

# Docker

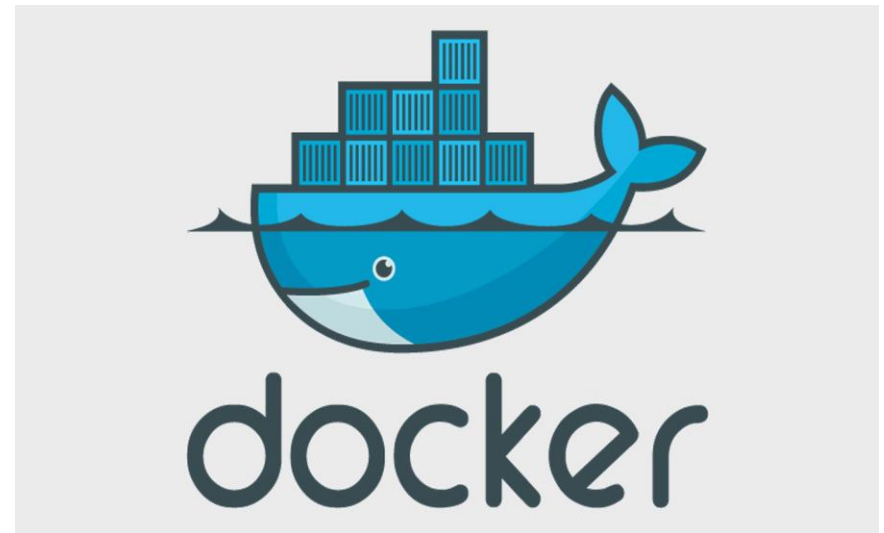
Presented by: Aarti Bhosale

## Agenda

- Docker Overview
- What & why Docker
- Docker container Vs VM
- Container
- Docker architecture
- Docker workflow
- Docker commands
- Use cases of Docker
- Dockerfile
- Demo

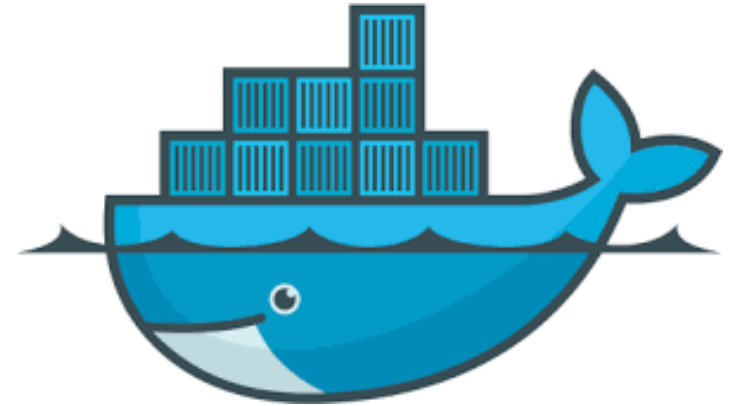
## Docker Introduction

- Started in France by Hykes within dotCloud
- Released as open source in March 2013
- Developed in Go programming language
- OS-Level virtualization
- Used by Millions of user



## Docker Community

- 3330+ contributors
- 242+ billion docker pull
- 3.1K GitHub stars
- 7M hub users
- 2.9M desktop installation
- 20-30% of companies adopted docker
- 70-75% deployment



**Reference:** <https://github.com/docker/docker.github.io>

## What is Docker?

It is tool used to automate deployment of application in the form of lightweight container so application can work efficiently on different platform.

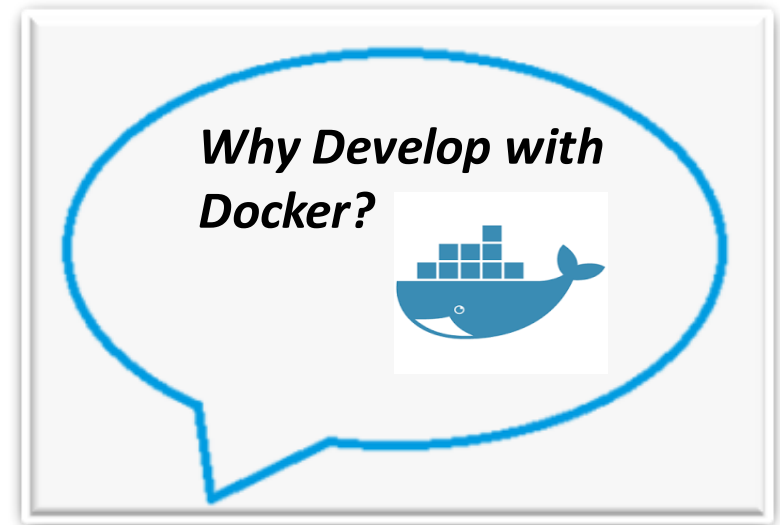
- It is open source tool
- Easy to create, deploy & run application using container
- Package your application along with all its dependencies together in form of container
- Able to manage infrastructure just like application
- Boost application performance and reduce size of application



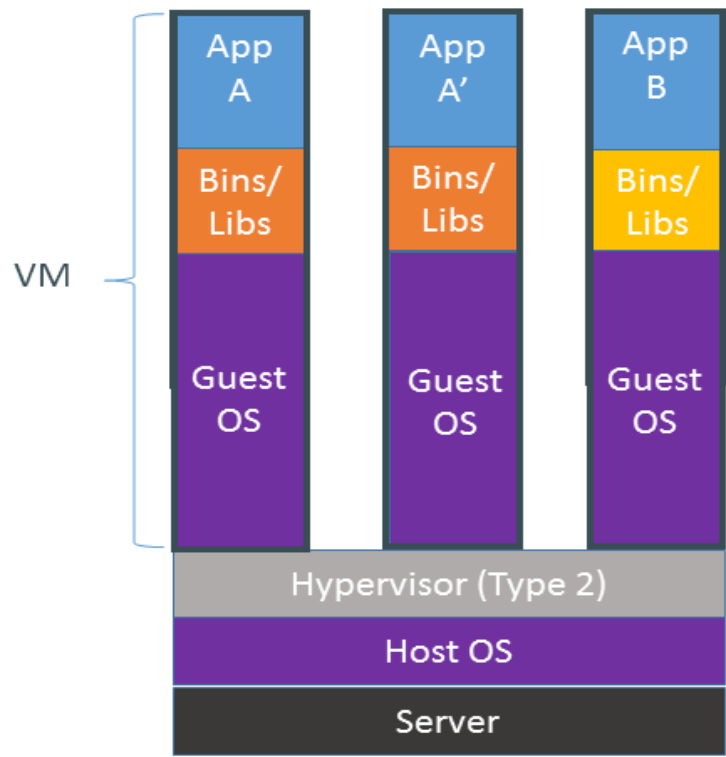
## Why Docker?

### Couple of reasons to use docker -

- Faster delivery
- Light weight
- Efficient use of system resources
- Portability
- Sharability
- Supported by all cloud platform

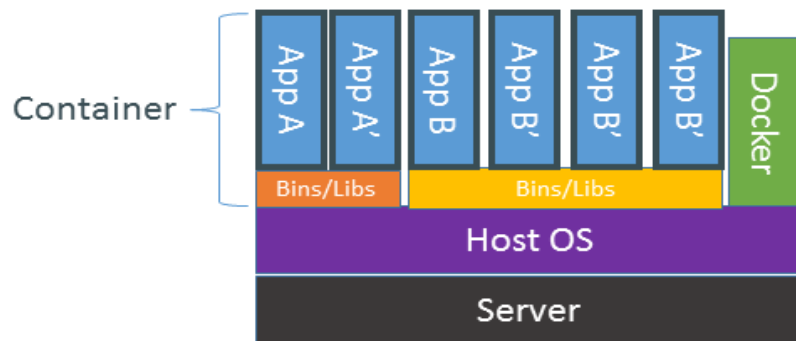


## Container vs VM





Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart



## Docker container vs VM

Criteria	 Virtual Machine	Docker 
OS support	Occupies a lot of memory space	Docker Containers occupy less space
Boot-up time	Long boot-up time	Short boot-up time
Performance	Running multiple virtual machines leads to unstable performance	Containers have a better performance as they are hosted in a single Docker engine
Scaling	Difficult to scale up	Easy to scale up
Efficiency	Low efficiency	High efficiency
Portability	Compatibility issues while porting across different platforms	Easily portable across different platforms
Space allocation	Data volumes cannot be shared	Data volumes can be shared and reused among multiple containers



## Why should we care for Docker

### For Developers

- Simple creation of Environments
- Portable App containers
- Build, test & package on production like environment
- Increase the quality of code produced by Developers
- Fast and simple
- 16,000+ apps available on Docker Hub

### For DevOps

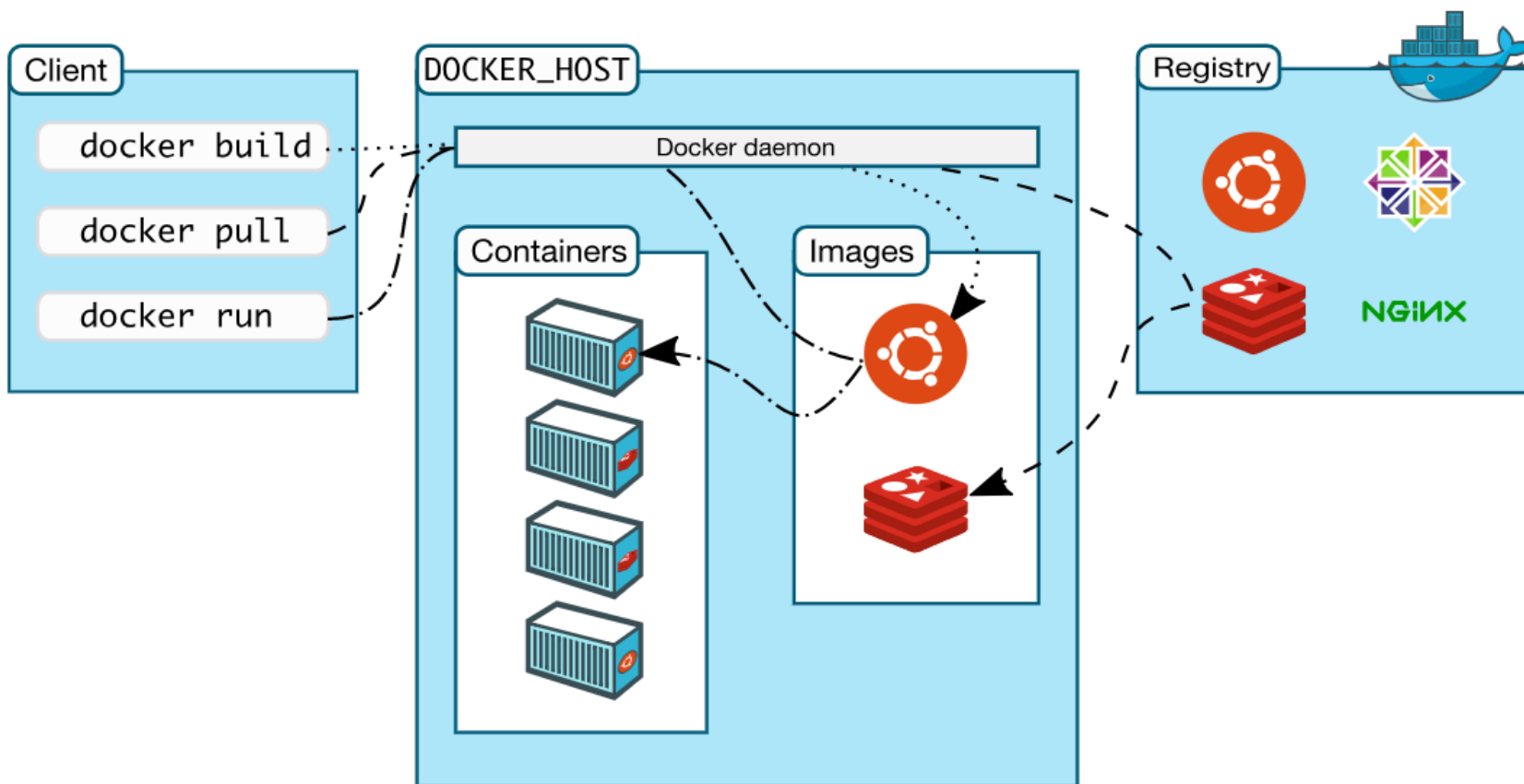
- Provide standard Environments for Dev, Test & Production
- Reducing “works on my machine” finger-pointing
- Helps Infra teams to deploy & run any App on any infrastructure
- Entire lifecycle more efficient and repeatable
- Compared to VM's, reduces costs & improves speed
- Deployment & Release Automation
- Perfect for DevOps – CI, CD, Release, Scaling, Clouds, etc

## What is containers?

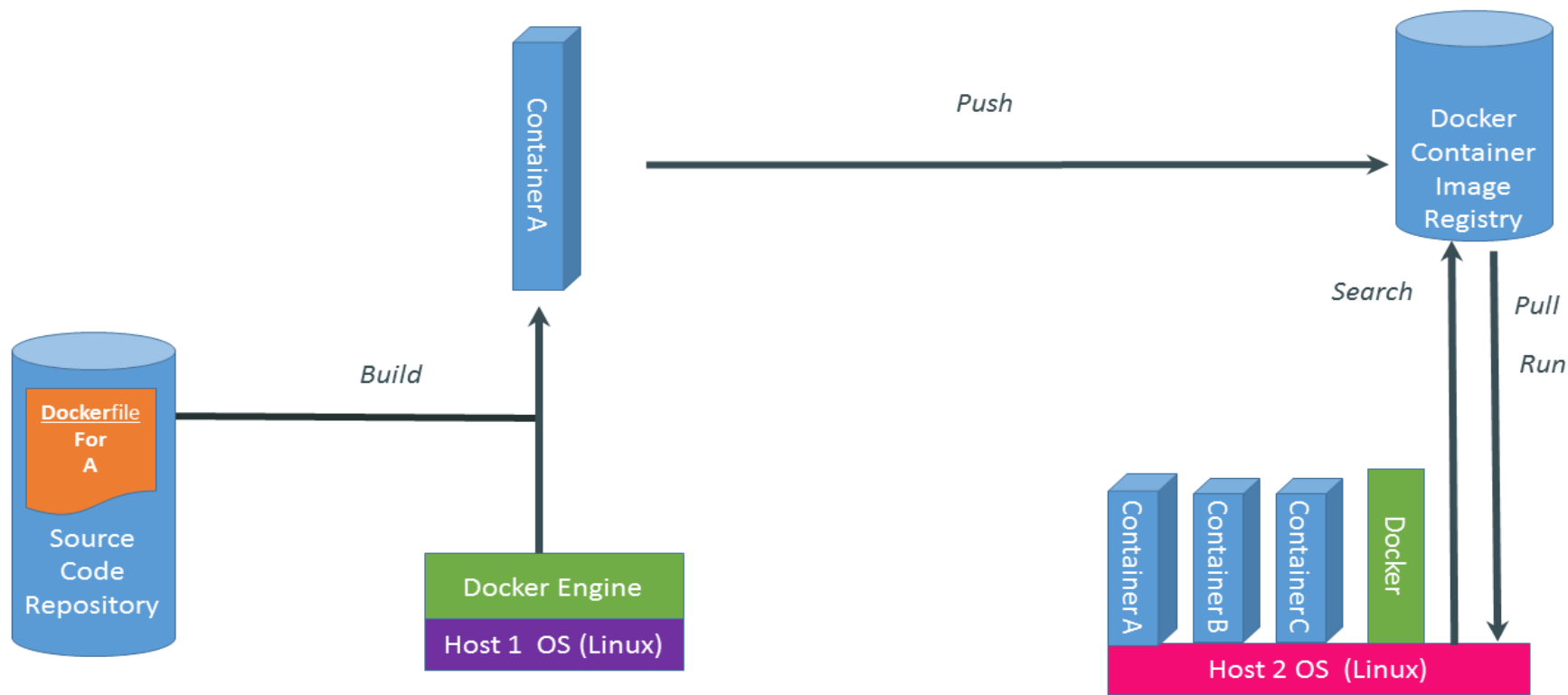
- Standardized unit of software
- Portable packaging for application
- Run everywhere
- Small, lightweight & fast



## Docker Architecture



## Docker Workflow



## Docker Commands

Few basic docker commands

Command	Description
version	Get the currently installed docker version
pull	Pull the images from docker repository
run	Create container from an image
ps	List the running container
ps -a	Show all the running and exited container
exec	Access the running container
stop	Stops a running container
kill	Kills the container by stopping its execution immediately
save	Save an image to a tar archive

## Docker Commands

Command	Description
commit	Creates a new image of an edited container on the local system
login	Login to the docker registry server
push	Push an image to the docker registry
images	Lists all the locally stored docker images
rm	Delete a stopped container
rmi	Delete an image
build	Build an image from specified dockerfile
logs	Fetch the logs of a container
start	Start a stop container
restart	Restart a running container

## What is Dockerfile?

Dockerfile is plain text file which contains all commands/instruction in order to, needed to build a given image.

Example:

```
# Use official ubuntu 16.04
FROM ubuntu:16.04

# Update & upgrade ubuntu
RUN apt-get update
RUN apt-get upgrade -y

# Install git
RUN apt-get install git -y

# Clone git repo
RUN git clone https://github.com/RKostadinov/honepot-project.git

# Make port 3306 available to the world outside this container
EXPOSE 3306

# Run start-up.sh when the container launches
CMD ["/honepot-project/Database/start-up.sh"]
```

## Dockerfile Instruction

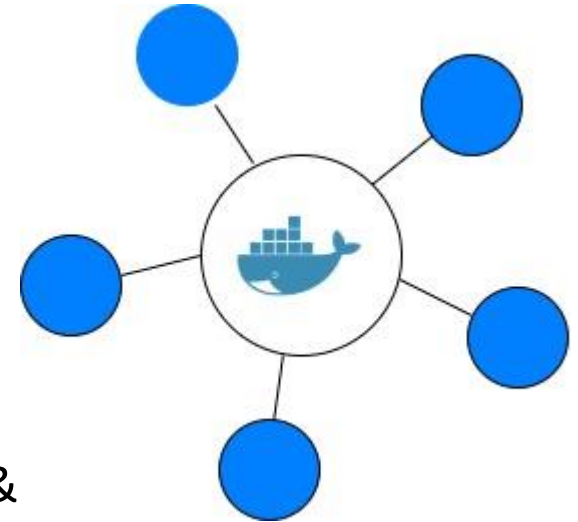
Basic instruction used in Dockerfile

Instruction	Description
FROM	Specify base image
WORKDIR	Set working directory
ADD	Copy files, directories or remote url files to container image
COPY	Copy files, directories to container image
RUN	Execute specific command
CMD	Specify command at the time of container execution
ENTRYPOINT	Configure and run container as an executable
ARG	Defines variable that user can pass at a build time
LABEL	Specify metadata information in key-value pair



## Use Cases for Docker

- Automate packaging & deployment of Applications
- Isolation
- Simplifying configuration
- Developer Productivity
- Rapid Deployment
- Continuous integration/deployment
- Better disaster recovery
- Scaling Web apps, databases and backend services
- Micro-services architecture / Service orchestration & discovery





Any  
questions ?

Thank You!