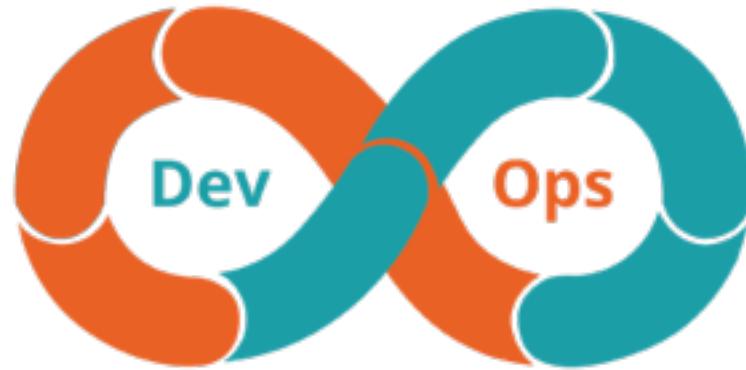


Docker Containerization Boot Camp



Welcome!

Logistics (breaks, facilities, lunch, etc.)

Rules of Engagement

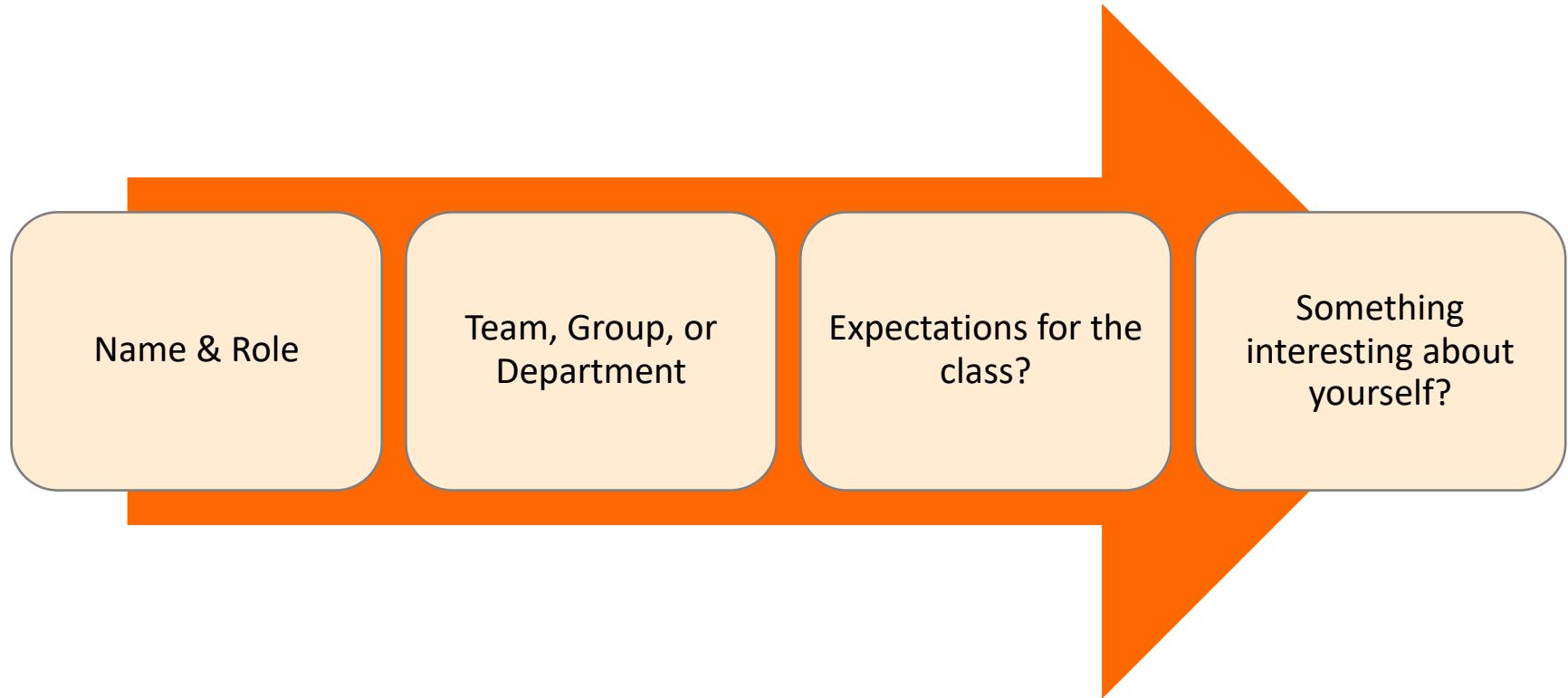
Introductions

Let's Get Started!



Who is your instructor?
A little about me...

Introductions



What to Expect from this Class

Flexibility

Conversations

Literacy and awareness on many of the principles, tools, and practices surrounding Docker as part of a DevOps architecture.

An effort to focus on your own situations and challenges so you can act on what you learn.

Introduction to Docker

Part 1

What is Docker?

The Docker Project

Open Source Project

- 2B+ Docker Image Downloads
- 2000+ contributors
- 40K+ GitHub stars
- 200K+ Dockerized apps
- 240 Meetups in 70 countries
- 95K Meetup members

Docker Inc

Containers as a Service provider

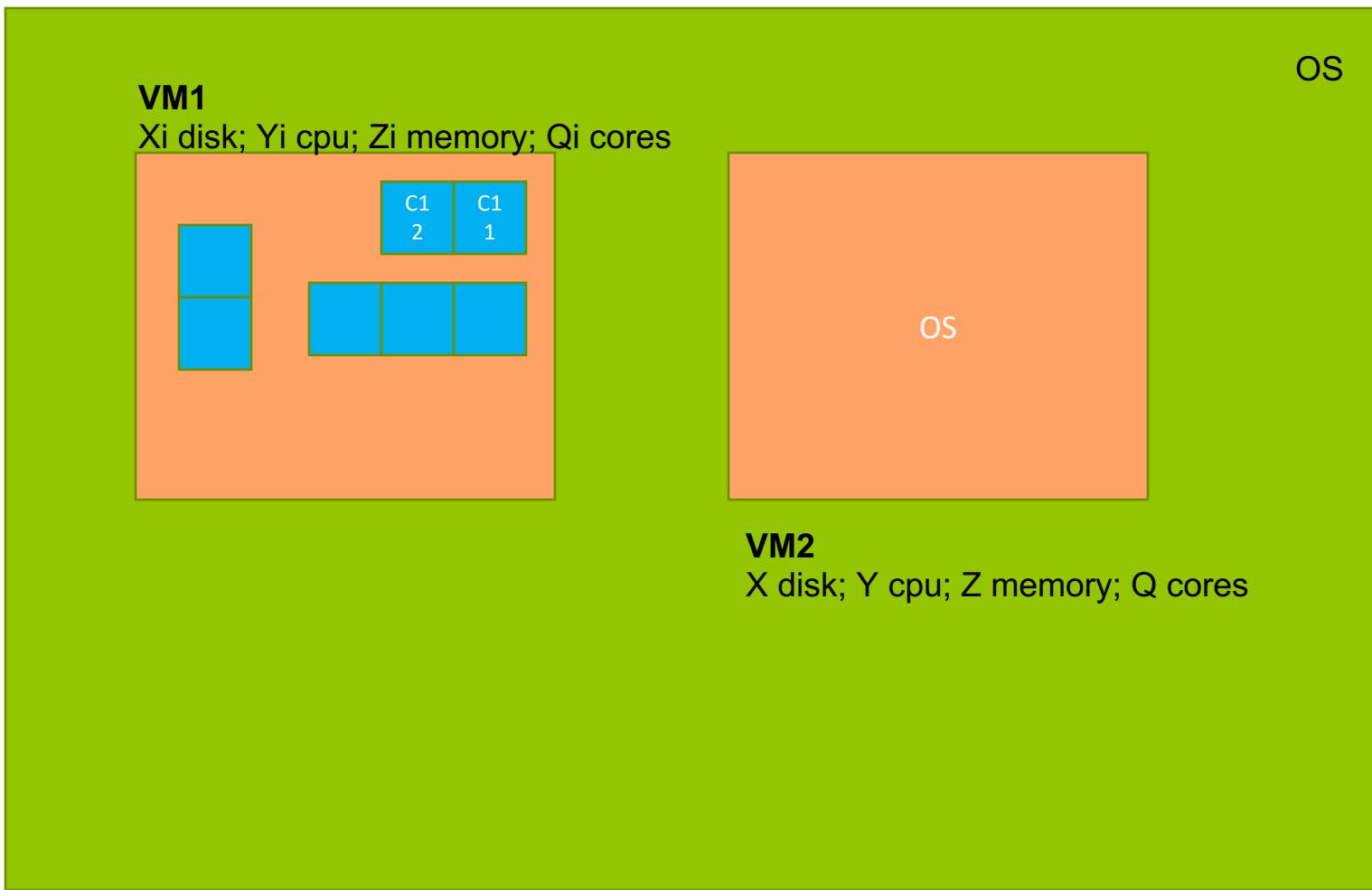
- Integrated platform for dev and IT
- Commercial technical support

Docker project sponsor

- Primary sponsor of Docker project
- Supports project maintainers

RACK SERVER

X disk; Y cpu; Z memory; Q cores



Docker Ecosystem



Docker Basics



Docker Image

- The basis of a Docker container



Docker Container

- The standard unit in which the application service resides



Docker Engine

- Creates, ships and runs Docker containers deployable on physical or virtual host locally, in a datacenter or cloud service provider



Docker Registry

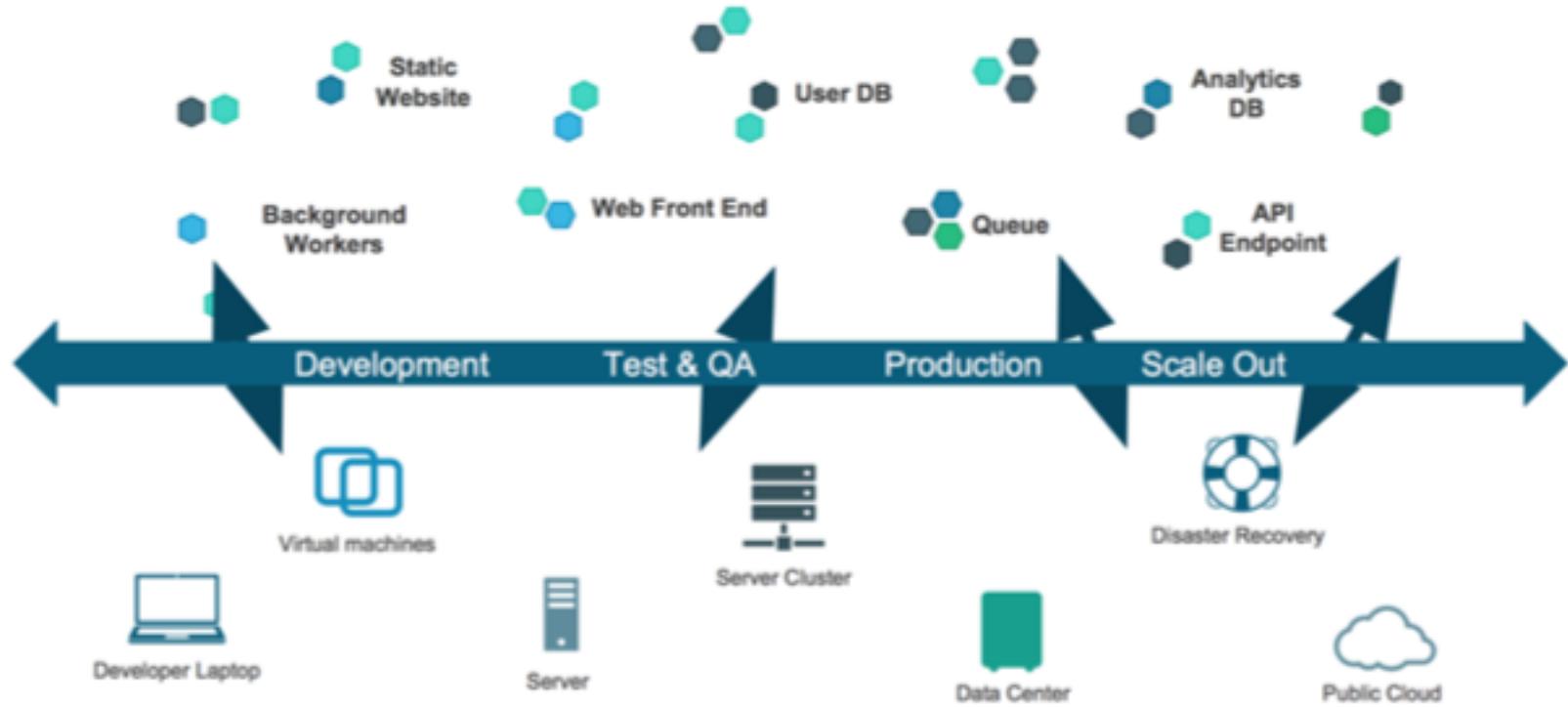
- On-premises registry for image storing and collaboration

Application Evolution



Challenge: The Dependency Matrix

“It works on MY machine.”



Notes

Docker solves the problem of dependency by giving the developer a simple way to express *all* their application's dependencies in one place, while streamlining the process of assembling them.

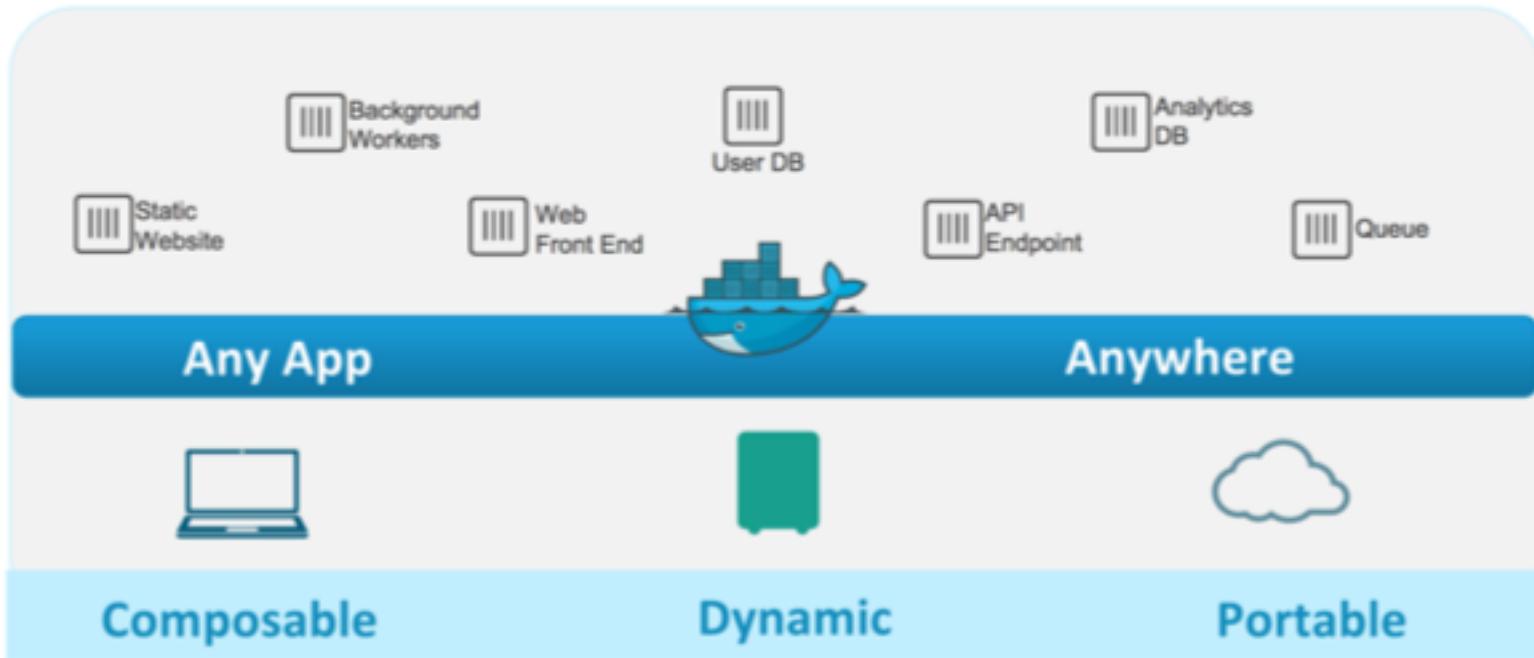
Containers as a Solution



Container

- Packages up software binaries and dependencies
- Isolates software from each other
- Container is a standard format
- Easily portable across environment
- Allows ecosystem to develop around its standard

Containers as a Solution



Developer Benefits

Build once...finally) run anywhere

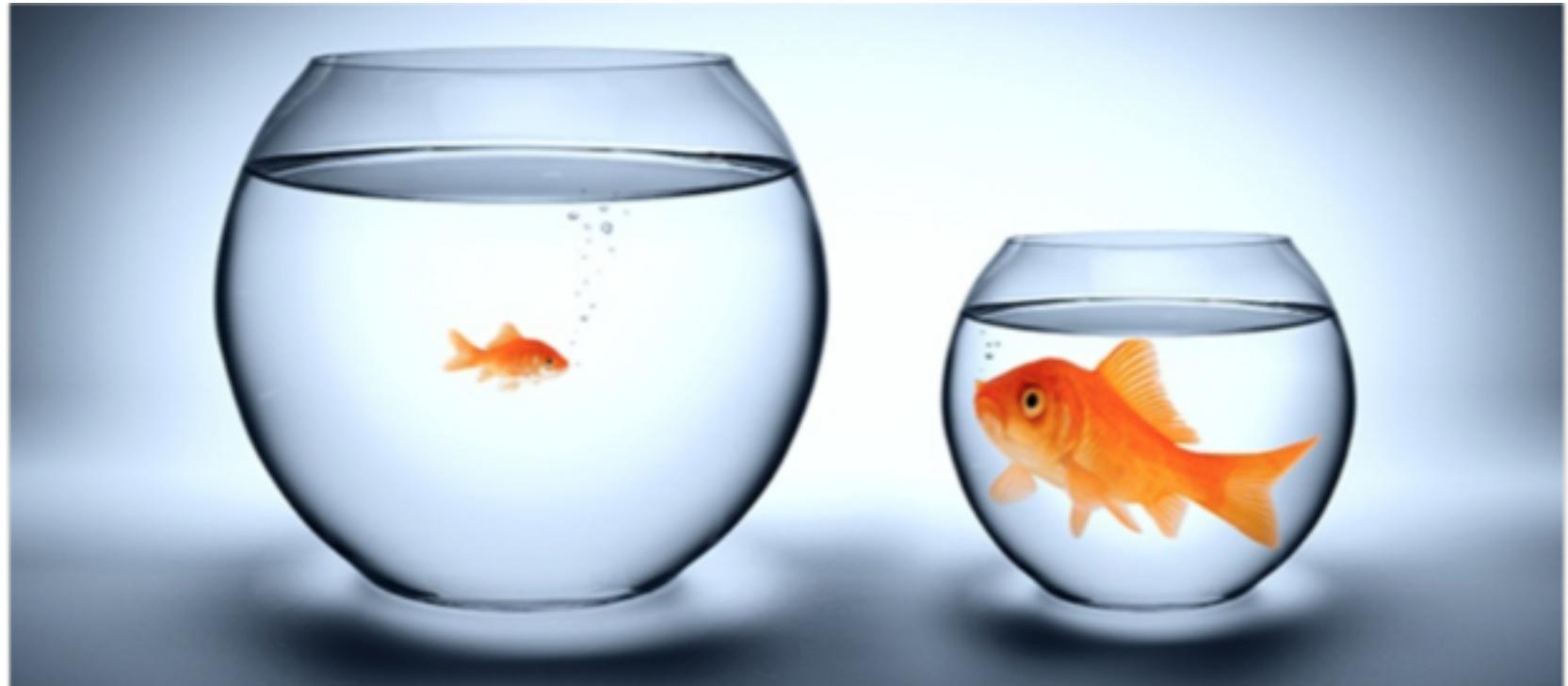
- A clean, safe, hygienic and portable runtime environment for your app.
- No worries about missing dependencies, packages and other pain points during subsequent deployments.
- Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
- Automate testing, integration, packaging...anything you can script
- Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.
- Cheap, zero-penalty containers to deploy services? A VM without the overhead of a VM? Instant replay and reset of image snapshots? That's the power of Docker

Operational Benefits

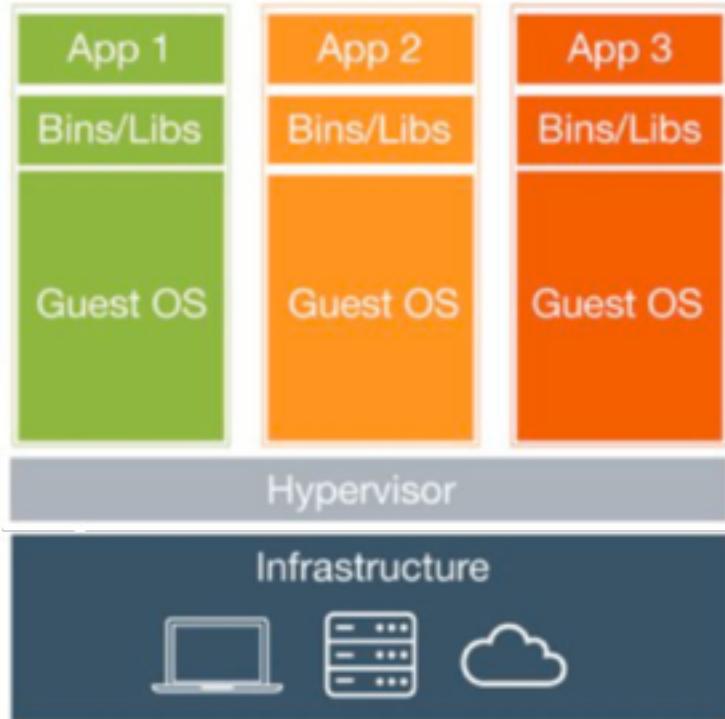
Configure once...run anything

- Make the entire lifecycle more efficient, consistent, and repeatable
- Increase the quality of code produced by developers.
- Eliminate inconsistencies between development, test, production, and customer environments
- Support segregation of duties
- Significantly improves the speed and reliability of continuous deployment and continuous integration systems
- Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VM

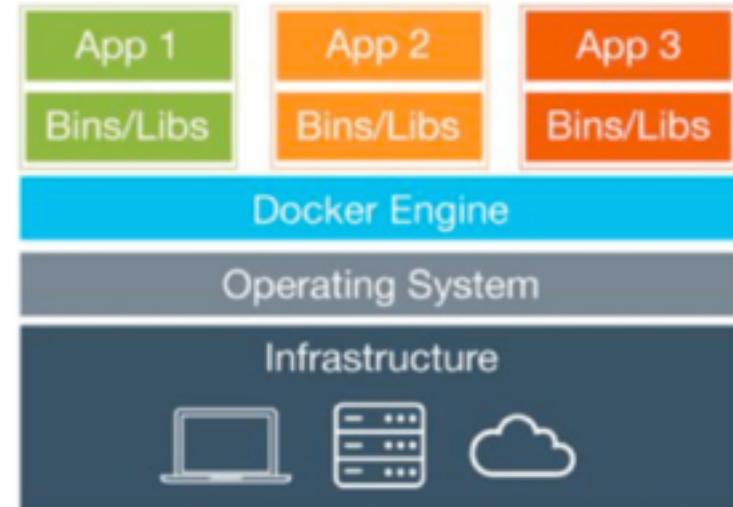
VMs vs. Containers



VMs vs. Containers



Virtual Machines



Containers

VMs vs. Containers

Virtual Machines (VMs)	Containers
Represents hardware-level virtualization	Represents operating system virtualization
Heavyweight	Lightweight
Slow provisioning	Real-time provisioning and scalability
Limited performance	Native performance
Fully isolated and therefore more secure (maybe)	Process-level isolation and therefore less secure (maybe)

VMs vs. Containers

Simulates a physical machine

Provides a local file system

Can be accessed over a network

Full and independent guest operating system

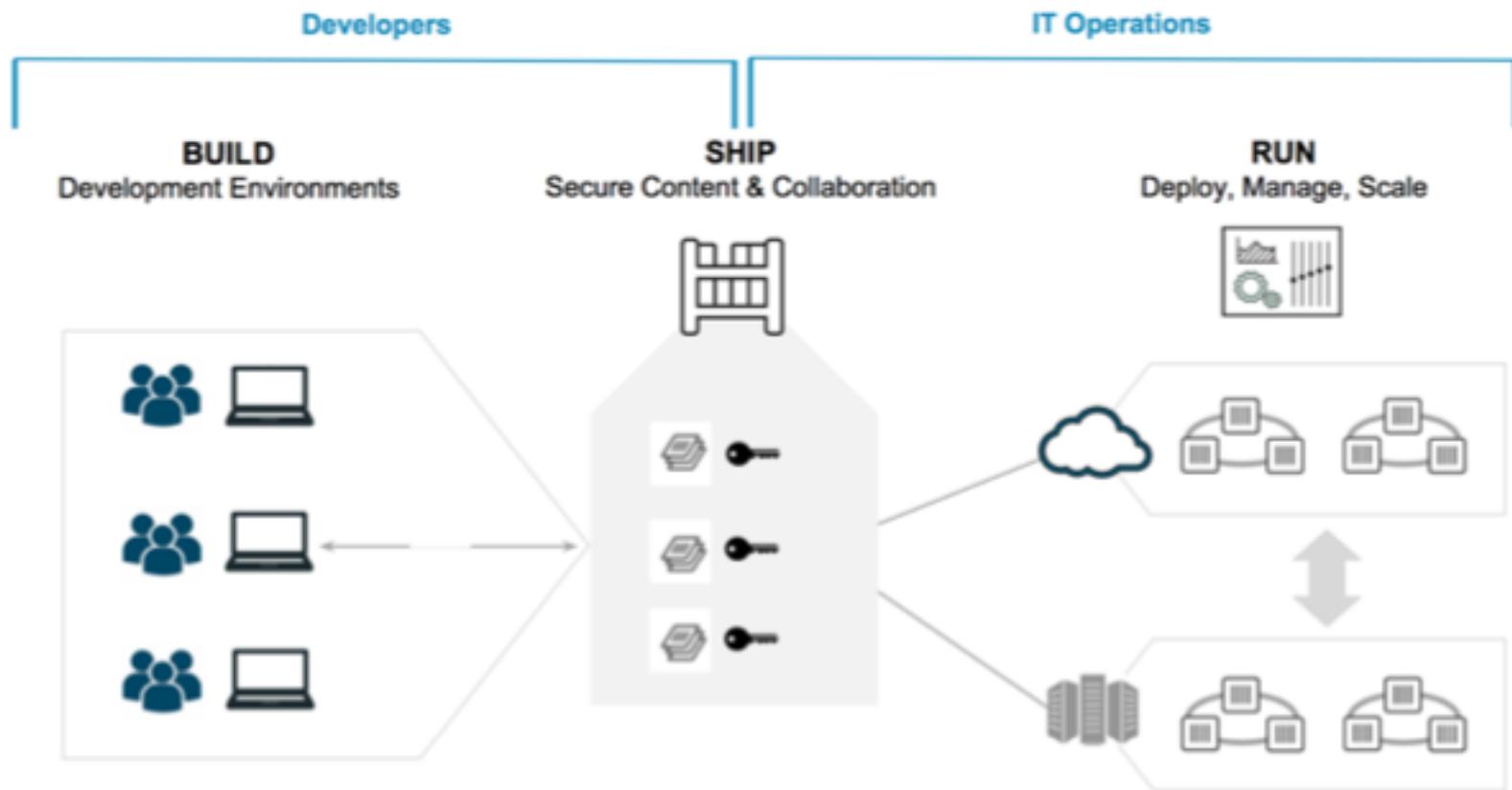
Virtualized device drivers

Strong resource and memory management

Huge memory foot-print

Needs a hypervisor

Dev/Ops Container Workflow



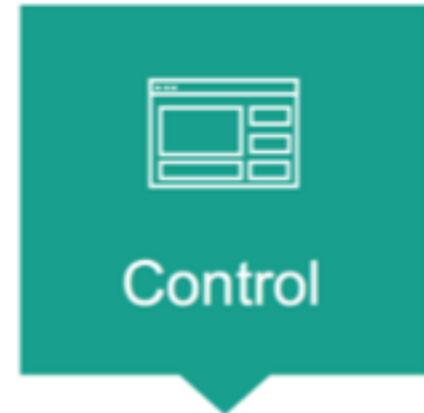
Container Benefits



Innovation
at speed



Frictionless
movement



Manage and
secure at scale

Docker CE vs. Docker EE

	COMMUNITY EDITION	ENTERPRISE EDITION BASIC	ENTERPRISE EDITION STANDARD	ENTERPRISE EDITION ADVANCED
Container engine and built in orchestration, networking, security	✓	✓	✓	✓
Docker Certified - Infrastructure, Plugins and ISV Containers		✓	✓	✓
Image Management (private registry, caching)	Cloud hosted repos		✓	✓
Docker Datacenter - Integrated container app management			✓	✓
Docker Datacenter - Multi-tenancy with RBAC, LDAP/AD support			✓	✓
Integrated secrets mgmt, image signing policy			✓	✓
Image security scanning	Preview			✓
Support	Community	Biz Day or 24x7	Biz Day or 24x7	Biz Day or 24x7
Pricing	Free	Few hundred per node per year	Few thousand per node per year	Few thousand per node per year

Installing Docker

- **Docker container engine is built from and on top of the Linux kernel, so it needs Linux to run natively***
 - Docker is increasingly being packaged with newer Linux distributions
 - Docker is also available on MacOS and Windows through the use of lightweight Linux VMs - HyperV-based on Windows, HyperKit-based on MacOS
- * **Native Windows containers can also be used with Docker for Windows if you have very specific versions of Windows.**

Prerequisite: Connect to Classroom VM



CLASSROOM WORK

- § \$ ssh student@FQDN
 - § student password = DockerStudentPW0

FQDN = The fully qualified domain name of the machine assigned to you by your instructor. This will commonly be in the form docker#.domain.com (where # is your student number as assigned to you by your instructor).

Prerequisite: Add & Update Repos



CLASSROOM WORK

```
$ sudo apt-get update -y
```

Install necessary packages

```
$ sudo apt-get -y install apt-transport-https ca-certificates curl
```

Add Docker official GPG key

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Setup the stable repository

```
$ sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
```

```
$ sudo apt-get update -y
```

Also at

<https://docs.docker.com/engine/installation/>

Install Docker-CE



CLASSROOM WORK

Install Docker

```
$ sudo apt-get -y install docker-ce
```

Test Docker

```
$ sudo docker run hello-world
```

Also at

<https://docs.docker.com/engine/installation/>

Docker Hello World(s)

```
vagrant@vagrant-ubuntu-trusty-64:~$ docker run -it hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world

c04b14da8d14: Pull complete
Digest: sha256:8256e8a36e2070f7bf2d0b0763dbabdd67798512411de4cdcf9431a1feb60fd9
Status: Downloaded newer image for hello-world:latest

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 Hub account:
https://hub.docker.com

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
vagrant@vagrant-ubuntu-trusty-64:~$
```

Installing Docker Script

It's good to install Docker the “hard” way at least once so you understand what is involved.

- There is a Docker-maintained community script that will install the latest release on your Linux machine (most popular flavors) available at <https://get.docker.com/>
- Run ‘curl -sSL https://get.docker.com/ | sh’ in your terminal
- The script will also install a Union File System driver and verify Docker engine functionality

Docker Basics

\$ sudo docker
version

```
cludwig — student@docker-proto: ~ — ssh student@...
student@docker-proto:~$ sudo docker version
Client:
  Version:      17.03.1-ce
  API version:  1.27
  Go version:   go1.7.5
  Git commit:   c6d412e
  Built:        Mon Mar 27 17:14:09 2017
  OS/Arch:      linux/amd64

Server:
  Version:      17.03.1-ce
  API version:  1.27 (minimum version 1.12)
  Go version:   go1.7.5
  Git commit:   c6d412e
  Built:        Mon Mar 27 17:14:09 2017
  OS/Arch:      linux/amd64
  Experimental: false
student@docker-proto:~$
```

Docker Basics

\$ sudo
docker info

```
cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 95x29
student@docker-proto:~$ sudo docker info
Containers: 1
Running: 0
Paused: 0
Stopped: 1
Images: 1
Server Version: 17.03.1-ce
Storage Driver: aufs
Root Dir: /var/lib/docker/aufs
Backing Filesystem: extfs
Dirs: 3
Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
Network: bridge host macvlan null overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 4ab9917febca54791c5f071a9d1f404867857fcc
runc version: 54296cf40ad8143b62dbcaa1d90e520a2136ddfe
init version: 949e6fa
Security Options:
apparmor
seccomp
Profile: default
Kernel Version: 4.4.0-78-generic
```

Docker Basics

Determine the status of the Docker service

- **\$ sudo service docker status**

(Hit “q” to come out)

```
[ubuntu@ubuntu-xenial:~/docker]$ sudo service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2016-11-29 14:12:54 UTC; 2h 47min ago
     Docs: https://docs.docker.com
 Main PID: 1284 (dockerd)
    Tasks: 22
   Memory: 82.9M
      CPU: 1min 31.242s
     CGroup: /system.slice/docker.service
             └─1284 /usr/bin/dockerd -H fd://
                  ├─1212 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --shim docker-containerd-shim --metrics-interval=0 --start-ti
                  └─1212 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --shim docker-containerd-shim --metrics-interval=0 --start-ti

Nov 29 16:29:33 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:29:33.147992827Z" level=info msg="Layer sha256:6fc2e4ad81348c14fd2d7e3880b5db2bbc2f699a5cdcf18b
Nov 29 16:29:36 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:29:36.424426912Z" level=info msg="Layer sha256:a21398f45083710727f6f382cf232d5d58d4dc9589d6ff5
Nov 29 16:30:09 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:30:09.788820388Z" level=info msg="Layer sha256:9359188bd45e8c8b98fc0f0297ba87b4f5ed64e9dc8fba7
Nov 29 16:30:09 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:30:09.913582571Z" level=info msg="Layer sha256:9359188bd45e8c8b98fc0f0297ba87b4f5ed64e9dc8fba7
Nov 29 16:30:28 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:30:28.557526233Z" level=info msg="Layer sha256:be3b076c597ddebb8e264d732927a160c2e91c78516c84a1
Nov 29 16:30:50 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:30:50.018714100Z" level=info msg="Layer sha256:47e85df6c4115d2974ab80ef823c215917dae0a9011f4262
Nov 29 16:30:56 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:30:56.628202459Z" level=info msg="Layer sha256:ce86475a45c5830b905647755ce0324b8c313582bf66ea0c
Nov 29 16:31:14 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:31:14.829527550Z" level=info msg="Layer sha256:2b9628b0eba7b5a870fd354986be4d15e605d51c8f9f7f2a
Nov 29 16:31:15 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:31:15.102270791Z" level=info msg="Layer sha256:c0f15147c2663dc3d68f177fffc0034bb2c04e25eddc6007
Nov 29 16:32:33 ubuntu-xenial dockerd[1284]: time="2016-11-29T16:32:33.001824138Z" level=info msg="Layer sha256:72deb7ae364e53981d0d1befdf7d9c5b15208f93cd8e169b
[Lines 1-22/22 (END)]
```

No 'sudo' Configuration

Run docker without sudo

- <https://docs.docker.com/engine/installation/linux/linux-postinstall/>

1. Create Docker group:

- `$ sudo groupadd docker`

2. Add your user to docker group:

- `$ sudo usermod -aG docker $USER`

3. Log out and log back in:

- `$ exit`

then log back in

Get to know Docker APIs



CLASSROOM WORK

```
$ docker --help  
$ docker run --help  
$ docker image --help  
$ docker container --help
```

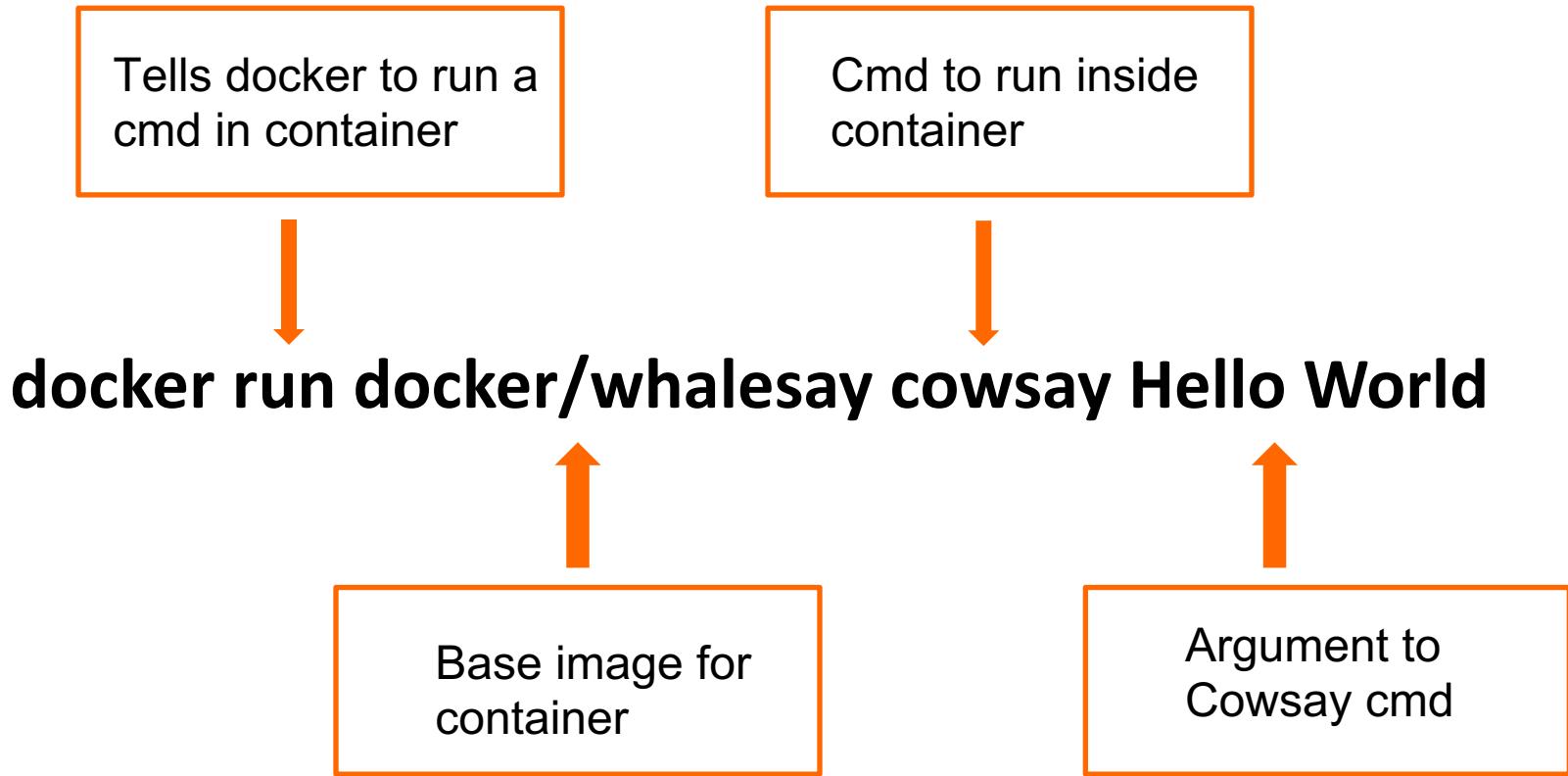
Docker Basics

- Standard command format:
 - docker [NOUN] verb
- \$ sudo docker --help #displays help and management subcommands (nouns)
- \$ sudo docker NOUN --help
- There are reasonable defaults and short forms of option flags for many things. i.e.:
 - “docker pull” assumes “image”
 - “docker stop” assumes “container”
 - can use “img” instead of “image”
 - can use “con” instead of “container”, etc.

Docker Hello World(s) cowsay

```
$ sudo docker run docker/whalesay cowsay Hello World
```

Docker Hello World(s)



What is 'Docker Engine'?



Using Docker using Python

This is not covered in the course, but always good to know your options

```
#Install docker package for python
```

```
!pip install docker
```

Run a container

```
import docker
client = docker.from_env()
client.containers.run("docker/whalesay", ["cowsay", "Hello", "World"])
print container.id
```

List all containers

```
import docker
client = docker.from_env()
for container in client.containers.list():
    print container.id
```

Using Docker using Java

This is not covered in the course, but always good to know your options

Add Maven dependency

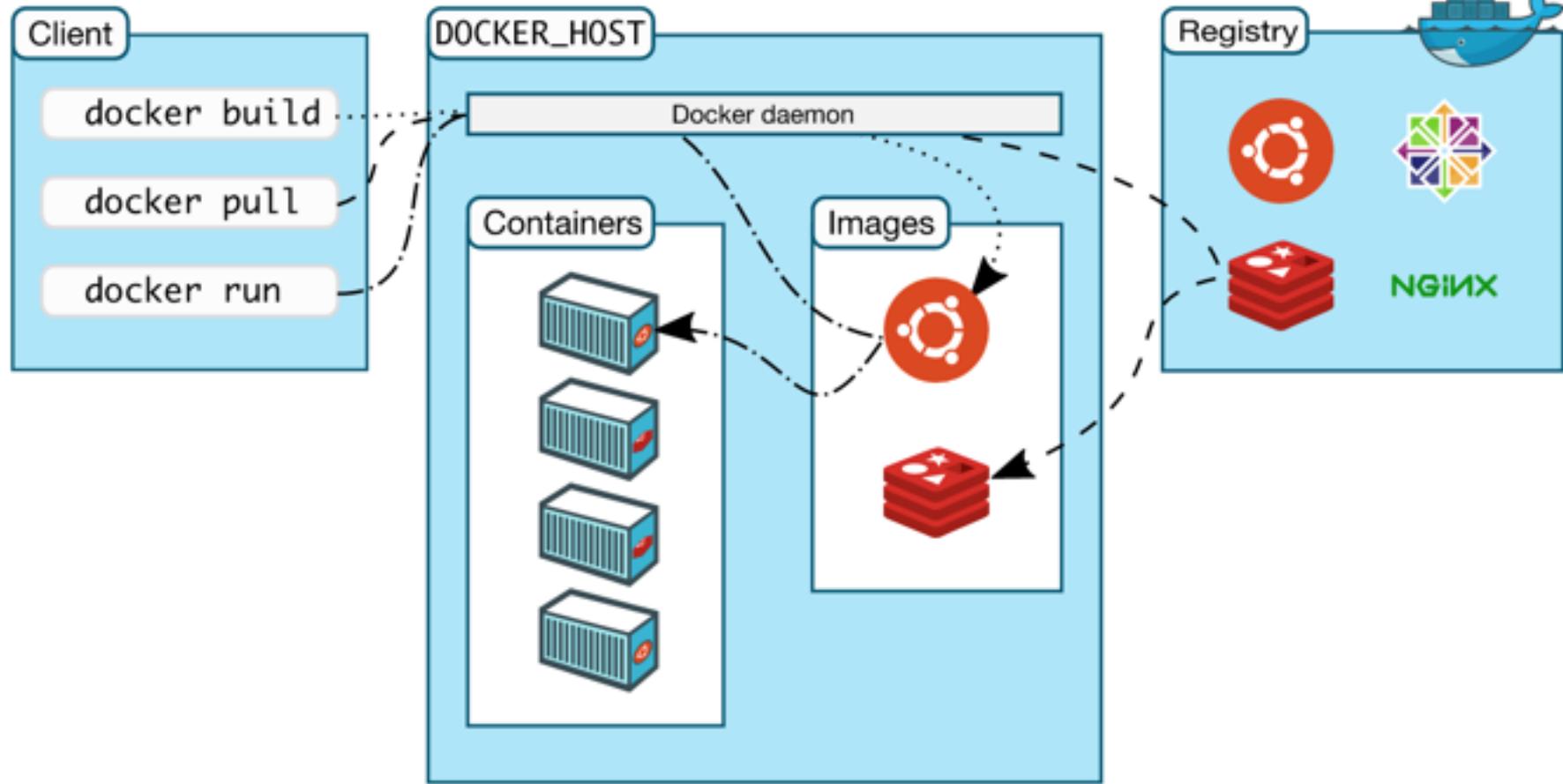
```
<dependency>
  <groupId>com.github.docker-java</groupId>
  <artifactId>docker-java</artifactId>
  <version>3.0.14</version>
</dependency>
```

```
## DockerClientBuilder takes all parameters
DockerClient dockerClient = DockerClientBuilder.getInstance().build();

## List all containers
List<Container> containers = dockerClient.listContainersCmd().exec();

## Create containers
CreateContainerResponse container
= dockerClient.createContainerCmd("docker/whalesay")
  .withCmd("").exec();
```

How is Docker Architected?



Docker Webapp & Hello World



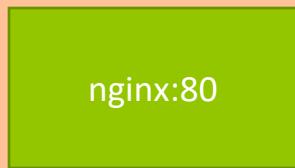
CLASSROOM WORK

- **Exercise 2.1 in Docker Labs**

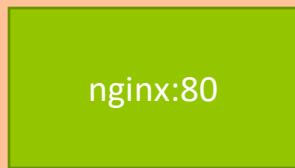
[https://github.com/docker/labs/blob/master
/beginner/chapters/webapps.md](https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md)

**** Run only 2.1 Run a static website in a container**

Ubuntu VM, host machine



82

8081
8080

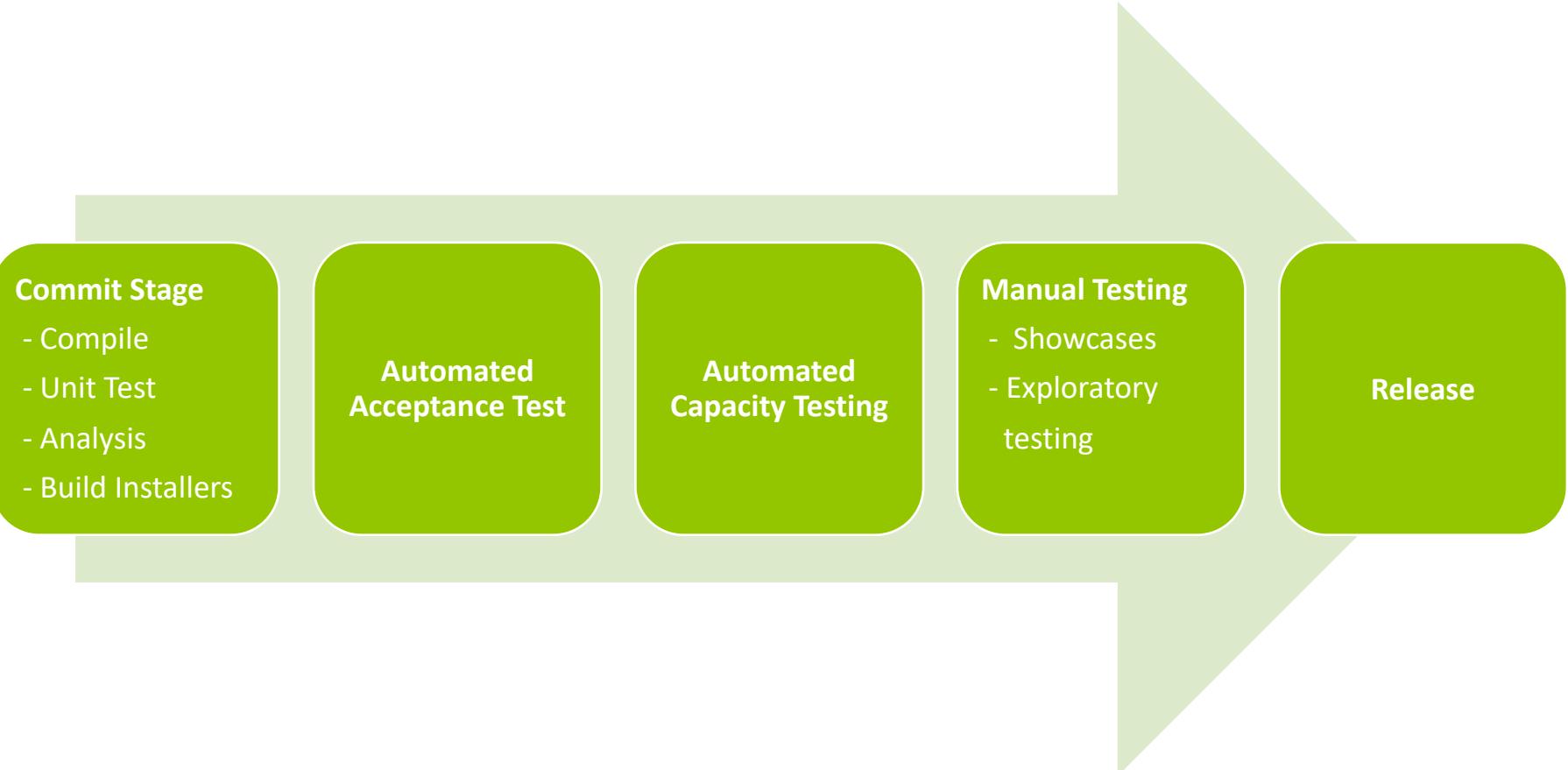
82

So far !!

- Basic difference between VMs and Containers
- Install Docker
- Run containers:-
 - Hello World
 - whalesay
 - static-website

Where Containers are used?

- For automated testing & CICD



Where Containers are used?

- **DevOps (Mostly in all types of Automation)**
- **Automated testing (Unit, acceptance and stress testing)**
- **CICD (if not tested on production-like environment, CICD will be risky)**
- **Microservices**
- **Less risky deployment architectures**
- **Chef kitchen test**
- **Not only used for testing but is used in production (along with container orchestration) in B2C and B2B organizations**

Advantage of using Containers

- Developers testing their code in Prod environments
- Lesser hand-offs and drastically lesser waiting time (to get new environment setup)
 - Reduce technical debt
- Overall system resilience - Service failure does not kill the application & may be invisible to users
- Scalability
 - Seamless replication
- High Availability
- Less risky patterns for rolling updates available
- Prevent server sprawl and Jenga infrastructure

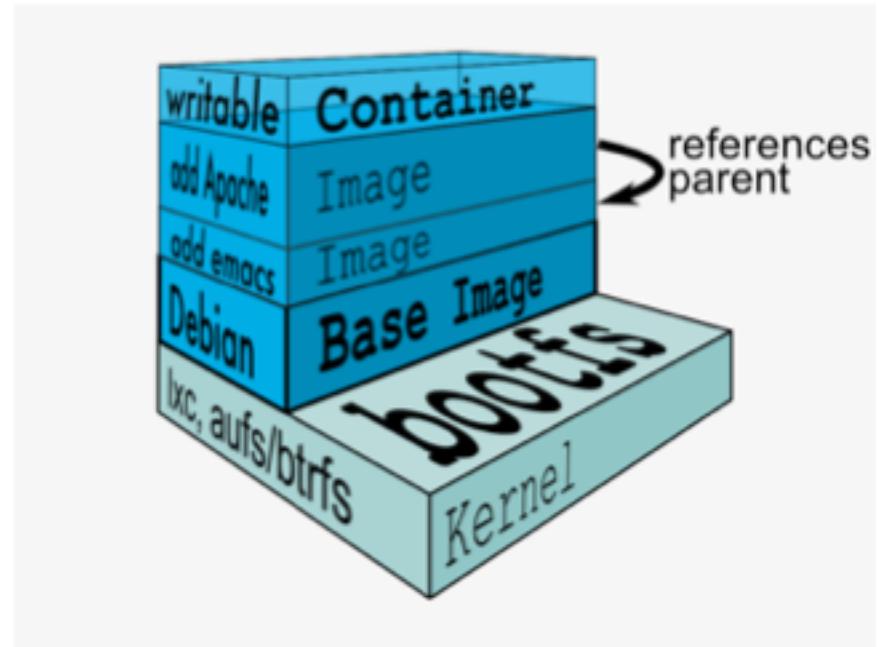
Docker Images

Part 2

Docker Union File System

AuFS (AnotherUnionFS) is a multi-layered filesystem that implements union mount

- Allows several filesystems or directories to be simultaneously mounted and visible through a single mount point
- Appears to be one filesystem to the end user



Example Docker Layers



Docker Layer – Each Docker image is composed of a series of layers. Docker uses Union File Systems to combine these into a single image. The combination of these layers gives the illusion of a traditional file system. Only the top layer is writeable.

Example Docker Layers

6

Image 1 – web app 1

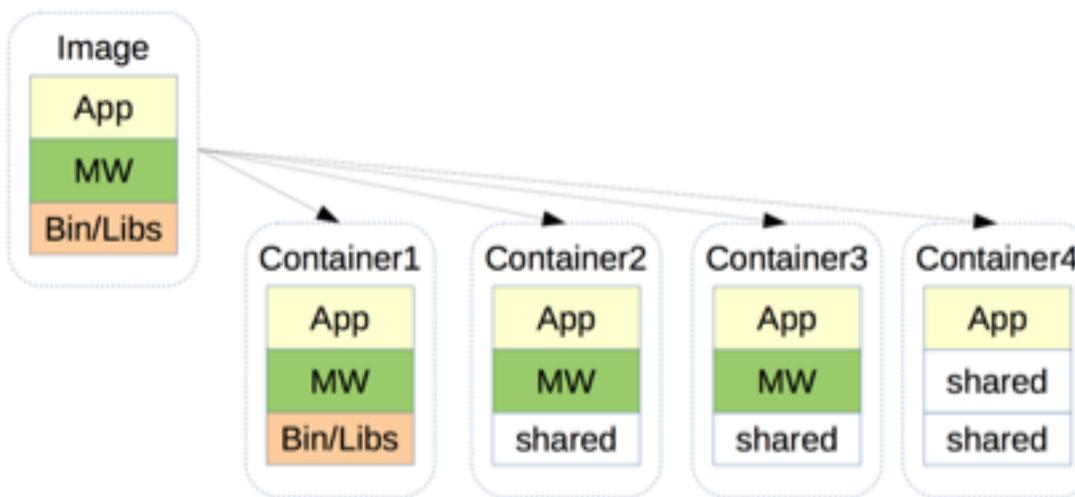


Image 2 – web app 2

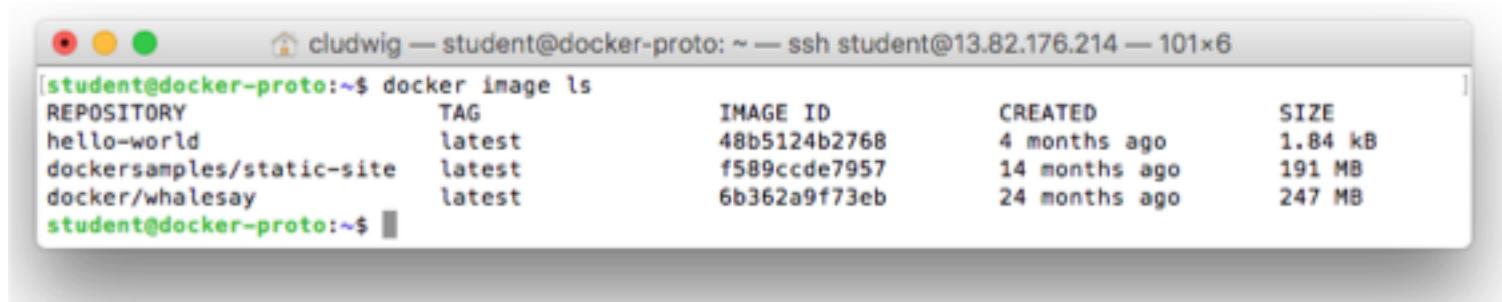


Benefits of Union File Systems

- Files are shared across containers
- Storage and memory spaces are saved
- Faster deployment of containers



\$ docker image ls



A screenshot of a terminal window titled "cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 101x6". The window contains the following text:

```
[student@docker-proto:~$ docker image ls
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
hello-world          latest     48b5124b2768  4 months ago  1.84 kB
dockersamples/static-site  latest   f589ccde7957  14 months ago  191 MB
docker/whalesay      latest     6b362a9f73eb  24 months ago  247 MB
student@docker-proto:~$ ]
```

Docker Hub

Docker's first-party cloud-based registry

- Hosts a broad repository of Docker images
- Contains nearly any popular open source technology including MariaDB, Jenkins, Cloudera, etc
- Contains 'Official images' from vendors such as Canonical, Oracle, Red Hat, etc
- Provides one free private Docker repo, more for a paid subscription

Docker Hub

A screenshot of a web browser showing the Docker Hub search results for 'mariadb'. The search bar at the top contains 'mariadb'. Below the search bar, there are 1096 repositories listed. The first four repositories are displayed in a grid:

Repository	Owner	Status	Stars	Pulls	Actions
mariadb	official	public automated build	935	5M+	> DETAILS
bitnami/mariadb	bitnami	public automated build	21	1M+	> DETAILS
million12/mariadb	million12	public automated build	9	10K+	> DETAILS
maxexcelo/mariadb	maxexcelo	public automated build	4	1.4K	> DETAILS

\$ Few Useful Commands

\$ docker search [searchterm]

```
cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 119x28
student@docker-proto:~$ docker search mariadb
NAME                           DESCRIPTION                                              STARS   OFFICIAL   AUTOMATED
mariadb                         MariaDB is a community-developed fork of M...    1346    [OK]
bitnami/mariadb                 Bitnami MariaDB Docker Image                   34      [OK]
paintedfox/mariadb              A docker image for running MariaDB 5.5, a ...    29      [OK]
million12/mariadb               MariaDB 10 on CentOS-7 with UTF8 defaults       14      [OK]
toughiq/mariadb-cluster          Dockerized Automated MariaDB Galera Cluste...    11      [OK]
webhippie/mariadb               Docker images for mariadb                      9       [OK]
panubo/mariadb-galera           MariaDB Galera Cluster                      7       [OK]
gists/mariadb                   MariaDB on Alpine                        7       [OK]
kakilangit/mariadb              Docker for MariaDB with OQGraph & TokuDB E...    6       [OK]
maxexcloo/mariadb              Service container with MariaDB installed a...    4       [OK]
tianon/mariadb                  DEPRECATED; use mariadb:* -- : "I just met..."    4       [OK]
takaomag/mariadb                docker image of archlinux (mariadb)            2       [OK]
desertbit/mariadb               This is an extended docker image of the of...    1       [OK]
drupaldocker/mariadb             MariaDB for Drupal                      1       [OK]
tcaxias/mariadb                 MariaDB container                      1       [OK]
jpc0/mariadb                    Mariadb, so I can have it on my raspberry        1       [OK]
lucidfrontier45/mariadb          Mariadb with some customizable properties       0       [OK]
vger/mariadb                     MariaDB image, based on Debian Jessie            0       [OK]
yannickvh/mariadb               Custom build of MariaDB based on the offic...    0       [OK]
dogstudio/mariadb               MariaDB Container for Dogs                  0       [OK]
objectstyle/mariadb              ObjectStyle MariaDB Docker Image                 0       [OK]
nimmis/mariadb                  MariaDB multiple versions based on nimmis/...    0       [OK]
rkrahlf/mariadb                 A docker image for MariaDB                      0       [OK]
mmckeen/mariadb                 MariaDB image based on openSUSE Tumbleweed       0       [OK]
danielsreichenbach/mariadb     Minimal MariaDB container to be used as co...    0       [OK]
student@docker-proto:~$
```

\$ docker image pull [imageName]

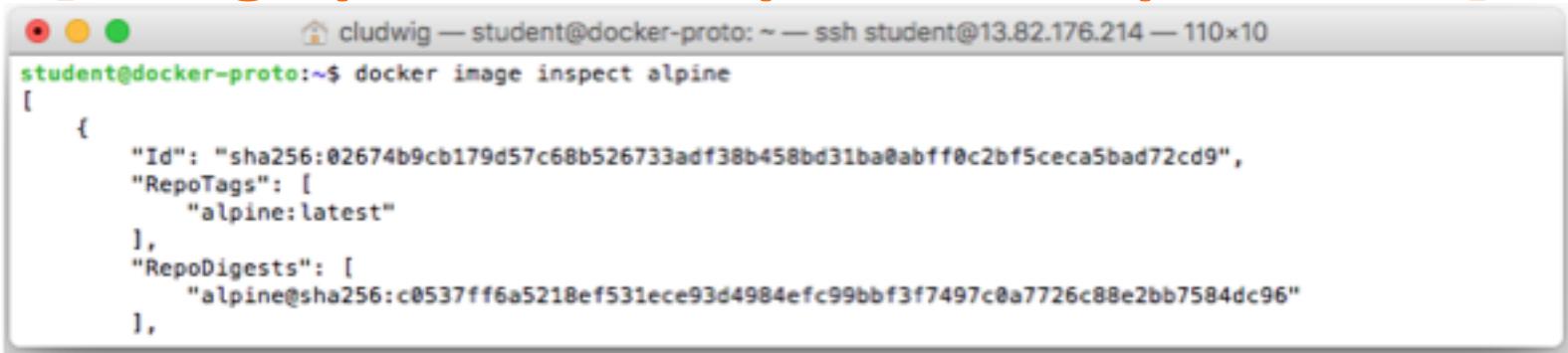
```
cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 119x6
student@docker-proto:~$ docker image pull alpine
Using default tag: latest
latest: Pulling from library/alpine
cfc728c1c558: Pull complete
Digest: sha256:c0537ff6a5218ef531ece93d4984efc99bbf3f7497c0a7726c88e2bb7584dc96
Status: Downloaded newer image for alpine:latest
```

Best Practice: Choose the smallest base images possible

Debian – 123 MB
Alpine – 5 MB

```
cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 119x7
student@docker-proto:~$ docker image list
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
alpine              latest   02674b9cb179  11 days ago  3.99 MB
hello-world         latest   48b5124b2768  4 months ago  1.84 kB
dockersamples/static-site  latest   f589ccde7957  14 months ago  191 MB
docker/whalesay     latest   6b362a9f73eb  24 months ago  247 MB
student@docker-proto:~$
```

\$ docker inspect [image|container|network|volume]



```
cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 110x10
student@docker-proto:~$ docker image inspect alpine
[
  {
    "Id": "sha256:02674b9cb179d57c68b526733adf38b458bd31ba@abff0c2bf5ceca5bad72cd9",
    "RepoTags": [
      "alpine:latest"
    ],
    "RepoDigests": [
      "alpine@sha256:c0537ff6a5218ef531ece93d4984efc99bbf3f7497c0a7726c88e2bb7584dc96"
    ],
  }
]
```

- **ID** The unique identifier for the container
- **State** This stanza has various status flags and the process id for the container. Using the `ExitCode` from this `State` element a [graceful shutdown](#) or recovery process could be initiated. The following format will return just the `ExitCode` of the most recently run container:

```
docker inspect -f '{{.State.ExitCode}}' $(docker ps -lq)
```
- **Image** The image this container is running.
- **NetworkSettings** The network environment for the container and therefore for the application(s) within the image.
- **LogPath** The system path to this container's log file.
- **RestartCount** Keeps track of the number of times the container has been restarted. This value is the key value used when defining a container's [restart policy](#)..
- **Name** The user defined name for the container.
- **Volumes** Defines the volume mapping between the host system and the container.
- **HostConfig** Key configurations for how the container will interact with the host system. These could take CPU and memory limits, networking values, or device driver paths.
- **Config** The runtime configuration options set when the docker run command was executed. Part of this configuration is another "Image" value. This image `{{.Config.Image}}` is the tagged image which may be different than the image listed in `{{.Image}}`

\$ docker image history docker/whalesay

```
cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 138x21
student@docker-proto:~$ docker image history docker/whalesay
IMAGE          CREATED      CREATED BY
6b362a9f73eb  24 months ago  /bin/sh -c #(nop) ENV PATH=/usr/local/bin:...
<missing>       24 months ago  /bin/sh -c sh install.sh
<missing>       24 months ago  /bin/sh -c git reset --hard origin/master
<missing>       24 months ago  /bin/sh -c #(nop) WORKDIR /cowsay
<missing>       24 months ago  /bin/sh -c git clone https://github.com/mo...
<missing>       24 months ago  /bin/sh -c apt-get -y update && apt-get in...
<missing>       2 years ago   /bin/sh -c #(nop) CMD ["/bin/bash"]
<missing>       2 years ago   /bin/sh -c sed -i 's/^#\!/\n' /etc/alternat...
<missing>       2 years ago   /bin/sh -c echo '#!/bin/sh' > /usr/sbin/po...
<missing>       2 years ago   /bin/sh -c #(nop) ADD file:f4d7b4b3402b5c5...
student@docker-proto:~$
```

Use history to view layers in an image

Docker Interactive Mode

Best Practice: Run containers in interactive mode while developing docker images

```
[ubuntu@instance-41:~/flask-app$ sudo docker run -t -i ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
6bbebedd9b76a4: Pull complete
fc19d60a83f1: Pull complete
de413bb911fd: Pull complete
2879a7ad3144: Pull complete
668604fde02e: Pull complete
Digest: sha256:2d44ae143feeb36f4c898d32ed2ab2dffeb3a573d2d8928646dfc9cb7deb1315
Status: Downloaded newer image for ubuntu:latest
root@a4b1111d9d0e:/# ]
```

Docker Interactive Mode

Create a terminal
To use

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

Interactive mode

Docker Interactive Mode

Attach to a running container and shell instance

```
docker attach some_container #by Name
```

Attach to a running container by starting a new shell instance

```
docker exec -it some_container /bin/bash #by Name
```

Docker Interactive Mode

Quick Tip: Be aware that alpine and other distros may have a different entry point (sh vs bash)

```
docker run -it alpine /bin/sh
```

See all containers

- \$ docker container ls #show running

```
student@docker-proto:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
e63ebe9ecaa2      dockersamples/static-site   "/bin/sh -c 'cd /u..."   8 hours ago        Up 8 hours         0.0.0.0:32768->443/tcp
69->80/tcp, 0.0.0.0:32768->443/tcp   dreamy_galileo
```

- `$ docker container ls -a` #show all containers (including running, stopped, exited, etc.)



A terminal window titled "cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 124x12" displays the command "docker container ls -a". The output lists five Docker containers with their respective IDs, images, commands, creation times, statuses, and ports.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORT
NAMES					
e63ebe9ecaa2	dockersamples/static-site	/bin/sh -c 'cd /u...'	8 hours ago	Up 8 hours	0.0.0.0:32769->80/tcp, 0.0.0.0:32768->443/tcp
16528921a105	dockersamples/static-site	/bin/sh -c 'cd /u...'	14 hours ago	Exited (137)	8 hours ago
edb964673b1b	docker/whalesay	"cowsay Hello World"	15 hours ago	Exited (0)	15 hours ago
31f3c64d31a1	hello-world	/hello	22 hours ago	Exited (0)	22 hours ago
admiring_goldberg					

\$docker container stats

```
cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 138x34
[student@docker-proto:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
NAMES
e63ebe9ecaa2      dockersamples/static-site   "/bin/sh -c 'cd /u..."   8 hours ago       Up 8 hours        0.0.0.0:32769->80/tcp, 0.0.0.0:32768->443/tcp
dreamy_galileo
[student@docker-proto:~$ docker container stats e63

cludwig — student@docker-proto: ~ — ssh student@13.82.176.214 — 138x34
CONTAINER          CPU %               MEM USAGE / LIMIT     MEM %           NET I/O          BLOCK I/O         PIDS
e63                0.00%              1.562 MiB / 667.2 MiB  0.23%          4.29 kB / 12.4 kB  73.7 kB / 12.3 kB  3
```

- Provides the CPU, memory, network, and other monitoring KPI's of a docker container.
- Multiple containers can be profiled as well
- All containers can be profiled by leaving out the container id(s)

Removing Docker Containers & Images

```
#!/bin/bash
```

```
# Remove all stopped containers  
docker container prune [-f]
```

```
# Remove all unused images  
docker image prune [-f] [-a]
```

-f option skips confirmation.

-a option (for images) removes ALL unused images, not just those that were left behind when a container was removed (dangling).

Best Practice: Prevent image inflation and regularly clean up images

Docker Working with Images and Containers



CLASSROOM WORK 2.1

45 minutes

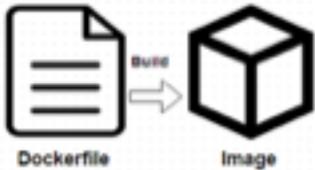
Practice Docker image and container commands

https://github.com/shekhar2010us/microservices_monolithic_docker/blob/master/docker_image_container.md

Docker Files

Part 3

Dockerfile



```
1 FROM ubuntu:14.04
2
3 # install cowsay, and move the "default.cow" out of the way so we can overwrite it with "docker.cow"
4 RUN apt-get update && apt-get install -y cowsay --no-install-recommends && rm -rf /var/lib/apt/lists/* \
5     && mv /usr/share/cowsay/cows/default.cow /usr/share/cowsay/cows/orig-default.cow
6
7 # "cowsay" installs to /usr/games
8 ENV PATH $PATH:/usr/games
9
10 COPY docker.cow /usr/share/cowsay/cows/
11 RUN ln -sv /usr/share/cowsay/cows/docker.cow /usr/share/cowsay/cows/default.cow
12
13 CMD ["cowsay"]
14
```

- Docker images are built, layer by layer, from a base image
- Images are composed of layers, each with a simple command such as
 - Run a shell command
 - Add a file or directory
 - Create an environment variable
 - Modify permissions
 - Execute process or script

Dockerfile

Dockerfile format

```
#Comment  
INSTRUCTION arguments
```

Command	Description
FROM	The base image to use in the build. This is mandatory and must be the first command in the file
MAINTAINER	An optional value for the maintainer of the script
RUN	Executes a command and save the result as a new layer This executes during build time
CMD	The command that runs when the container starts Doesn't execute during built time whereas RUN executes during build time
ENV	Sets an environment variable in the new container
COPY	Copies a file from source to destination in the file system of the container
ADD	Remote URL support and Copies a file from the host system onto the container
ENTRYPOINT	Configure container to run as an executable
EXPOSE	Opens a port for linked containers
ONBUILD	A command that is triggered when the image in the Dockerfile is used as a base for another image It behaves as if a RUN instruction is inserted immediately after the FROM instruction of the downstream Dockerfile
USER	Sets the default user within the container
VOLUME	Creates a shared volume that can be shared among containers or by the host machine
WORKDIR	Set the default working directory for the container for other instructions like RUN, CMD, COPY, ADD, ENTRYPOINT
LABEL	Add metadata to the image; LABEL key=value

Dockerfile

Dockerfile format

```
#Comment  
INSTRUCTION arguments
```

FROM command

The first instruction in a Dockerfile is a FROM command, to specify the Image from which you are building from.

```
FROM <image>  
FROM <Image>:<tag>
```

```
FROM ubuntu:14.04
```

Dockerfile

MAINTAINER command

This lets you know who to consult (or blame) for any dockerfile issues

MAINTAINER Santa Claus
Santa@northpole.org

ENV command

Used to provide environment variables for containers.

```
ENV <key> <value>
ENV AWS-KEY 123ABC12345ABCDEF
```

Dockerfile

RUN command

Executes commands and commits the results in a new layer.

RUN <command>

RUN apt-get update

CMD command

Can only be one in a dockerfile. The CMD instruction provides defaults to an executing container and can be overridden with arguments to docker run.

CMD ["cowsay"]

CMD ["/bin/bash"]

Dockerfile

ADD/COPY command

Places files onto a file system. ADD performs the same as copy, but also allows the <src> to be a URL. It will also unpack recognized compression formats.

```
COPY <src> <dest>
```

```
ADD <src> <dest>
```

EXPOSE command

Informs Docker that the container listens on a network port at runtime. Ports of a container are still not accessible to the host unless –p flag is passed to the Docker run command when the container is invoked.

Dockerfile

Best Practices for writing Dockerfiles:

- Use `.dockerignore` to exclude any unnecessary files and improve Docker's performance
- Run a single process per **container** to simplify scale and reuse
- Minimize layers per container
- Sharing base images is a common among development teams
- When possible, base from tiny images such as Alpine Linux
- For production: specify version of the base image (do not use latest)
- **For production: use image sha digest rather than image name**



Dockerfile

- Dockerfile build syntax

```
docker build .
```

```
docker build -t <namespace/image name:>versiontag> /path
```

```
docker build -t myimage/kafka .
```

***** Be aware that when you run 'docker build .', all files in the current folder get uploaded to docker engine. This may be undesirable, and use of .dockerignore will assist with this situation.***

- Then run the newly created image

```
Docker run –t myname/jenkins
```

Dockerfile Exercises



CLASSROOM WORK

Exercise 2.2 & 2.3 in Docker Labs

<https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md>

NB: You can skip the second part of step 2.3.4 (“Push Your Image”) unless you want to first create yourself an account at hub.docker.com.

Dockerfile Optimization

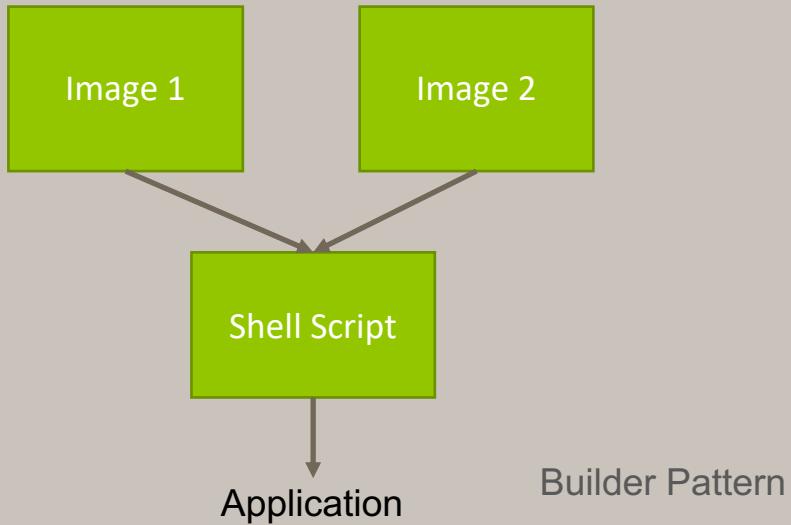


CLASSROOM WORK

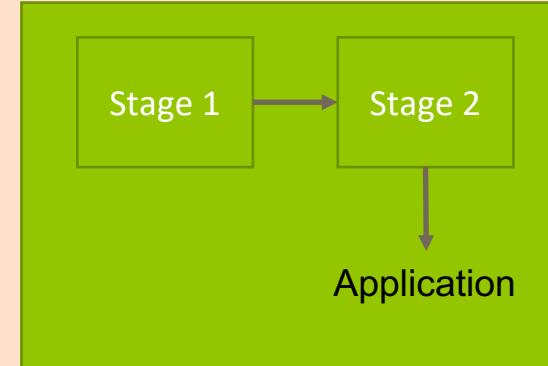
This exercise covers optimizing dockerfile to reduce the number of layers

https://github.com/shekhar2010us/aws_key/blob/master/docker/Docker%20File%20Optimization.md

Multi-stage Build



Builder Pattern



Multi Stage Build

- **Builder Pattern:** Common practice to maintain two dockerfiles. One for development and other for production that contains application and only things needed to run the application

This is not ideal !!!!

- **Multi-Stage Build:** One stage to another

Multi-stage Build

- New as of Docker 17.05
- Allows “ship artifacts, not build environments”
- Ideal for languages like Go, where binaries are built in a heavyweight environment, but can ship the binary in a lightweight container.

```
# first stage does the building
# for UX purposes, I'm naming this stage `build-stage`

FROM golang:1.8 as build-stage
WORKDIR /go/src/github.com/codeship/go-hello-world
COPY hello-world.go .
RUN go build -o hello-world .

# starting second stage
FROM alpine:latest

# copy the binary from the `build-stage`
COPY --from=build-stage /go/src/github.com/codeship/go-hello-world/hello-world /bin

CMD hello-world
```

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
single-stage	latest	58328409dbf7	2 minutes ago	704MB
multi-stage	latest	9af3c2a2bf40	23 minutes ago	5.54MB

Docker Hub/Mongo



CLASSROOM WORK

Assignment –

- Stand up a Mongo database in a container
- Connect to it
- Insert and query some rows

Pointers –

- Like Redis, mongo shell can be started in terminal. You can do insert & search queries in mongo shell
- Default port is 27017, you might want to publish this port
- About Mongo:
 - Database is called database
 - Table is called collection
 - Rows are called documents

Docker Monitoring

<https://github.com/google/cadvisor> -

Monitor the system from inside the Docker container!



```
sudo docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --  
volume=/sys:/sys:ro --volume=/var/lib/docker:/var/lib/docker:ro --  
publish=8080:8080 --detach=true --name=cadvisor google/cadvisor:latest
```

yourdockerhost:8080

The screenshot shows the cAdvisor web interface running on port 8080. The top navigation bar shows the URL as 13.82.178.214:8080/containers/. Below the header, there are two main sections: "Usage" and "Processes".

Usage section:

- Contains five circular performance meters for CPU, Memory, Network, Disk I/O, and Network I/O.

Processes section:

User	PID	PPID	Start Time	CPU %	MEM %	RSS	Virtual Size	Status	Running Time	Command	Container
root	11,956	11,939	13:05	7.60	3.80	25.69 MB	307.25 MB	S1	00:00:01	cadvisor	/dockerc/7d2e7e08c4d694428
root	5,160	1	May21	0.20	0.90	59.48 MB	398.33 MB	S1	00:03:18	dockerd	/system.slice/docker-e
root	1,571	1,151	May21	0.10	3.50	23.83 MB	212.40 MB	S1	00:01:30	python3	/system.slice/walinuxagen
root	1	0	May21	0.00	0.80	5.87 MB	37.18 MB	S0	00:00:23	systemd	/init.
root	9	8	May21	0.00	0.00	0.00 B	0.00 B	S0	nn:nn:nn	rhombard	

Running a Private Docker Registry

- docker run -d -p 5000:5000 registry

```
[ubuntu@instance-41:~/app$ sudo docker pull registry
Using default tag: latest
latest: Pulling from library/registry
3690ec4760f9: Already exists
930045f1e8fb: Pull complete
feeaa90cbdbc: Pull complete
61f85310d350: Pull complete
b6082c239858: Pull complete
Digest: sha256:1152291c7f93a4ea2ddc95e46d142c31e743b6dd70e194af9e6ebe530f782c17
Status: Downloaded newer image for registry:latest
[ubuntu@instance-41:~/app$ sudo docker run -d -p5000:5000 registry
64730d7011f71d463d480982a765624b4f353e2f2c7003779281cf453c8b2178
```



Push/Pull enabled
from private registry

Docker Registry

- Tag an image
 - `docker pull ubuntu`
 - `docker tag ubuntu localhost:5000/myubuntu:0.1`
- Push the tagged image to registry
 - `docker push localhost:5000/myubuntu:0.1`
- Verify the pushed image
 - `docker rmi -f ubuntu`
 - `docker rmi -f localhost:5000/myubuntu:0.1`
 - `docker pull localhost:5000/myubuntu:0.1`
 - Run a container from the above image

Docker Registry Search

- **Search Registry**

- <https://docs.docker.com/registry/spec/api/#listing-repositories>

```
curl -XGET http://localhost:5000/v2/_catalog
```

- **List Image Tags**

- <https://docs.docker.com/registry/spec/api/#listing-image-tags>

```
curl -XGET http://localhost:5000/v2/myubuntu/tags/list
```

Docker Create your first image



CLASSROOM WORK 3.1

45 minutes

https://github.com/shekhar2010us/microservices_monolithic_docker/blob/master/docker_create_image.md

Docker Monitoring/Registry Exercise



CLASSROOM WORK

20 minutes

https://github.com/shekhar2010us/microservices_monolithic_docker/blob/master/docker_monitor_registry.md

Docker Volumes

Part 4

Data volumes

A *data volume* is a specially-designated directory within one or more containers that bypasses the [Union File System](#). Data volumes provide several useful features for persistent or shared data:

- Volumes are initialized when a container is created. If the container's base image contains data at the specified mount point, that existing data is copied into the new volume upon volume initialization. (Note that this does not apply when [mounting a host directory](#).)
- Data volumes can be shared and reused among containers.
- Changes to a data volume are made directly.
- Changes to a data volume will not be included when you update an image.
- Data volumes persist even if the container itself is deleted.

Data volumes are designed to persist data, independent of the container's lifecycle. Docker therefore *never* automatically deletes volumes when you remove a container, nor will it "garbage collect" volumes that are no longer referenced by a container.

Adding a data volume

- You can add a **data volume** to a container using the -v flag with the docker create and docker run command. You can use the -v multiple times to mount multiple data volumes. Now, mount a single volume in your web application container.
 - `docker run -it --name test1 -v data2:/data ubuntu`
 - In addition to creating a volume using the -v flag you can also **mount a directory** from your Docker engine's host into a container
 - `docker run -it --name test1 -v /home/ubuntu/src:/data ubuntu`
- This command mounts the host directory, /home/ubuntu/src, into the container at /data.

Docker volume commands

Command	Description
docker volume create	Create a volume
docker volume ls	List volumes
docker volume inspect	Detailed information on volume
docker volume rm	Remove volume

Working with Volumes (docker volumes)

```
ubuntu@ip-172-31-35-117:~$ docker volume ls
DRIVER          VOLUME NAME
local           0aaf0f814ce772146fbffbe8d67da39bc00e25d7f728a3a43b1f7080499342b
local           4e37dd17754c6f0ba9f43ebc8052be0c18db1c5b56b44f539b283406afc35e5c
local           b5f8b253293cc20b10133335eed74848fbf466b5d055fa5e3b16976766a4c618
ubuntu@ip-172-31-35-117:~$ 
ubuntu@ip-172-31-35-117:~$ docker volume create myvolume
myvolume
ubuntu@ip-172-31-35-117:~$ docker volume ls
DRIVER          VOLUME NAME
local           0aaf0f814ce772146fbffbe8d67da39bc00e25d7f728a3a43b1f7080499342b
local           4e37dd17754c6f0ba9f43ebc8052be0c18db1c5b56b44f539b283406afc35e5c
local           b5f8b253293cc20b10133335eed74848fbf466b5d055fa5e3b16976766a4c618
local           myvolume

ubuntu@ip-172-31-35-117:~$ docker volume inspect myvolume
[ {
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/myvolume/_data",
    "Name": "myvolume",
    "Options": {},
    "Scope": "local"
} ]
```

Working with Volumes (docker volumes)

```
ubuntu@ip-172-31-35-117:~$ docker run -it -v myvolume:/myvolume --name vlab nginx  
/bin/bash  
root@5476f2fa7f09:/# cd myvolume/  
root@5476f2fa7f09:/myvolume# ls  
root@5476f2fa7f09:/myvolume#  
root@5476f2fa7f09:/myvolume# echo "content from container" >> file.1  
root@5476f2fa7f09:/myvolume# ls  
file.1  
root@5476f2fa7f09:/myvolume# exit  
exit
```

```
ubuntu@ip-172-31-35-117:~$ sudo su -  
root@ip-172-31-35-117:~# ll /var/lib/docker/volumes/myvolume/_data  
total 12  
drwxr-xr-x 2 root root 4096 Aug  7 04:19 ./  
drwxr-xr-x 3 root root 4096 Aug  7 04:18 ../  
-rw-r--r-- 1 root root    23 Aug  7 04:19 file.1  
root@ip-172-31-35-117:~# cat /var/lib/docker/volumes/myvolume/_data/file.1  
content from container  
root@ip-172-31-35-117:~# exit  
logout
```

Working with Volumes (absolute path)

```
ubuntu@ip-172-31-35-117:~$ mkdir data_dir
ubuntu@ip-172-31-35-117:~$ echo "file.1" >> data_dir/file.1
ubuntu@ip-172-31-35-117:~$ echo "file.2" >> data_dir/file.2
ubuntu@ip-172-31-35-117:~$ ll data_dir/
total 16
drwxrwxr-x 2 ubuntu ubuntu 4096 Aug  7 04:25 .
drwxr-xr-x 7 ubuntu ubuntu 4096 Aug  7 04:25 ..
-rw-rw-r-- 1 ubuntu ubuntu     7 Aug  7 04:25 file.1
-rw-rw-r-- 1 ubuntu ubuntu     7 Aug  7 04:25 file.2
ubuntu@ip-172-31-35-117:~$
ubuntu@ip-172-31-35-117:~$ docker run -it -v /home/ubuntu/data_dir:/mydata --name vlab2 nginx /bin/bash
root@4fd5e542d67c:/# ls mydata/
file.1 file.2
root@4fd5e542d67c:/# cat mydata/file.1
file.1
root@4fd5e542d67c:/# exit
exit
```

Working with Volumes (--volumes-from)

```
ubuntu@ip-172-31-35-117:~$ docker volume create --name backup  
backup  
ubuntu@ip-172-31-35-117:~$ docker volume create --name logs  
logs  
ubuntu@ip-172-31-35-117:~$ docker run -it --name master -v backup:/backup -v  
logs:/logs ubuntu bash  
root@cc963e4d24b9:/# echo "back up content" >> backup/back.file  
root@cc963e4d24b9:/# echo "log file content" >> logs/log.file  
root@cc963e4d24b9:/# exit  
exit  
ubuntu@ip-172-31-35-117:~$ docker run -it --name slave --volumes-from master ubuntu  
bash  
root@f027d96030ba:/# ll backup/  
total 12  
drwxr-xr-x 2 root root 4096 Aug 7 04:30 ./  
drwxr-xr-x 1 root root 4096 Aug 7 04:31 ../  
-rw-r--r-- 1 root root 16 Aug 7 04:30 back.file  
root@f027d96030ba:/# ll logs/  
total 12  
drwxr-xr-x 2 root root 4096 Aug 7 04:30 ./  
drwxr-xr-x 1 root root 4096 Aug 7 04:31 ../  
-rw-r--r-- 1 root root 17 Aug 7 04:30 log.file  
root@f027d96030ba:/# exit  
exit
```

Dangling Volumes

- If container is deleted with a volume attached, the volume remains. Sometimes removing all such ‘dangling’ volumes is desired
 - `$ docker volume prune`

Dangling Volumes

- Before

```
[ubuntu@instance-41:~$ docker volume ls
DRIVER          VOLUME NAME
local           030f652ad7ea6766f3d17a421233896b78296cf89d9bc816c69e6084df1aa24d
local           1e16af70af2319707350ce672c0af925c010561a5f75f83393674b5f20df7074
local           94ed6b916617b932638619fcbdbfc77d843175a57c520d02d3162f5750916409
```

- After

```
ubuntu@instance-41:~$ docker volume prune
030f652ad7ea6766f3d17a421233896b78296cf89d9bc816c69e6084df1aa24d
1e16af70af2319707350ce672c0af925c010561a5f75f83393674b5f20df7074
94ed6b916617b932638619fcbdbfc77d843175a57c520d02d3162f5750916409
ubuntu@instance-41:~$ docker volume ls
DRIVER          VOLUME NAME
```

Docker Volumes Exercise



CLASSROOM WORK

30 minutes

https://github.com/shekhar2010us/aws_key/blob/master/docker/Docker%20Volume.md

Docker Volume Plugin - Flocker

<https://clusterhq.com/flocker/>



When container moves, its data volume stays in place.
Database starts on new server without any data.



When container moves, data volume moves with it.
Your database gets to keep its data!

Docker Cheat Sheet



Docker Cheat Sheet



Glossary

Layer - a set of read-only files to provision the system.

Image - a read-only layer that is the basis of your container. Might have a parent image.

Container - a runnable instance of the image.

Registry / Hub - central place where images live.

Docker machine - a VM to run Docker containers (Linux does this natively).

Docker compose - a utility to run multiple containers as a system.

Useful one-liners

Download an Image

```
docker pull image_name
```

Start and stop the container

```
docker start|stop container_name
```

Create and start container, run command

```
docker run -it --name container_name image_name command
```

Create and start container, run command, destroy container

```
docker run --it -t image_name command
```

Example filesystem and port mappings

```
docker run -it --rm -p 8080:8080 -v /path/to/app:/agent.jar -e JAVA_OPTS="-Dmanagement.agent=/agent.jar" tomcat:8.0.29-jdk
```

Docker cleanup commands

Kill all running containers

```
docker kill $(docker ps -q)
```

Delete dangling images

```
docker rmi $(docker images -q -f dangling=true)
```

Remove all stopped containers

```
docker rm $(docker ps -a -q)
```

Docker machine commands

Use docker-machine to run the containers

Start a machine

```
docker-machine start machine_name
```

Configure docker to use a specific machine

```
eval $(docker-machine env machine_name)
```

Docker compose syntax

docker-compose.yml file example

```
version: "2"
services:
  web:
    container_name: "web"
    image: java:8 # image name
    # command to run
    command: java -jar /app/app.jar
    ports: # map ports to the host
      - "4567:4567"
    volumes: # map filesystem to the host
      - ./app:/app:/app/app.jar
  mongo:
    container_name: "mongo"
    image: mongo # image name
```

Create and start containers

```
docker-compose up
```

Interacting with a container

Run a command in the container

```
docker exec -it container_name command sh
```

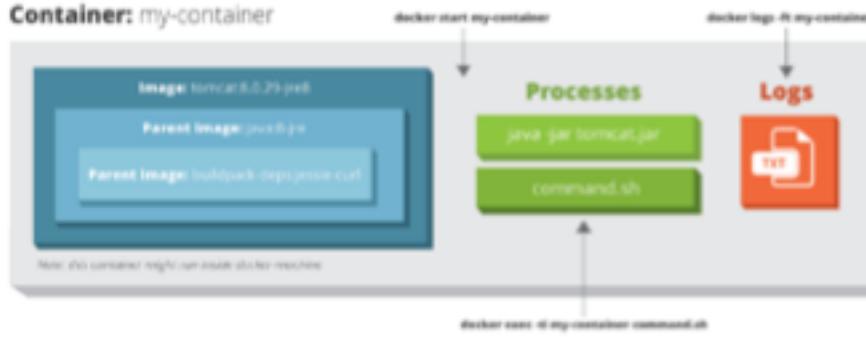
Follow the container logs

```
docker logs -f container_name
```

Save a running container as an image

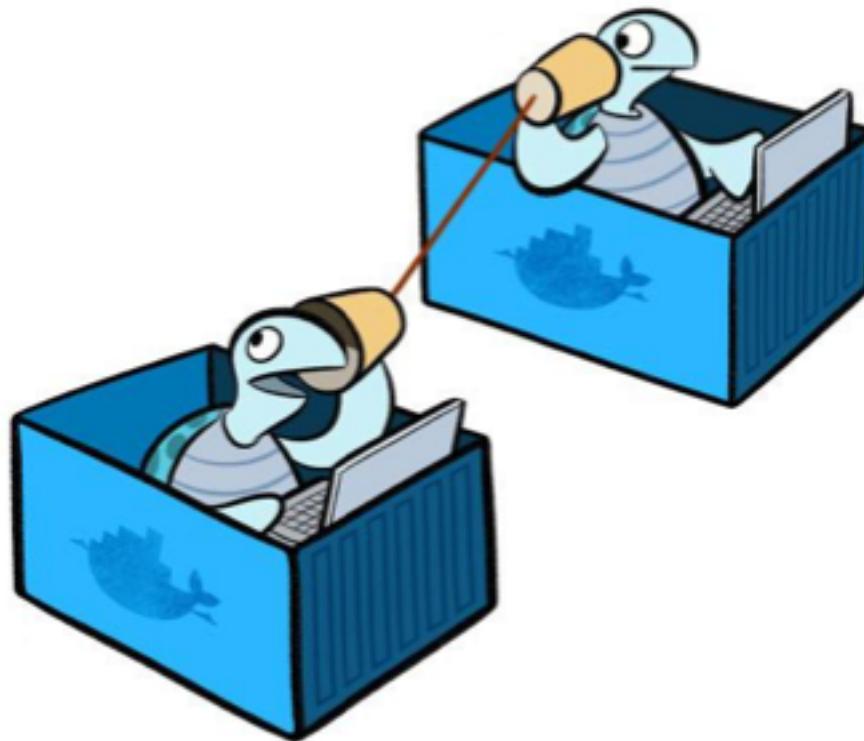
```
docker commit -c "commit message" -n "author" container_name username/image_name:tag
```

Container: my-container



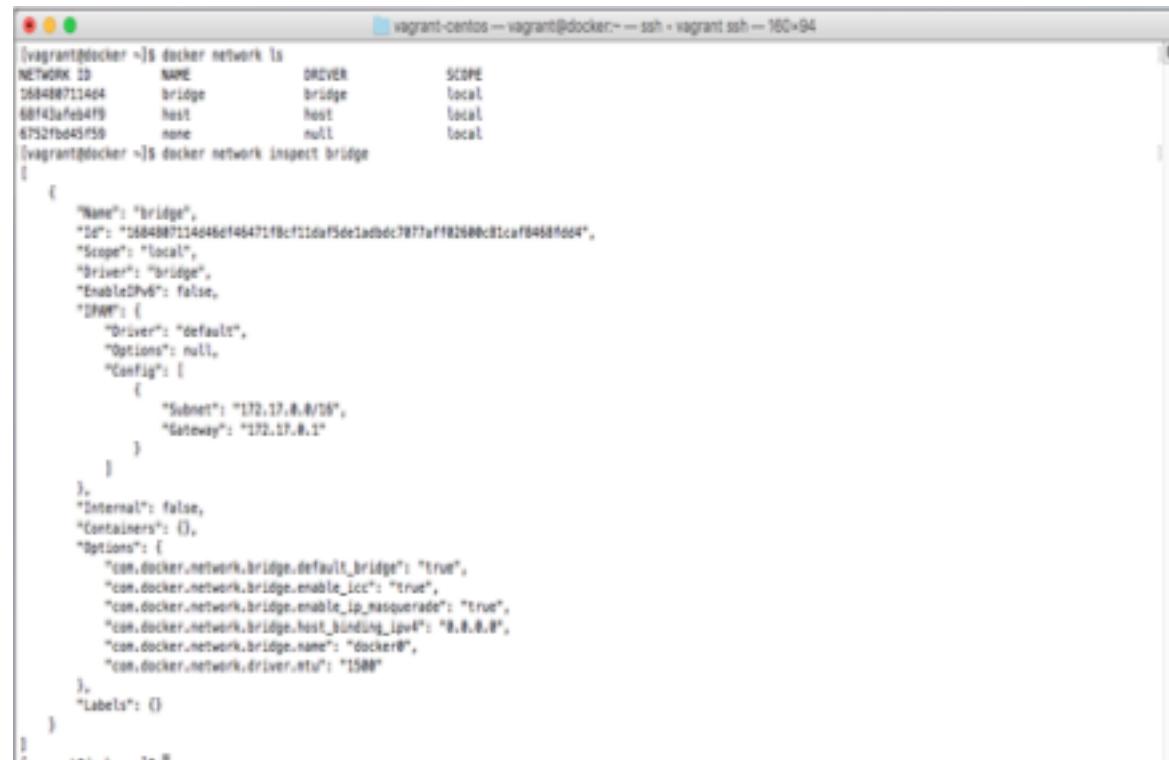
BOUGHT TO YOU BY
JRebel

Docker Networking



Docker Network Defaults

- Three networks are created by default (bridge, host & none)
 - **docker network ls**
- We can easily inspect the bridge network
 - **docker network inspect bridge**
- We see that a 172.17.0.0/16 address space is allocated. Created Containers are given an IP in this space by default.



```
vagrant@vagrant:~$ docker network ls
NETWORK ID      NAME    DRIVER      SCOPE
568488711484    bridge   bridge      local
6814afeb84f9    host     host       local
6752fb45f59    none     null       local

[vagrant@vagrant:~$ docker network inspect bridge
{
  "Name": "bridge",
  "Id": "5684887114848ef48471f8cf11da5de1adbc7877aff82680c81caf8488f6d4",
  "Scope": "local",
  "Driver": "bridge",
  "EnableIPv6": false,
  "IPAM": {
    "Driver": "default",
    "Options": null,
    "Config": [
      {
        "Subnet": "172.17.0.0/16",
        "Gateway": "172.17.0.1"
      }
    ]
  },
  "Internal": false,
  "Containers": {},
  "Options": {
    "com.docker.network.bridge.default_bridge": "true",
    "com.docker.network.bridge.enable_icc": "true",
    "com.docker.network.bridge.enable_ip_masquerade": "true",
    "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
    "com.docker.network.bridge.name": "docker0",
    "com.docker.network.driver.mtu": "1500"
  },
  "Labels": {}
}
```

Docker Network Defaults

```
vagrant@vagrant:~$ docker run --name some-ghost -p 8080:2368 -d ghost
53efddde5d54cadb3cfaab85476f6981f373cf3e6825c2af8f5cc140e62a464
vagrant@vagrant:~$ docker network inspect bridge
{
    "Name": "bridge",
    "Id": "53efddde5d54cadb3cfaab85476f6981f373cf3e6825c2af8f5cc140e62a464",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
        "Driver": "default",
        "Options": null,
        "Config": [
            {
                "Subnet": "172.17.0.0/16",
                "Gateway": "172.17.0.1"
            }
        ]
    },
    "Internal": false,
    "Containers": {
        "53efddde5d54cadb3cfaab85476f6981f373cf3e6825c2af8f5cc140e62a464": {
            "Name": "some-ghost",
            "EndpointID": "4e026b09883d98ee592bbf228e467e9f8f815948a7abc79543524152148fa7f",
            "MacAddress": "02:42:ac:11:00:02",
            "IPv4Address": "172.17.0.2/16",
            "IPv6Address": ""
        }
    },
    "Options": {
        "com.docker.network.bridge.default_bridge": "true",
        "com.docker.network.bridge.enable_icc": "true",
        "com.docker.network.bridge.enable_ip_masquerade": "true",
        "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
        "com.docker.network.bridge.name": "docker0",
        "com.docker.network.driver.iface": "eth0"
    },
    "Labels": {}
}
vagrant@vagrant:~$ ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.053 ms
...
--- 172.17.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 200ms
rtt min/avg/max/mdev = 0.049/0.184/0.183/0.057 ms
```

- Created container is attached to bridge network by default
 - Can be attached to user created networks as well with `--network` flag
- Container is assigned 172.17.0.2 ip in the bridge network and appears in the docker inspect command
- The container can then be pinged at 172.17.0.2

Docker network commands

Command	Description
docker network create	Create a network
docker network ls	List networks
docker network inspect	Detailed information on network
docker network rm	Remove network
docker network disconnect	Disconnect container from network
docker network connect	Connect container to network

Working with Networks

Run redis container with default network

```
ubuntu@ip-172-31-35-117:~$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
bcddd713372f    bridge    bridge      local
94a5aa84bbcd    host      host       local
e785de4a1213    none      null       local
ubuntu@ip-172-31-35-117:~$ docker run -d -p 1000:6379 --name red1 redis
82233b85cb84e2372e0e3fcdf03c5f46642c8faf0e201a7b0c43f7439c2ca02e
```

```
ubuntu@ip-172-31-35-117:~$ docker inspect red1 | grep IPAddress
  "SecondaryIPAddresses": null,
  "IPAddress": "172.17.0.3",
  "IPAddress": "172.17.0.3",
```

Private IP
of the container

Run ubuntu container with default network

```
ubuntu@ip-172-31-35-117:~$ docker run -it --name ub1 ubuntu
root@65b7ce9e6028:/# apt-get update && apt-get install -y redis-tools
```

Working with Networks

Check how to access redis container from ubuntu container → we can access

```
root@65b7ce9e6028:/# redis-cli -h <private_ip_container> -p 6379
172.17.0.2:6379> set fool bar1
OK
172.17.0.2:6379> quit
root@65b7ce9e6028:/#
root@65b7ce9e6028:/# redis-cli -h <public_ip_host> -p 1000
34.216.137.67:1000> get fool
"bar1"
34.216.137.67:1000> quit
root@65b7ce9e6028:/#
root@65b7ce9e6028:/# redis-cli -h 34.216.137.67 -p 6379
Could not connect to Redis at 34.216.137.67:6379: Connection refused
Could not connect to Redis at 34.216.137.67:6379: Connection refused
not connected> quit
root@65b7ce9e6028:/#
root@65b7ce9e6028:/# redis-cli -h 172.17.0.2 -p 1000
Could not connect to Redis at 172.17.0.2:1000: Connection refused
Could not connect to Redis at 172.17.0.2:1000: Connection refused
not connected>
not connected> quit
root@65b7ce9e6028:/# CNTRL P Q (to exit ubuntu, but not kill it)
```

This confirms that you can access redis container with either
<private ip and private port> or <public ip or public port>

Working with Networks

Create a new bridge network

```
ubuntu@ip-172-31-35-117:~$ docker network create bridgea  
1f0afe0c3f5529403c5baa17e16e5534d0bf2f1966fd75c385902a7e97e5f61f  
ubuntu@ip-172-31-35-117:~$ docker network ls  
NETWORK ID      NAME        DRIVER      SCOPE  
bcddd713372f    bridge      bridge      local  
1f0afe0c3f55    bridgea     bridge      local  
94a5aa84bcd    host        host        local  
e785de4a1213    none        null       local
```

Run ubuntu container in the new network, install redis-tools

```
ubuntu@ip-172-31-35-117:~$ docker run -it --name ub2 --network bridgea ubuntu  
root@34461322da59:/# apt-get update && apt-get install -y redis-tools
```

Redis (bridge network) is not accessible from ubuntu (bridgea) network

```
root@34461322da59:/# redis-cli -h <private_ip_container> -p 6379  
^Z  
[2]+  Stopped                  redis-cli -h 172.17.0.2 -p 6379  
root@34461322da59:/#
```

Working with Networks

Connect ub2 container to bridge network

```
ubuntu@ip-172-31-35-117:~$ docker network connect bridge ub2  
ubuntu@ip-172-31-35-117:~$ docker inspect ub2
```

```
"Networks": {  
    "bridge": {  
        "Gateway": "172.17.0.1",  
        "IPAddress": "172.17.0.3",  
    },  
    "bridgea": {  
        "Gateway": "172.18.0.1",  
        "IPAddress": "172.18.0.2",  
    }  
}
```

Now Redis (bridge network) is accessible from ubuntu (bridgea + bridge) network

```
ubuntu@ip-172-31-35-117:~$ docker attach ub2  
root@34461322da59:/# redis-cli -h 172.17.0.2 -p 6379  
172.17.0.2:6379> keys *  
1) "foo1"  
172.17.0.2:6379> get foo1  
"bar1"  
172.17.0.2:6379> quit
```

Docker Networks Exercise



CLASSROOM WORK

20 minutes

- Run a redis container with default network
- Run a ubuntu container with default network
- Install redis-tools in host and check if you can connect to redis container and add some data
- Install redis-tools in ubuntu container and check if you can connect to redis container. (optional: create an image with redis-tools and ubuntu)
- Create a bridge network (bridgea)
- Run a ubuntu container with bridgea network and the new image, check if you can connect to redis – probably not
- Connect this new ubuntu container with “bridge” network, and check again if you can connect to the redis container

**** keep inspecting network and container after every step**

Docker Compose/Swarm

Part 5

Launching Multiple Containers is Clumsy



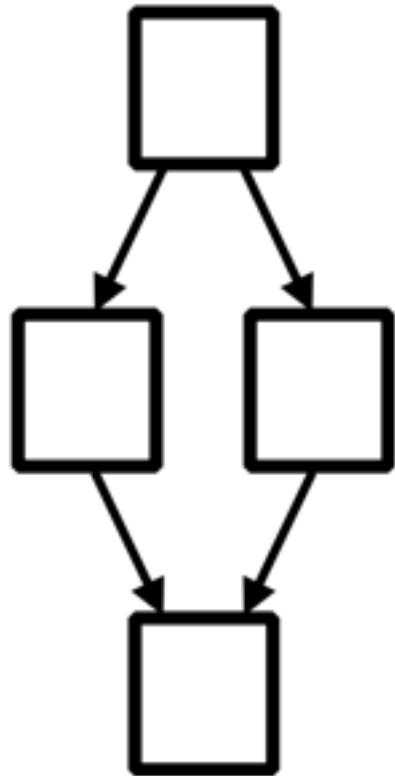
```
$ docker pull redis:latest
```

```
$ docker build -t web .
```

```
$ docker run -d --name=db redis:latest redis-server  
--appendonly yes
```

```
$ docker run -d --name=web --link db:db -p  
5000:5000 -v `pwd`:/code web python app.py
```

Launching Multiple Containers is Clumsy



\$ docker pull ...

\$ docker pull ...

\$ docker build ...

\$ docker build ...

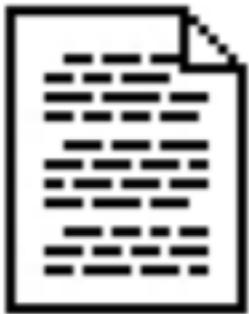
\$ docker run ...

\$ docker run ...

\$ docker run ...

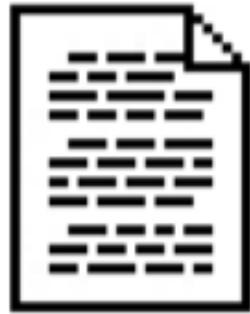
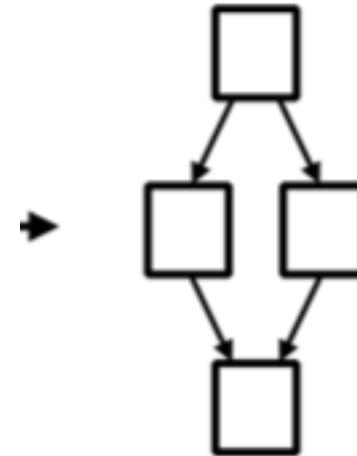
\$ docker run ...

Docker Compose or Swarm Launches app



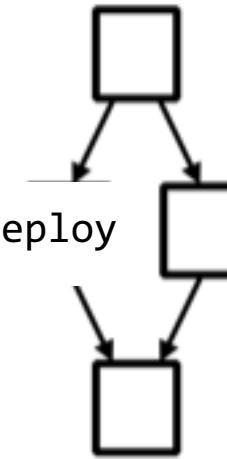
Text file

→ \$ docker-compose up



Text file

→ \$ dock \$ docker stack deploy



Docker Compose Commands

Command	Description
<code>docker-compose up</code>	(Re)build services
<code>docker-compose kill</code>	Kill the containers
<code>docker-compose logs</code>	Show the logs of the containers
<code>docker-compose down</code>	Stop and remove images, containers, volumes and networks
<code>docker-compose rm</code>	Remove stopped containers

Docker Compose to deploy a cluster



CLASSROOM WORK

1. Install docker-compose:

<https://github.com/docker/compose/releases>

- sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
- sudo chmod +x /usr/local/bin/docker-compose
- docker-compose version

2. Create web app using docker-compose:

<https://docs.docker.com/compose/gettingstarted/>

Step 1-4: mandatory

Step 5-8: Optional

Also connect to the redis container and check the entry in the database

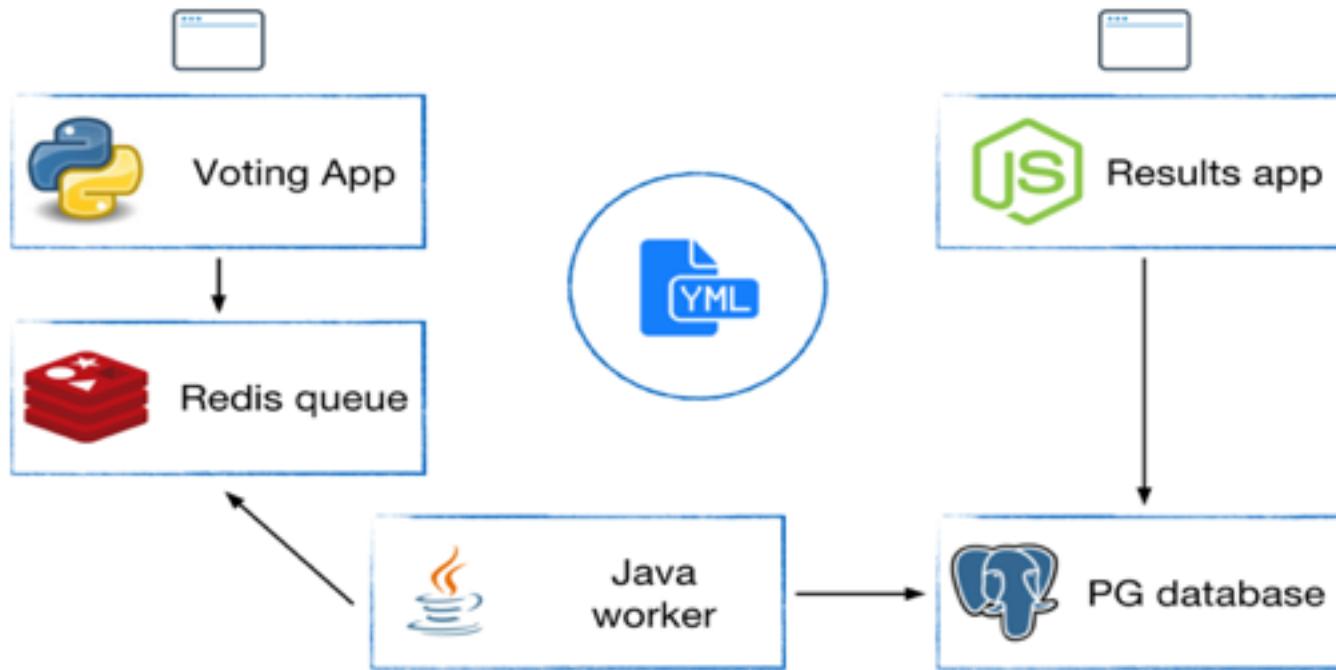
Docker Swarm Commands

Command	Description
<code>docker swarm init</code>	Create a swarm
<code>docker stack deploy</code>	Deploy the swarm
<code>docker stack services</code>	Show swarm services
<code>docker stack rm</code>	Remove stack

Web App with multiple Tech



CLASSROOM WORK



Web App with multiple Tech



CLASSROOM WORK

Exercise 3.0 in Docker Labs

<https://github.com/docker/labs/blob/master/beginner/chapters/votingapp.md>

(1) Deploy using docker-compose

\$ docker-compose up -d (wait until all done and then test in browser)

After test is complete: docker-compose down

** kill all containers and remove all image

docker ps -a | awk -F" " '{print \$1}' | xargs docker rm -f

docker images | awk -F" " '{print \$3}' | xargs docker rmi -f

(2) Deploy using docker swarm

\$ docker swarm init

\$ docker stack deploy --compose-file docker-stack.yml vote

\$ docker stack services vote (wait until all done and then test in browser)

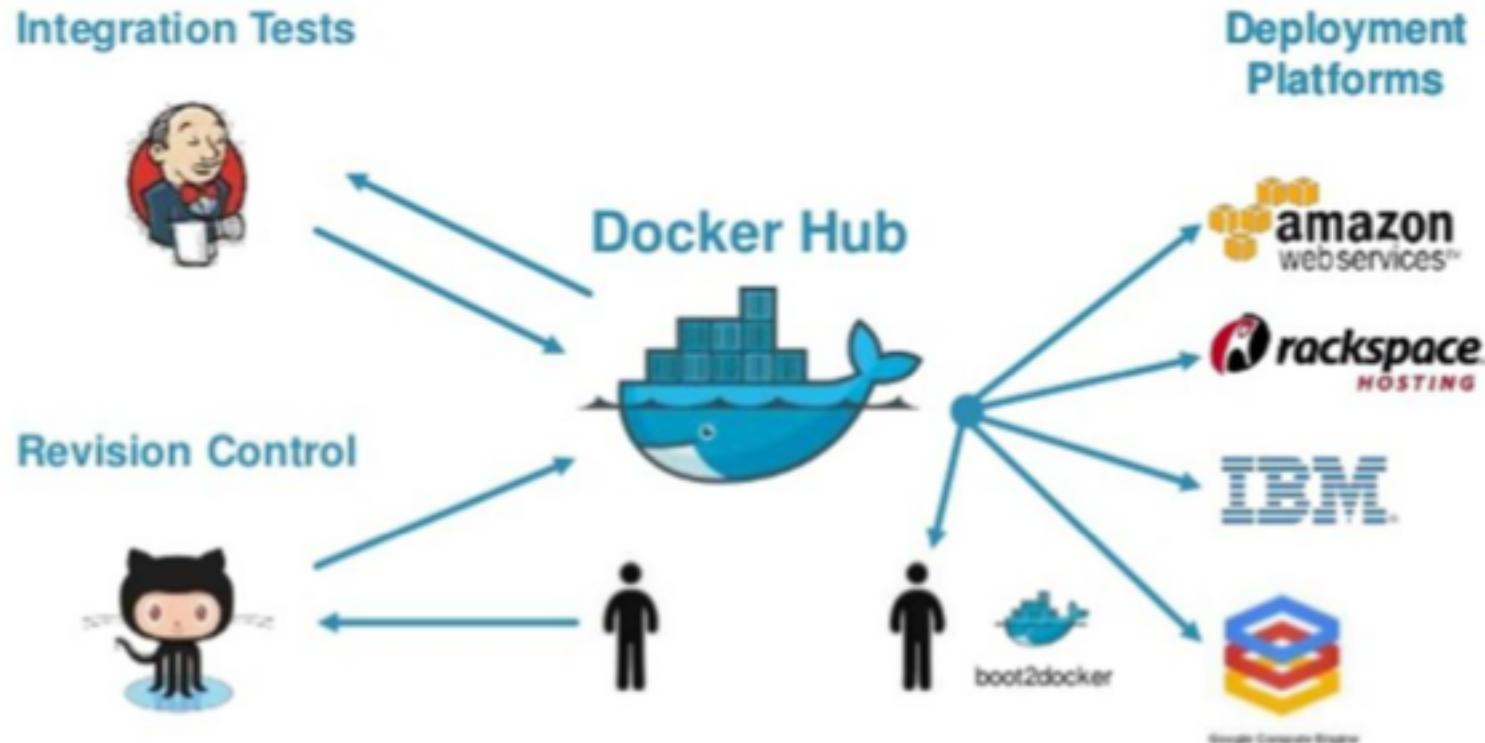
After test is complete: docker stack rm vote

(3) Kubernetes deployment is not in scope – but you can check how it works, the project has all relevant Kubernetes code

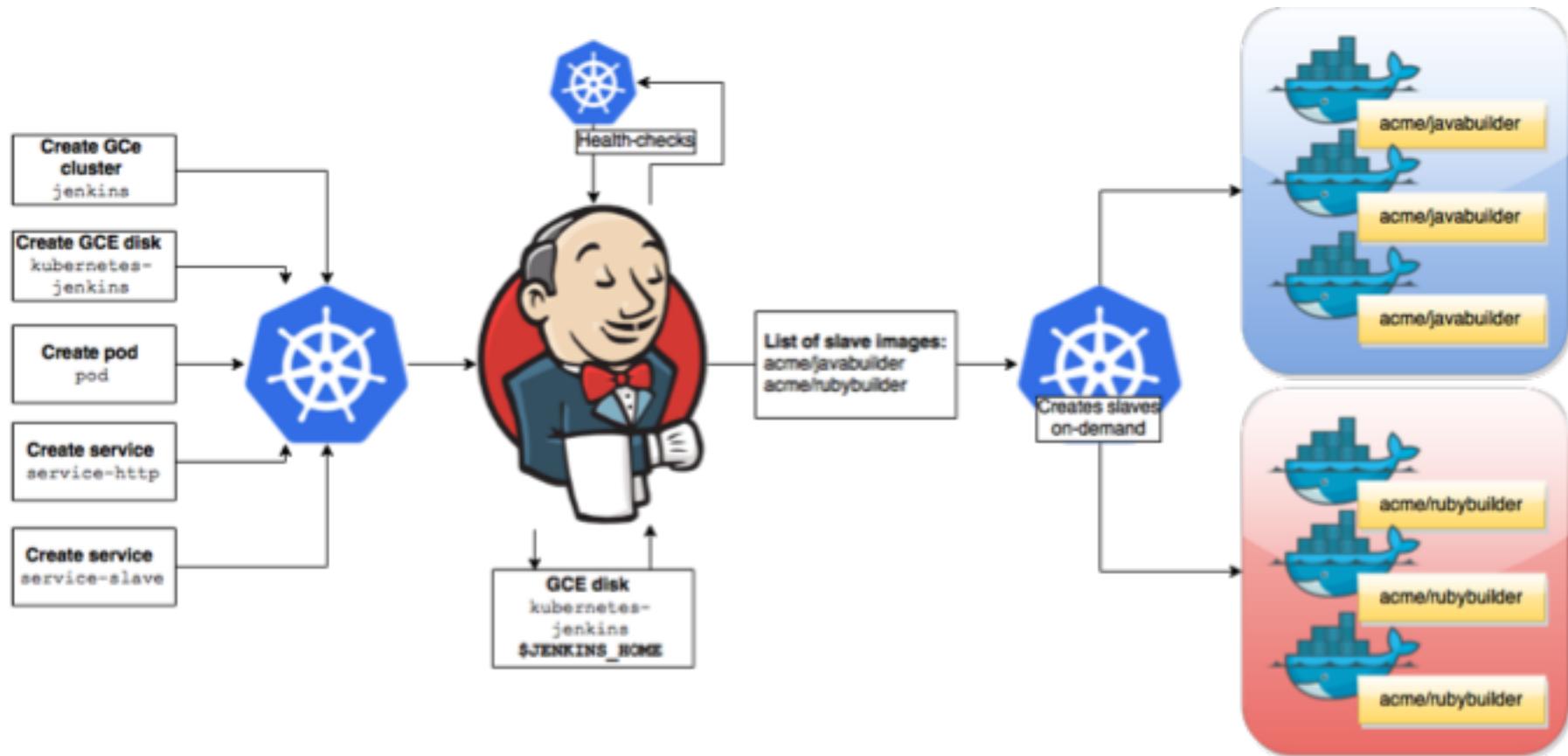
Docker Patterns

Part 6

Docker Patterns

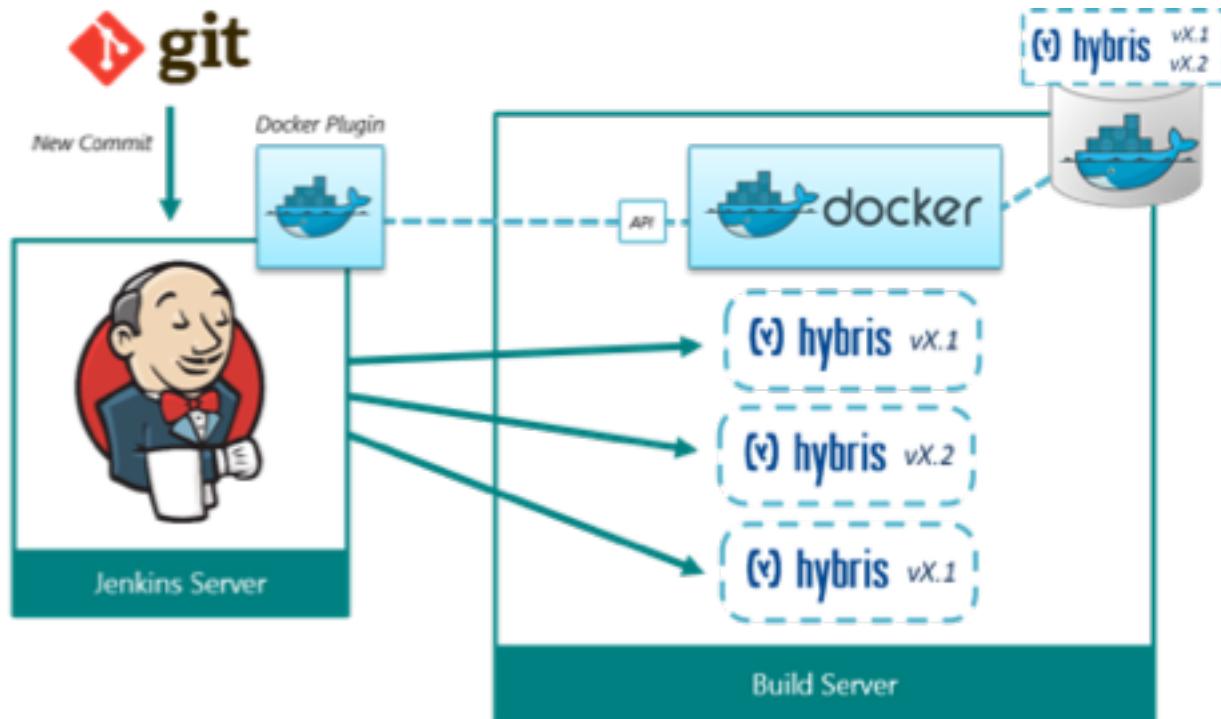


Jenkins Continuous Integration-Cloudbees



- Jenkins master runs from a Docker image and is part of a Kubernetes Jenkins cluster.
- Master has persistent volume that stores job/system configuration and credentials
- `$JENKINS_HOME` is in slave node
- This separation of master and slave enable easy management and easy upgrade of master

According to Michael Duke



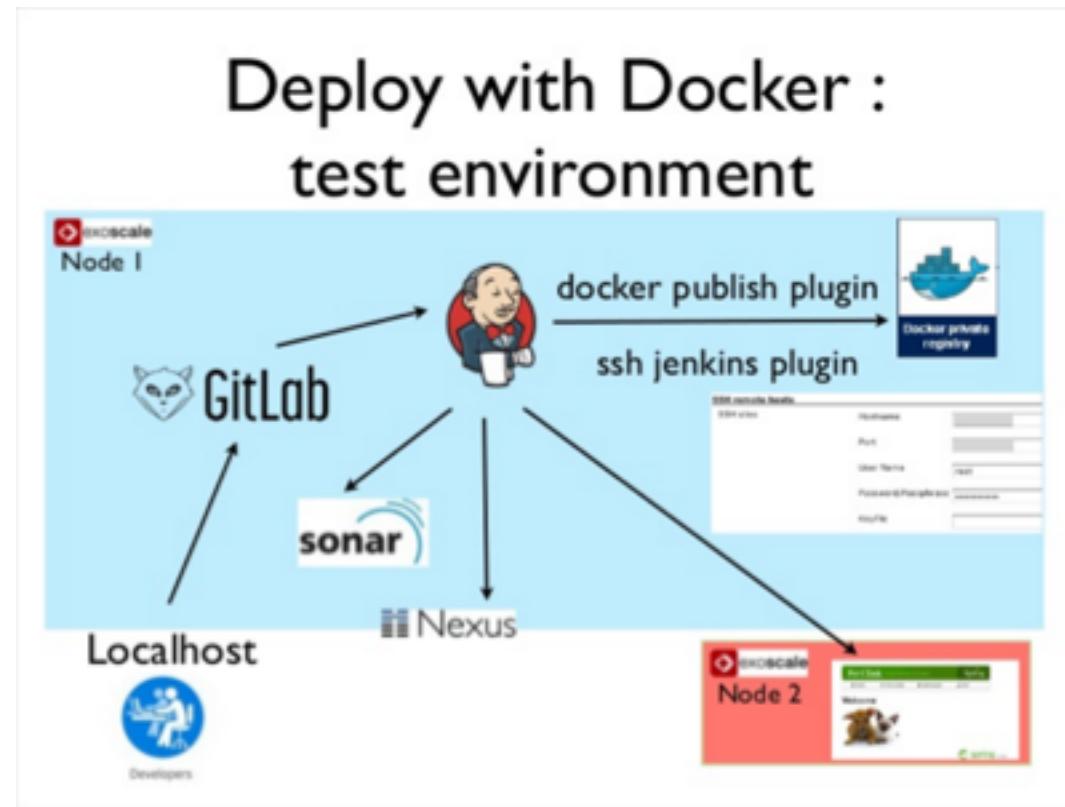
- Docker plugin for Jenkins
- Allow Jenkins to provision new containers as Jenkins slave
- To compile code for a particular version of Hybris

Generally

- Hybris can compile one app at a time (takes 45+ minutes)
- And cannot cross-compile app for diff versions of the platform

Link to Blog: Docker as Jenkins Build Solution <http://bit.ly/2a57Za8>

According to Julia Mateo



- CICD workflow using Jenkins and Docker plugin
- Publish and pull images onto prod
- Code passes SonarQube quality and artifacts stored in Nexus

Link to Presentation: Continuous Delivery

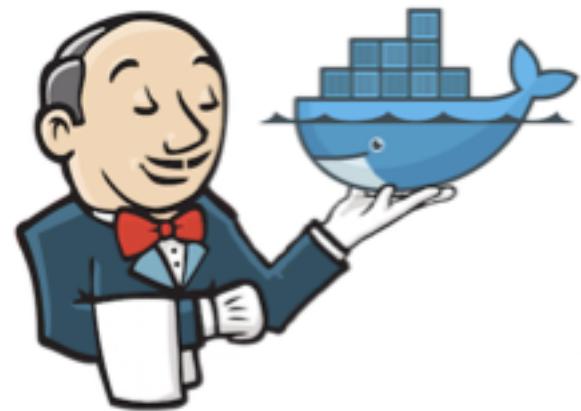
<http://bit.ly/2aHZj90>

Docker and Jenkins

Docker and Jenkins, a popular open source continuous integration tool, is a fairly common pattern, enough so that it is worth going into an example.

Spin up Jenkins in
a Docker
Container

Enable Jenkins to
create other
containers on the
Docker Host computer



How to run a ‘Host’ Docker command inside a ‘Client’ Docker container?

Mount a Docker socket

- Issuing a Docker run command with the Docker socket will enable you to manage host containers from within the client container
- ‘`docker run -v /var/run/docker.sock:/var/run/docker.sock`’

Docker and Jenkins



CLASSROOM WORK

Jenkins Exercise in Docker Tutorials

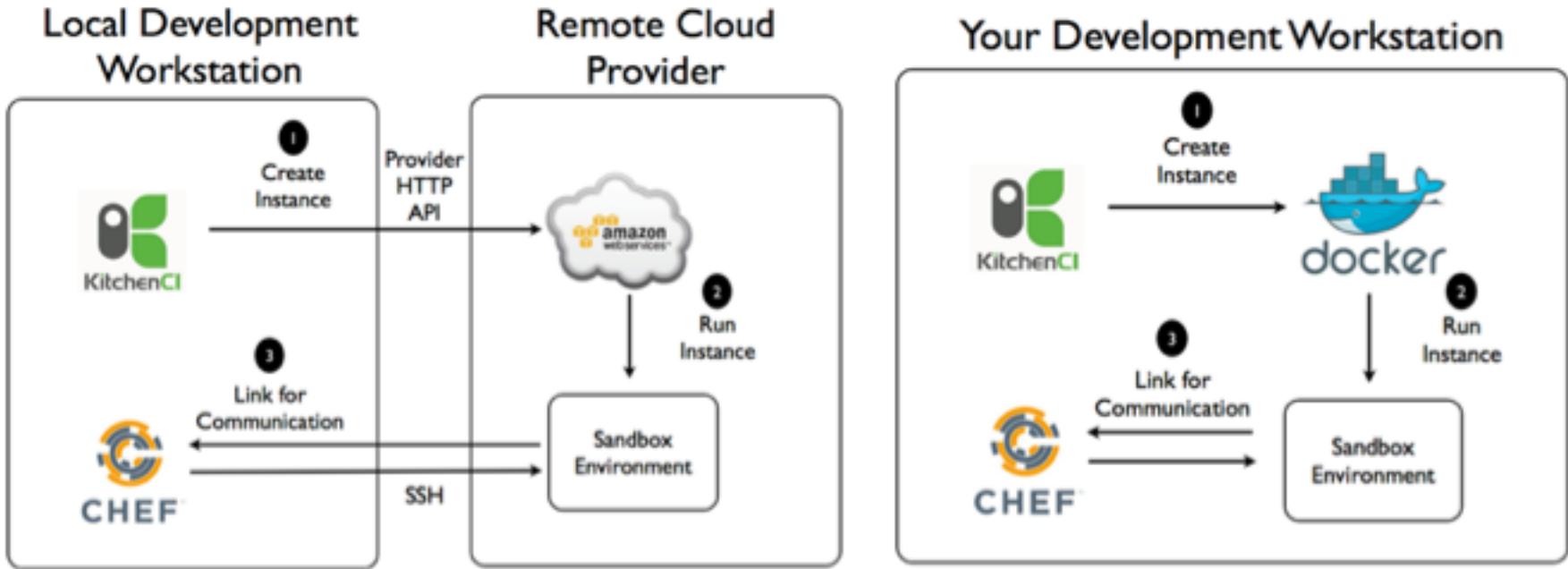
This assignment is to set up a Jenkins server using Docker in a container. And to create a simple project to build.

Estimated time to complete: 20-30 min

Solution:

https://github.com/shekhar2010us/aws_key/blob/master/docker/Jenkins%20%26%20Docker.md

According to Mischa Taylor

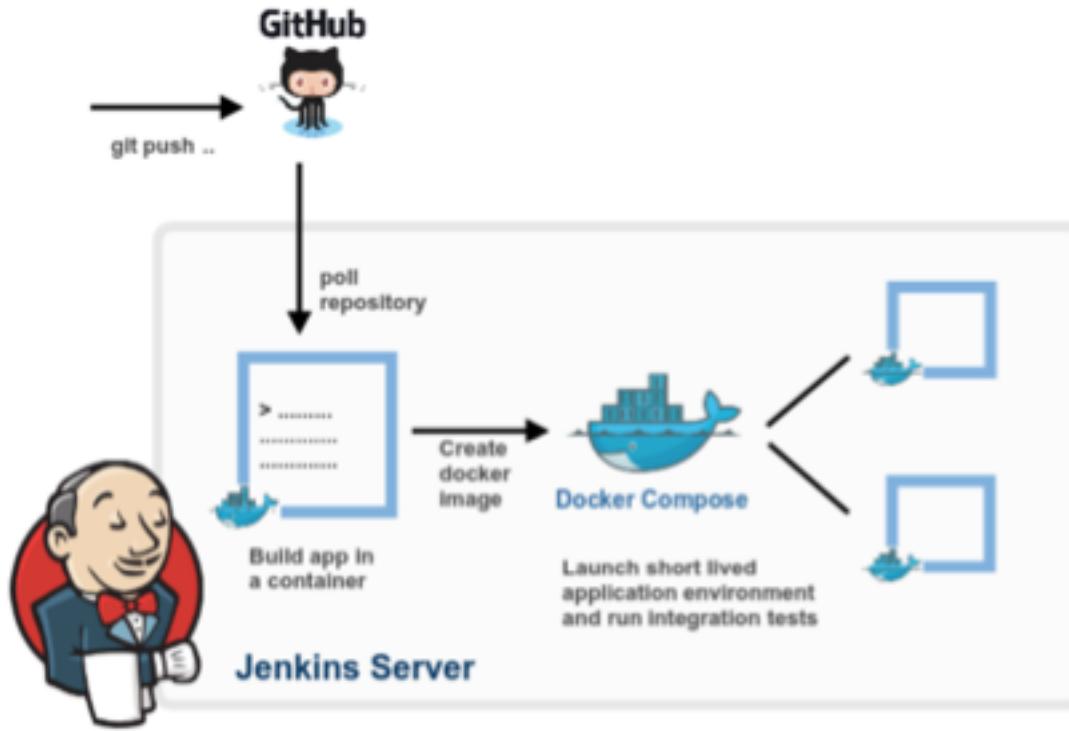


- Chef SCM Test Kitchen harness environment running against various deployment environments – including a Docker based system

Link to Blog: Survey of Test Kitchen Providers

<http://bit.ly/2a56kS2>

According to Rancher

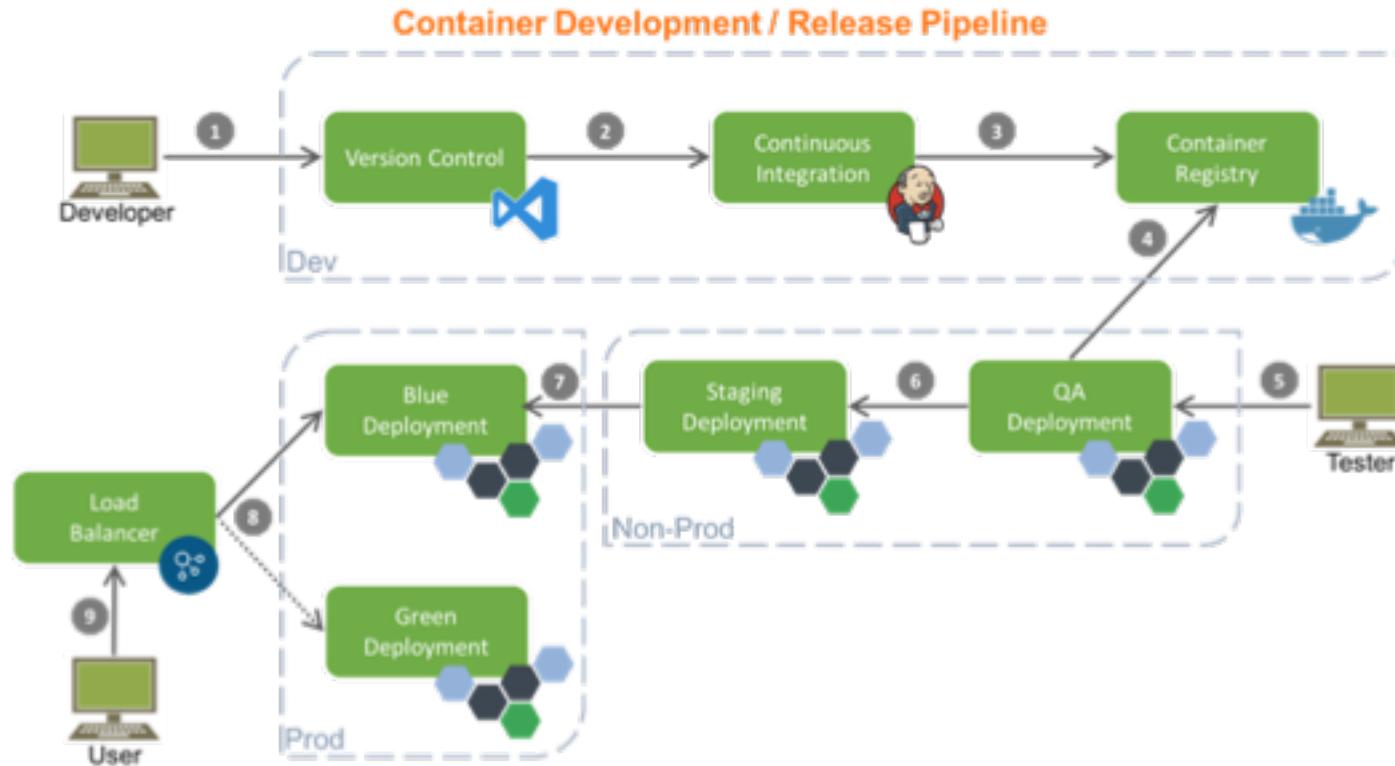


- Jenkins and Docker to build a CI pipeline based around the git-flow branching model (with develop(ment) branch and master branch)

Link to Blog: Docker-based Build Pipelines

<http://bit.ly/2aHtuOK>

According to Robert Greiner

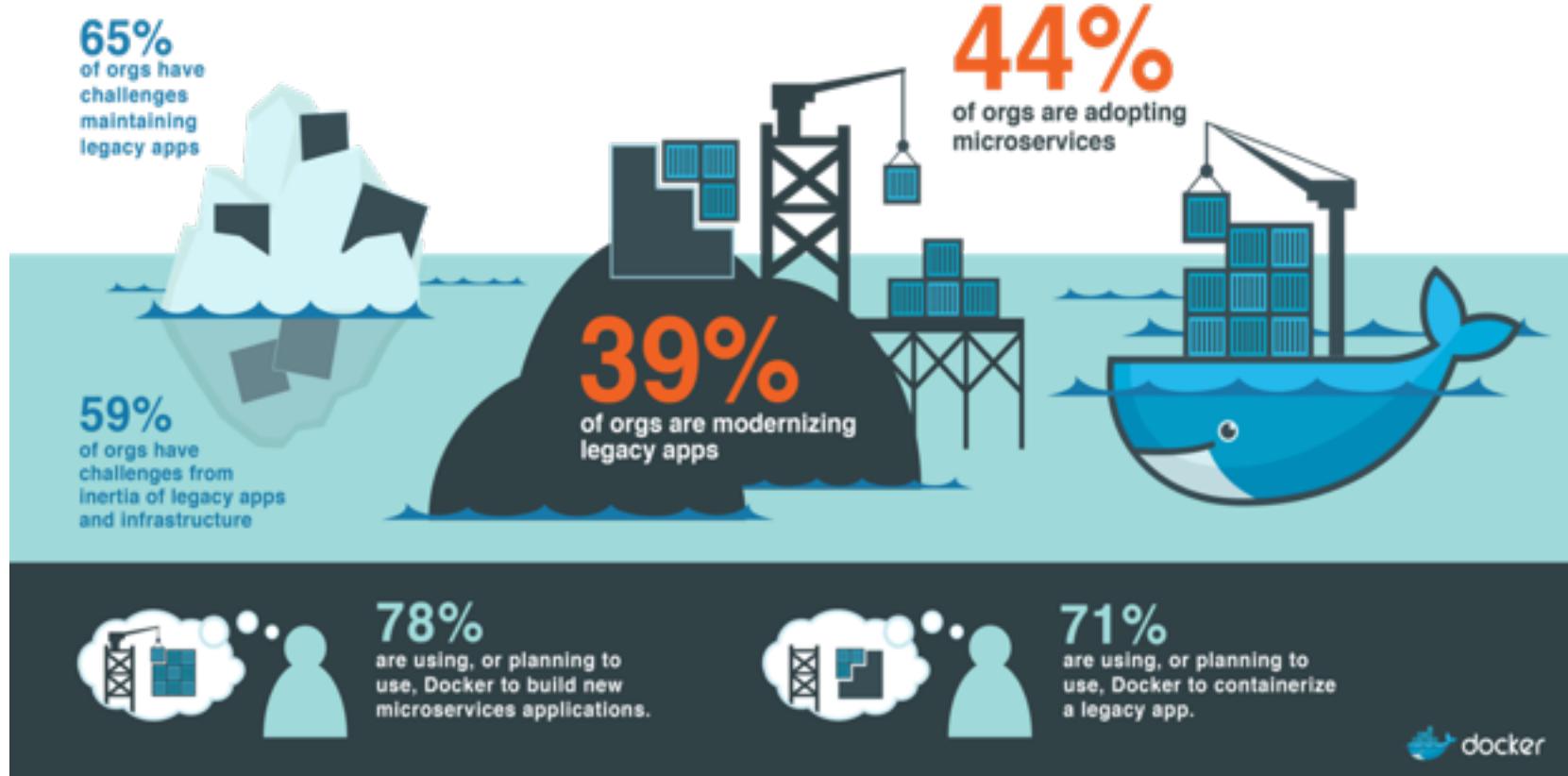


- Blue Green Deployment using Docker

Link to Blog: Continuous Integration with Docker

<http://bit.ly/2aeA1io>

Docker Adoption & Usage Data

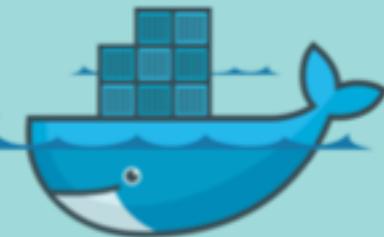
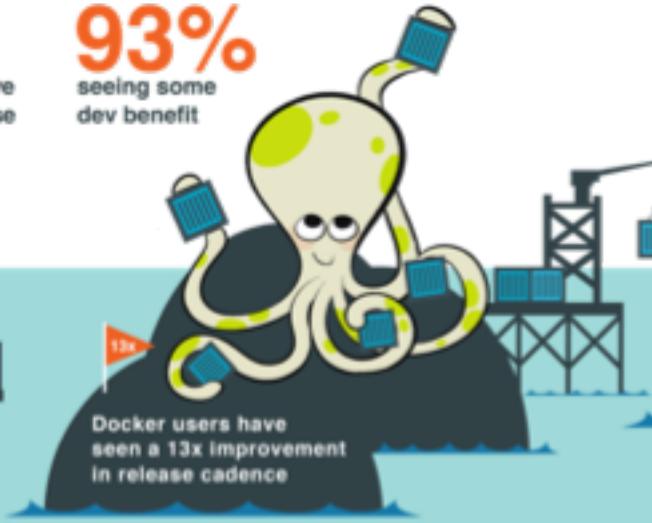


Docker Adoption & Usage Data

45%
of Docker users have
been able to increase
the frequency of
software releases

93%
seeing some
dev benefit

57%
Docker users
have seen
improvements
in operational
environment
management



70%
of Docker users say
*'Docker has dramatically
transformed... etc'*



62%
have seen improved MTTR
on software issues.



Docker Security

Part 7

Security For Docker Images

- Getting trustworthy images
 - Trusted Content
 - Pull by sha
- **Docker Bench Security**
- Docker Trusted Registry
- **Docker Security scanning**
- **Clair by CoreOS**

Trusted Content

```
ubuntu@ip-172-31-35-117:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
ubuntu@ip-172-31-35-117:~$ export DOCKER_CONTENT_TRUST=1
ubuntu@ip-172-31-35-117:~$ docker pull alpine:latest
Pull (1 of 1): alpine:latest@sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430: Pulling from library/alpine
8e3ba11ec2a2: Pull complete
Digest: sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Status: Downloaded newer image for
alpine@sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Tagging alpine@sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430 as alpine:latest
ubuntu@ip-172-31-35-117:~$
ubuntu@ip-172-31-35-117:~$ docker pull tenstartups/alpine
Using default tag: latest
Error: remote trust data does not exist for docker.io/tenstartups/alpine: notary.docker.io does not have trust data for
docker.io/tenstartups/alpine
ubuntu@ip-172-31-35-117:~$
ubuntu@ip-172-31-35-117:~$ export DOCKER_CONTENT_TRUST=0
ubuntu@ip-172-31-35-117:~$ docker pull tenstartups/alpine
Using default tag: latest
latest: Pulling from tenstartups/alpine
ff3a5c916c92: Pull complete
938de623aa55: Pull complete
Digest: sha256:31dc8b12e0f73a1de899146c3663644b7668f8fd198cfe9b266886c9abfa944b
Status: Downloaded newer image for tenstartups/alpine:latest
ubuntu@ip-172-31-35-117:~$
```

Trusted Content

```
ubuntu@ip-172-31-35-117:~$ docker images
REPOSITORY      TAG          IMAGE ID   CREATED        SIZE
ubuntu@ip-172-31-35-117:~$ export DOCKER_CONTENT_TRUST=0
ubuntu@ip-172-31-35-117:~$ docker pull alpine:latest
latest: Pulling from library/alpine
8e3ba11ec2a2: Pull complete
Digest: sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Status: Downloaded newer image for alpine:latest
ubuntu@ip-172-31-35-117:~$
ubuntu@ip-172-31-35-117:~$ docker rmi -f alpine:latest
Untagged: alpine:latest
Untagged: alpine@sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Deleted: sha256:11cd0b38bc3ceb958ffb2f9bd70be3fb317ce7d255c8a4c3f4af30e298aa1aab
Deleted: sha256:73046094a9b835e443af1a9d736fcfc11a994107500e474d0abf399499ed280c
ubuntu@ip-172-31-35-117:~$
ubuntu@ip-172-31-35-117:~$ export DOCKER_CONTENT_TRUST=1
ubuntu@ip-172-31-35-117:~$ docker pull alpine:latest
Pull (1 of 1): alpine:latest@sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430: Pulling from library/alpine
8e3ba11ec2a2: Pull complete
Digest: sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Status: Downloaded newer image for
alpine@sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Tagging alpine@sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430 as alpine:latest
ubuntu@ip-172-31-35-117:~$
```

Docker Content Trust



CLASSROOM WORK

In this exercise, we'll learn how to enable Docker Content Trust and sign images.

https://github.com/shekhar2010us/aws_key/blob/master/docker/Lab-Distribution%20and%20Trust.md

Docker Bench Security

Docker Bench is an automated script that checks for dozens of best-practices around deploying Docker in production

<https://github.com/docker/docker-bench-security>

Run with the following command
(available at the github page above)

```
docker run -it --net host --pid host --cap-add  
audit_control \  
-v /var/lib:/var/lib \  
-v /var/run/docker.sock:/var/run/docker.sock \  
-v /usr/lib/systemd:/usr/lib/systemd \  
-v /etc:/etc --label docker_bench_security \  
docker/docker-bench-security
```





Security

10-15 Minutes

Docker Bench Results Discussion

<https://www.nearform.com/blog/securing-docker-containers-on-aws/>

<https://www.cisecurity.org/cis-benchmarks/>

https://github.com/shekhar2010us/aws_key/blob/master/CIS_Docker_Community_Edition_Benchmark_v1.1.0.pdf

(Some) Docker Security Best Practices

- Docker containers can attach to volumes in read only mode with :ro option
 - Docker run -d -v /some/volume:ro jenkins
 - Start Docker containers with the -u flag so that they run as an ordinary user instead of root.
 - Mitigate by limiting CPU, RAM, Sockets that each container can consume
 - Use secure computing (seccomp) to block system calls at kernel level. Use strace to determine kernel calls made, then create a profile file in json format and start the container using that profile file
 - `docker run --rm -it --security-opt seccomp=custom_profile.json custom_app`
 - Log to stdout / stderr - so you can see with docker logs and docker can move to syslogs (for capture/rotate by ELK, etc.)
 - `docker inspect --format='{{.LogPath}}' <containerid> => push to ELK for monitoring`

Docker Security Scanning (formerly Project Nautilus)

- Docker Security Scanning conducts binary level scanning of your images before they are deployed, provides a detailed bill of materials (BOM) that lists out all the layers and components, continuously monitors for new vulnerabilities, and provides notifications when new vulnerabilities are found.
- Available for private Docker Hub repositories (potentially free)
- Available as a paid service to Docker Cloud customers, maintainers of “Official” repositories on Docker Hub, and Docker EE Advanced

<https://github.com/docker/labs/blob/master/security/scanning/README.md>

Clair by CoreOS

- Robust free and open source image security scanning tool
- <https://github.com/coreos/clair>
- Static analysis of images for known vulnerabilities
- Integrates into CI pipelines

Docker Security

**Stay up to date with the discussion at
Docker's user group, forum, github issues,
and IRC channel**

- <https://groups.google.com/forum/#!forum/docker-dev>
- <https://forums.docker.com/>
- <https://github.com/docker/docker/issues>
- <https://docs.docker.comopensource/get-help/>
- **IRC #docker & #docker-dev**

Use Docker to Set Up an ELK Stack



CLASSROOM WORK

Use Docker to (quickly!) set up an ELK (Elasticsearch, Logstash, Kibana) stack

https://github.com/shekhar2010us/aws_key/blob/master/docker/Lab-%20Setup%20ELK.md

Secrets Management

- ENV variables - NOT recommended
- General-purpose Key/Value Pair solutions
 - Vault
 - Keywhiz
- Embedded in orchestration
 - Docker 1.13+ - recommended
 - Kubernetes secrets
- Custom solutions
 - Make sure you care as much as Docker et al.

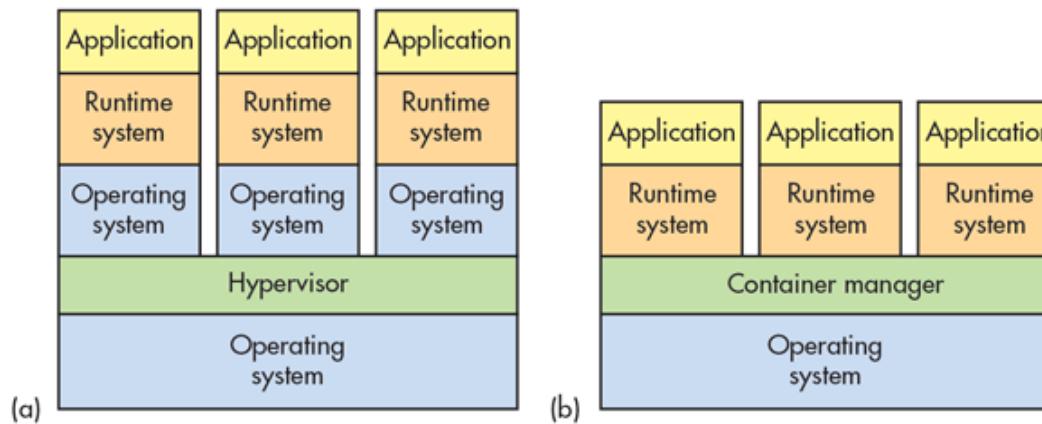
Docker is So Much Fun!

- **So easy to get started**
 - Got an open-source or commercial service or application that you're considering? Docker makes ~~playing with~~ researching it as easy as:
 - \$ docker run coolnewtoy:latest
- **So easy to scale:**
 - docker service scale webfrontend=5
- **Some more fun:**
 - <https://github.com/docker/dockercraft>
 - <https://www.youtube.com/watch?v=eZDIJgJf55o>

Kubernetes

Container Advantages

- Portable
- Isolated
- Lighter footprint & overhead (vs VMs)
- Simplify Devops practices
- Speed up Continuous Integration
- Empower Microservice architectures and adoption



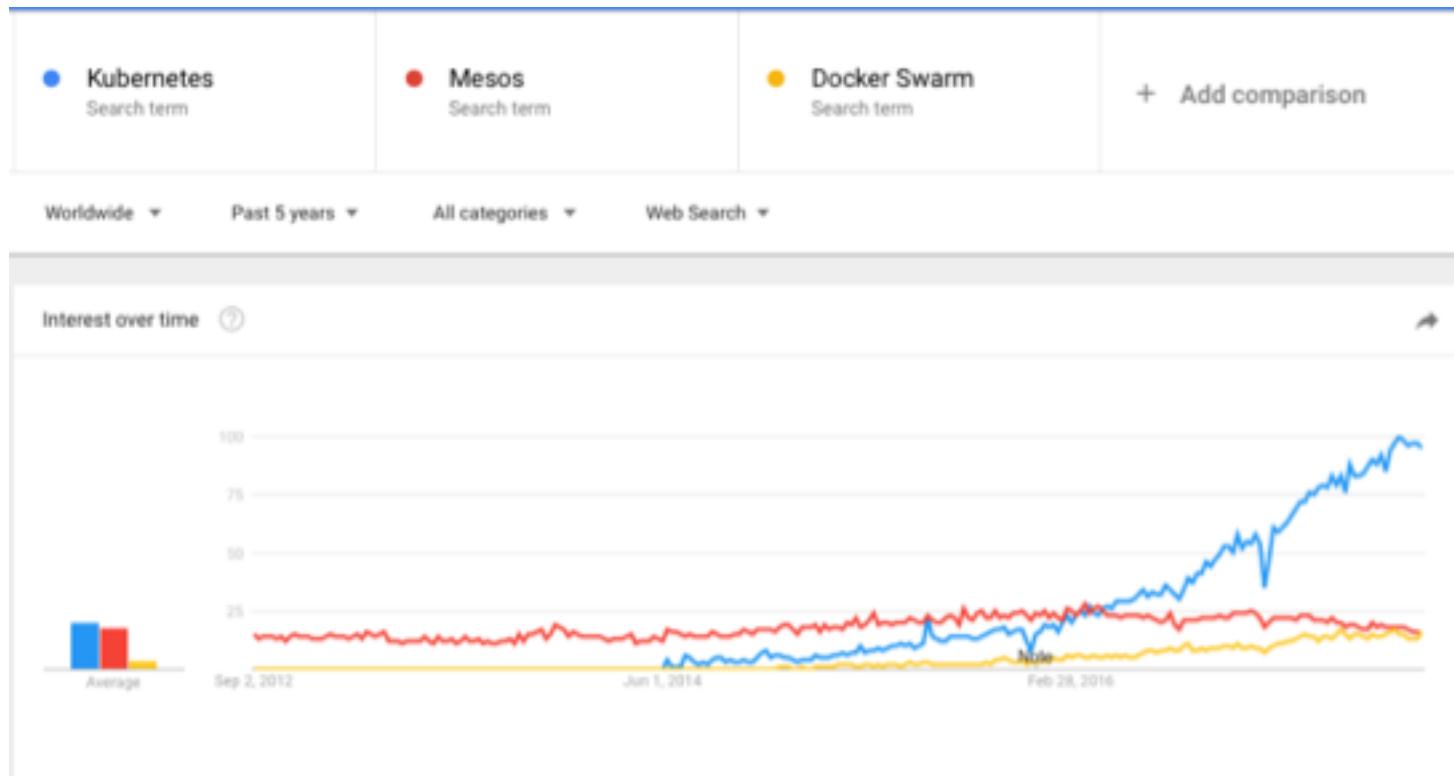
Challenges with multiple containers



- How to scale?
- Once I scale, where are they?
- How do my containers find each other?
- How should I manage port conflicts?
- What if a host fails?
- How to update them? Health checks?
- How will I track their logs?

Container Orchestration Tools

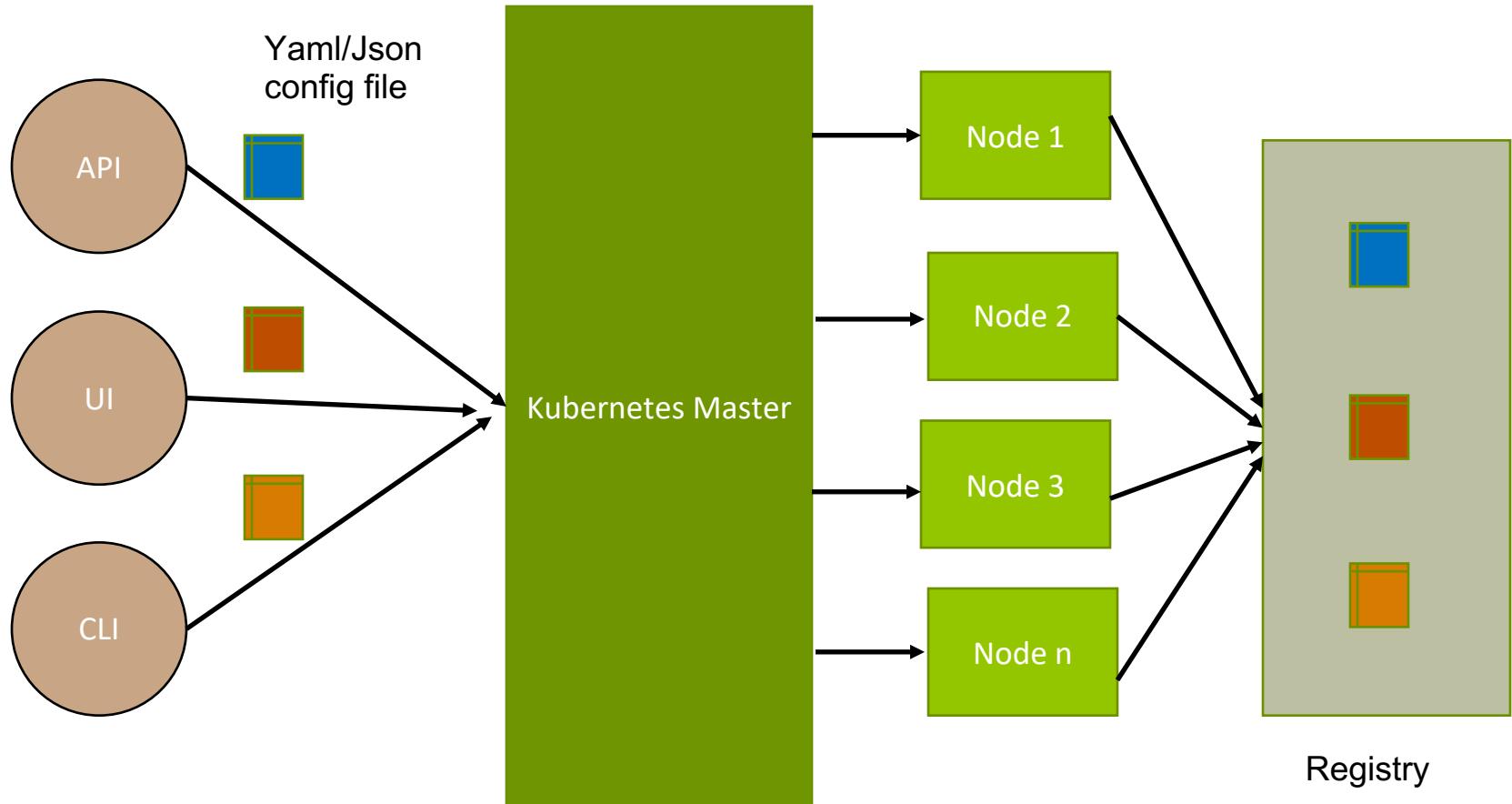
- The three most popular are: Kubernetes, Docker Swarm & Mesos
- Kubernetes has become the unofficial standard



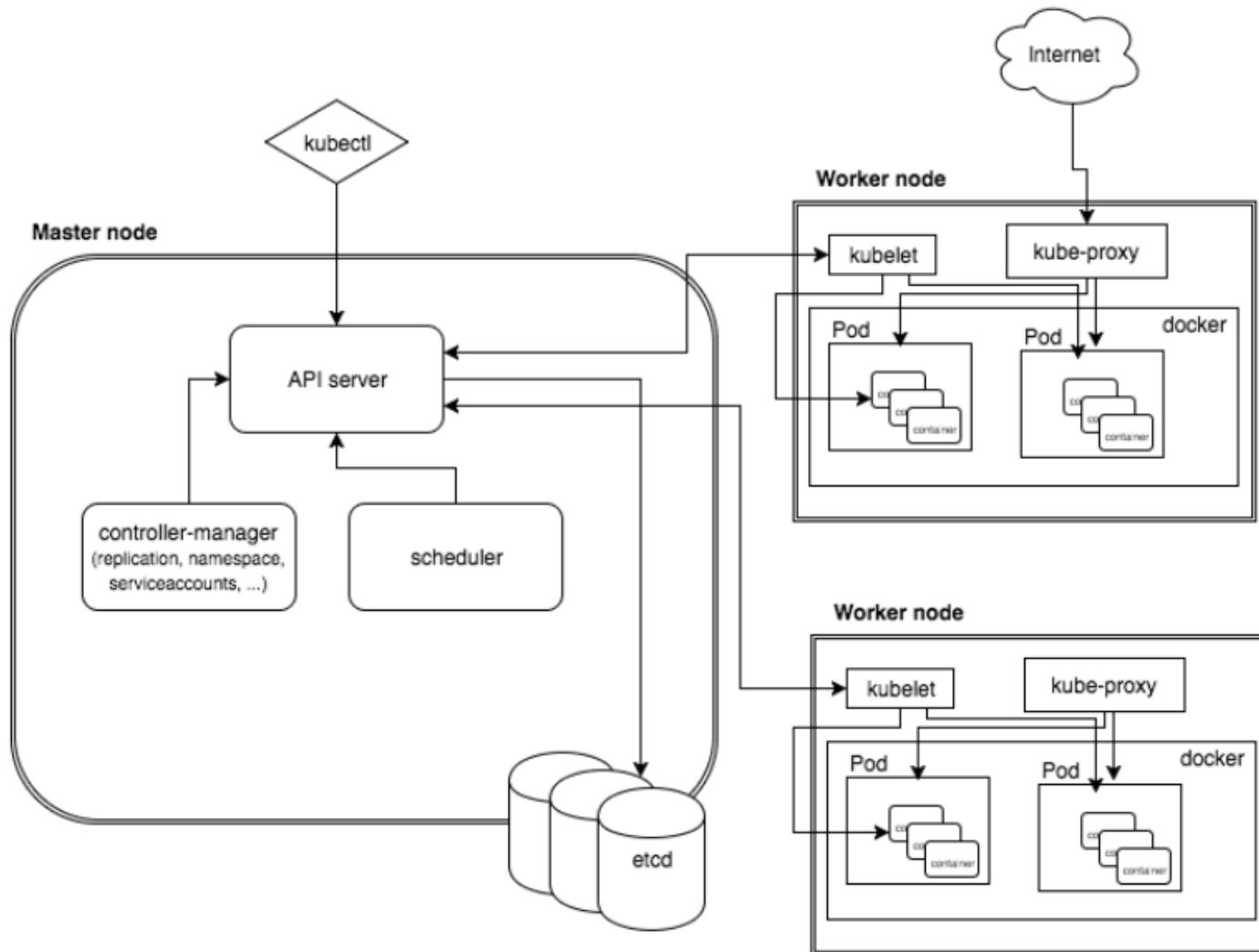
What is Kubernetes?

- **Kubernetes is inspired from an internal Google project called Borg**
- **Open source project managed by the Linux Foundation**
- **Unified API for deploying web applications, batch jobs, and databases**
- **Decouples applications from machines through containers**
- **Declarative approach to deploying applications**
- **Automates application configuration through service discovery**
- **Maintains and tracks the global view of the cluster**
- **APIs for deployment workflows**
 - **Rolling updates, canary deploys, and blue-green deployments**

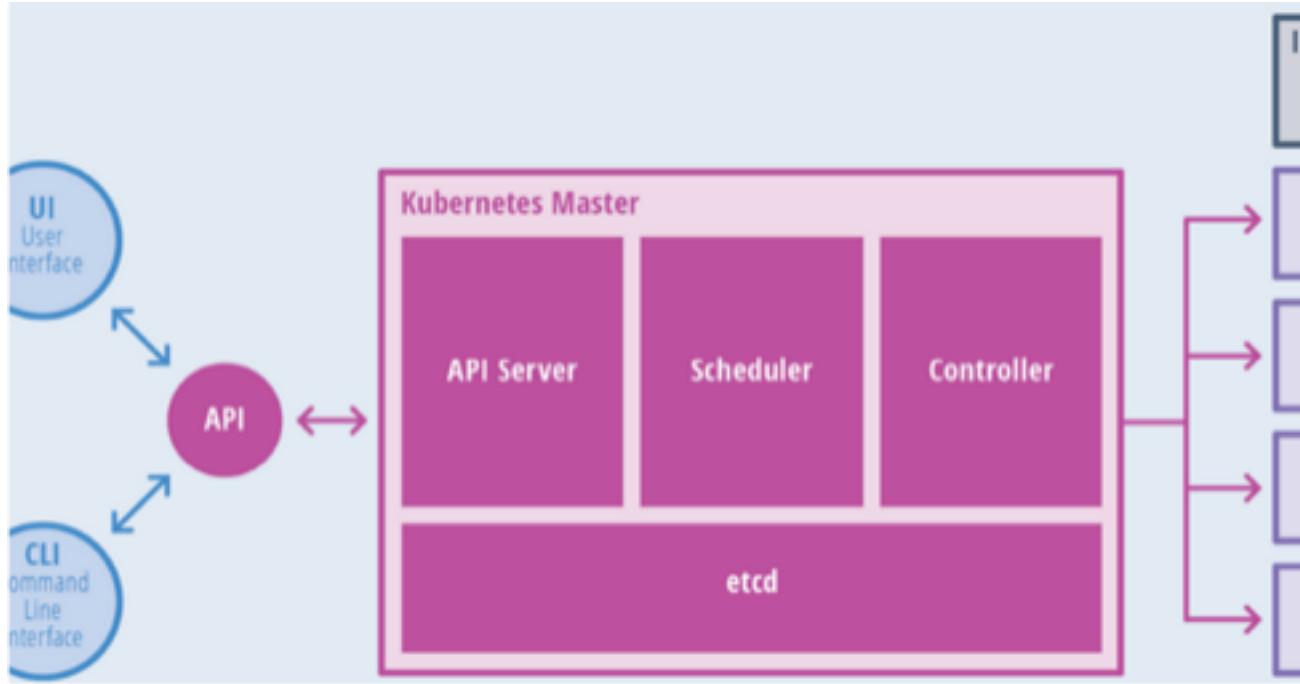
Kubernetes Architecture



K8s Architecture - detailed



Kubernetes Master



Components of master:

- API Server
- Scheduler
- Controller
- etcd

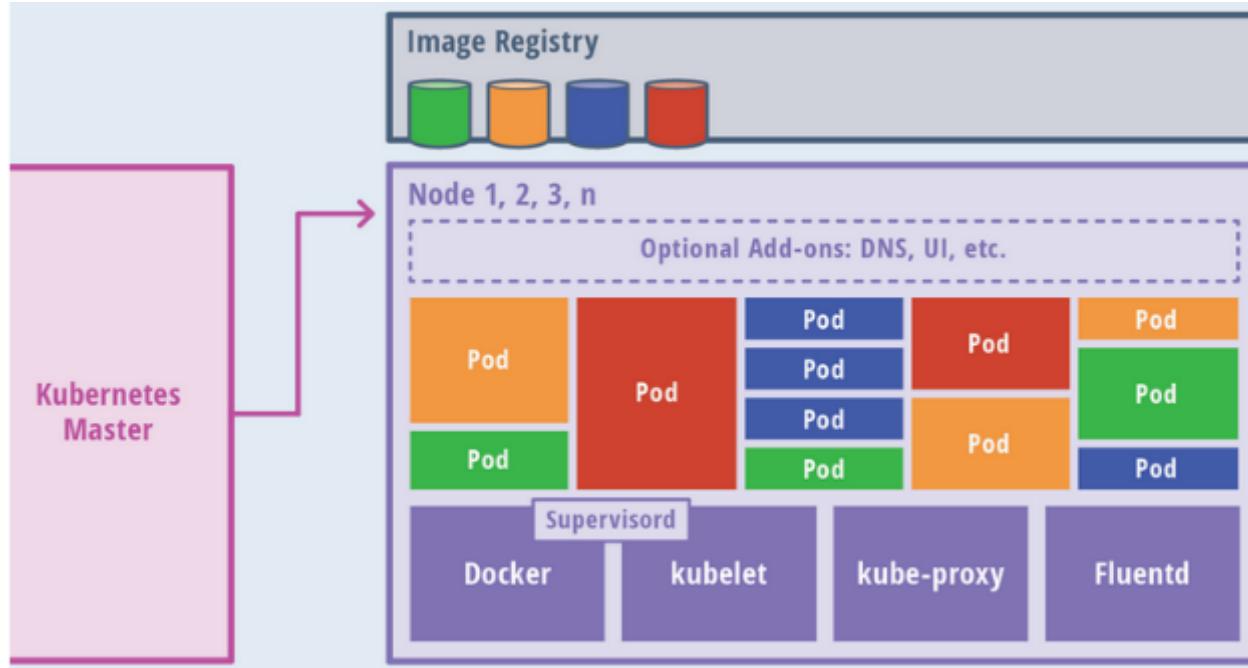
Kubernetes Master

- **The API server** is the entry points for all the REST commands used to control the cluster. It processes REST requests, validates them, and executes the bound business logic. The result state has to be persisted in the “etcd” component.
- **Etcd** is an open source, distributed key-value database; it acts as a single source of truth (SSOT) for all components of the Kubernetes cluster. Masters query etcd to retrieve various parameters of the state of the nodes, pods and containers. Etcd is considered a metadata service in Kubernetes.

Kubernetes Master

- **Controller Manager** is responsible for most of the collectors that regulate the state of the cluster. In general, a controller can be considered a daemon that runs in nonterminating loop and is responsible for collecting and sending information to the API server. It works toward getting the shared state of cluster and then making changes to bring the current status of the server to the desired state. The key controllers are **replication controller**, **endpoint controller**, **namespace controller**, and **service account controller**. The controller manager runs different kind of controllers to handle nodes, endpoints, etc.
- **Scheduler** is one of the key components of Kubernetes master. It is responsible for distributing the workload, tracking resource utilization on cluster nodes and selecting the nodes for the workloads to run. In other words, this is the mechanism responsible for allocating pods to available nodes.

Kubernetes Nodes



Components of a node:

- kubelet
- Kube-proxy
- Docker
- Fluentd

Kubernetes Node - Kubelet

- **Kubelet Service** on each node is responsible for relaying information to and from the control plane service. It interacts with etcd store to read configuration details and to write values **via api-server**. It communicates with the master component to receive commands and work. The kubelet process then assumes responsibility for maintaining the state of work and the node server. It manages network rules, port forwarding, etc.

Kubernetes Node - Kubeproxy

- **Kubernetes Proxy Service** is a proxy service which runs on each node and helps in making services available to the external host. It helps in forwarding the request to correct containers and is capable of performing primitive load balancing. It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well. It manages pods on node, volumes, secrets, creating new containers' health checkup, etc.

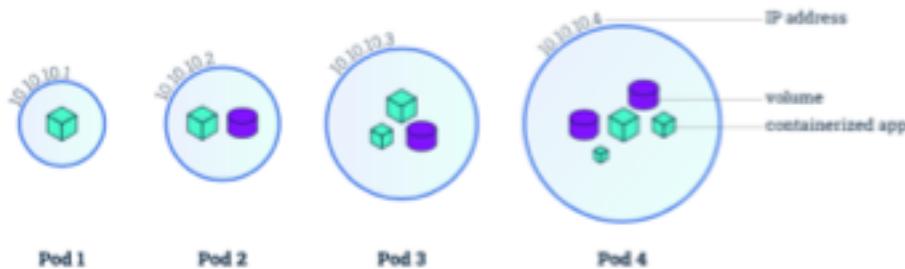
Kubernetes Installation

https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/kubernetes_single_node_cluster_installation.md

Kubernetes in bare metal (Just 1 master and is configured to use master to deploy pods)

Kubernetes: Pods

- The smallest unit that can be scheduled to be deployed through K8s is called a *pod*.
- Homogeneous group of containers.
- This group of containers would share storage, Linux namespaces, cgroups, IP addresses. These are co-located, hence share resources and are always scheduled together.
- Pods are not intended to live long. They are created, destroyed and re-created on demand, based on the state of the server and the service itself.



- Containers in a pod share
 - IP address and port space
 - Filesystem
 - Storage (Volumes)
 - Labels
 - Secrets

pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-apparmor
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

```
[root@ip-172-31-68-96:~/code# kubectl create -f pod.yml
pod/nginx-apparmor created
[root@ip-172-31-68-96:~/code# kubectl get pods
NAME          READY     STATUS    RESTARTS   AGE
nginx-apparmor  1/1      Running   0          4s
root@ip-172-31-68-96:~/code# ]
```

Note: This is just creation of the pod. In order to expose it, a service must be created. In order for it to be respawned, a deployment or replication control must be created.

Kubernetes: Deployment

Abstraction that use Replication controller (Controller manager of kubernetes master) to manage replica-sets of pods

- If object {pod} is used, and it dies, it will not start again
- If object {deployment} is used to start pods, if the pod dies, deployment use replica-set to start the pods again to make sure desired number of pods are always alive
- Try deleting a pod (deployed using deployment) and check if it comes back?

dep.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

```
root@ip-172-31-68-96:/code# kubectl create -f dep.yml
deployment.apps/nginx-deployment created
root@ip-172-31-68-96:/code# kubectl get pods,deployments
NAME                                         READY   STATUS    RESTARTS   AGE
pod/nginx-apparmor                           1/1     Running   0          52s
pod/nginx-deployment-67594d6bf6-7jc78       1/1     Running   0          11s
pod/nginx-deployment-67594d6bf6-tf5df        1/1     Running   0          11s
pod/nginx-deployment-67594d6bf6-xzwww        1/1     Running   0          11s

NAME                               DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
deployment.extensions/nginx-deployment   3         3         3           3          11s
root@ip-172-31-68-96:/code# █
```

Kubernetes: Services

Abstraction regarding a set of pods which enables load balancing, traffic exposure, load balancing and service discovery.

- Abstraction that allows pods to die and replicate in Kubernetes without affecting your application.
- Enable loose coupling between different Pods.
- Provides a stable virtual IP and port
- Services allow Pods to receive traffic.
 - Each Pod has a unique IP but those IP addresses are not exposed outside the Pod without a service.

Kubernetes: Types of Services

- **Cluster IP**:- Expose the service on a cluster-internal IP. Using this makes the service only reachable from within the cluster.
- **NodePort** :- Expose the service on each Node's IP at a static port. You will be able to contact the NodePort Service, from outside the cluster by <NodeIP>:<NodePort>
- **LoadBalancer** :- Expose the service externally using load balancer

```
root@ip-172-31-68-96:~# kubectl get svc
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP  10.96.0.1    <none>        443/TCP   37m
root@ip-172-31-68-96:~#
root@ip-172-31-68-96:~# kubectl expose deployment nginx-deployment --type=LoadBalancer --name=nginx-service
service/nginx-service exposed
root@ip-172-31-68-96:~#
root@ip-172-31-68-96:~# kubectl get svc
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP  10.96.0.1    <none>        443/TCP   37m
nginx-service LoadBalancer  10.102.89.83  <pending>    80:31095/TCP 3s
root@ip-172-31-68-96:~#
```

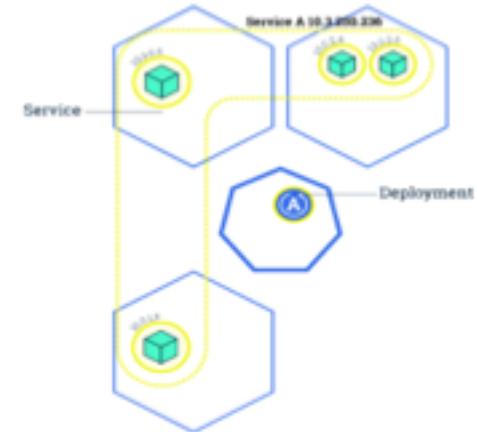
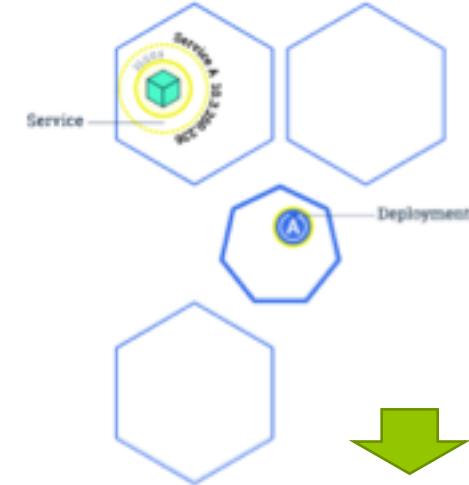
CronJob Example

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              args:
                - /bin/sh
                - --c
                - date; echo Hello from the Kubernetes cluster
        restartPolicy: OnFailure
```

Kubernetes: Scaling

Changing the number of resources to meet a desired state

- Accomplished by adjusting the number of replicas in a deployment
- Accommodates both scaling up and scaling down to a minimum of 0
- Traffic is automatically sent to newly created instances through the service load balancer
- Can be used to enable rolling updates without downtime



```

root@ip-172-31-0-112:~/code# kubectl run nginx --image nginx --port 80
deployment.apps/nginx created
root@ip-172-31-0-112:~/code#
root@ip-172-31-0-112:~/code# kubectl get deployments
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx         1          1          1           1           8s
nginx-deployment 3          3          3           3           17m
redis          1          1          1           1           33m
root@ip-172-31-0-112:~/code#
root@ip-172-31-0-112:~/code# kubectl scale deployments/nginx --replicas=3
deployment.extensions/nginx scaled
[root@ip-172-31-0-112:~/code# kubectl get deployments
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx         3          3          3           3           34s
nginx-deployment 3          3          3           3           18m
redis          1          1          1           1           33m
[root@ip-172-31-0-112:~/code# kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-6f858d4d45-b4cv2            1/1     Running   0          42s
nginx-6f858d4d45-g42mg            1/1     Running   0          16s
nginx-6f858d4d45-169jv            1/1     Running   0          16s
nginx-apparmor                     1/1     Running   0          30m
nginx-deployment-67594d6bf6-7v5lk  1/1     Running   0          12m
nginx-deployment-67594d6bf6-fdqx2  1/1     Running   0          18m
nginx-deployment-67594d6bf6-sx62p  1/1     Running   0          18m
redis-7869f8966-sq8xm             1/1     Running   0          33m

```

Kubernetes: AutoScaling

HPA – Horizontal Pod Autoscaler:

Based on memory and CPU utilization by a pod, autoscaling scales up or down the number of pods

```
kubectl autoscale deployment nginx --min=2 --max=10
```

```
kubectl autoscale deployment nginx-deployment --max=5 --cpu-percent=80
```

```
kubectl get hpa
```

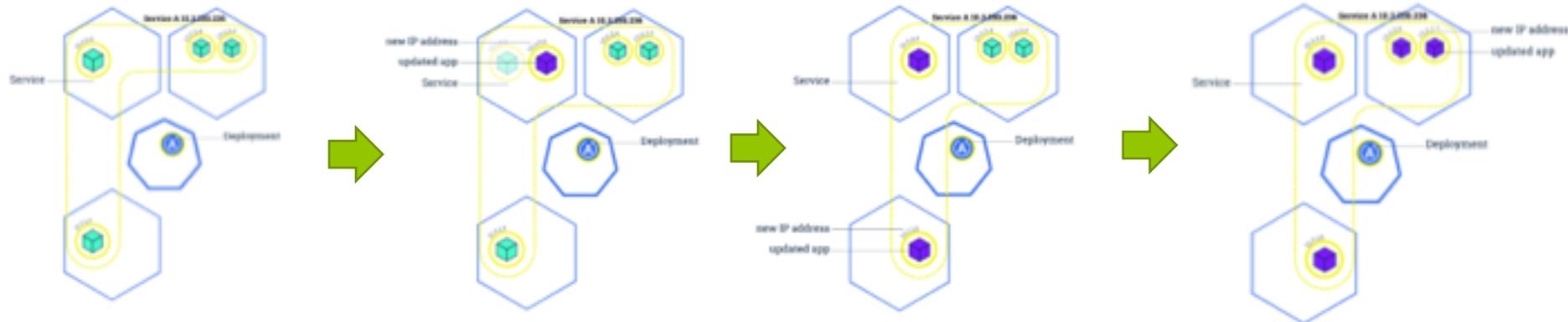
```
kubectl get hpa
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS
AGE					
nginx	Deployment/nginx	<unknown>/80%	1	4	3
nginx-deployment	Deployment/nginx-deployment	<unknown>/80%	2	5	0
	5s				

Kubernetes: Rolling Update

Updates the pods in a serial manner will load balancing traffic to available instances

- If deployment is exposed publicly, the service will load-balance traffic to only active and available pods
- Used to enable zero downtime updates
- Allows greater application update frequency
- Can also be leveraged to rollback to previous versions



```
root@ip-172-31-0-112:~/code# kubectl run nginx --image nginx:1.12.1 --port 80
deployment.apps/nginx created
root@ip-172-31-0-112:~/code# kubectl set image deployments/nginx nginx=nginx:latest
deployment.extensions/nginx image updated
root@ip-172-31-0-112:~/code# kubectl describe deployment nginx
Name:                  nginx
Namespace:             default
CreationTimestamp:     Thu, 19 Jul 2018 04:00:55 +0000
Labels:                run=nginx
Annotations:           deployment.kubernetes.io/revision=2
Selector:              run=nginx
Replicas:              1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:          RollingUpdate
MinReadySeconds:       0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  run=nginx
  Containers:
    nginx:
      Image:      nginx:latest
      Port:       80/TCP
      Host Port:  0/TCP
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
  Conditions:
    Type        Status  Reason
    ----        ----   -----
    Available   True    MinimumReplicasAvailable
    Progressing True    NewReplicaSetAvailable
  OldReplicaSets: <none>
  NewReplicaSet:  nginx-6c486b77db (1/1 replicas created)
Events:
  Type        Reason          Age        From               Message
  ----        ----   -----   ----   -----
  Normal     ScalingReplicaSet  43s       deployment-controller  Scaled up replica set nginx-7f9bc86464 to 1
  Normal     ScalingReplicaSet  11s       deployment-controller  Scaled up replica set nginx-6c486b77db to 1
  Normal     ScalingReplicaSet  10s       deployment-controller  Scaled down replica set nginx-7f9bc86464 to 0
  Normal     ScalingReplicaSet  9ms      deployment-controller  Scaled down replica set nginx-6c486b77db to 0
```

```

root@ip-172-31-0-112:~/code# kubectl rollout undo deployments/nginx
deployment.extensions/nginx
root@ip-172-31-0-112:~/code# kubectl describe deployments/nginx
Name:           nginx
Namespace:      default
CreationTimestamp: Thu, 19 Jul 2018 04:00:55 +0000
Labels:         run=nginx
Annotations:   deployment.kubernetes.io/revision=3
Selector:       run=nginx
Replicas:      1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:  RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  run=nginx
  Containers:
    nginx:
      Image:      nginx:1.12.1
      Port:       80/TCP
      Host Port:  0/TCP
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
  Conditions:
    Type        Status  Reason
    ----        ----  -----
    Available   True    MinimumReplicasAvailable
    Progressing True    NewReplicaSetAvailable
    OldReplicaSets: <none>
    NewReplicaSet:  nginx-7f9bc86464 (1/1 replicas created)
  Events:
    Type        Reason          Age            From           Message
    ----        ----          ----  -----
    Normal     ScalingReplicaSet  1m             deployment-controller  Scaled up replica set nginx-6c486b77db to 1
    Normal     ScalingReplicaSet  1m             deployment-controller  Scaled down replica set nginx-7f9bc86464 to 0
    Normal     ScalingReplicaSet  18s (x2 over 2m) deployment-controller  Scaled up replica set nginx-7f9bc86464 to 1
    Normal     DeploymentRollback 18s            deployment-controller  Rolled back deployment "nginx" to revision 1
    Normal     ScalingReplicaSet  17s            deployment-controller  Scaled down replica set nginx-6c486b77db to 0

```

Note: Rollbacks can also be enacted with zero-downtime

kubectl rollout status deployment nginx

kubectl rollout history deployment nginx

kubectl rollout history deployment/nginx --revision=3

kubectl rollout undo deployment/nginx --to-revision <num>

Kubernetes: Edit

Edit the configuration of a running entity

```
[root@ip-172-31-68-96:~# kubectl get deploy
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx         4          4          4           4           28m
nginx-deployment 2          2          2           2           37m
[root@ip-172-31-68-96:~# kubectl edit deploy/nginx
deployment.extensions/nginx edited
[root@ip-172-31-68-96:~# kubectl get deploy
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx         2          2          2           2           28m
nginx-deployment 2          2          2           2           38m
root@ip-172-31-68-96:~# ]
```

Kubernetes: Pod Priority

Set the priority for pod scheduling, cannot ask never to kill a pod

Example PriorityClass

```
apiVersion: scheduling.k8s.io/v1alpha1
kind: PriorityClass
metadata:
  name: high-priority
  value: 1000000
  globalDefault: false
  description: "This priority class should be used for XYZ service pods only."
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  priorityClassName: high-priority
```

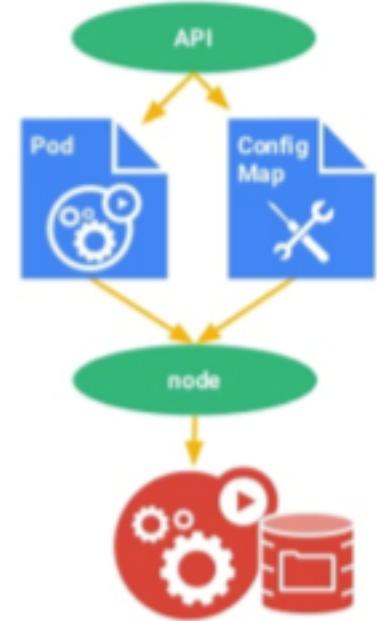
Kubernetes: Config Map

A set of key-value pairs that serve as configuration data for pods. They solve the problem of how to pass config data such as environment variables to pods.

Config maps are commonly composed of:

- Command line arguments
- Environment variables
- Files in a volume

Config maps function similar to secrets, but values are stored as strings and are more readily readable.



Kubernetes: Config Map

```
[root@ip-172-31-68-96:~# kubectl create configmap test-config --from-literal=key1=config1 --from-literal=key2=config2
configmap/test-config created
[root@ip-172-31-68-96:~# kubectl get configmap
NAME        DATA      AGE
my-config   2         1m
test-config 2         7s
[root@ip-172-31-68-96:~#
[root@ip-172-31-68-96:~# kubectl describe configmap/test-config
Name:           test-config
Namespace:      default
Labels:         <none>
Annotations:    <none>

Data
-----
key1:
-----
config1
key2:
-----
config2
Events:  <none>
root@ip-172-31-68-96:~# ]
```

Kubernetes: Config Map

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-cm
  labels:
    name: test-cm
data:
  key1: value1
  key2: value2
```

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-cm
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: test-cm
              key: key1
  restartPolicy: Never
```

Kubernetes: Persistent Volumes

- PV: Storage in the cluster that has been provisioned by an administrator (static or dynamic) and is independent of any individual pod that uses the PV.

Following volumes are supported by Kubernetes:

GCEPersistentDisk ; AWSElasticBlockStore

AzureFile ; AzureDisk

FC (Fibre Channel) ; FlexVolume

Flocker ; NFS

iSCSI ; RBD (Ceph Block Device)

CephFS ; Cinder (OpenStack block storage)

Glusterfs ; VsphereVolume

Portworx Volumes ; ScaleIO Volumes

StorageOS

Kubernetes: Persistent Volumes

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

Kubernetes: Persistent Volume Claims

- PVC: Request for a storage by a user, it is similar to pod and it can request specific size and access modes (e.g., can be mounted once read/write or many times read-only).

https://github.com/shekhar2010us/kubernetes_teach_git/tree/master/ex6

Example

<https://github.com/dockersamples/example-voting-app>

kubectl create -f k8s-specifications/

UFT Pointers

Available UFT Docker images:

- functionaltesting/uft
- functionaltesting/leanft
- functionaltesting/leanft-chrome
- functionaltesting/leanft-firefox

LeanFT:

- Functional test automation solution enables automating applications of common technologies, including Web applications.
- Cross-browser and cross-platform solution that allow developers and dev-testers to create and execute functional tests on their chosen environment.

UFT Pointers

Run UFT tests in a Windows Docker container

- https://admhelp.microfocus.com/uft/en/14.50-14.53/UFT_Help/Content/User_Guide/docker.htm

Execute UFT Pro (LeanFT) tests in Docker

- <https://community.microfocus.com/t5/Application-Delivery-Management/Execute-your-UFT-Pro-LeanFT-tests-in-Docker/ba-p/1610193>

Run tests in Docker containers

- <https://admhelp.microfocus.com/uftdv/en/15.0/HelpCenter/Content/HowTo/Run-tests-in-docker.htm>

Run Selenium in Docker



CLASSROOM WORK

Run Selenium Test in Docker

https://github.com/shekhar2010us/aws_key/blob/master/docker/Run%20Selenium%20Test.md

eval(this.class);

- Please fill out our online course evaluation, which gives you immediate access to your certificate of completion.
 - All responses are confidential even though an email address is requested.
1. Go to: **www.metricsthatmatter.com/ASPE**
 2. From the drop down menu choose **Docker Containerization Boot Camp, site, instructor name, class start date**
 3. Enter email address, answer the questions, provide desired written feedback, and click submit.

Feedback Email

- If you haven't already, you will receive an email similar to the one here asking for your feedback on the course.
- This Evaluation will ask your feedback on many aspects of your training, including the course materials, instructor facilitation and labs where applicable.
- We appreciate you completing this survey to help us continue to better our content and deliveries in order to serve you better.

We'd Love to Have Your Feedback!

ASPE
LEARN. TRANSFORM. SUCCEED.

**Thank You for Coming to ASPE for Your Training Needs.
We'd Love to Have Your Feedback!**

Thank you for attending training with ASPE. Your feedback is very important to us. Please complete our online course evaluation by following the link below.

[Complete Evaluation](#)

Working toward certification with PMI, Scrum Alliance, or the IIBA?

Need to report your Continuing Education Units? You can access your own Certificate of Completion for download as a PDF after completing the survey.



114 Edinburgh South Dr., Suite 200
Cary, NC 27511
Toll Free: 877-800-5221 | Fax: 919-816-1710

Questions

