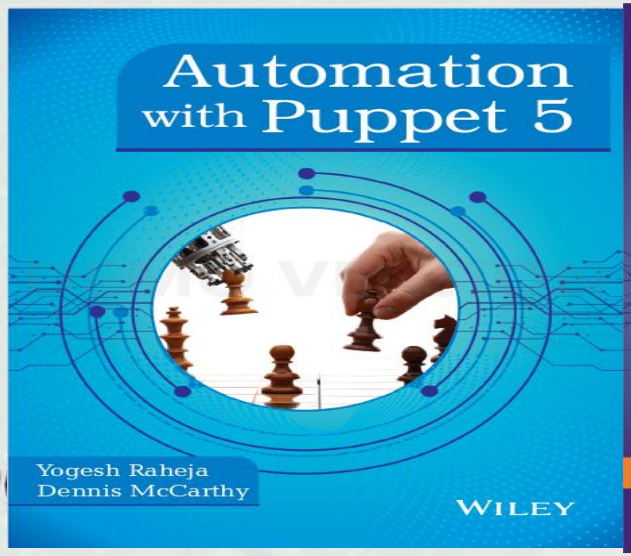
Docker Fundamentals

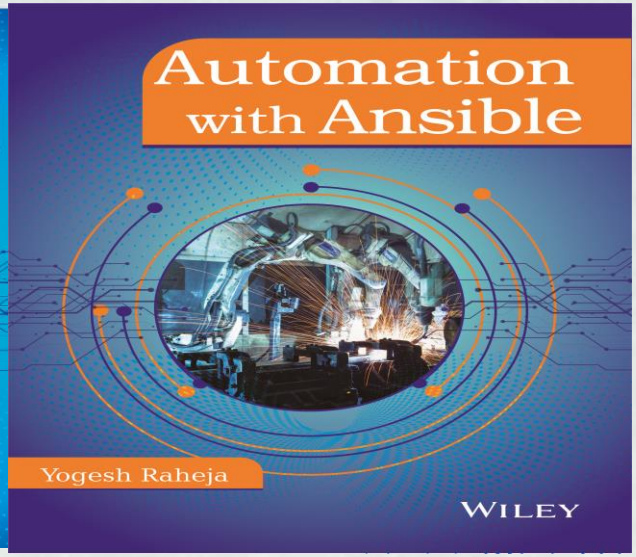
THINKNYX
TECHNOLOGIES

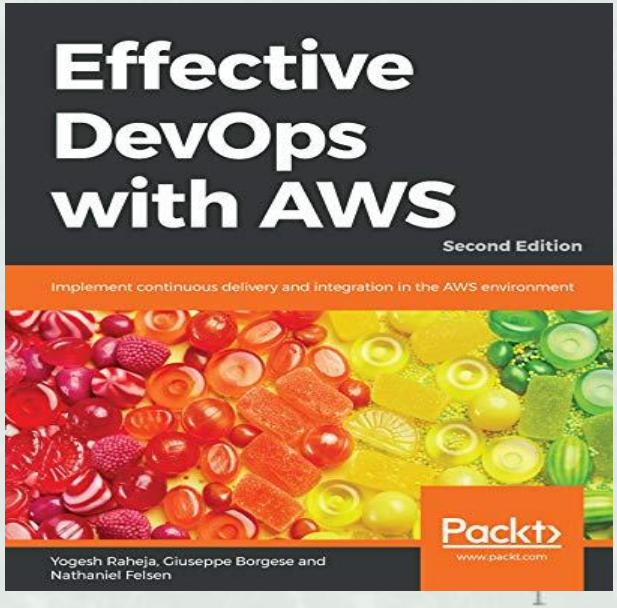
Yogesh Raheja
+91-9810344919
yogesh.raheja@thinknyx.com

WHO AM I

YOGESH RAHEJA







Agenda

- Introduction
- Docker Components
- Classroom Environment
- Containers
- Docker – Images
- Docker - Building Images
- Deep Dive – Images
- Deep Dive – Containers
- Container Network Model
- Docker Volumes

Session: 1

Introduction

About Docker Inc.

Thinknyx

- Docker Inc. used to be 'dotCloud' Inc.
- 'dotCloud' Inc. used to be a French company.
- Docker Inc. is the primary sponsor and contributor to the Docker Project:
  - Hires maintainers and contributors.
  - Provides infrastructure for the project.
  - Runs the Docker Hub.
- HQ in San Francisco.
- Backed by more than 100M in venture capital.

Results

Thinknyx

Speed

- No OS to boot = applications online in seconds

Portability

- Less dependencies between process layers = ability to move between infrastructure

Efficiency

- Less OS overhead
- Improved VM density

Adoption in Just 4 years

Thinknyx

14M
Docker Hosts

900K
Docker apps

77K%
Growth in Docker job listings

12B
Image pulls Over 390K% Growth

3300
Project Contributors

Session: 2

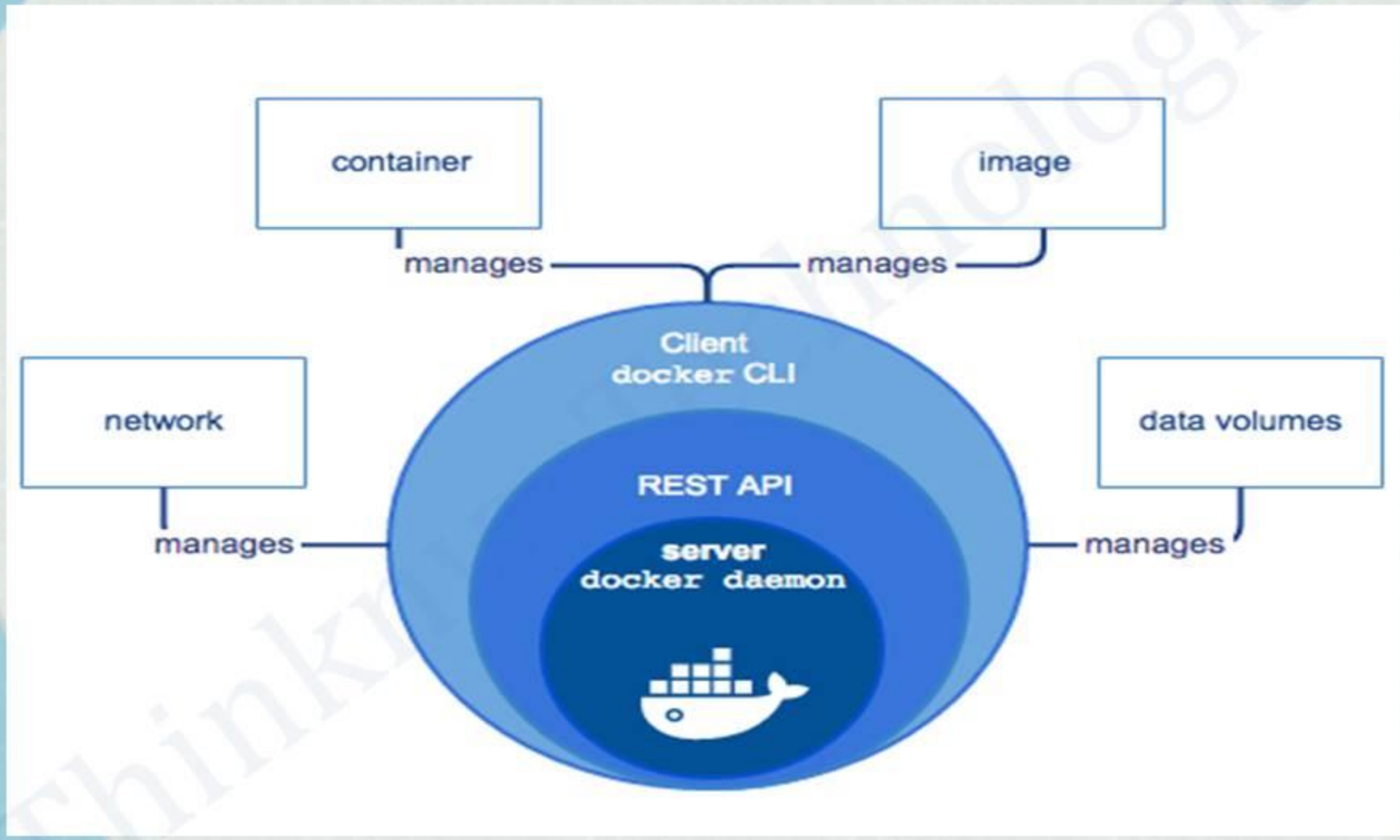
Docker Components

Docker Overview

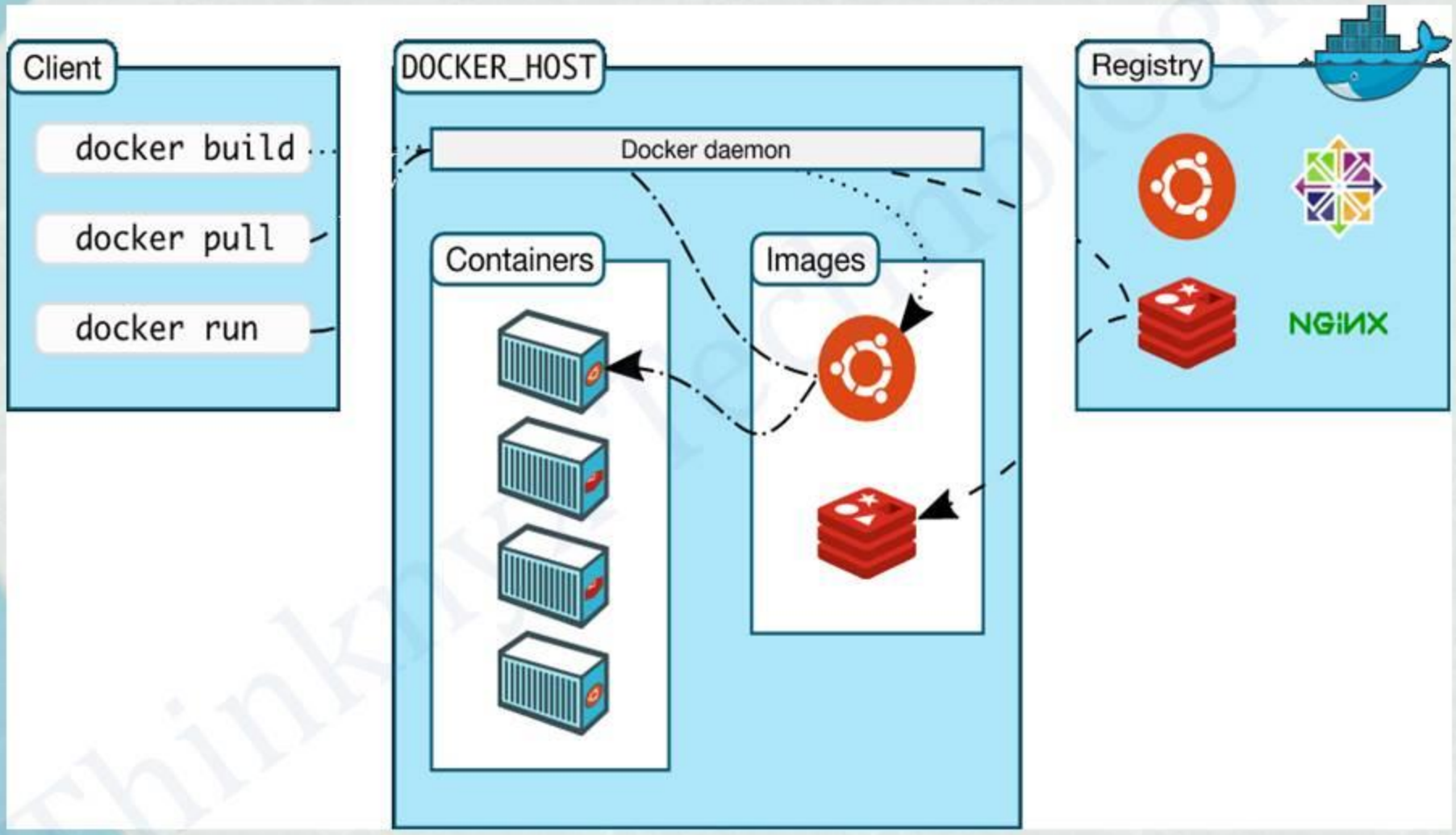
Thinknyx

- Docker is an open platform for developing, shipping, and running applications.
- Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- With Docker, you can manage your infrastructure in the same ways you manage your applications.
- By using Docker's methodologies for shipping, testing, and deploying, you can reduce time of customer delivery.

Docker Engine



Docker Architecture



Docker Architecture

Thinknyx

- Docker uses a client-server architecture.
- The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers.
- The Docker client and 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 using a REST API, over UNIX sockets or a network interface.

Docker Architecture

Image

The basis of a Docker container. The content at rest.

Container

The image when it is 'running.' The standard unit for app service

Engine

The software that executes commands for containers. Networking and volumes are part of Engine. Can be clustered together.

Registry

Stores, distributes and manages Docker images

Control Plane

Management plane for container and cluster orchestration

Docker Images

Thinknyx

- A Docker image is a read-only template with instructions for creating a Docker container.
- For example, an image might contain an Ubuntu operating system with Apache web server and your web application installed. You can build or update images from scratch or download and use images created by others.
- A docker image is described in text file called a Dockerfile, which has a simple, well-defined syntax.
- Docker images are the build component of Docker.

Docker Containers

Thinknyx

- A Docker container is a running instance of a Docker image.
- You can run, start, stop, move, or delete a container using Docker API or CLI commands.
- When you run a container, you can provide configuration metadata such as networking information or environment variables.
- Each container is an isolated and secure application platform, but can be given access to resources running in a different host or container, as well as persistent storage or databases.
- Docker containers are the run component of Docker.

Docker Registeries

Thinknyx

- A docker registry is a library of images.
- A registry can be public or private, and can be on the same server as the Docker daemon or Docker client, or on a totally separate server.
- Docker registries are the distribution component of Docker.
- "Docker Hub" is known as global registry.

Docker Orchestration

Thinknyx

- Three tools for orchestrating distributed application with Docker

- Docker Machine
- This is a tool that provisions Docker hosts and install the Docker Engine on them

- Docker Swarm
- This is a tool that clusters multiple Docker Engines and do the Scheduling of containers

- Docker Compose
- This is a tool that create and manage multi-container application i.e combine different container for an application

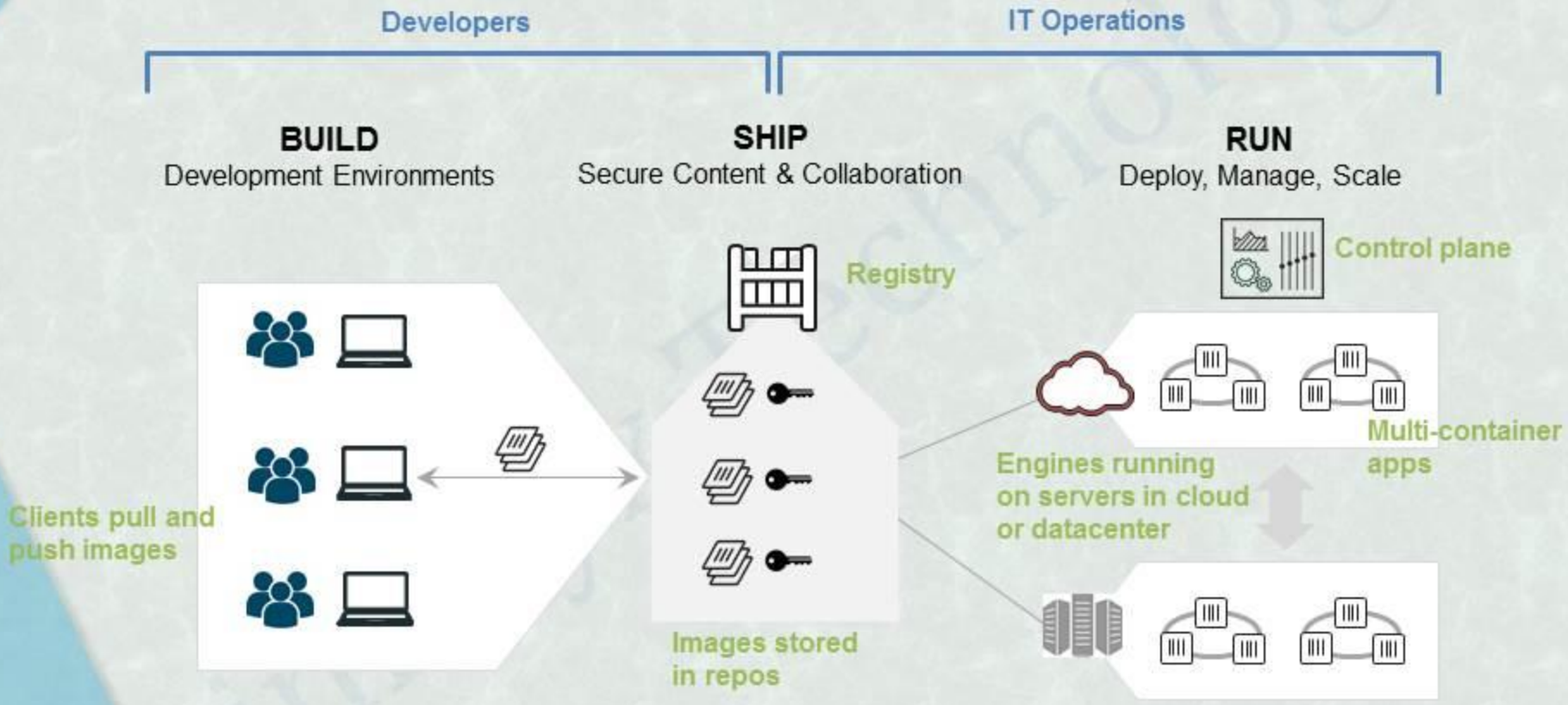
Docker Features

Thinknyx

- Lightweight
  - Containers running on a single machine all share the same operating system kernel so they start instantly and make more efficient use of RAM. Images are constructed from layered filesystems so they can share common files, making disk usage and image downloads much more efficient.
- Open
  - Docker containers are based on open standards allowing containers to run on all major Linux distributions and Microsoft operating systems with support for every infrastructure.
- Secure
  - Containers isolate applications from each other and the underlying infrastructure while providing an added layer of protection for the application.

Container as a Service

Thinknyx



Session: 3

Classroom Environment

Docker Engine Install Demo

Thinknyx

- Docker Engine/Client would be installed on Training Environment as demo LAB.

- https://docs.docker.com/engine/installation/linux/centos/

Manage Docker as a non-root user

Thinknyx

- The docker daemon binds to a Unix socket instead of a TCP port.
- By default that Unix socket is owned by the user "root" and other users can only access it using sudo.
- The docker daemon always runs as the root user.
- If you don't want to use sudo when you use the docker command, add users to Unix group called "docker" example: usermod -aG docker <user-name>
- When the docker daemon starts, it makes the ownership of the Unix socket read/writable by the docker group.

Docker Test

Thinknyx

- docker –version
- Test the docker functioning: docker run hello-world

Session: 4

Containers

How Container works

Thinknyx

- A container uses the host machine's Linux kernel, and consists of any extra files you add when the image is created, along with metadata associated with the container at creation or when the container is started.
- Each container is built from an image.
- The image defines the container's contents, which process to run when the container is launched, and a variety of other configuration details.
- 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 UnionFS) in which your application runs.

How Container works

Thinknyx

- When you use the "docker run" CLI command, the Docker Engine client instructs the Docker daemon to run a container.

- This example tells the Docker daemon to run a container using the centos Docker image, to remain in the foreground in interactive mode (-i), and to run the /bin/bash command.

```
docker run -i -t centos /bin/bash
```

```
docker run ubuntu ps ax
```

How Container works

Thinknyx

[root@thinknyx libcontainerd]# docker run -i -t centos /bin/bash
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
d9aaf4d82f24: Pull complete
Digest: sha256:eba772bac22c86d7d6e72421b4700c3f894ab6e35475a34014ff8de74c10872e
Status: Downloaded newer image for centos:latest
[root@9a06b1a61fc5 /]#

[root@thinknyx overlay]# ls -lrt /var/lib/docker/image/overlay/repositories.json
-rw-------. 1 root root 545 Sep 18 03:17 /var/lib/docker/image/overlay/repositories.json

How Container works

Thinknyx

[root@thinknyx overlay]# docker run ubuntu ps ax

[root@thinknyx overlay]# cat /var/lib/docker/image/overlay/repositories.json

[root@thinknyx overlay]# docker image list

[root@thinknyx overlay]# docker image inspect image-name

Hello World - Container

Thinknyx

- In your Docker environment, just run the following command:

  docker run –i –t centos echo “My name is Yogesh Raheja”
  docker run busybox echo “hello world”

- We used one of the smallest, simplest images available: busybox.
- We ran a single process and echo'ed hello world.

More Useful - Container

Thinknyx

- In your Docker environment, just run the following command:

  docker run -it centos

- This is a brand new container.
- It runs a bare-bones, no-frills centos system.
- -i tells Docker to connect us to the container's stdin.
- -t tells Docker that we want a pseudo-terminal.

More Useful - Container

Thinknyx

- Do something in our container:

  Lets suppose we try to use “talk” for communication.

- Let’s check how many packages are installed:

  rpm –qa | wc –l

- Install a package in our container

  yum -y update
  yum install talk

More Useful - Container

Thinknyx

- Exiting our container:

  exit

- Our container is now in a *stopped state.*
- What if we create a same container with same image again???...

Starting Another - Container

Thinknyx

- In your Docker environment, just run the following command:

  docker run -it centos

  Now try to use talk package utilities

- We started a *brand new container.*
- The basic centos image was used, and “talk” is not here.

A non interactive - Container

Thinknyx

- In your Docker environment, just run the following command:

  docker run jpetazzo/clock

- This container just displays the time every second.
- This container will run forever.
- To stop it, press ^C.
- Docker has automatically downloaded the image jpetazzo/clock.

Run in background - Container

Thinknyx

- Containers can be started in the background, with the -d flag (daemon mode):

  docker run -d jpetazzo/clock

- We don't see the output of the container.
- But don't worry: Docker collects that output and logs it!
- docker ps -a
- docker logs <conatiner-id>
- Docker gives us the ID of the container.

List Running Containers

Thinknyx

- With docker ps, just like the UNIX ps command, lists running processes.

  docker ps
  docker ps –l
  docker ps -a

- The (truncated) ID of our container.
- The image used to start the container.
- That our container has been running (Up) for a couple of minutes.
- Now, start multiple containers and use “docker ps” to list them.

List Running Containers - Flags

Thinknyx

- To see only the last container that was started:

  docker ps -l

- To see only the ID of containers:

  docker ps –q

- We can combine the flags to see ID of last container started as well.

Logs of Container

Thinknyx

- Logs of container can be seen using:

  docker logs 068 [[where 068 is prefix of ID]]

- We specified a *prefix of the full container ID.*
- You can, of course, specify the full ID.
- The logs command will output the *entire logs of the container.*
- To avoid being spammed with pages of output, we can use the –tail option:

  docker logs --tail 3 068

Stop our Container

Thinknyx

- There are two ways we can terminate our detached container.
  - Killing it using the docker "kill" command.
  - Stopping it using the docker "stop" command.
- The first one stops the container immediately, by using the KILL signal.
- The second one is more graceful. It sends a TERM signal, and after 10 seconds, if the container has not stopped, it sends KILL.

Stop our Container

Thinknyx

- Let's stop one of our containers:

  docker stop 47d6

- This might take 10 seconds.
- Docker sends the TERM signal.
- If, the container doesn't react to this signal, 10 seconds later, since the container is still running, Docker sends the KILL signal; this terminates the container.

Killing Container

Thinknyx

- Let's kill all our containers:

  docker kill 068 57ad

- The stop and kill commands can take multiple container IDs.
- Those containers will be terminated immediately (without the 10 seconds delay).
- Do check running containers now.

List stopped Container

Thinknyx

- Let's list all of the stopped containers:

```
docker ps –a
```

Removing Container

Thinknyx

- Let's remove our container:

  docker rm <yourContainerID>

Detach Sequence - Container

Thinknyx

- If you have started an *interactive container (with option -it),* you can detach from it.
- The "detach" sequence is ^P^Q.
- You should not detach by hitting ^C, as this would deliver SIGINT to the container.
- You can also change the detach sequence by:

```
docker run -ti --detach-keys ctrl-x,x jpetazzo/clock
```

Detach by hitting ^X x. (This is ctrl-x then x, not ctrl-x twice!)

- Now, do verify that our container is still running.

Attaching to a Container

Thinknyx

- You can attach to a container:

  docker attach <containerID>

- The container must be running.

- There *can be multiple clients attached to the same container.*

- If you don't specify --detach-keys when attaching, it defaults back to ^P^Q.

Restarting a Container

Thinknyx

- When a container has exited, it is in stopped state.
- It can then be restarted with the “start” command.

  docker start <containerID>

- The container must be running.
- There *can be multiple clients attached to the same container.*
- If you don't specify --detach-keys when attaching, it defaults back to ^P^Q.

Debugging a Container

Thinknyx

- You can SSH into a container using “docker exec”:

  docker exec -it <yourContainerId> bash

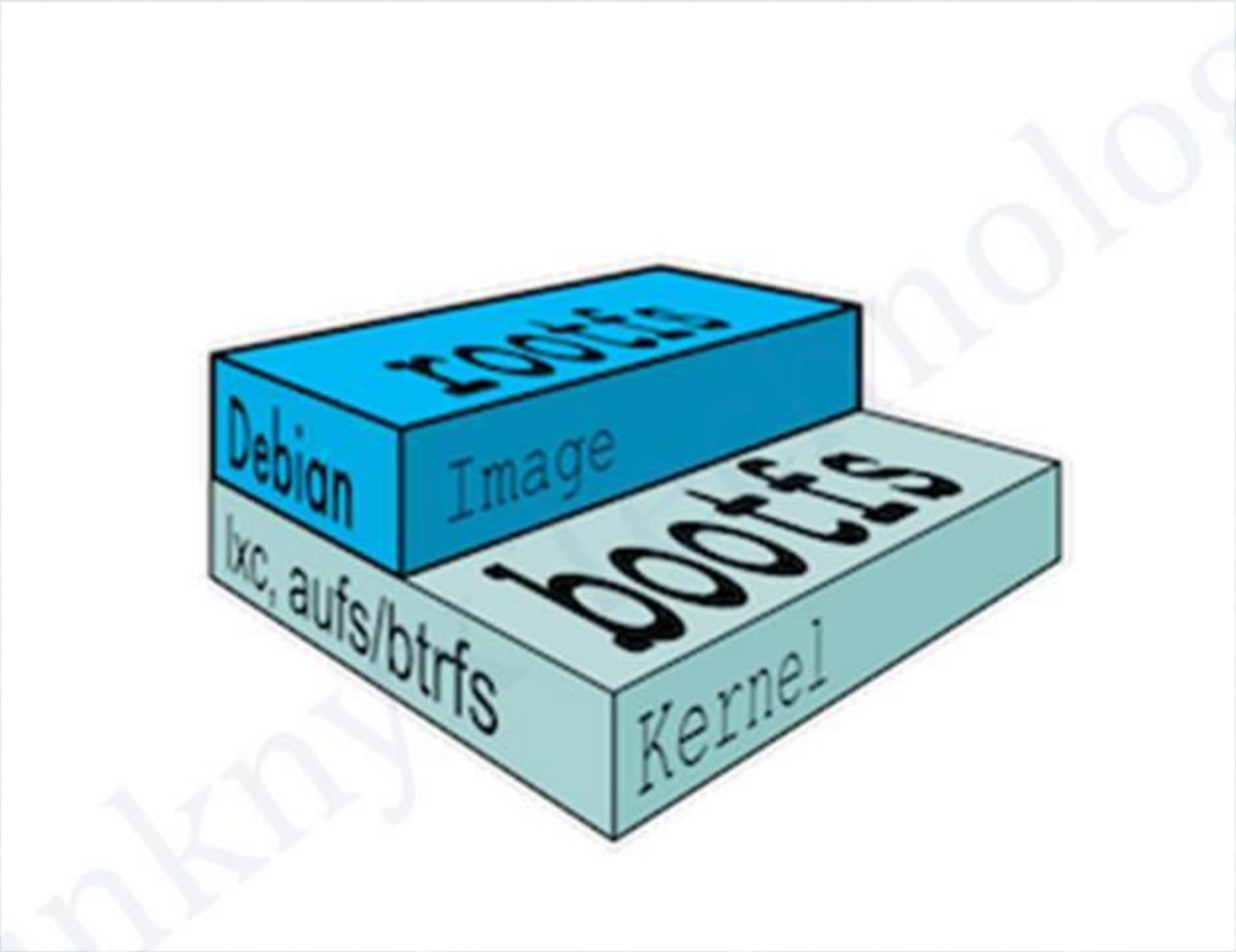
Thinknyx

Session: 5

Docker - Images

Docker Images

Thinknyx



What is an image?

Thinknyx

- An image is a collection of files + some meta data.
- Images are made of *layers, conceptually stacked on top of each other.*
- Each layer can add, change, and remove files.
- Images can share layers to optimize disk usage, transfer times, and memory use.
- Example:
  - CentOS
  - JRE
  - Tomcat
  - Dependencies
  - Application JAR
  - Configuration

Container vs Image

Thinknyx

- An image is a read-only filesystem.
- A container is an encapsulated set of processes running in a read-write copy of that filesystem.
- docker run starts a container from a given image.
- Its “A chicken-and-egg problem”.
- The only way to create an image is by "freezing" a container.
- The only way to create a container is by instantiating an image.

Store & manage images

Thinknyx

- Images can be stored:
  - On your Docker host.
  - In a Docker registry.
- You can use the Docker client to download (pull) or upload (push) images.
- To be more accurate: you can use the Docker client to tell a Docker server to push and pull images to and from a registry.

Showing current images

Thinknyx

- Let's look at what images are on our host now.

  docker images

Searching for images

Thinknyx

- We cannot list all images on a remote registry, but we can search for a specific keyword:

  docker search zookeeper

- "Stars" indicate the popularity of the image.

Downloading images

Thinknyx

- There are two ways to download images.
  - Explicitly, with “docker pull”.
  - Implicitly, when executing “docker run” and the image is not found locally.

- Pulling an image.

  docker pull debian:jessie

- Images can have tags.
- Tags define image versions or variants.
- “docker pull ubuntu” will refer to “ubuntu:latest”.
- The :latest tag is generally updated often.

Docker Image Information

Thinknyx

- Docker Image details:
- docker image list

- Detailed Infromation about docker image:
- ls –lrt /var/lib/docker/image/overlay/imagedb/content/sha256

- High level certificate sign information (SHA):
- cat /var/lib/docker/image/overlay/repositories.json

- All details with command line:
- docker image inspect

Session: 6

Building Images

Building Images Interactively

Thinknyx

- Let's have a Use Case:
  - We will build an image that has httpd.
  - First, we will do it manually with docker commit.
  - Then, we will use a Dockerfile and "docker build".

Create a new container

Thinknyx

- Let's start from base image "centos":

  docker run -it centos
  yum -y update
  yum install httpd
  exit

- Inspect the changes:

  docker diff <yourContainerId>

- Commit the changes:

  docker commit <yourContainerId>

Real way:

docker commit -m "Added HTTPD" -a "Yogesh Raheja" 1bb937745e8e yogeshraheja/centostesting:v1

Run & Tag the image

Thinknyx

- Let's run the new images:

  docker run -it <newImageId>
  rpm –qa | grep –i httpd

- Tagging images:

  docker tag <newImageId> newhttpd

- Run it using Tag:

  docker run -it newhttpd

Dockerfile overview

Thinknyx

- A Dockerfile is a build recipe for a Docker image.
- It contains a series of instructions telling Docker how an image is constructed.
- The “docker build” command builds an image from a Dockerfile.

First Dockerfile

Thinknyx

- Create a directory to hold our Dockerfile.
  mkdir myimage

- Create a Dockerfile inside this directory.

  cd myimage
  vim Dockerfile

- Write below in our Dockerfile

  FROM centos
  RUN yum update -y
  RUN yum install –y httpd

First Dockerfile

Thinknyx

- "FROM" indicates the base image for our build.
- Each "RUN" line will be executed by Docker during the build.
- Our "RUN" commands **must be non-interactive.**
- No input can be provided to Docker during the build.

First Dockerfile

Thinknyx

- Build the Dockerfile:

  docker build -t httpd .

- -t indicates the tag to apply to the image.

- . indicates the location of the Directory of Dockerfile.

Run & Tag the image

Thinknyx

- Let's run the new images:

  docker run -it <newImageId>

  rpm –qa | grep –i httpd

Using Image & viewing history

Thinknyx

- The history command lists all the layers composing an image.
- For each layer, it shows its creation time, size, and creation command.
- When an image was built with a Dockerfile, each layer corresponds to a line of the Dockerfile.

docker history httpd

Using JSON Syntax - Dockerfile

Thinknyx

- Change our Dockerfile:

```
FROM centos
RUN yum update
RUN [“yum", "install", “-y”, “httpd"]
```

- Now, Build & run it.

Uploading images to Docker Hub

- We can share our images through the Docker Hub.
- Below are steps:
  - have an account on the Docker Hub
  - tag our image accordingly (i.e. username/imagename)
  - docker push username/imagename
- Anybody can now docker run username/imagename from any Docker host.

Thinknyx

Session: 8

Deep Dive - Containers

Default Names - Containers

Thinknyx

- When we create a container, if we don't give a specific name, Docker will pick one for us.
- It will be the concatenation of:
  - A mood (furious, goofy, suspicious, boring...)
  - The name of a famous inventor (tesla, darwin, wozniak...)
- Examples: happy_curie, clever_hopper, jovial_lovelace ...

Specify Name - Containers

Thinknyx

- You can set the name of the container when you create it.

  docker run --name ticktock jpetazzo/clock

- If you specify a name that already exists, Docker will refuse to create the container.

- You can rename containers with “docker rename”.

- This allows you to "free up" a name without destroying the associated container.

Inspecting Containers

Thinknyx

- The docker inspect command will output a very detailed JSON map.

  docker inspect <containerID>

- To get specific detail:

  docker inspect --format '{{ json .Created }}' <containerID>

Thinknyx

Session: 9

Container Network Model

Network Model

Thinknyx

- [root@thinknyx yogesh]# docker run -it tomcat

- [root@thinknyx ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

dc6481d7f9b1 tomcat "catalina.sh run" About a minute ago Up About a minute **8080/tcp** vigorous_carson

- [root@thinknyx ~]# docker inspect dc6481d7f9b1

- Go to browser and check the web page

- For more details check below link (also you can use –P or –p to bind specific ports to containers)
- https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/#environment-variables

Container Network Model

Thinknyx

- The CNM was introduced in Engine 1.9.0 (November 2015).
- The CNM adds the notion of a network, and a new top-level command to manipulate and see those networks: “docker network”

```
docker network ls
```

What's in a Network

Thinknyx

- Conceptually, a network is a virtual switch.
- It can be local (to a single Engine) or global (across multiple hosts).
- A network has an IP subnet associated to it.
- A network is managed by a *driver.*
- A network can have a custom IPAM (IP allocator).
- Containers with explicit names are discoverable via DNS.

Creating a Network

Thinknyx

- Let's create a network called dev.
  docker network create dev

- The network can be seen with “network ls” command.

- We will create a named container on this network.
  docker run -d --name search --net dev tomcat

Communication between Containers

- Now, create another container on this network.

  docker run -ti --net dev alpine sh

- From this new container, we can resolve and ping the other one, using its assigned name:

  ping search

Connecting multiple Containers

Thinknyx

- Another Use Case, to connect containers.
- Let's try to run an application that requires two containers.
- The first container is a web server.
- The other one is a redis data store.
- We will place them both on the dev network created before.

Session: 10

Docker - Volumes

Working with Volumes

Thinknyx

- Docker volumes can be used to achieve many things, including:
  - Bypassing the copy-on-write system to obtain native disk I/O performance.
  - Bypassing copy-on-write to leave some files out of docker commit.
  - Sharing a directory between multiple containers.
  - Sharing a directory between the host and a container.
  - Sharing a single file between the host and a container.

Working with Volumes

Thinknyx

- Volumes can be declared in two different ways.
  - Within a Dockerfile, with a VOLUME instruction.

    VOLUME /uploads

  - On the command-line, with the -v flag for “docker run”.

    docker run -d -v /uploads myapp

Sharing Volumes

Thinknyx

- You can start a container with exactly the same volumes as another one.
- The new container will have the same volumes, in the same directories.
- They will contain exactly the same thing, and remain in sync.
- Under the hood, they are actually the same directories on the host anyway.
- This is done using the --volumes-from flag for docker run.

```
docker run -it --name alpha -v /var/log ubuntu bash
docker run --volumes-from alpha ubuntu cat /var/log/now
```

Listing Volumes

Thinknyx

- If a container is stopped, its volumes still exist and are available.
- Since Docker 1.9, we can see all existing volumes:

  docker volume ls

Sharing Directory/File

Thinknyx

- We can specific directory or file or directory from host to container as follows:

  docker run -d -v /path/on/the/host:/path/in/container image

- Example:

  docker run -itd –v /var/lib:/usr/local –p 9090:8080 --name helloWebApp tomcat

Questions & Answers

THANK YOU

For any queries or questions, please contact:

support@thinknyx.com
yogesh.raheja@thinknyx.com
yogeshraheja07@gmail.com