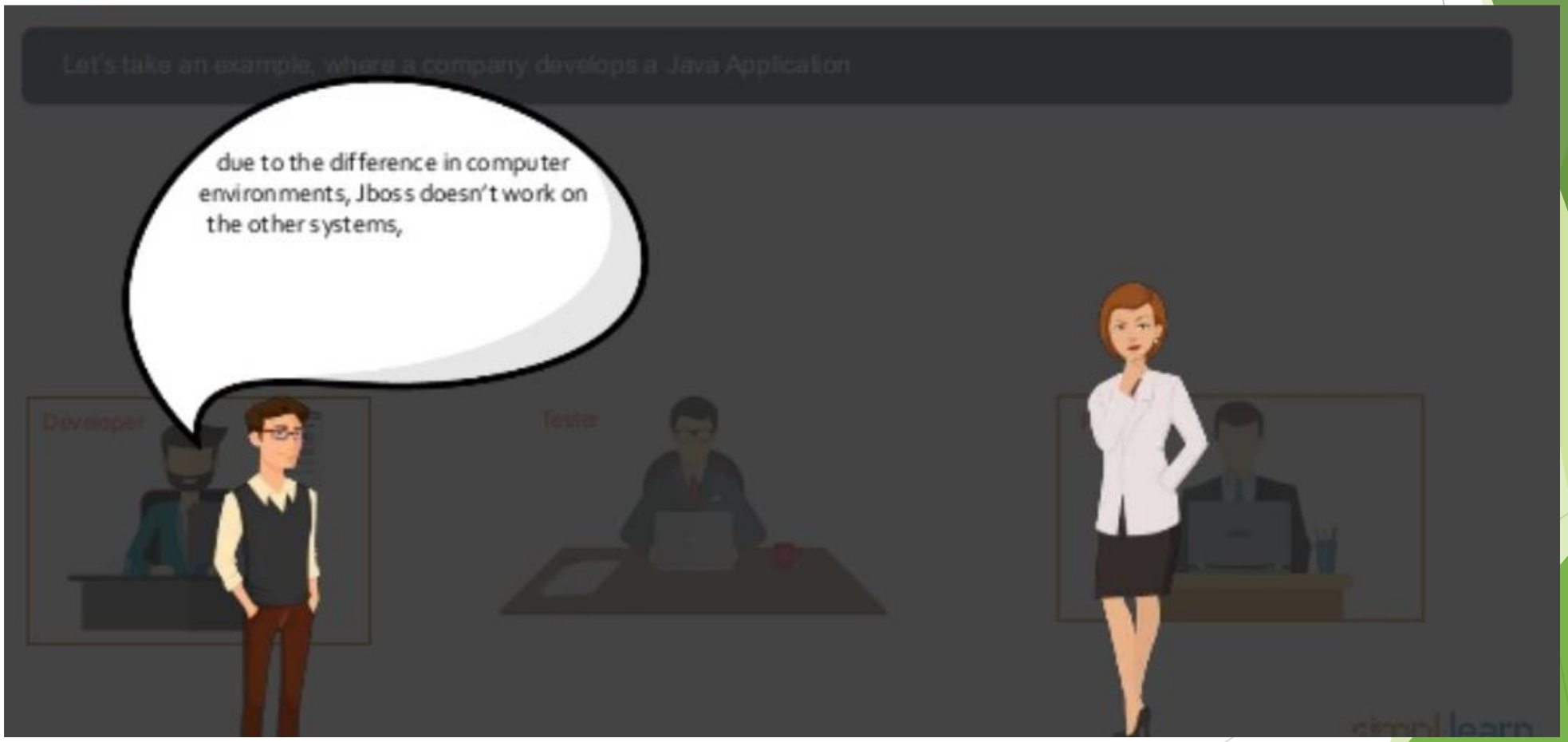
Why Docker Container?

Some Questions



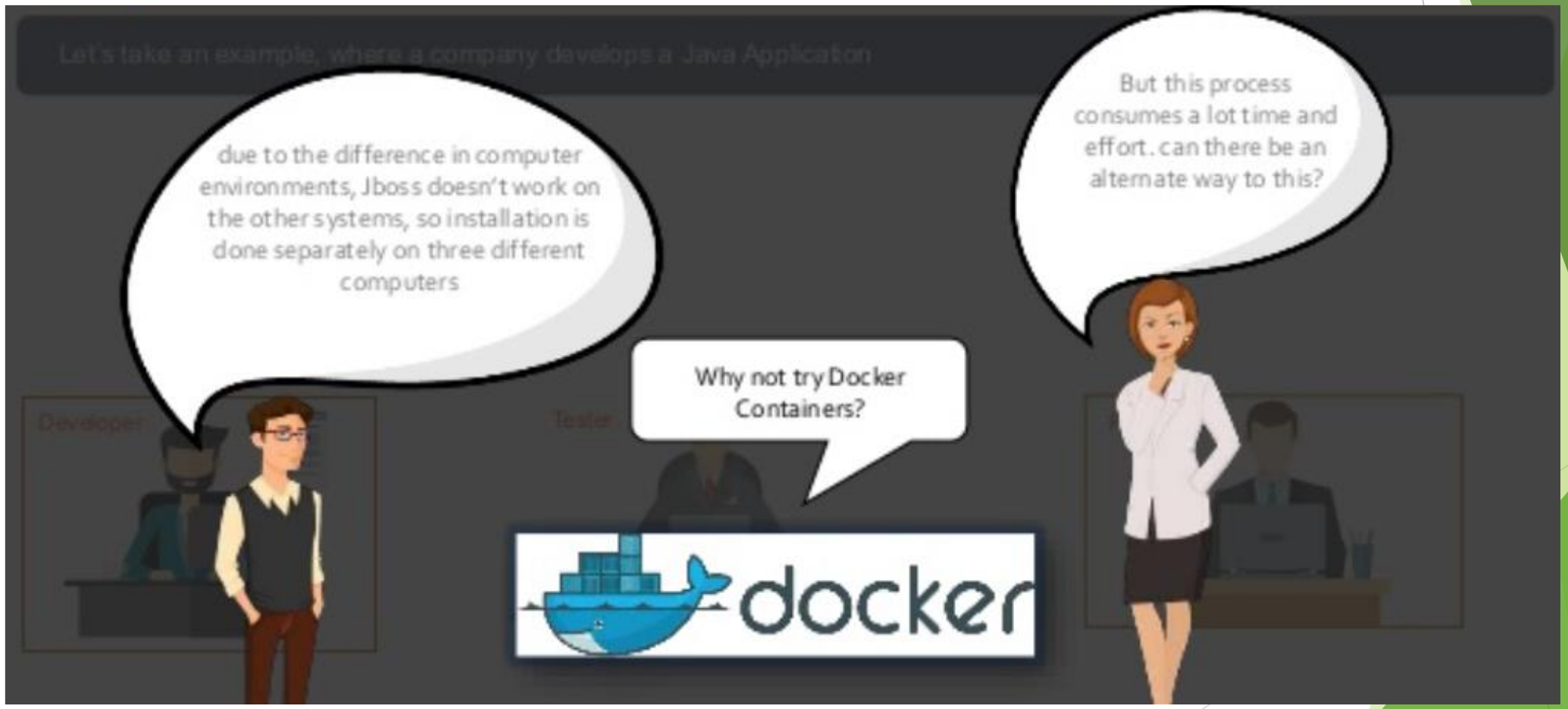
Why Docker Container?

Some Questions



Why Docker Container?

Some Questions



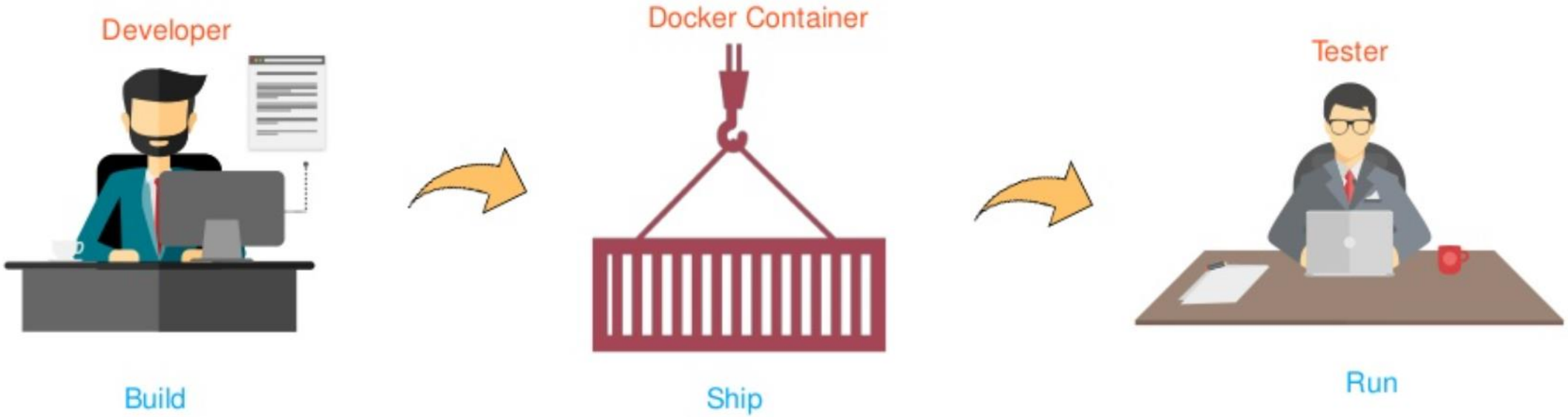
What is in it for you?

Let's get started

- What is Docker?
- Architecture of Docker
- What is a Docker Container?
- How to create a Docker Container?
- Benefits of Docker Containers
- Basic Commands of Containers
- Demo

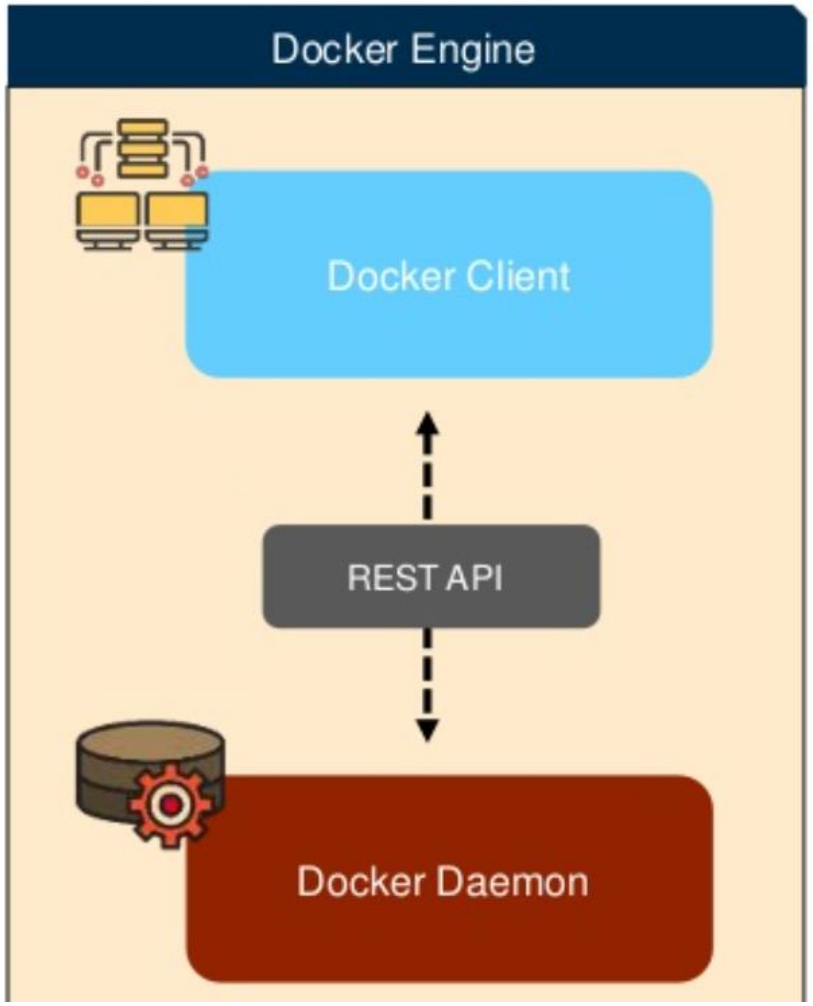
What is Docker?

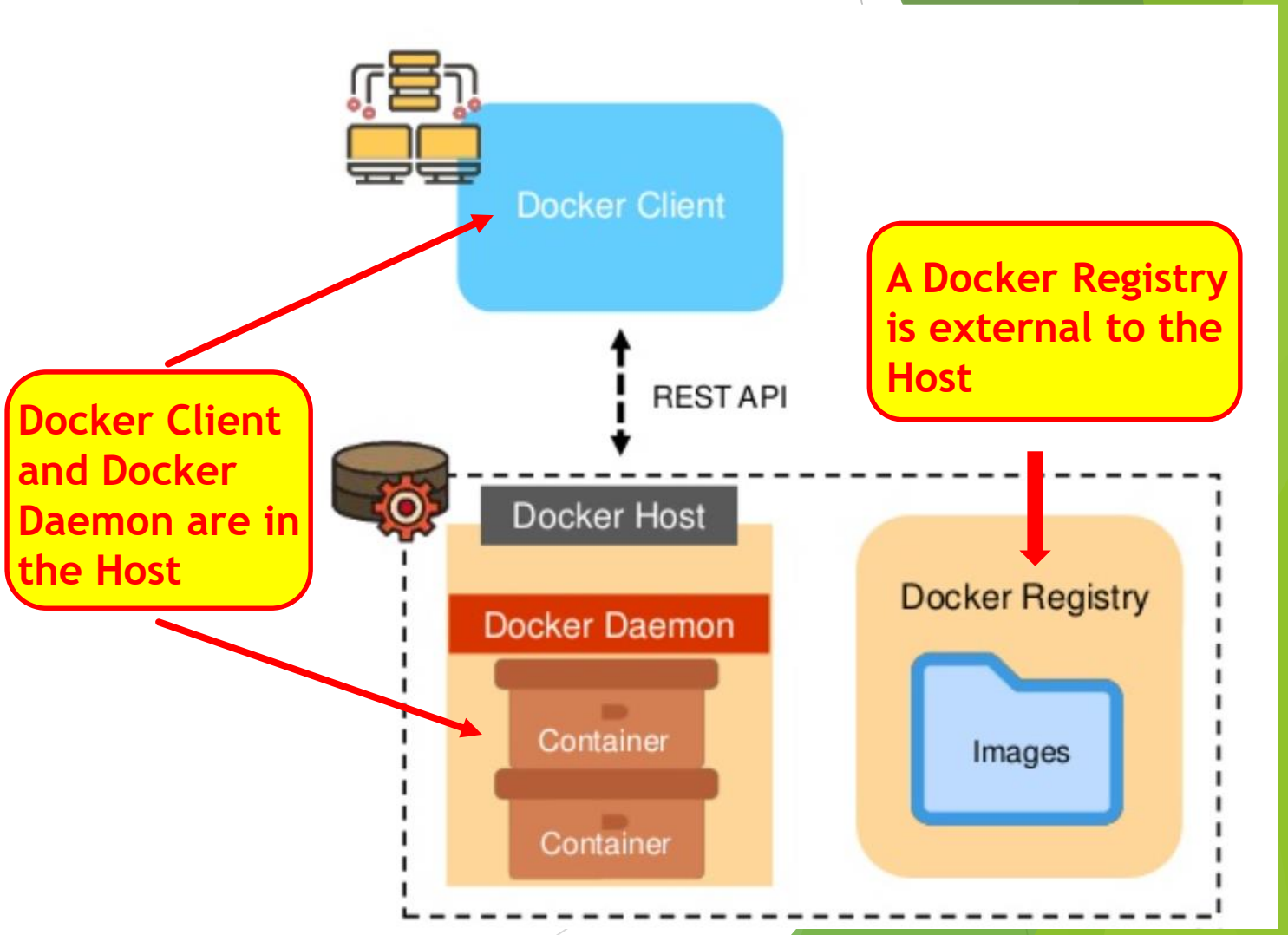
Docker is an open source platform that helps a user to package an application and it's dependencies into a Docker Container for the development and deployment of software



Architecture of Docker

Architecture of Docker



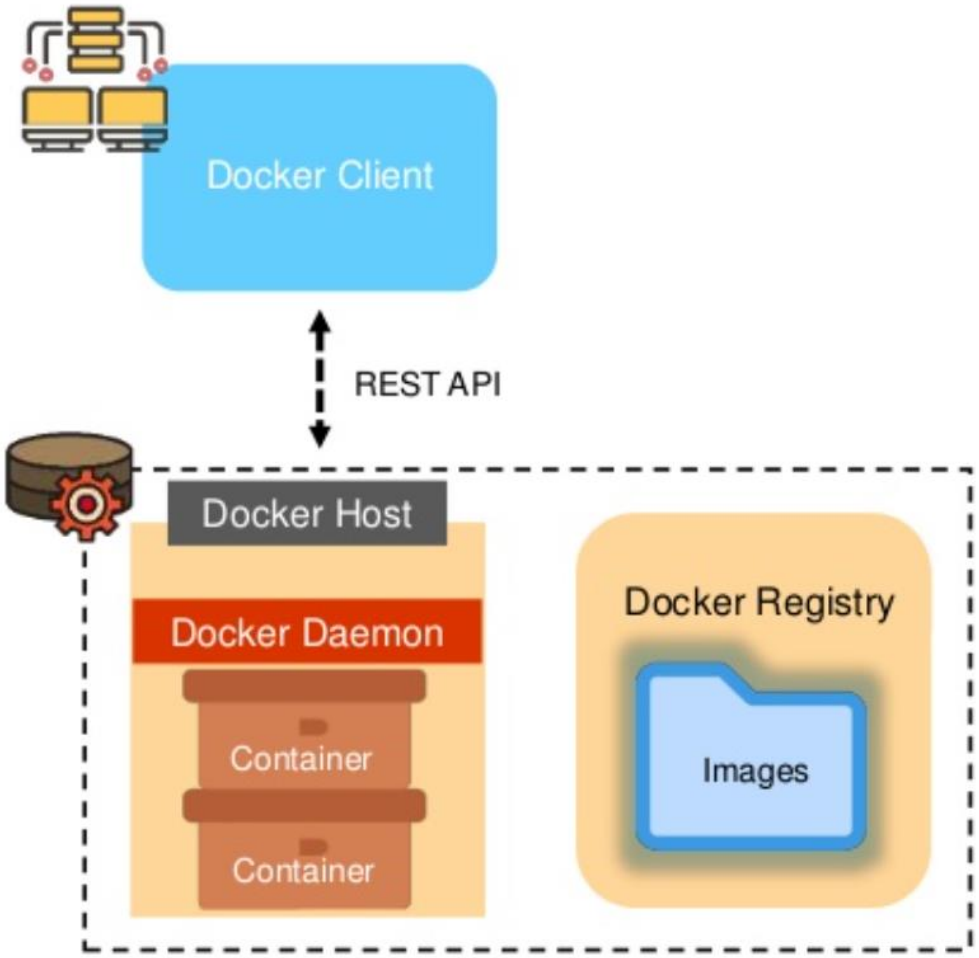


Architecture of Docker

Docker Client is a service which uses REST API to send commands to Docker Daemon through CLI commands

Docker Daemon checks the client request and communicates with the Docker components in order to perform a service

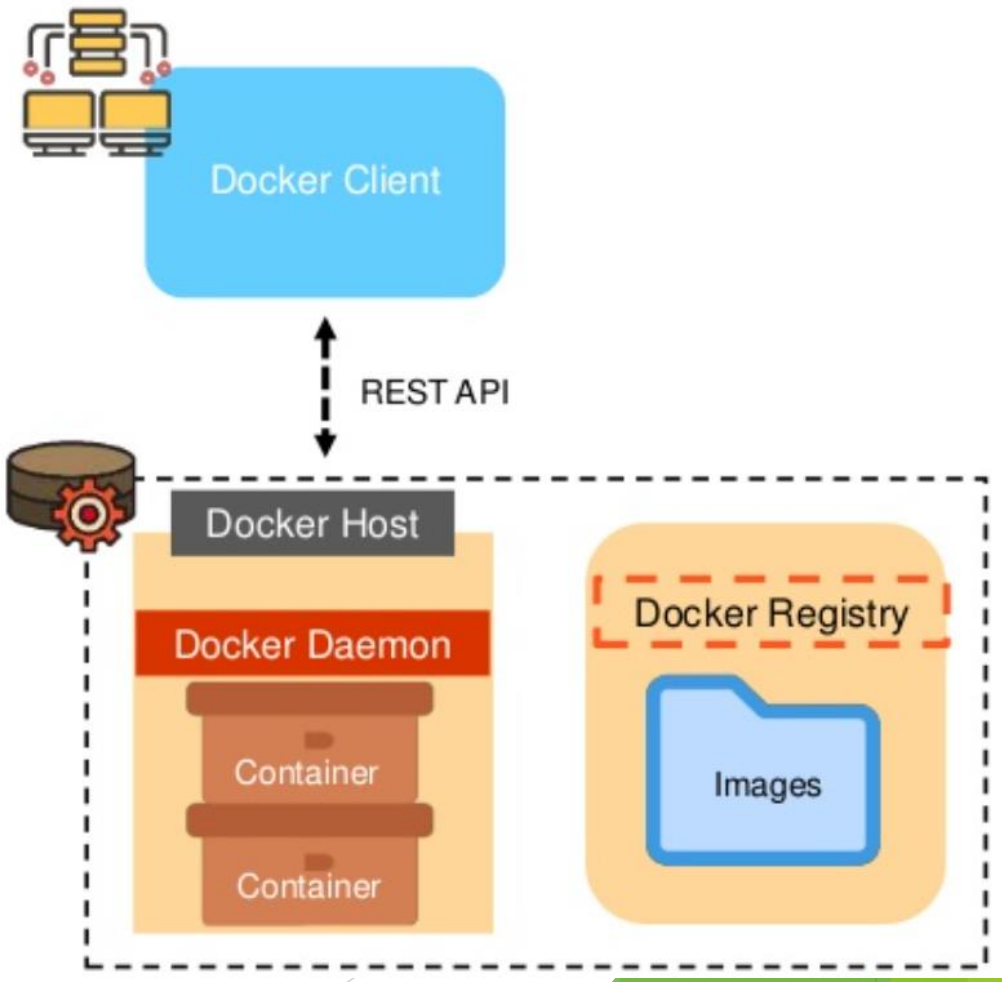
A Docker Image is a file of instructions which is used to create Containers



Architecture of Docker

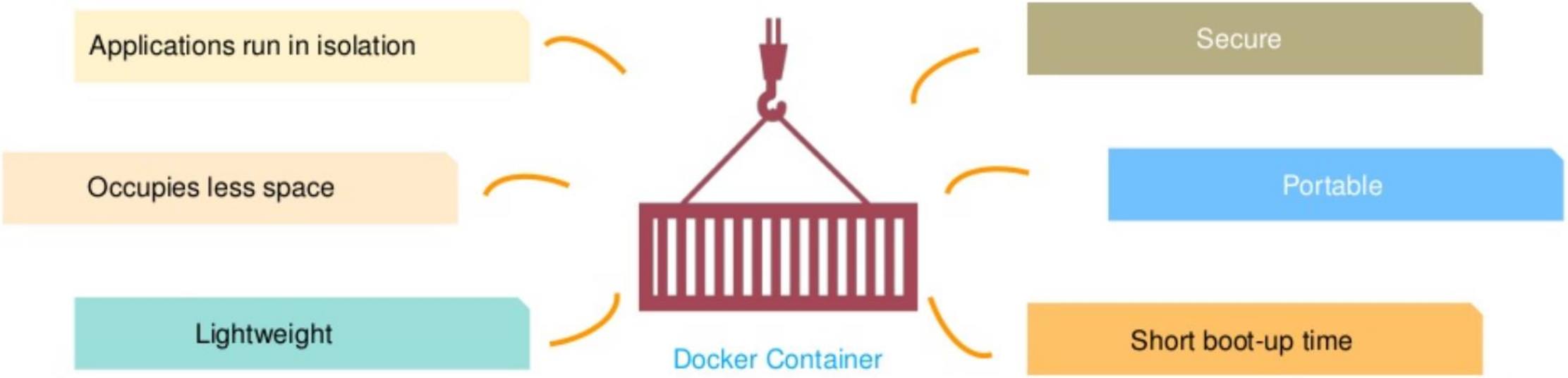
Docker Container is a portable executable package which includes applications and their dependencies

Docker Registry is a service used for hosting and distributing Docker images among users



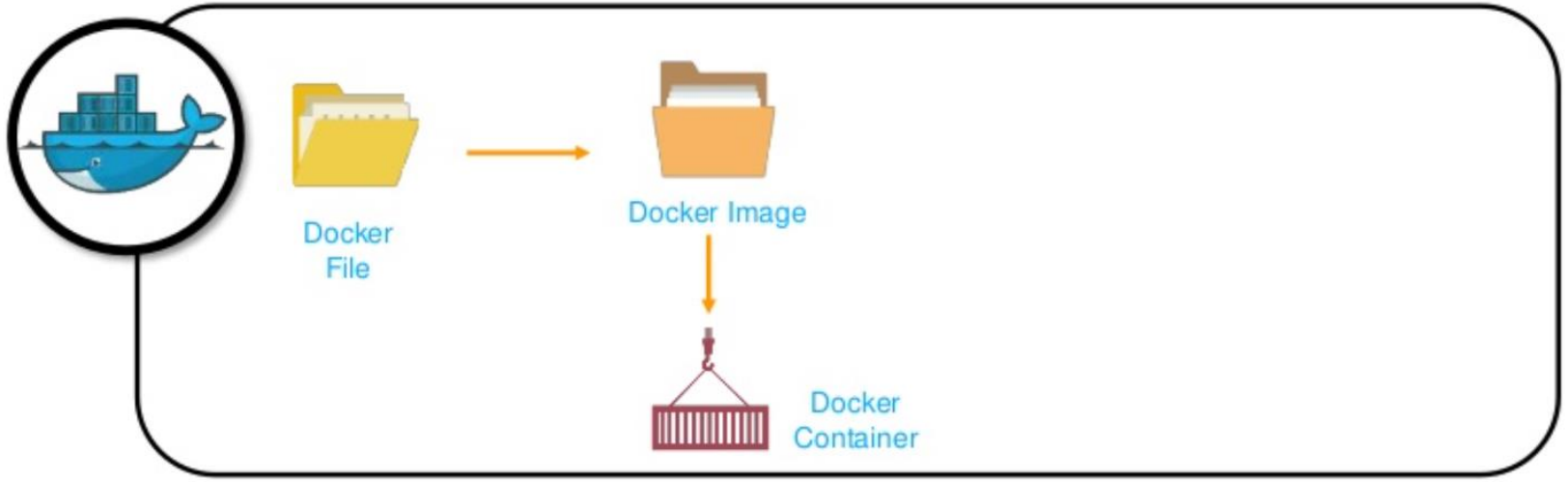
What is a Docker Container?

- Docker Container is an executable software package that includes all dependencies (frameworks, libraries, etc.) required to execute an application
- With Docker Containers, applications can work efficiently in different computer environments



How to create a Docker Container?

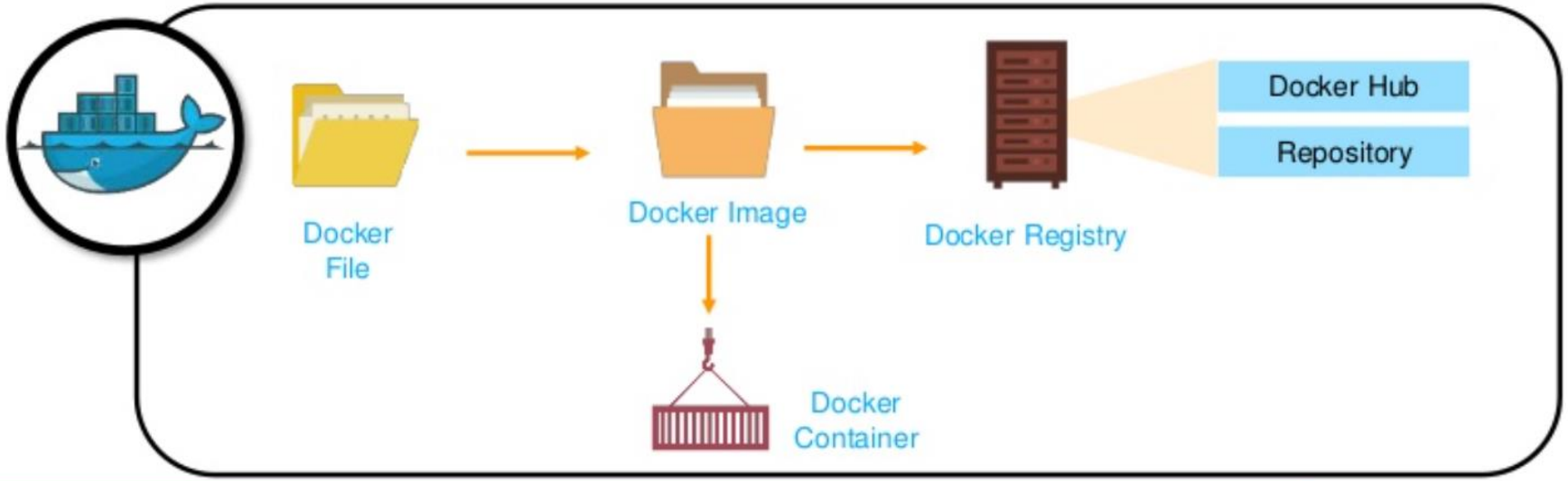
- Docker File creates a Docker Image using the build command
- A Docker Image contains of all the project's code
- With Docker Image, a user can run the code in order to create Docker Containers



Note: Command to run a Docker Container is Docker run <image-id>

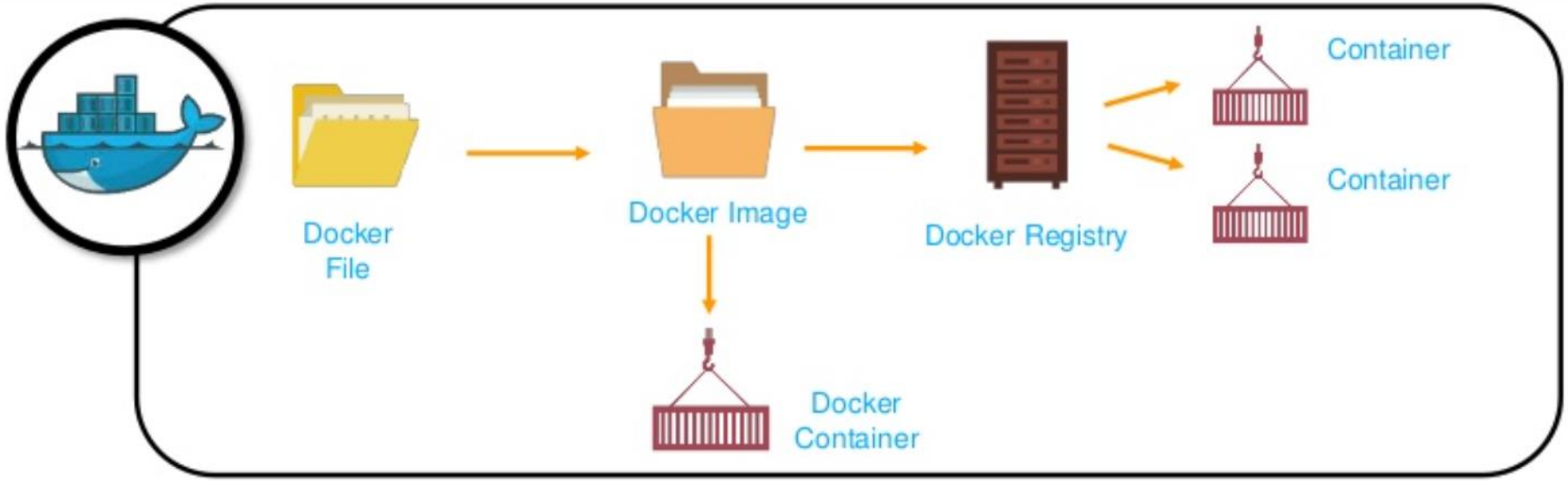
How to create a Docker Container?

- Once the Docker Image is created, it can be stored in the Docker Registry using Docker push command (Docker push image_name)
- When a Docker Image is created, it gets stored in a Docker Hub or in a Repository

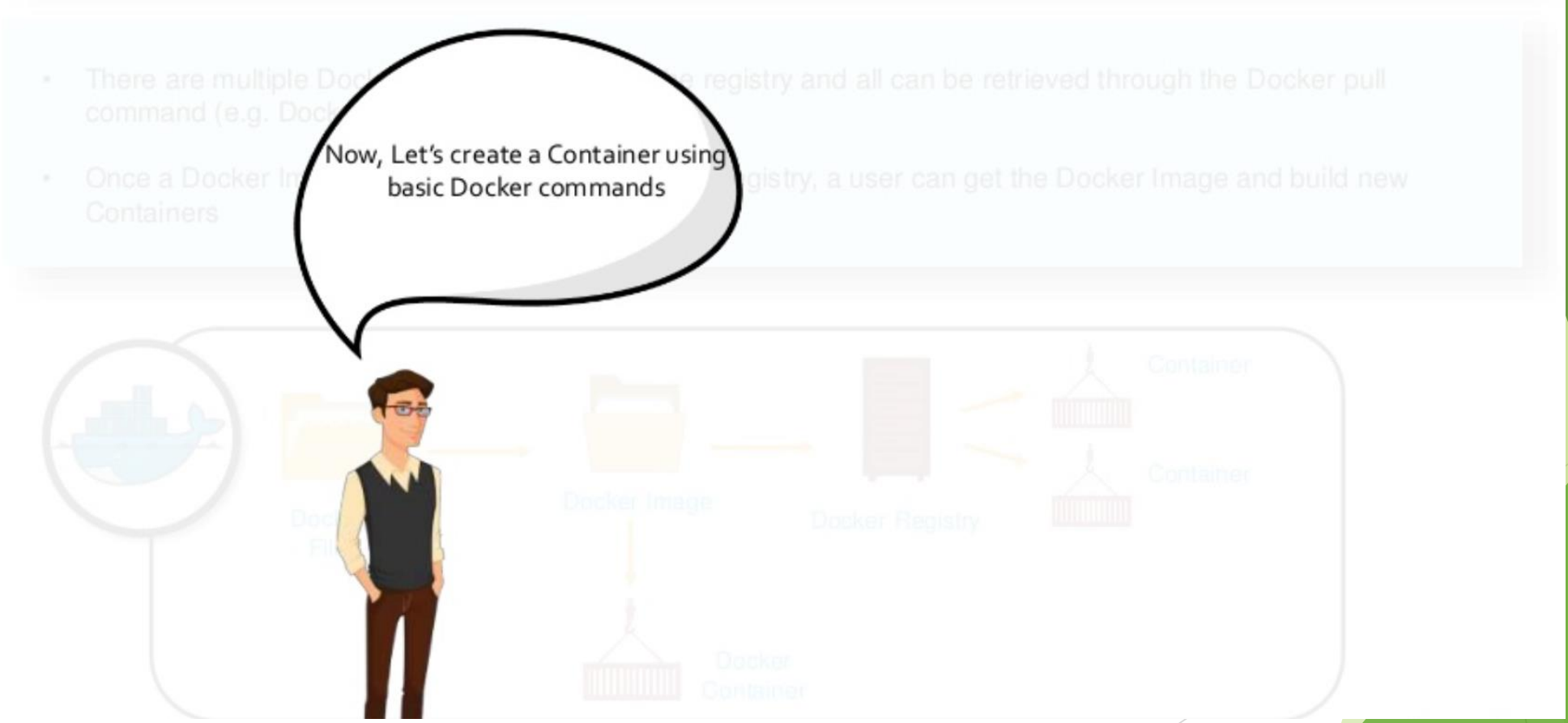


How to create a Docker Container?

- There are multiple Docker images available in the registry and all can be retrieved through the Docker pull command (e.g. Docker pull image_name)
- Once a Docker Image is retrieved from the Docker Registry, a user can build new Containers

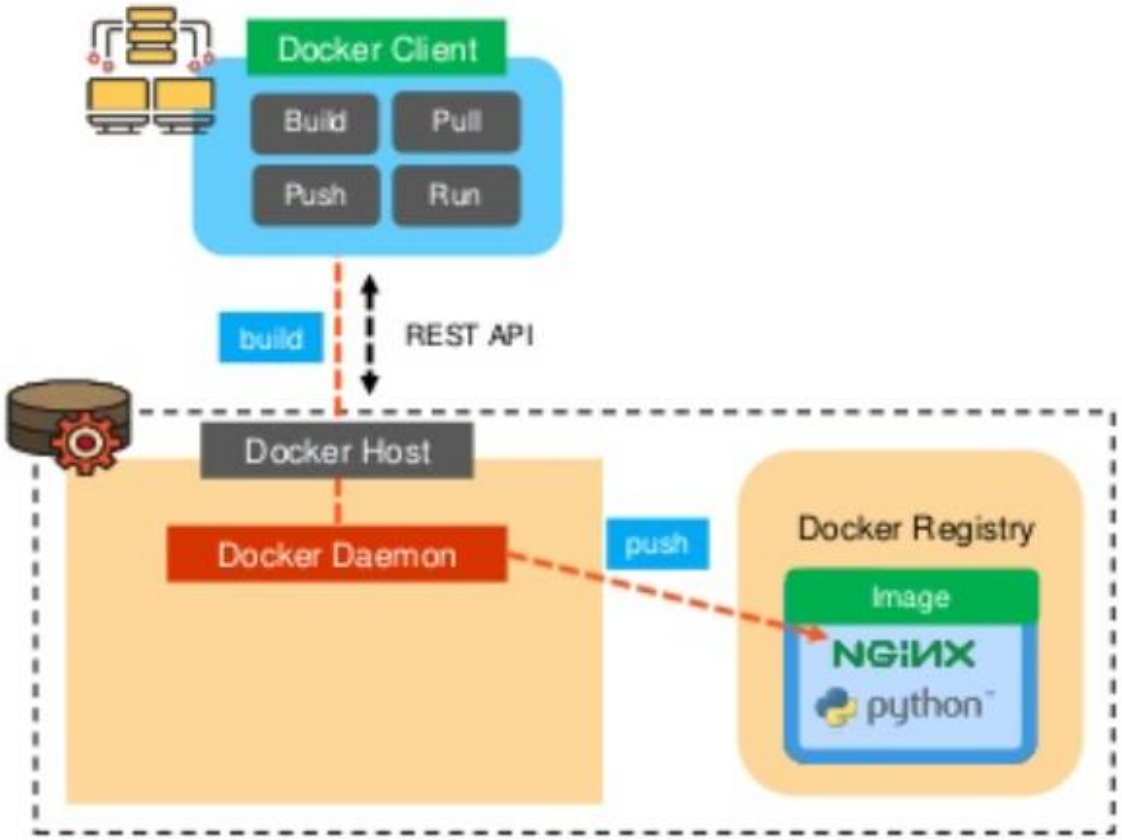


How to create a Docker Container?



How to create a Docker Container?

In Docker, a Dockerfile is used to build the image using *build command* and that image is stored into the registry using *push command*

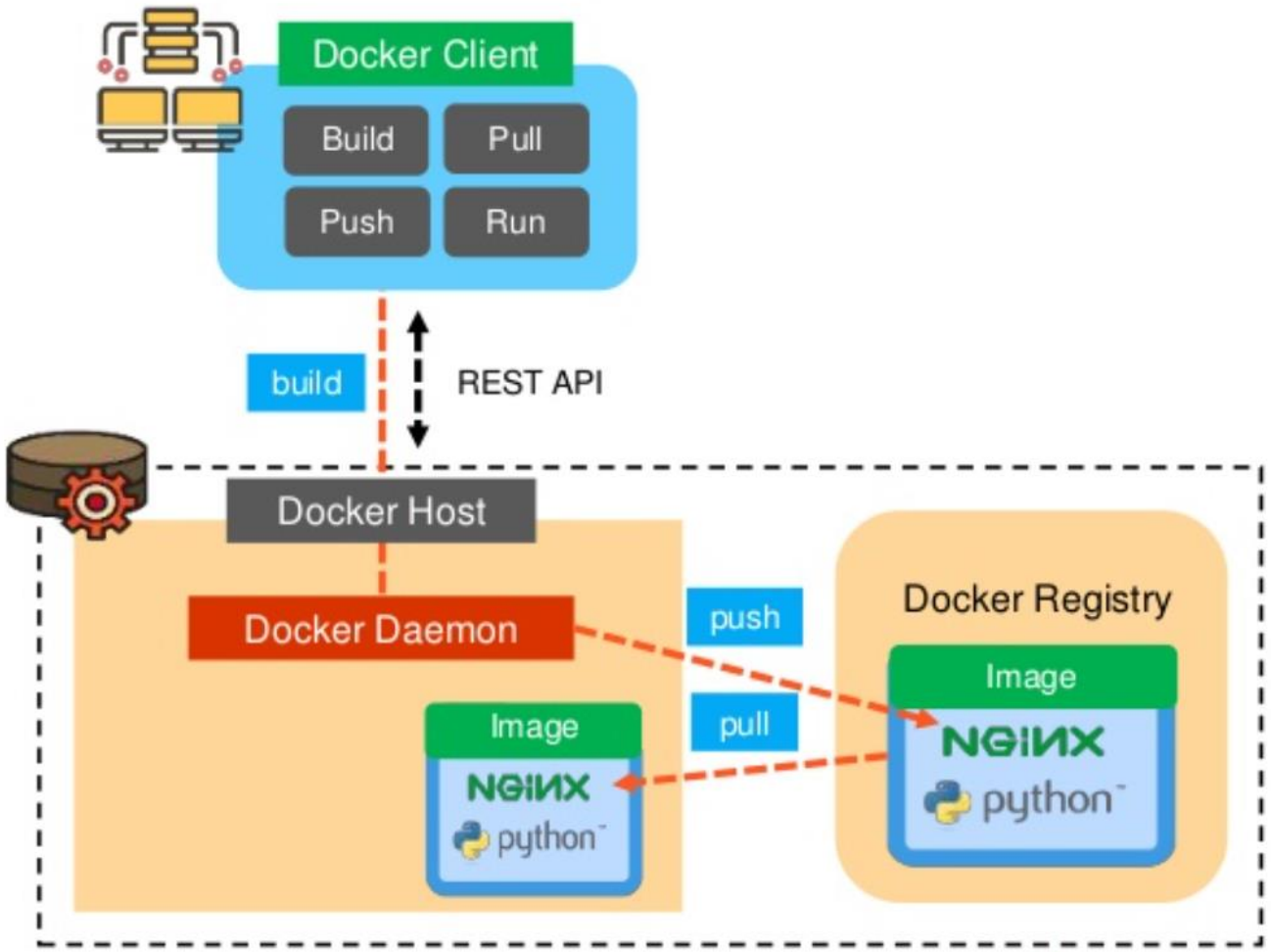


Note: Docker uses Docker images to run your code, not the Dockerfile

How to create a Docker Container?

In Docker, a Dockerfile is used to build the image using *build command* and that image is stored into the registry using *push command*

When you run *pull command*, Docker Image (NGNIX) is retrieved from the registry

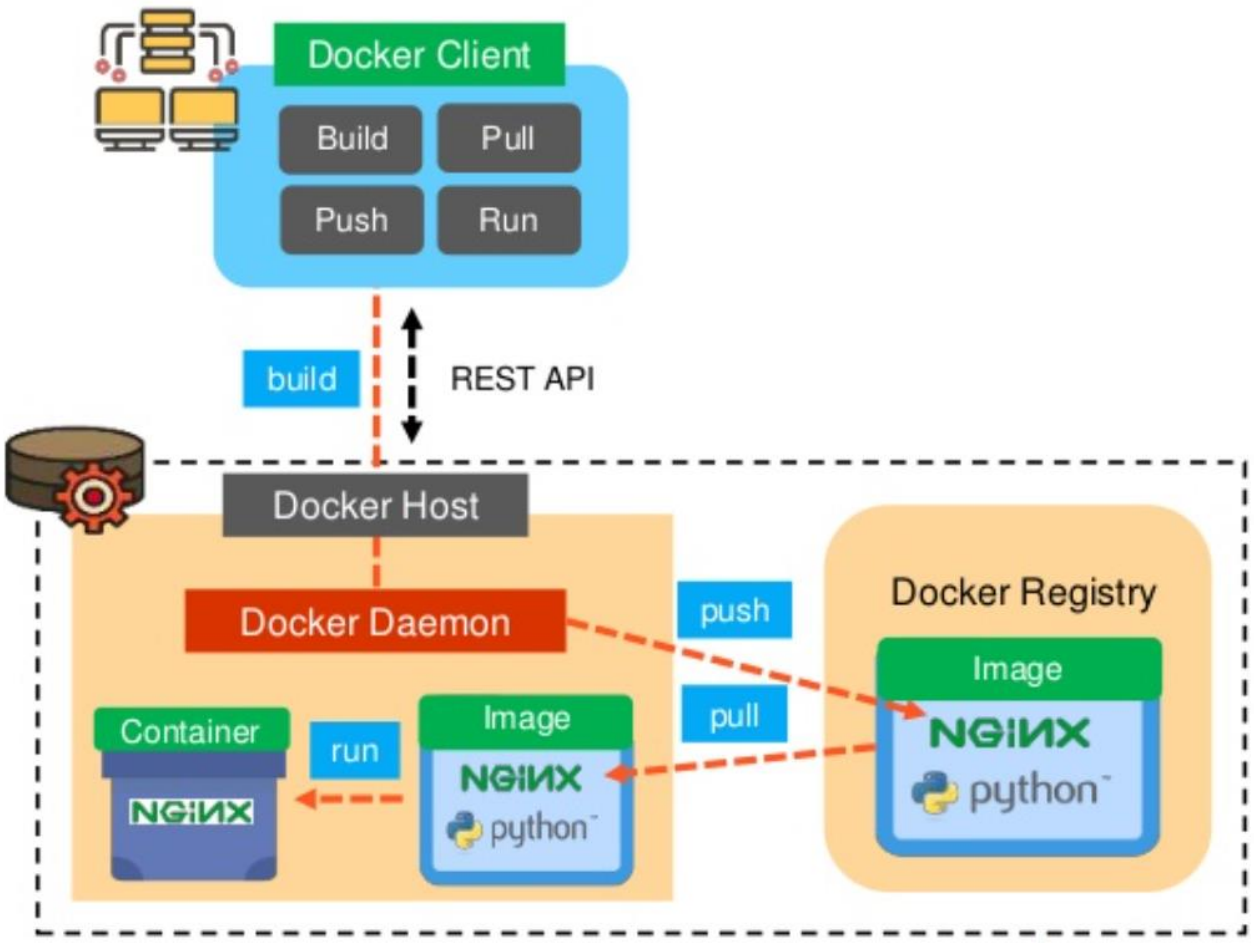


How to create a Docker Container?

In Docker, a Dockerfile is used to build the image using *build command* and that image is stored into the registry using *push command*

When you run *pull command*, Docker Image (NGNIX) is retrieved from the registry

Finally, a single Container (NGNIX) is built using Docker Image through the *run command*



How to create a Docker Container?

DID YOU KNOW?

- When a Container is created, a new layer is formed on top of the Docker Image layers called Container layer
- Each Container has a separate (R/W) Container layer and any changes made in a Docker Container is reflected upon the particular Container layer
- In case a Container is deleted, the Container layer also gets deleted

Note: Docker Image has multiple image layers and each Image layer is created by executing each command in the Dockerfile

Benefits of Containers

Containers have no external dependency for applications to run

As Containers are light-weight, they are easily shipped (deployed) to other computers and get executed on other computer environments regardless of their host operating systems

Data volumes can be shared and reused among multiple Containers

Containers run applications in isolation and also share the OS kernel with other Containers

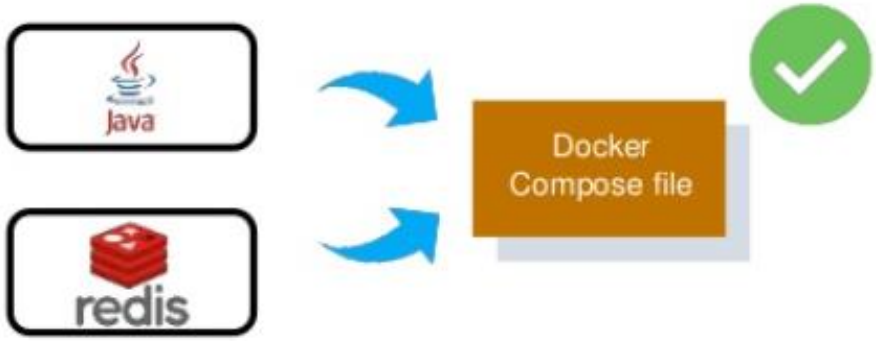
Benefits of Containers

Docker Compose

Docker Compose can be used to run multiple Containers in a single service

For example

Consider an instance where you have an application which requires Apache Tomcat and redis. Now, you can easily create one Docker Compose file to run both Containers in a single service



Summary

Docker Image
Contains all the codes that can be used to build a single container

Dockerfile
A config file for building a single container

Docker Compose
A config file to build multiple containers

Basic Docker Container Commands

Basic commands for Docker

- Docker Container commit - Command to create a new Docker image from the changes made in Container
- Docker Container cp - Command to copy files between the local filesystem and a Docker Container
- Docker Container prune - Command to remove all stopped Containers
- Docker Container kill - Command to terminate one or more running Containers
- Docker Container exec - Command to run a new command in a running Container
- Docker Container ls - Command to list Docker Containers
- Docker Container rm - Command to remove one or more Containers
- Docker Container restart - Command to restart one or more Containers

The End