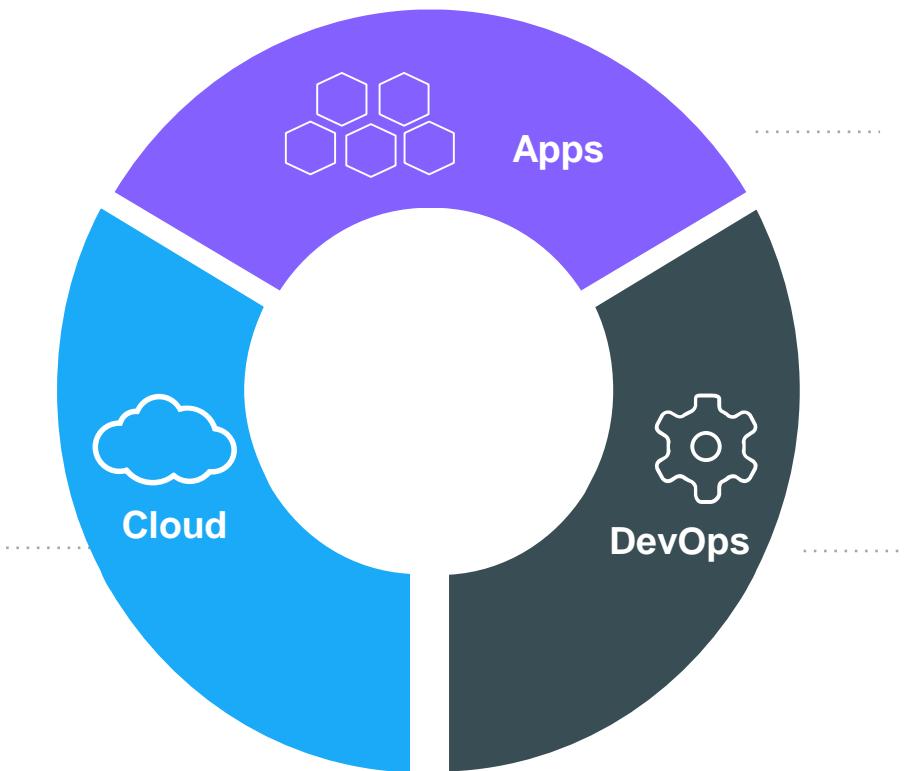


Introduction to Docker



The IT Landscape is Changing



Movement in the cloud



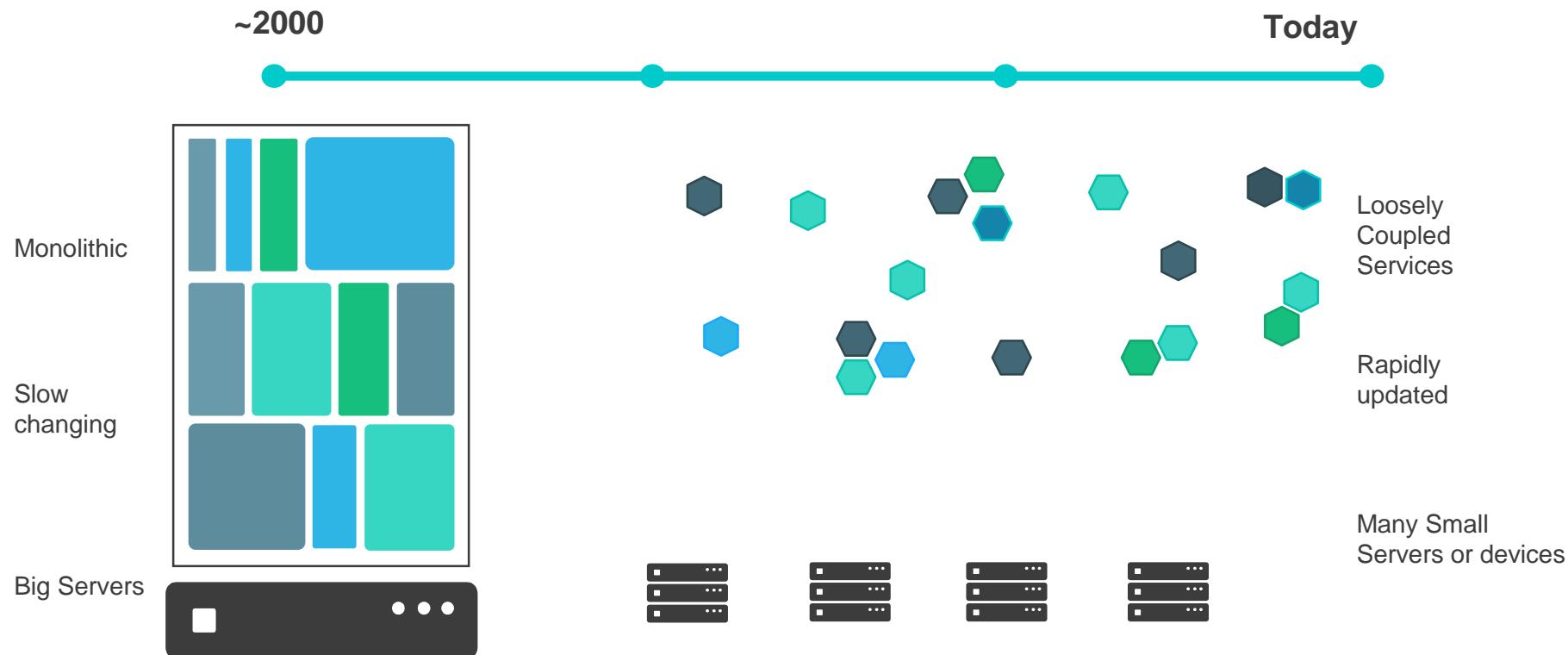
80%

Migrate workloads to cloud

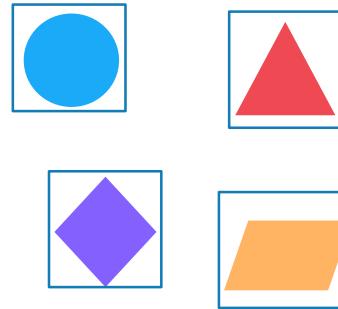
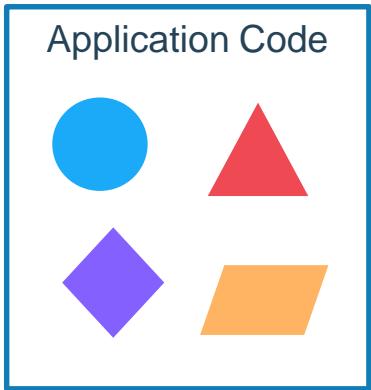
Portability across environments

Want to avoid cloud vendor lock-in

Applications are transforming



Application Modernization



Developer Issues:

- Minor code changes require full re-compile and re-test
- Application becomes single point of failure
- Application is difficult to scale

Microservices: Break application into separate operations

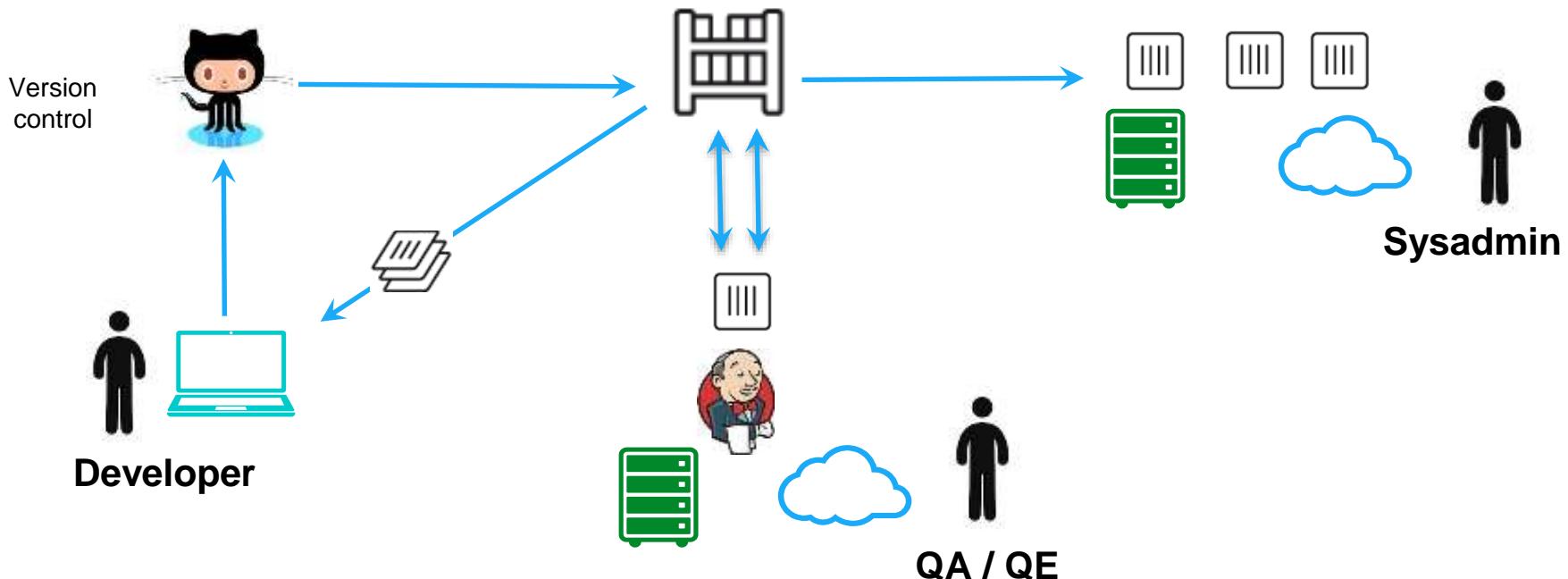
12-Factor Apps: Make the app independently scalable, stateless, highly available by design

Continuous Integration and Delivery

1. Development

2. Test

3. Stage / Production



Tug of War Between Developers and Ops



Developers

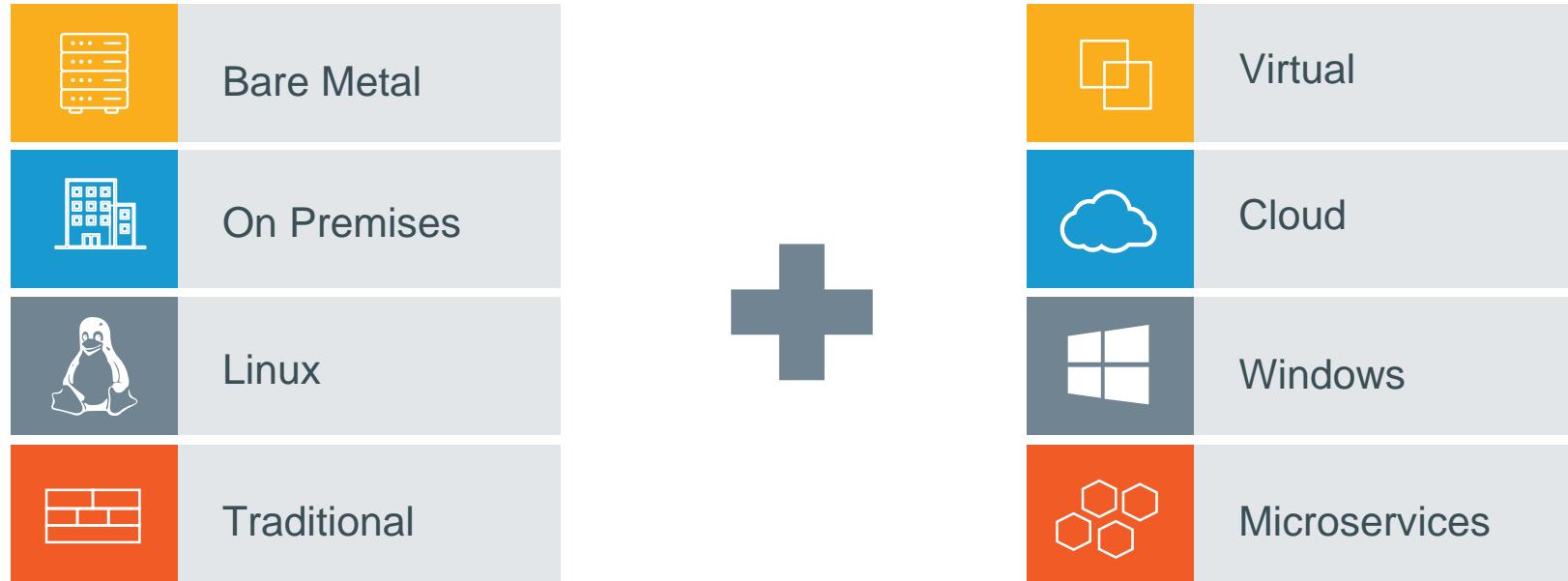
- Freedom to create and deploy apps fast
- Define and package application needs



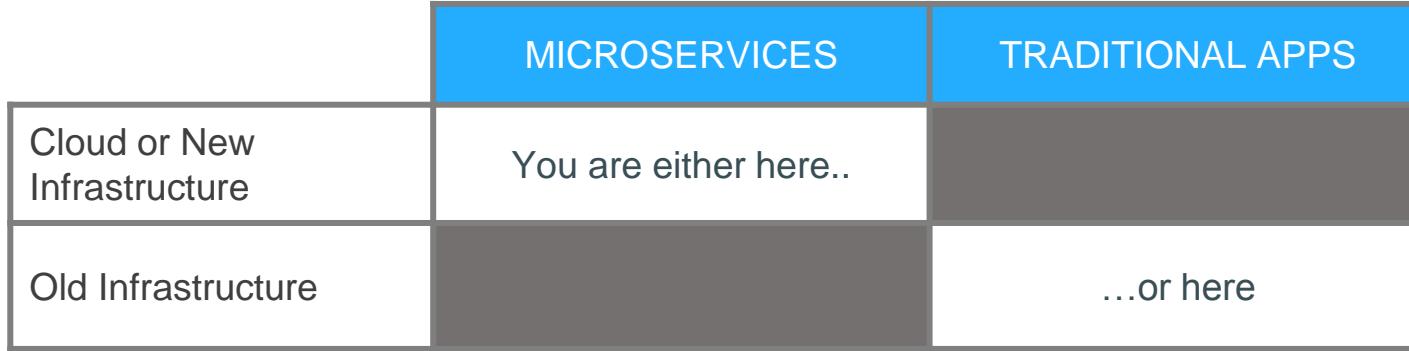
IT Operations

- Quickly and flexibly respond to changing needs
- Standardize, secure, and manage

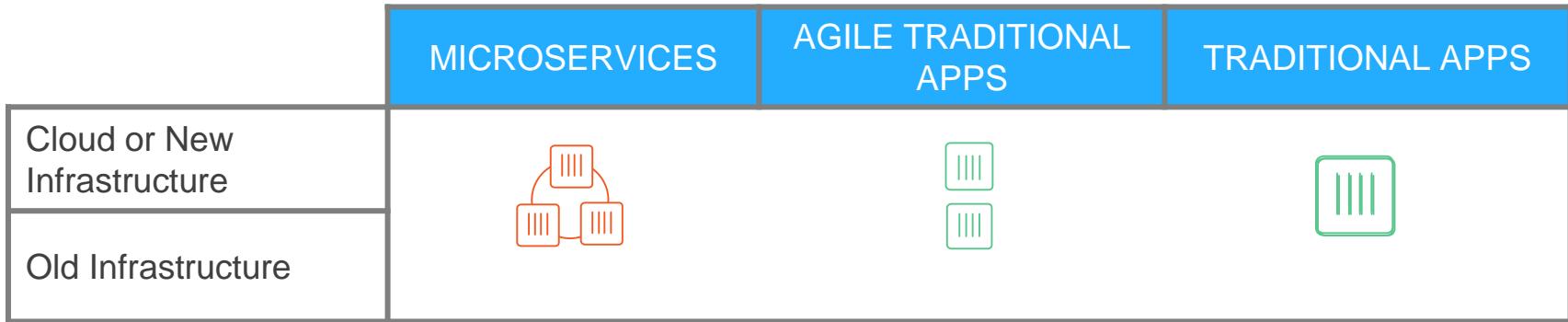
Organizations Must Deal with Diverse Technology



The Myth of Bi-Modal IT



Enabling a Journey

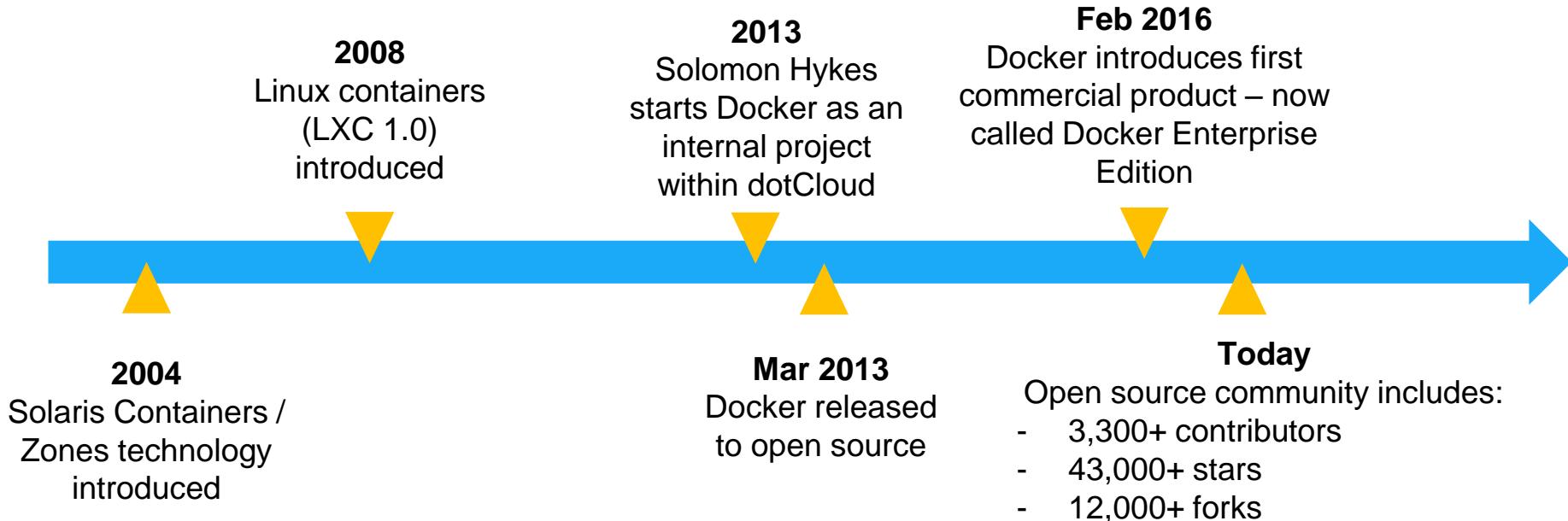


...that is past AND future proof

Docker and Container Overview



History of Docker

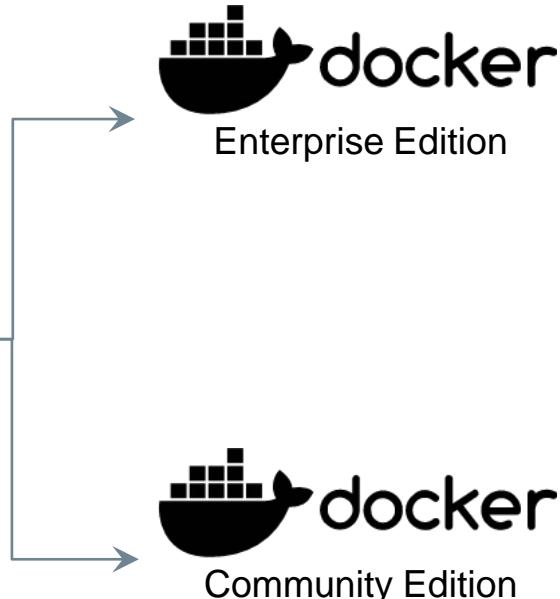


The Docker Family Tree



Open source **framework** for assembling core components that make a container platform

Intended for:
Open source contributors + ecosystem developers



Subscription-based, commercially supported **products** for delivering a secure software supply chain

Intended for:
Production deployments + Enterprise customers



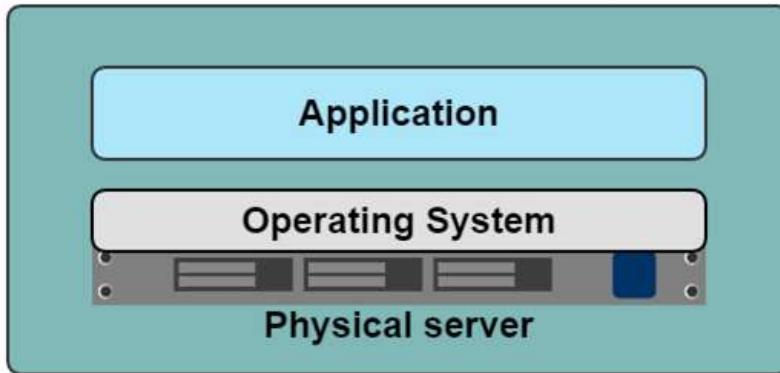
Free, community-supported **product** for delivering a container solution

Intended for:
Software dev & test

A History Lesson

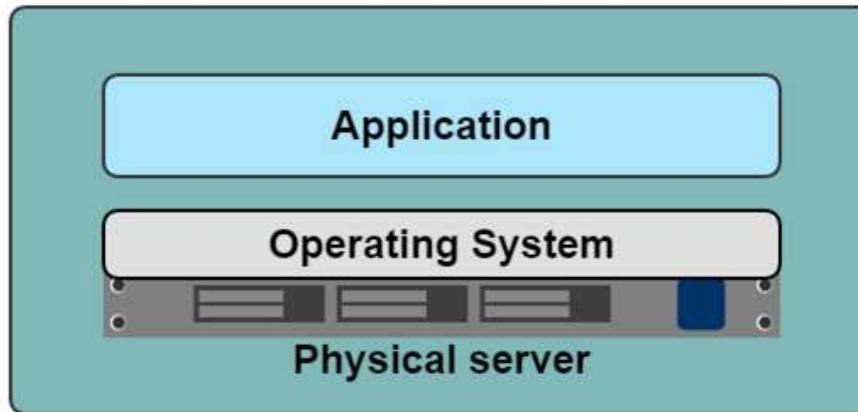
In the Dark Ages

One application on one physical server



Historical limitations of application deployment

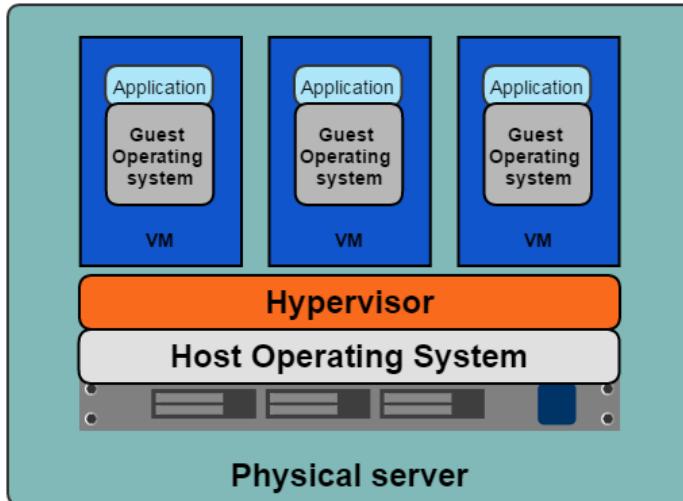
- Slow deployment times
- Huge costs
- Wasted resources
- Difficult to scale
- Difficult to migrate
- Vendor lock in



A History Lesson

Hypervisor-based Virtualization

- One physical server can contain multiple applications
- Each application runs in a virtual machine (VM)



Benefits of VMs

- Better resource pooling
 - One physical machine divided into multiple virtual machines
- Easier to scale
- VMs in the cloud
 - Rapid elasticity
 - Pay as you go model

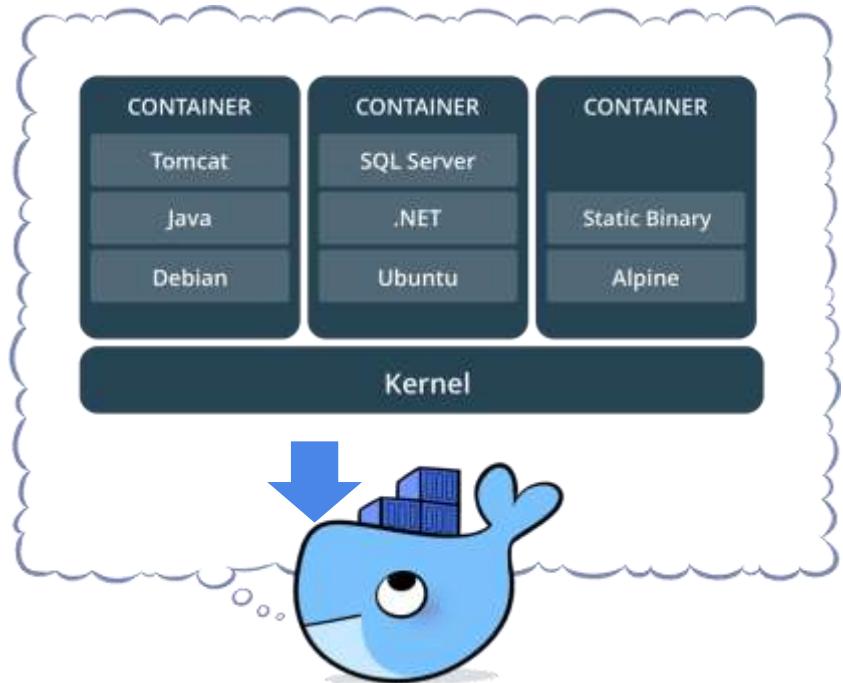


Limitations of VMs

- Each VM stills requires
 - CPU allocation
 - Storage
 - RAM
 - An entire guest operating system
- The more VMs you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed



What is a container?

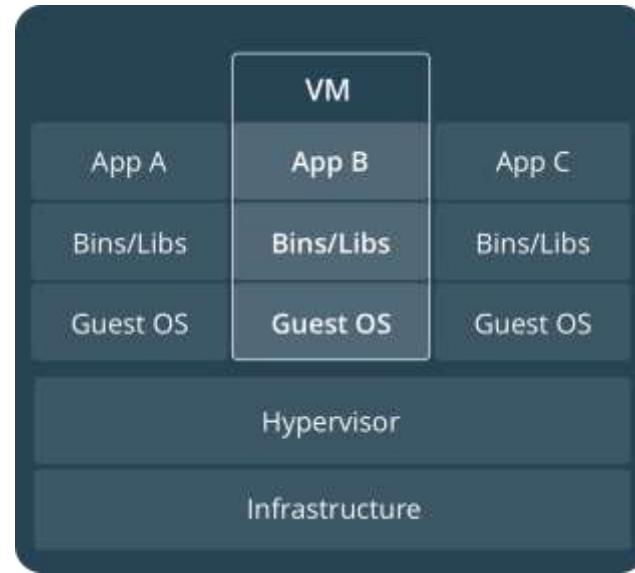


- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works with all major Linux and Windows Server

Comparing Containers and VMs

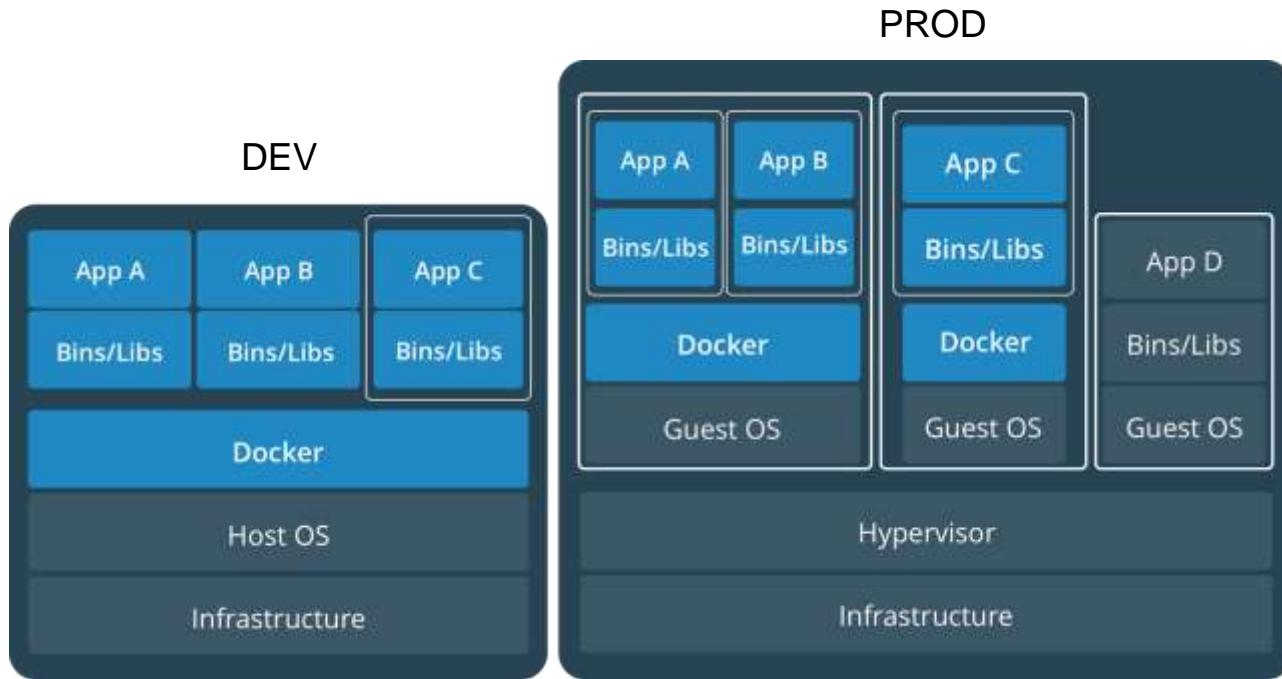


Containers are an app level construct



VMs are an infrastructure level construct to turn one machine into many servers

Containers and VMs together



Containers and VMs together provide a tremendous amount of flexibility for IT to optimally deploy and manage apps.

Key Benefits of Docker Containers

Speed

- No OS to boot = applications online in seconds

Portability

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

Efficiency

- Less OS overhead
- Improved VM density

Container Solutions & Landscape



Docker Basics



Image

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



Container

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



Engine

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



Registry

Stores, distributes and manages Docker images



Control Plane

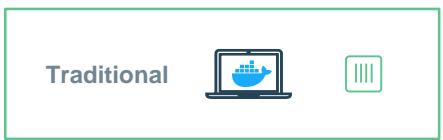
Management plane for container and cluster orchestration

Foundation: Docker Engine

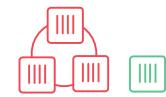
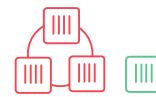


Docker Engine

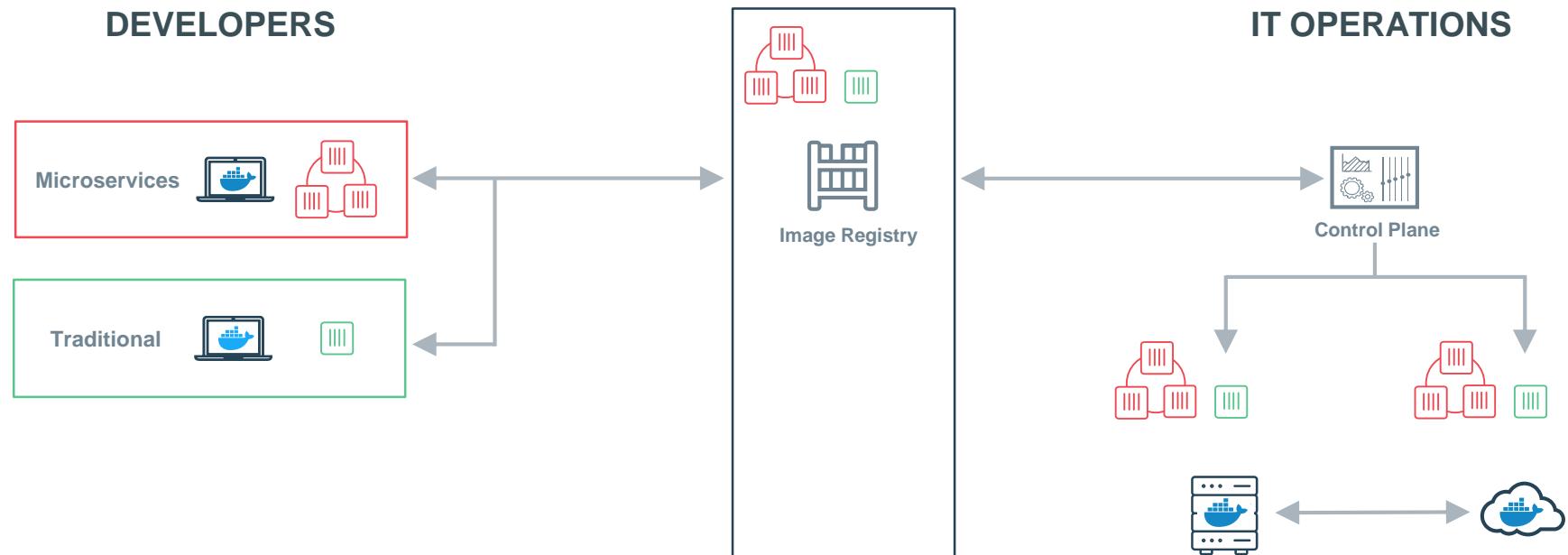
DEVELOPERS



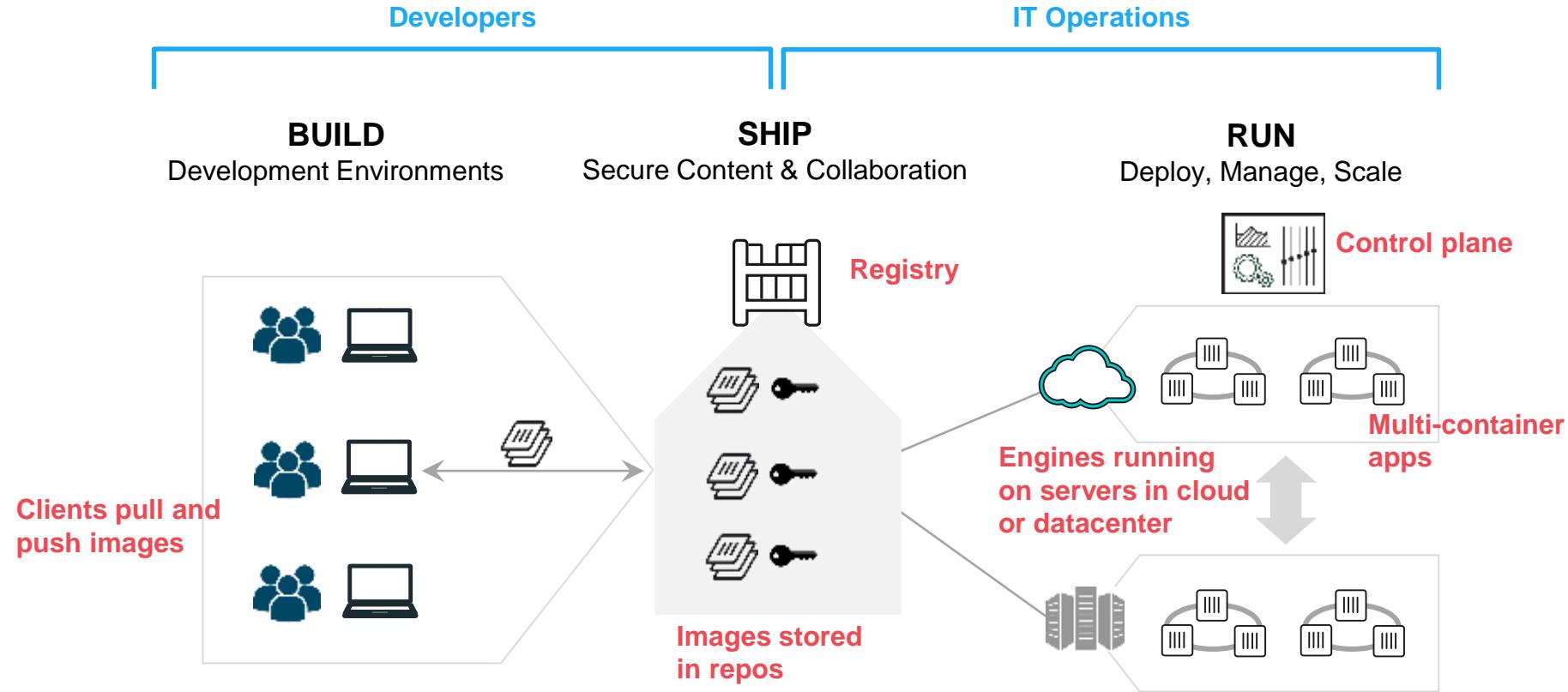
IT OPERATIONS



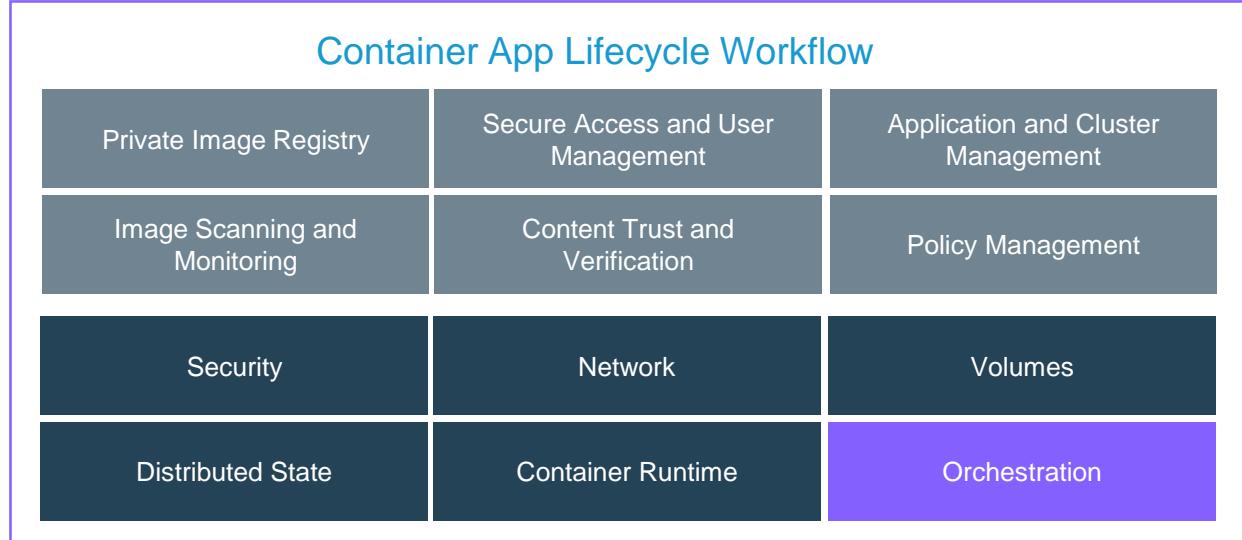
Building a Software Supply Chain



Containers as a Service



Building a Secure Supply Chain



Enterprise Edition



Docker Engine



Usable
Security



Trusted
Delivery



Portable

Basic Docker Commands

```
$ docker pull mikegcoleman/catweb:latest  
$ docker images  
$ docker run -d -p 5000:5000 --name catweb mikegcoleman/catweb:latest  
$ docker ps  
$ docker stop catweb (or <container id>)  
$ docker rm catweb (or <container id>)  
$ docker rmi mikegcoleman/catweb:latest (or <image id>)
```

How to find my Docker version?

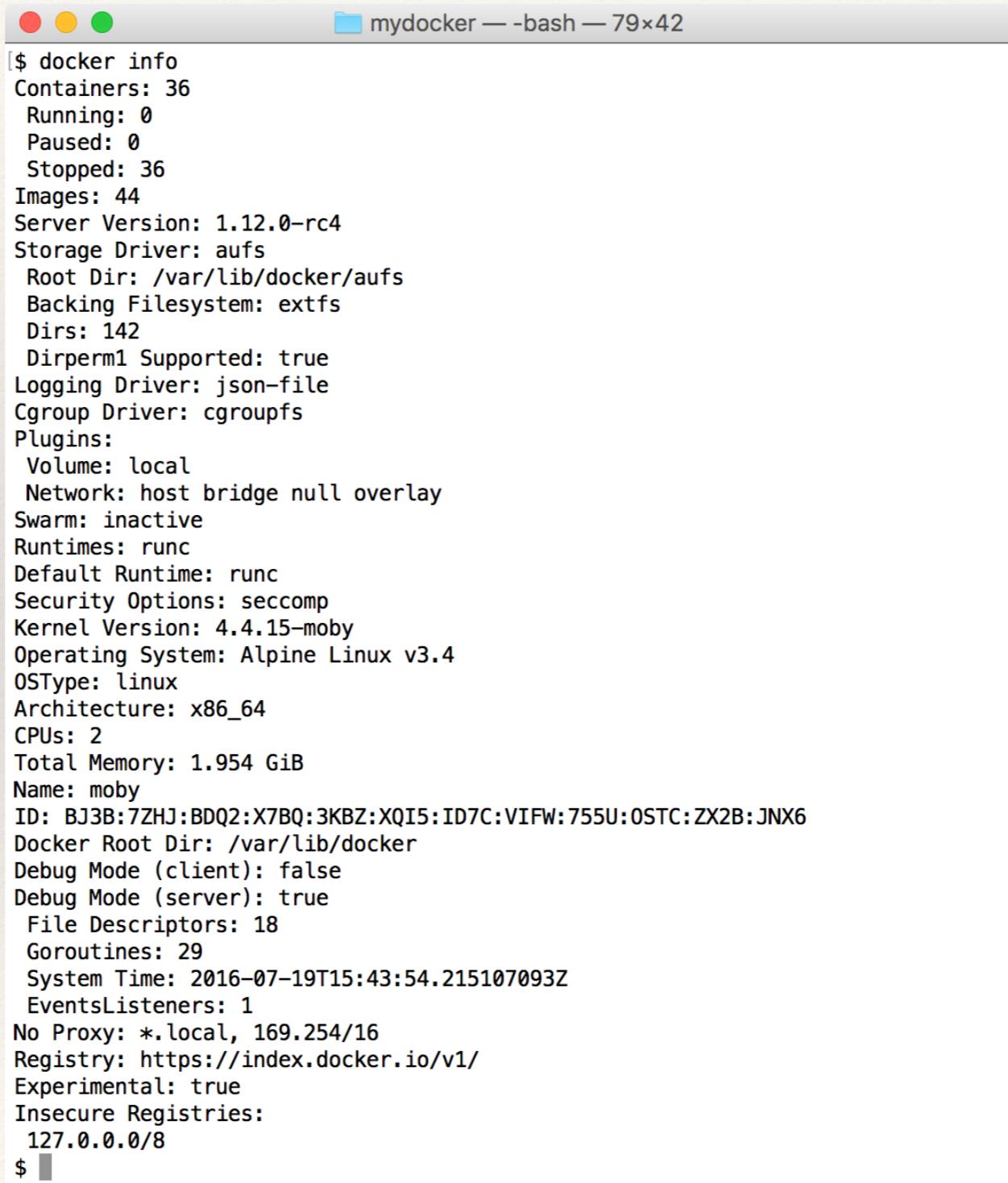
```
mydocker — -bash — 47x19
$ docker version
Client:
  Version:      1.12.0-rc4
  API version:  1.24
  Go version:   go1.6.2
  Git commit:   e4a0dbc
  Built:        Wed Jul 13 03:28:51 2016
  OS/Arch:      darwin/amd64
  Experimental: true

Server:
  Version:      1.12.0-rc4
  API version:  1.24
  Go version:   go1.6.2
  Git commit:   e4a0dbc
  Built:        Wed Jul 13 03:28:51 2016
  OS/Arch:      linux/amd64
  Experimental: true
$
```

How to find my Docker version?

```
$ docker -v  
Docker version 1.12.0-rc4, build e4a0dbc, experimental
```

How to find details of my Docker installation?



The screenshot shows a terminal window titled "mydocker — -bash — 79x42". The window contains the output of the "docker info" command. The output provides detailed information about the Docker daemon's configuration and environment. Key details include:

- Containers: 36 (Running: 0, Paused: 0, Stopped: 36)
- Images: 44
- Server Version: 1.12.0-rc4
- Storage Driver: aufs (Root Dir: /var/lib/docker/aufs, Backing Filesystem: extfs, Dirs: 142, Dirperm1 Supported: true)
- Logging Driver: json-file
- Cgroup Driver: cgroupfs
- Plugins:
 - Volume: local
 - Network: host bridge null overlay
- Swarm: inactive
- Runtimes: runc
- Default Runtime: runc
- Security Options: seccomp
- Kernel Version: 4.4.15-moby
- Operating System: Alpine Linux v3.4
- OSType: linux
- Architecture: x86_64
- CPUs: 2
- Total Memory: 1.954 GiB
- Name: moby
- ID: BJ3B:7ZHJ:BDQ2:X7BQ:3KBZ:XQI5:ID7C:VIFW:755U:0STC:ZX2B:JNX6
- Docker Root Dir: /var/lib/docker
- Debug Mode (client): false
- Debug Mode (server): true
 - File Descriptors: 18
 - Goroutines: 29
 - System Time: 2016-07-19T15:43:54.215107093Z
 - EventsListeners: 1
- No Proxy: *.local, 169.254/16
- Registry: https://index.docker.io/v1/
- Experimental: true
- Insecure Registries:
 - 127.0.0.0/8

\$

Can I install Docker from cmdline?

Yes! from get.docker.com

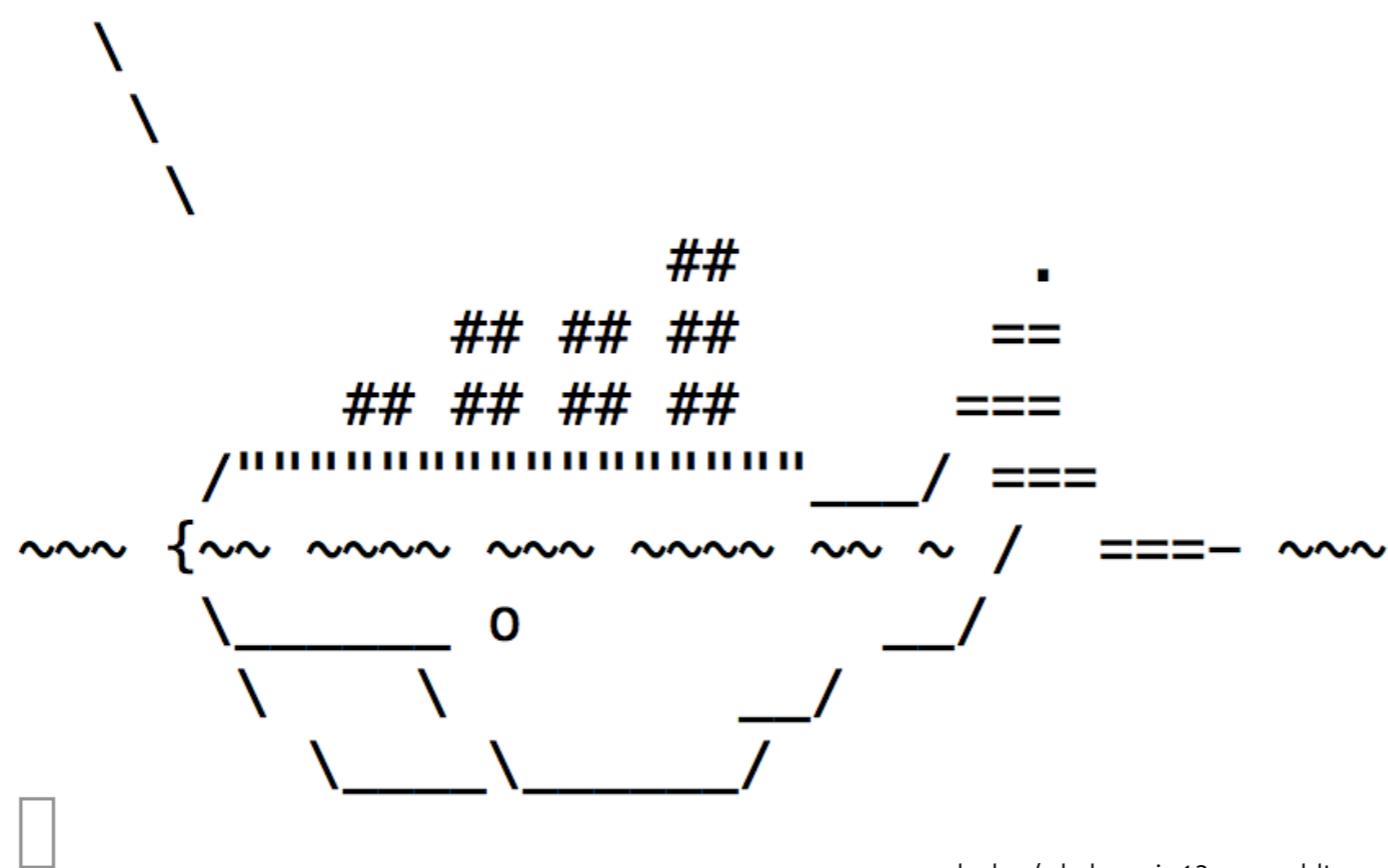
```
# This script is meant for quick & easy install via:  
# 'curl -sSL https://get.docker.com/ | sh'  
# or:  
# 'wget -qO- https://get.docker.com/ | sh'
```

How to do “hello world” in Docker?

```
$ docker run docker/whalesay cowsay Hello world
```

```
[ $ docker run docker/whalesay cowsay Hello world
```

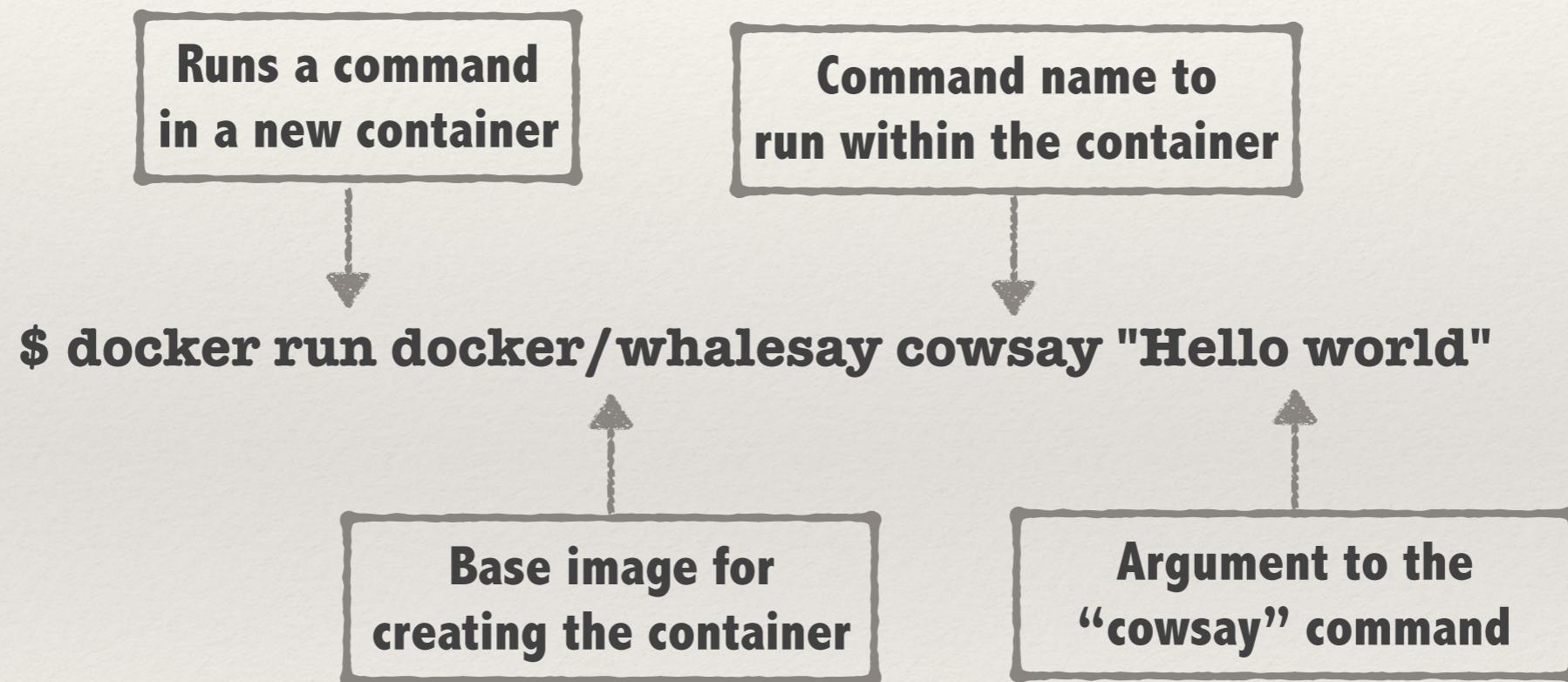
```
< Hello world >
```



```
$ █
```

docker/whalesay is 13 years old!
<https://hub.docker.com/r/docker/whalesay/>

How to do “hello world” in Docker?



How to do “hello world” in Docker?

\$ docker run -it hello-world

```
$ docker run -it hello-world
```

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/>

The term “Docker” can be confusing: the client and the daemon both are called Docker!

How to get help on commands to use?

Use “docker -h” command, as in:

```
$ docker -h
Usage: docker [OPTIONS] COMMAND [arg...]
      docker [ --help | -v | --version ]
```

A self-sufficient runtime for containers.

Options:

| | |
|----------------------|---------------------------------|
| --config=~/.docker | Location of client config files |
| -D, --debug | Enable debug mode |
| -H, --host=[] | Daemon socket(s) to connect to |
| -h, --help | Print usage |
| -l, --log-level=info | Set the logging level |

...

Commands:

| | |
|--------|-------------------------------|
| attach | Attach to a running container |
|--------|-------------------------------|

Docker commands look like Linux commands - so familiarity with Linux commands can really help to get up to speed quickly with Docker.

Docker Images

How to get list of images?

| \$ docker images | REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|------------------|---------------------------|--------|--------------|---------------|----------|
| | prasantk/average | latest | de756e5f8b96 | 10 days ago | 34.22 MB |
| | <none> | <none> | 3d02168f00fc | 10 days ago | 34.22 MB |
| | <none> | <none> | d3a03f4665ab | 10 days ago | 34.22 MB |
| | <none> | <none> | 827527375287 | 10 days ago | 34.22 MB |
| | <none> | <none> | 264584ac458e | 10 days ago | 745.6 MB |
| | python | 3.5 | 7fd24fb1b492 | 10 days ago | 686 MB |
| | example/docker-node-hello | latest | 92baf9b777b8 | 2 weeks ago | 658.6 MB |
| | centos | latest | 05188b417f30 | 2 weeks ago | 196.8 MB |
| | hello-world | latest | c54a2cc56cbb | 2 weeks ago | 1.848 kB |
| | ubuntu | latest | 0f192147631d | 2 weeks ago | 132.8 MB |
| | node | 0.10 | 8beaba2e26be | 3 weeks ago | 642.4 MB |
| | alpine | latest | 4e38e38c8ce0 | 3 weeks ago | 4.799 MB |
| | fedora | latest | f9873d530588 | 4 weeks ago | 204.4 MB |
| | nginx | latest | 0d409d33b27e | 6 weeks ago | 182.8 MB |
| | docker/whalesay | latest | 6b362a9f73eb | 14 months ago | 247 MB |
| | training/webapp | latest | 6fae60ef3446 | 14 months ago | 348.8 MB |
| | progrium/stress | latest | db646a8f4087 | 2 years ago | 281.8 MB |

How to search for an image?

```
$ docker search debian
```

| NAME | DESCRIPTION | STARS | OFFICIAL | AUTOMATED |
|------------------------------|---|-------|----------|-----------|
| debian | Debian is a Linux distribution that's comp... | 1503 | [OK] | |
| neurodebian | NeuroDebian provides neuroscience research... | 25 | [OK] | |
| armbuild/debian | ARMHF port of debian | 8 | | [OK] |
| jesselang/debian-vagrant | Stock Debian Images made Vagrant-friendly ... | 8 | | [OK] |
| eboraas/debian | Debian base images, for all currently-avai... | 5 | | [OK] |
| mschuerig/debian-subsonic | Subsonic 5.1 on Debian/wheezy. | 4 | | [OK] |
| reinblau/debian | Debian with usefully default packages for ... | 2 | | [OK] |
| webhippie/debian | Docker images for debian | 1 | | [OK] |
| datenbetrieb/debian | minor adaption of official upstream debian... | 1 | | [OK] |
| opennsm/debian | Lightly modified Debian images for OpenNSM | 1 | | [OK] |
| lucasbarros/debian | Basic image based on Debian | 1 | | [OK] |
| lephare/debian | Base debian images | 1 | | [OK] |
| eeacms/debian | Docker image for Debian to be used with EE... | 1 | | [OK] |
| maxexcloo/debian | Docker base image built on Debian with Sup... | 1 | | [OK] |
| servivum/debian | Debian Docker Base Image with Useful Tools | 1 | | [OK] |
| konstruktoid/debian | Debian base image | 0 | | [OK] |
| icedream/debian-jenkinsslave | Debian for Jenkins to be used as slaves. | 0 | | [OK] |
| visono/debian | Docker base image of debian 7 with tools i... | 0 | | [OK] |
| nimmis/debian | This is different version of Debian with a... | 0 | | [OK] |
| vcatechnology/debian | A Debian image that is updated daily | 0 | | [OK] |
| ustclug/debian | debian image for docker with rustic mirror | 0 | | [OK] |
| fike/debian | Debian Images with language locale installed. | 0 | | [OK] |
| pl31/debian | Debian base image. | 0 | | [OK] |
| mariorez/debian | Debian Containers for PHP Projects | 0 | | [OK] |
| gnumoksha/debian | [PT-BR] Imagem básica do Debian com ajust... | 0 | | [OK] |

```
$ □
```

How to get an image?

Use “docker pull <image_name>” command

```
$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
5c90d4a2d1a8: Already exists
Digest: sha256:8b1fc3a7a55c42e3445155b2f8f40c55de5f8bc8012992b26b570530c4bded9e
Status: Downloaded newer image for debian:latest
```

In my case debian image was already pulled. If it were not there, Docker would have pulled it afresh

How to get details of an image?

Use “docker inspect <image_name>” command

```
docker inspect debian
[
{
  "Id": "sha256:1b088884749bd93867ddb48ff404d4bbff09a17af8d95bc863efa5d133f87b78",
  "RepoTags": [
    "debian:latest"
  ],
  "RepoDigests": [
    "debian@sha256:8b1fc3a7a55c42e3445155b2f8f40c55de5f8bc8012992b26b570530c4bded9e"
  ],
  "Parent": "",
  "Comment": "",
  "Created": "2016-06-09T21:28:43.776404816Z",
  "Container": "2f3dc897cf758418389d50784c73b43b1fd7db09a80826329496f05eef7b377",
  "ContainerConfig": {
    "Hostname": "6250540837a8",
    "Domainname": "",
    "User": "",
    "AttachStdin": false,
    "AttachStdout": false,
    "AttachStderr": false,
    "Tty": false,
    "OpenStdin": false,
    // ...
  }
}
```

How to see “layers” in an image?

Use “docker history <image_name>” command

| IMAGE | CREATED | CREATED BY | SIZE | COMMENT |
|--------------|-------------|---|----------|---------|
| a0b516dc1799 | 5 weeks ago | /bin/sh -c set -x && dpkg-divert --divert /u | 8.657 MB | |
| <missing> | 5 weeks ago | /bin/sh -c echo '/usr/local/lib64' > /etc/ld. | 49.69 kB | |
| <missing> | 5 weeks ago | /bin/sh -c buildDeps='flex' && set -x && ap | 841.3 MB | |
| <missing> | 5 weeks ago | /bin/sh -c #(nop) ENV GCC_VERSION=6.1.0 | 0 B | |
| <missing> | 5 weeks ago | /bin/sh -c set -xe && for key in \$GPG_KEYS; | 140.8 kB | |
| <missing> | 5 weeks ago | /bin/sh -c #(nop) ENV GPG_KEYS=B215C1633BCA04 | 0 B | |
| <missing> | 6 weeks ago | /bin/sh -c apt-get update && apt-get install | 318.5 MB | |
| <missing> | 6 weeks ago | /bin/sh -c apt-get update && apt-get install | 131.2 MB | |
| <missing> | 6 weeks ago | /bin/sh -c apt-get update && apt-get install | 44.69 MB | |
| <missing> | 6 weeks ago | /bin/sh -c #(nop) CMD ["/bin/bash"] | 0 B | |
| <missing> | 6 weeks ago | /bin/sh -c #(nop) ADD file:76679eeb94129df23c | 125.1 MB | |

Each of these lines are layers and the size column shows the exact size of each layer in the image

How can I load and store images?

Use “docker save” and “docker load” commands

```
$ docker save nginx -o nginx.tar
$ ls -ltr nginx.tar
-rw----- 1 gsamarthyam staff 190775808 Jul 20 11:04 nginx.tar
$ docker load -i ./nginx.tar
Loaded image: nginx:latest
```

How do I delete an image?

Use “`docker rmi <image-tag>`”

```
$ docker images alpine
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
alpine          latest   4e38e38c8ce0  4 weeks ago  4.799 MB
$ docker rmi alpine
Untagged: alpine:latest
Untagged: alpine@sha256:3dcdb92d7432d56604d4545cbd324b14e647b313626d99b889d0626de158f73a
$ docker images alpine
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
```

How to delete all docker images?

Use “\$docker rmi \$(docker images -q)”

docker images -q
lists all image ids

How to find “dangling images”?

Use “`docker images -f "dangling=true"`”

| \$ docker images -f "dangling=true" | | | | |
|-------------------------------------|--------|--------------|---------------|----------|
| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
| <none> | <none> | 777f9424d24d | 7 minutes ago | 125.2 MB |
| <none> | <none> | 3d02168f00fc | 12 days ago | 34.22 MB |
| <none> | <none> | 0f192147631d | 3 weeks ago | 132.8 MB |

How to remove “dangling images”?

Use “`docker rmi $(docker images -f "dangling=true" -q)`”

How can I create my own Docker image?

Create “Dockerfile” - its like Makefile for Docker

```
$ cat myimage/Dockerfile
FROM ubuntu
RUN echo "my first image" > /tmp/first.txt
$ docker build myimage
Sending build context to Docker daemon 2.048 kB
Step 1 : FROM ubuntu
--> ac526a356ca4
Step 2 : RUN echo "my first image" > /tmp/first.txt
--> Running in 18f62f47d2c8
--> 777f9424d24d
Removing intermediate container 18f62f47d2c8
Successfully built 777f9424d24d
$ docker images | grep 777f9424d24d
<none>           <none>          777f9424d24d      4 minutes ago    125.2 MB
$ docker run -it 777f9424d24d
root@2dcd9d0caf6f:/# ls
bin  boot  core  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv
sys  tmp   usr   var
root@2dcd9d0caf6f:/# cat /tmp/first.txt
my first image
root@2dcd9d0caf6f:/# exit
exit
$
```

How to name/tag an image when building?

Use “`docker build <<dirname>> -t"imagename:tag"` command

```
$ docker build myimage -t"myfirstimage:latest"
Sending build context to Docker daemon 2.048 kB
Step 1 : FROM ubuntu
--> ac526a356ca4
Step 2 : RUN echo "my first image" > /tmp/first.txt
--> Using cache
--> 777f9424d24d
Successfully built 777f9424d24d
$ docker images myfirstimage
REPOSITORY          TAG           IMAGE ID      CREATED        SIZE
myfirstimage        latest        777f9424d24d   58 minutes ago  125.2 MB
$
```

Docker Containers

How to get list of containers?

| \$ docker ps -a | CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|-----------------|---------------|------------------|------------------------|-------------|------------------------|-------------------------|-----------------------|
| | 6322b8204a5d | ubuntu | "/bin/bash" | 6 days ago | Exited (0) 6 days ago | | desperate_aryabhata |
| | 1e95fcfd15893 | fedora | "/bin/bash" | 6 days ago | Exited (0) 6 days ago | | stoic_agnesi |
| | 73c773524c8a | nginx | "nginx -g 'daemon off" | 10 days ago | Exited (0) 9 days ago | | serene_hoover |
| | 70f0d1e4cd08 | nginx | "-bridge:my-bridge-ne" | 10 days ago | Created | | happy_hawking |
| | c2e2f1fd2352 | ubuntu | "/bin/bash -c export" | 10 days ago | Exited (0) 10 days ago | | sad_galileo |
| | 77ded5de4b2f | prasantk/average | "node average.js 1 2 " | 10 days ago | Exited (0) 10 days ago | | mad_darwin |
| | c676058126b1 | prasantk/average | "node average.js 1 2 " | 10 days ago | Exited (0) 10 days ago | | ecstatic_lalande |
| | 6bddbe7885f5 | prasantk/average | "node average.js" | 10 days ago | Exited (0) 10 days ago | | big_thompson |
| | 0a3ad84c2221 | 3d02168f00fc | "node average.js" | 10 days ago | Exited (0) 10 days ago | | peaceful_minsky |
| | 47e697c3fc12 | 3d02168f00fc | "node average.js" | 10 days ago | Exited (0) 10 days ago | | hungry_lamport |
| | 78797aa37937 | 3d02168f00fc | "-e '1 2 3'" | 10 days ago | Created | | drunk_mayer |
| | 8c13665a8ca8 | 3d02168f00fc | "1" | 10 days ago | Created | | loving_carson |
| | 2afe5e6f1384 | 3d02168f00fc | "1 2 3" | 10 days ago | Created | | nostalgic_leavitt |
| | 5d7403525309 | 3d02168f00fc | "1 2 3" | 10 days ago | Created | | happy_newton |
| | 1045734ecd5c | 3d02168f00fc | "node average.js" | 10 days ago | Exited (0) 10 days ago | | reverent_williams |
| | 8989e7fc7d7a | nginx | "nginx -g 'daemon off" | 10 days ago | Exited (0) 9 days ago | | kickass_lichterman |
| | a08eb10ae2fa | nginx | "nginx -g 'daemon off" | 10 days ago | Exited (0) 10 days ago | | docker-nginx |
| | e447a0763ff1 | nginx | "nginx -g 'daemon off" | 10 days ago | Exited (0) 10 days ago | | hopeful_dijkstra |
| | 9a1eab880312 | alpine | "/bin/sh" | 10 days ago | Exited (0) 10 days ago | | hungry_sinoussi |
| | 1932e28ef5cc | alpine | "/bin/bash" | 10 days ago | Created | | modest_engelbart |
| | 454adf9473f9 | alpine | "echo hello" | 10 days ago | Exited (0) 10 days ago | | elated_curran |
| | ad834018d0a3 | alpine | "echo hello" | 10 days ago | Exited (0) 10 days ago | | sleepy_davinci |
| | 4a678e2c1c11 | hello-world | "/hello" | 10 days ago | Exited (0) 10 days ago | | thirsty_keller |
| | 369e76f97dd7 | training/webapp | "python app.py" | 13 days ago | Exited (0) 13 days ago | 0.0.0.0:32771->5000/tcp | boring_roentgen |
| | 826204fae788 | hello-world | "/hello" | 13 days ago | Exited (0) 13 days ago | | happy_kalam |
| | 4c8bacdd231a | docker/whalesay | "cowsay foobar" | 13 days ago | Exited (0) 13 days ago | | big_carson |
| | 60809ce0320d | docker/whalesay | "cowsay boo" | 13 days ago | Exited (0) 13 days ago | | distracted_wilson |
| | 1e5d8f24be78 | ubuntu | "/bin/bash" | 2 weeks ago | Exited (0) 2 weeks ago | | fervent_agnesi |
| | 8a23c6c978f3 | ubuntu:latest | "/bin/bash" | 2 weeks ago | Created | | berserk_pare |
| | 64c20bcac482 | ubuntu | "echo hello" | 2 weeks ago | Exited (0) 2 weeks ago | | gloomy_darwin |
| | 213605afcc24 | ubuntu | "hello" | 2 weeks ago | Created | | goofy_jones |
| | 6575e1b2ae09 | ubuntu | "hello" | 2 weeks ago | Created | | condescending_shirley |
| | 8345b4e82a5b | ubuntu | "hello" | 2 weeks ago | Created | | serene_jepsen |
| | ce31cb01a791 | ubuntu | "echo hello" | 2 weeks ago | Exited (0) 2 weeks ago | | small_bhaskara |
| | cf3b1580e3d1 | ubuntu | "hello" | 2 weeks ago | Created | | evil_heyrovsky |
| | f46a4880894e | ubuntu | "hello" | 2 weeks ago | Created | | sick_lamport |

How to run a container?

Use “`docker run OPTIONS <<image-tag>> CMD ARGS`”

```
$ docker run fedora /bin/echo 'Hello world'  
Hello world  
$
```

The diagram illustrates the components of a Docker run command. It shows the command `docker run fedora /bin/echo 'Hello world'` with callouts pointing to its parts:

- Image name:** Points to the word `fedora`.
- Command name:** Points to the word `/bin/echo`.
- Command argument:** Points to the string `'Hello world'`.

```
docker run fedora /bin/echo 'Hello world'
```

How to run a container interactively?

```
$ docker run -t -i fedora /bin/bash
[root@00eef5289c91 /]# pwd
/
[root@00eef5289c91 /]# whoami
root
[root@00eef5289c91 /]# ls
bin  boot  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv
sys  tmp   usr  var
[root@00eef5289c91 /]# cc
bash: cc: command not found
[root@00eef5289c91 /]# gcc
bash: gcc: command not found
[root@00eef5289c91 /]# java
bash: java: command not found
[root@00eef5289c91 /]# tar
bash: tar: command not found
[root@00eef5289c91 /]# exit
exit
$
```

Create a terminal
to interact with

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

short for “—interactive”

How to run a container in the background?

short for “—detach” and it runs
container in the background

```
$ docker run -d ubuntu /bin/sh -c "while true; do echo current date and time is: $(date); sleep 10; done"
9128bf57e03c3b32f0bf784a92332953996236d7e358a77c62c10bdec95fd5b9
$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS
9128bf57e03c        ubuntu      "/bin/sh -c 'while tr'"   About a minute ago   Up About a
minute              lonely_einstein
$ docker logs 9128bf57e03c3b32f0bf784a92332953996236d7e358a77c62c10bdec95fd5b9
current date and time is: Fri Jul 22 15:42:49 IST 2016
current date and time is: Fri Jul 22 15:42:49 IST 2016
current date and time is: Fri Jul 22 15:42:49 IST 2016
current date and time is: Fri Jul 22 15:42:49 IST 2016
// output elided
```

How to attach to a running container?

short for “—detach” and it runs
container in the background

```
$ docker run -d ubuntu /bin/sh -c "while true; do echo current date and time is: $(date); sleep 10; done"  
acc349675098a0133366076f2082db6171ee4a0cd2e1e45ada9a485684ea4c01  
$ docker attach acc349675098a0133366076f2082db6171ee4a0cd2e1e45ada9a485684ea4c01  
current date and time is: Mon Aug 1 10:30:13 IST 2016  
current date and time is: Mon Aug 1 10:30:13 IST 2016
```

The “attach” command attaches
to a running container

How do I create an image from a running container?

Use “docker commit” command

```
$ docker run -d alpine echo "hello world"
9884347880f62f7c5d43702c3d701e3b87a49f9bdde5843380af1479f4dc0755
$ docker logs 9884347880f62f7c5d43702c3d701e3b87a49f9bdde5843380af1479f4dc0755
hello world
$ docker commit -m "my first image from container"
9884347880f62f7c5d43702c3d701e3b87a49f9bdde5843380af1479f4dc0755 myalpine:latest
sha256:b707ef35394c365bece70240213942e43da7f882107d30482ad6bec6b4bacfb7
$ docker images
REPOSITORY           TAG      IMAGE ID      CREATED
SIZE
myalpine             latest   b707ef35394c  18 hours ago
4.799 MB
$ docker run -it b707ef35394c365bece70240213942e43da7f882107d30482ad6bec6b4bacfb7
hello world
$
```

How to get list of containers?

Use “docker ps” command

| \$ docker ps | CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|--------------|------------------|------------------------|------------|-----------|----------------------|-------------------------|
| | 3651758ff308 | wordpress:latest | "/entrypoint.sh apach" | 2 days ago | Up 2 days | 0.0.0.0:8000->80/tcp | mywordpress_wordpress_1 |
| | b9538054539 | mysql:5.7 | "docker-entrypoint.sh" | 2 days ago | Up 2 days | 3306/tcp | mywordpress_db_1 |

How do I see all the containers?

Use “docker ps -a” command

| \$ docker ps -a | CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|-----------------|--------------|------------------|------------------------|----------------|---------------------------|----------------------|-------------------------|
| | 2c378c6b84b1 | fedora | "/bin/echo 'Hello wor" | 4 minutes ago | Exited (0) 4 minutes ago | | grave_thompson |
| | c4b2db95f268 | hello-world | "/hello" | 5 minutes ago | Exited (0) 5 minutes ago | | amazing_jones |
| | 2dcd9d0caf6f | 777f9424d24d | "/bin/bash" | 42 minutes ago | Exited (0) 42 minutes ago | | prickly_khorana |
| | 3651758ff308 | wordpress:latest | "/entrypoint.sh apach" | 2 days ago | Up 2 days | 0.0.0.0:8000->80/tcp | mywordpress_wordpress_1 |
| | b95388054539 | mysql:5.7 | "docker-entrypoint.sh" | 2 days ago | Up 2 days | 3306/tcp | mywordpress_db_1 |
| | 4b984664f9aa | golang:latest | "go run myapp.go" | 2 days ago | Exited (1) 2 days ago | | mydocker_app_1 |
| | 63cd7661a8ad | hello-world | "/hello" | 2 days ago | Exited (0) 2 days ago | | adoring_sammet |
| | c191fbeae884 | ubuntu | "/bin/bash" | 2 days ago | Exited (0) 2 days ago | | clever_mcclintock |
| | 08e173332d46 | docker/whalesay | "cowsay Hello world" | 2 days ago | Exited (0) 2 days ago | | tender_joliot |
| | 6322b8204a5d | 0f192147631d | "/bin/bash" | 9 days ago | Exited (0) 9 days ago | | desperate_aryabhata |
| ... | | | | | | | |

How do I remove a container?

Use “docker rm” command

```
$ docker stop mywordpress_db_1  
mywordpress_db_1  
$ docker rm mywordpress_db_1  
mywordpress_db_1
```

You have to first stop a container before trying to remove it

How do I remove all the containers?

Use “`docker stop $(docker ps -a -q)`” and
“`docker rm $(docker ps -a -q)`” commands

```
$ docker stop $(docker ps -a -q)
00eef5289c91
8553eebfab94
696a04db91db
// rest of the output elided
$ docker rm $(docker ps -a -q)
00eef5289c91
8553eebfab94
696a04db91db
// rest of the output elided
$ docker ps -a
CONTAINER ID        IMAGE NAMES          COMMAND      CREATED      STATUS
```

Note how the output
shows no containers