Why Docker?

docker
Build, Ship, Run, Any App Anywhere
From Dev
To Ops
Any App
Microsoft
ASP.NET
.NET
Java
MORE
docker
Any OS
Windows
Linux
Anywhere
Physical
Virtual
Cloud

Why Docker?

BUILD
DEVELOPER WORKFLOWS
Docker allows you to compose your application from microservices, without worrying about inconsistencies between development and production environments, and without locking into any platform or language.
SHIP
REGISTRY SERVICES
Docker lets you design the entire cycle of application development, testing and distribution, and manage it with a consistent user interface.
RUN
MANAGEMENT
Docker offers you the ability to deploy scalable services, securely and reliably, on a wide variety of platforms.
DOCKER ENGINE
INFRASTRUCTURE

What is Docker?

Docker is an open-source project that automates the deployment of applications inside software containers.

What is Docker?

"an open platform for developers and sysadmins to build, ship, and run distributed applications"

What is Docker?

Docker

Original author(s)	Solomon Hykes
Developer(s)	Docker, Inc.
Initial release	13 March 2013; 3 years ago
Stable release	1.11.2[1] / 31 May 2016; 36 days ago
Written in	Go[2]
Operating system	Linux[a]
Platform	x86-64 with modern Linux kernel
Type	Operating-system-level virtualization
License	Apache License 2.0
Website	www.docker.com

Source: https://en.wikipedia.org/wiki/Docker_(software)

Pop quiz

Docker is written in:

A. Java language

B. C language

C. D language

D. Go language

Pop quiz: answer

Good read: Why Did We Decide to Write Docker in Go?

VMs vs. Docker

VMs vs. Docker

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

Docker accesses virtualisation features of Linux

Native Docker support on Windows

Source: https://i2.wp.com/blog.docker.com/wp-content/uploads/windows.png?w=975&ssl=1

https://i2.wp.com/blog.docker.com/wp-content/uploads/windows.png?w=975&ssl=1

Docker for DevOps

Docker becoming popular over time

Google Trends: https://www.google.co.uk/trends/explore?q=%2Fm%2F0wkcjgj&hl=en-US

What's covered in this session?

CodeOps

Getting Started

Essential Docker components

Installing Docker

Install it from “https://www.docker.com/products/overview"

INSTALL THE PLATFORM

Install Docker with easy to use installers for the major desktop and cloud platforms.

MAC

A native Mac application with a user interface and auto-update capabilities, that is deeply integrated with OS X native virtualization.

Learn More

WINDOWS

A native Windows application with a user interface and auto-update capabilities, that is deeply integrated with Windows native virtualization.

Learn More

LINUX

Install Docker on nodes which have a Linux distribution already installed.

Learn More

AWS

Quickly deploy, scale, and manage Docker on AWS. Docker for AWS takes optimal advantage of the underlying infrastructure, while providing a modern Docker platform that can be used to deploy portable apps.

Learn More

AZURE

Quickly deploy, scale and manage Docker on Azure. Docker for Azure takes optimal advantage of the underlying infrastructure, while providing a modern Docker platform that can be used to deploy portable apps.

Learn More

OTHERS

For platforms without a Docker installer, easily setup Docker using Docker Machine.

Learn More

Official Docker images?

From “https://hub.docker.com/explore/"

nginx official	3.5K STARS	10M+ PULLS	DETAILS
busybox official	737 STARS	10M+ PULLS	DETAILS
ubuntu official	4.3K STARS	10M+ PULLS	DETAILS
redis official	2.4K STARS	10M+ PULLS	DETAILS
registry official	950 STARS	10M+ PULLS	DETAILS
swarm official	410 STARS	10M+ PULLS	DETAILS
mongo official	2.1K STARS	10M+ PULLS	DETAILS

Finding Docker version

mydocker — -bash — 47×19

```
$ 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
```

Finding details of a Docker installation

mydocker — -bash — 79×42

```
$ docker info
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:OSTC: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 commandline?

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 >
 -------------
    \
     \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""""""""""\___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \_______ o          __/
        \     \         __/
          \____\_______/
$
```

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/
```

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.

CodeOps

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

BEST PRACTICE

Choose smaller images

- Prefer choosing a smaller base image that provides equivalent functionality (for your requirement) instead of choosing a larger one

- Example: Choose Alpine vs. Fedora (5 MB vs. 205 MB)

```
alpine        latest    4e38e38c8ce0        4 weeks ago        4.799 MB
fedora        latest    f9873d530588        4 weeks ago        204.4 MB
```

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": "2f3dcd897cf758418389d50784c73b43b1fd7db09a80826329496f05eef7b377",
        "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

```
$ docker history gcc
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

Use tools to visualise layers

- You can use the online tool imagelayers.io to visualise the layers of an 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

Avoid “image sprawl”

- Remove unused images and release disk space

```
// output from the docker bench security tool

[INFO] 6.4 - Avoid image sprawl

   [INFO]      * There are currently: 60 images
```

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
```

Remove “dangling images”

- Remove “dangling images” using the command “$docker rmi $(docker images -f "dangling=true" -q)”

Using Registry & Repository

How to push my image to Docker Hub?

```
$ docker tag myjavaapp gsamarthyam/myfirstjavaprog:latest
$ docker push gsamarthyam/myfirstjavaprog:latest
The push refers to a repository [docker.io/gsamarthyam/myfirstjavaprog]
a97e2e0314bc: Pushed
3b9964bc9417: Pushed
de174b528b56: Pushed
// elided the output
latest: digest: sha256:1618981552efb12afa4e348b9c0e6d28f0ac4496979ad0c0a821b43547e13c13 size: 2414
$
```

How to pull my image from Docker Hub?

```
$ docker pull gsamarthyam/myfirstjavaprog:latest
latest: Pulling from gsamarthyam/myfirstjavaprog
Digest: sha256:1618981552efb12afa4e348b9c0e6d28f0ac4496979ad0c0a821b43547e13c13
// output elided …
$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
myjavaapp                     latest              0d7a3a12ba9d        About an hour ago   669.2 MB
gsamarthyam/myfirstjavaprog   latest              0d7a3a12ba9d        About an hour ago   669.2 MB
// output elided …
$ docker run gsamarthyam/myfirstjavaprog
hello world
$
```

How do I create and use my own registry?

Pull the “registry” image and run the container

```
$ docker pull registry
Using default tag: latest
latest: Pulling from library/registry

e110a4a17941: Already exists
2ee5ed28ffa7: Pull complete
d1562c23a8aa: Pull complete
06ba8e23299f: Pull complete
802d2a9c64e8: Pull complete
Digest: sha256:1b68f0d54837c356e353efb04472bc0c9a60ae1c8178c9ce076b01d2930bcc5d
Status: Downloaded newer image for registry:latest

$ docker run -d -p5000:5000 registry
7768bed98a5e1916a820c84906e1f21cfc84888a934c140ad22e19cee5e2541d
```

You can now push/pull images from this private registry

CodeOps

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
1e95fcd15893        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                     80/tcp, 443/tcp          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
$
```

Image name

Command argument

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

Command name

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"

Cobb's totem - the top

Running a container - Totem?

Running a container - Totem?

```
$ hostname
ganesh
$ docker run -it alpine /bin/sh
/ # hostname
  b4ebae46b156
/ # ps -a
    PID   USER     TIME   COMMAND
    1 root       0:00 /bin/sh
    6 root       0:00 ps -a
/ # exit
$ ps -a
PID TTY           TIME CMD
15327 ttys001    0:00.02 login -pf gsamarthyam
15328 ttys001    0:00.27 -bash
```

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               NAMES
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 expose a port?

host port (machine on which this command is run)

mapped port - nginx

```
$ docker run -d -p 80:80 nginx
9128bf57e03c3b32f0bf784a92332953996236d7e358a77c62c10bdec95fd5b9
```

```
$ docker inspect 9128bf57e03c3b32f0bf784a92332953996236d7e358a77c62c10bdec95fd5b9
```

```
  "PortBindings": {
                "80/tcp": [
                    {
                        "HostIp": "",
                        "HostPort": "80"
                    }
                ]
            },
```

Using Nginx

Type http://localhost:80 in your browser window

How to expose a port?

host port (machine on which this command is run); since no explicit mapped port number is provided, a random port number is assigned

```
$ docker run -d -p 80 --name minenginx nginx
9128bf57e03c3b32f0bf784a92332953996236d7e358a77c62c10bdec95fd5b9

$ docker inspect minenginx
```

```
"Ports": {
                "443/tcp": null,
                "80/tcp": [
                    {
                        "HostIp": "0.0.0.0",
                        "HostPort": "32770"
                    }
                ]
            },
```

randomly assigned and mapped port number (by docker)

How to expose all exposed ports?

-P publishes all exposed ports to random ports; in this case, ports 443 and 80 are respectively mapped to 32771 and 32772

```
$ docker run -d -P --name minenginx nginx
6b873116f198f4235e3eee1b2085e0312eaa0067217da614c62e2ce55a8c8d4e
$ docker port minenginx
443/tcp -> 0.0.0.0:32771
80/tcp -> 0.0.0.0:32772
```

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 to detach from a running container (without exiting)?

From docker documentation

To detach the tty without exiting the shell,
use the escape sequence Ctrl-p + Ctrl-q
note: This will continue to exist in a stopped state once exited (see "docker ps -a")

```
$ docker run -it alpine /bin/sh
/ # echo "hello"
hello
/ # $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
99d946d9b4e0        alpine              "/bin/sh"           15 seconds ago      Up 14 seconds
gloomy_mcclintock
$
```

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
$
```

Avoid “docker commit”

- Avoid creating docker images manually (e.g., using “docker commit”); rather automate the image build process (using Dockerfile and “docker build”)

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
b95388054539        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
...
```

Explicitly remove exited containers

- Explicitly use --rm to remove the container from the file system - otherwise, even if the container exits, it is not cleaned up yet (and will hog memory).

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                COMMAND              CREATED              STATUS
PORTS               NAMES
```

Note how the output shows no containers

State transition

Using nginx

Nginx exposes ports 80 and 443; -P maps them randomly in the ports range 49153 and 65535

```
$ docker run --name mynginx -P -d nginx
561e15ac1848cf481f89bb161c23dd644f176b8f142fe617947e06f095e0953f
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                        NAMES
561e15ac1848        nginx               "nginx -g 'daemon off"   18 hours ago
Up About a minute   0.0.0.0:32771->80/tcp, 0.0.0.0:32770->443/tcp   mynginx
$ curl localhost:32771
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
// rest of the output elided ...
```

Using nginx

Type “localhost:80” in the browser address bar

```
$ cat Dockerfile
FROM nginx:latest
MAINTAINER Ganesh Samarthyam

ADD ./index.html /usr/share/nginx/html/index.html
EXPOSE 80
$ cat index.html
<h1> welcome to Dockerizing apps! <h1>
$ docker build .
Sending build context to Docker daemon 3.072 kB
// output elided ...
Removing intermediate container b043a75a4e1c
Successfully built 1aae04309f8b
$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
<none>                        <none>              1aae04309f8b        6 seconds ago       182.8 MB
$ docker run -p 80:80 -d 1aae04309f8b
984c179231188445289e70d854250e4e981b77a899208360db4466e73930be42
$ curl localhost:80
<h1> welcome to Dockerizing apps! <h1>
$
```

How do I run a C program?

```
$ docker pull gcc
Using default tag: latest
latest: Pulling from library/gcc
5c90d4a2d1a8: Already exists
ab30c63719b1: Already exists
c6072700a242: Already exists
abb742d515b4: Already exists
d32a4c04e369: Pull complete
276c31cf0a4c: Pull complete
a455d29f9189: Pull complete
dcfe5869552b: Pull complete
Digest: sha256:35256b5f4e4d5643c9631c92e3505154cd2ea666d2f83812b418cfdb1d5866e8
Status: Downloaded newer image for gcc:latest
$
```

How do I run a C program?

```
$ docker pull ubuntu:latest
latest: Pulling from library/ubuntu
43db9dbdcb30: Pull complete
85a9cd1fcca2: Pull complete
c23af8496102: Pull complete
e88c36ca55d8: Pull complete
Digest: sha256:7ce82491d6e35d3aa7458a56e470a821baecee651fba76957111402591d20fc1
Status: Downloaded newer image for ubuntu:latest

$ docker run -i -t ubuntu /bin/bash
root@c191fbeae884:/# gcc
bash: gcc: command not found

root@c191fbeae884:/# apt-get update
// elided the output
root@c191fbeae884:/# apt-get install gcc
// elided the output
root@c191fbeae884:/# cat > hello.c
int main() { printf("hello world\n”); }
root@c191fbeae884:/# gcc -w hello.c
root@c191fbeae884:/# ./a.out
hello world
root@c191fbeae884:/#
```

How do I run a C program?

```
$ cat Dockerfile
FROM gcc:latest
MAINTAINER Ganesh Samarthyam version: 0.1

COPY . /usr/src/mycapp

WORKDIR /usr/src/mycapp

RUN gcc -o first first.c
CMD ["./first"]

$ cat first.c
#include <stdio.h>

int main() { printf("hello world\n"); }

$ docker build . -t"mycapp:latest"
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM gcc:latest
 ---> a0b516dc1799
// .. steps elided ...
Step 6 : CMD ./first
 ---> Using cache
 ---> f99e7f18fa42
Successfully built f99e7f18fa42
$ docker run -it mycapp
hello world
```

How do I run a Java program?

```
$ cat Dockerfile
FROM java:latest
COPY . /usr/src/
WORKDIR /usr/src/
RUN javac hello.java
CMD ["java", "hello"]
$ cat hello.java
class hello {
   public static void main(String []args) {
      System.out.println("hello world");
   }
}
$ docker build . -t"myjavaapp:latest"
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM java:latest
 ---> 264282a59a95
// intermediate steps elided
Successfully built 0d7a3a12ba9d
$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
myjavaapp                   latest              0d7a3a12ba9d        About an hour ago   669.2 MB
$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
myjavaapp                   latest              0d7a3a12ba9d        About an hour ago   669.2 MB
<none>                      <none>              7cfb4bdf47a7        About an hour ago   669.2 MB
// rest of the output elided
$  docker run myjavaapp
hello world
```

Beware of "container sprawl"

- Application broken to run in "too many containers" can be difficult to deal with!

"Breaking deployments into more functional discrete parts is smart, but that means we have MORE PARTS to manage. There's an inflection point between separation of concerns and sprawl."

-- Rob Hirschfeld

(CEO of RackN and OpenStack Foundation board member)

Pop quiz

What happens when you execute this on the command-line?

```
docker run debian /bin/sh
```

A. A prompt from the shell of created container will be thrown to you

B. A container is created and then exited immediately

C. A container is created and executes in the detached mode; you can attach to it later using the container id

D. Docker CLI issues the error: Error response from daemon: No command specified.

Pop quiz: answer

When you execute this command,

docker run debian /bin/sh

A container is created and then exited immediately.

```
$ docker ps -a
CONTAINER ID        IMAGE                                    COMMAND
CREATED             STATUS                        PORTS              NAMES
4c12998fd392        debian                                   "/bin/bash"
6 seconds ago       Exited (0) 5 seconds ago                         sick_panini
```

Pop quiz

What happens when you execute this on the command-line?

```
docker run -itd debian
```

A. You get "Error response from daemon: No command specified."

B. The created container runs in the detached mode

C. You get "unknown shorthand flag: -itd"

D. A shell from the created container is returned to you

Pop quiz: answer

When you execute

docker run -itd debian

the created container runs in the detached mode!

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS               NAMES
a53779a74904        debian              "/bin/bash"              2 minutes ago
Up 2 minutes                            agitated_aryabhata
```

CodeOps

Building images using Dockerfile

Different ways to create images

docker commit	Build an image from a container
docker build	Create an image from a Dockerfile by executing the build steps given in the file
docker import	Create a base image by importing from a tarball. [import is mainly used for creating base-images; first two options are widely used]

Dockerfile - key instructions

FROM	The base image for building the new docker image; provide “FROM scratch” if it is a base image itself
MAINTAINER	The author of the Dockerfile and the email
RUN	Any OS command to build the image
CMD	Specify the command to be started when the container is run; can be overridden by the explicit argument when providing docker run command
ADD	Copies files or directories from the host to the container in the given path
EXPOSE	Exposes the specified port to the host machine

How can I create an image from a Dockerfile?

Use docker build command
"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
$
```

Dockerfile for running a Java program

```
$ cat HiHello.java
import java.io.*;
import java.net.InetSocketAddress;
import com.sun.net.httpserver.*;

public class HiHello {
    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress(8080), 0);
        server.createContext("/hi", (HttpExchange t) ->
        {
        try {
           String response = "hello\n";
                   t.sendResponseHeaders(200, response.length());
                   try(OutputStream os = t.getResponseBody()) {
                      os.write(response.getBytes());
                }
        } catch (IOException ioe) {
           System.err.println("Error writing to outputstream: " + ioe);
           System.exit(-1);
        }
        }
   );
        server.start();
    }
}

$ curl localhost:8080/hi
hello
```

Dockerfile for running a Java program

```
$ cat Dockerfile
FROM java:latest
COPY HiHello.class /
EXPOSE 8080
ENTRYPOINT ["java"]
CMD ["HiHello"]
```

```
$ docker build .
Sending build context to Docker daemon 6.656 kB
Step 1 : FROM java:latest
 ---> 264282a59a95
// ... Successfully built 60a14f519720
$ docker run -d -p 8080:8080 60a14f519720
16f6d7eca560c96b995be9f0c6d68167930ab7501451a452818e04ce29ec177f
$ curl localhost:8080/hi
hello
```

Pop quiz

Which command do you use “to find layers and their sizes” in an image using Docker CLI?

A. Use “docker images -layers <<imageid>>” command

B. Use “docker layers <<imageid>> command

C. Use “docker history <<imageid>> command

D. There is no way you can find layers and their sizes using Docker CLI - you need to use external tools

Pop quiz: answer

To find layers and their sizes in an image using Docker CLI, use“docker history <<imageid>> command.

```
$  docker history google/cadvisor
IMAGE               CREATED             CREATED BY                                      SIZE
COMMENT
106e303be3a4        2 weeks ago         /bin/sh -c #(nop) ENTRYPOINT ["/usr/bin/cadvi   0 B
<missing>           2 weeks ago         /bin/sh -c #(nop) EXPOSE 8080/tcp               0 B
<missing>           2 weeks ago         /bin/sh -c #(nop) ADD file:1bde294f31142b3dee   25.87 MB
<missing>           2 weeks ago         /bin/sh -c apk --no-cache add ca-certificates   17.13 MB
<missing>           2 weeks ago         /bin/sh -c #(nop) ENV GLIBC_VERSION=2.23-r3     0 B
<missing>           2 weeks ago         /bin/sh -c #(nop) MAINTAINER dengnan@google.c   0 B
<missing>           3 months ago        /bin/sh -c #(nop) ADD file:852e9d0cb9d906535a   4.799 MB
```

Pop quiz

Which command do you use “recreate the Dockerfile that was used to build that image” from a given image id/tag using Docker CLI?

A. Use “docker images -dockerfile <<imageid>>” command

B. Use “docker build -reverse <<imageid>> command

C. Use “docker history --no-trunc --out:<filename> <<imageid>> command

D. There is no way to recreate the Dockerfile that was used to build that image from a given image id/tag using Docker CLI

Pop quiz: answer

There is **NO** way to recreate the Dockerfile that was used to build that image from a given image id/tag using Docker CLI.

Think about Makefile: can you recreate the Makefile that was used to build that executable file? No.

However, you can see the commands used to create the layers in the image. Pass "—no-trunc" option to "docker history" command.

Example: "docker history --no-trunc google/cadvisor"

Try it now!

Docker Volumes

Docker volume commands

Command	Description
docker volume create	Create a volume
docker volume inspect	Display detailed information on one or more volumes
docker volume ls	List the available volumes
docker volume rm	Remove one or more volumes

Commands for Docker volumes

```
$ docker volume create --name myvolume
myvolume
$ docker volume ls
local               myvolume
$ docker volume inspect myvolume
[
    {
        "Name": "myvolume",
        "Driver": "local",
        "Mountpoint": "/var/lib/docker/volumes/myvolume/_data",
        "Labels": {},
        "Scope": "local"
    }
]
$ docker volume rm myvolume
myvolume
```

How to persist data?

Use -v option to “mount volumes”

```
$ docker run -v /volumetesting --name="persistdata" alpine /bin/sh -c
"echo testing persistence with volumes > /volumetesting/textfile.txt”

$ docker run --volumes-from=persistdata alpine /bin/sh -c "cat /
volumetesting/textfile.txt"
testing persistence with volumes
```

How to get info on volumes?

Use “docker volume ls and inspect” options

```
$ docker volume ls
DRIVER                  VOLUME NAME
local                   081bf425dd953c6b13f8e36f24540d191792e51dbd9c327eadae131ded5da432
local                   3357f5522da19b47c3996db5e129b52d4be420ccec25d60d4473602cd25f473b

$ docker volume inspect
081bf425dd953c6b13f8e36f24540d191792e51dbd9c327eadae131ded5da432
[
    {
        "Name": "081bf425dd953c6b13f8e36f24540d191792e51dbd9c327eadae131ded5da432",
        "Driver": "local",
        "Mountpoint": "/var/lib/docker/volumes/
081bf425dd953c6b13f8e36f24540d191792e51dbd9c327eadae131ded5da432/_data",
        "Labels": null,
        "Scope": "local"
    }
]
```

Removing volumes

Use “docker volume rm” option

```
$ docker volume rm 081bf425dd953c6b13f8e36f24540d191792e51dbd9c327eadae131ded5da432
081bf425dd953c6b13f8e36f24540d191792e51dbd9c327eadae131ded5da432
```

Removing containers with volumes

- When the container is removed, the volumes will not be removed. If the volumes also need to be removed, you have to use the -v option, as in: docker rm -v <<sha>>

Clean up volumes

- You can “clean up” the volumes if you aren't using them. Use the command “docker volume rm $(docker volume ls -q)“ to remove all the volumes.

Use Flocker (data volume manager)

See: https://clusterhq.com/flocker/

source: https://clusterhq.com/assets/images/diagrams/diagram-1.jpg

Docker Machine

Docker Machine

Create and manage machines running Docker (cloud or on your computer)

```
$ docker-machine create --driver=virtualbox myhost
Running pre-create checks...
(myhost) Default Boot2Docker ISO is out-of-date, downloading the latest release...
(myhost) Latest release for github.com/boot2docker/boot2docker is v1.12.2
(myhost) Downloading /Users/gsamarthyam/.docker/machine/cache/boot2docker.iso from https://github.com/
boot2docker/boot2docker/releases/download/v1.12.2/boot2docker.iso...
// ...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run:
docker-machine env myhost

$ docker-machine env myhost
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.100:2376"
export DOCKER_CERT_PATH="/Users/gsamarthyam/.docker/machine/machines/myhost"
export DOCKER_MACHINE_NAME="myhost"
# Run this command to configure your shell:
# eval $(docker-machine env myhost

$  docker-machine ls
NAME     ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER    ERRORS
myhost   -        virtualbox   Running   tcp://192.168.99.100:2376           v1.12.2
```

Docker Compose

docker-compose commands

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

Creating multiple Docker containers

Step 1. Create a docker-compose.yml file (or docker-compose.yaml file)
Step 2. Execute “docker-compose up -d”
Step 3. Execute “docker-compose logs” from another shell (but from same dir)
Step 4. Execute “docker-compose down”

docker-compose commands

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

Docker voting app

Docker voting app

Step 1. Download .zip or clone: https://github.com/docker/example-voting-app
Step 2. Unzip the file and go to that directory from your shell
Step 3. Type “docker-compose up -d”
Step 4. From another shell, go to the same directory & type “docker-compose logs”
Step 5. In browser address bar, type “**http://localhost:5000**"
Step 6. In browser address bar, type “**http://localhost:5001**"
Step 7. From the shell, in that directory, type “docker-compose down”

CodeOps

Docker voting app

Docker Networking

Getting the ip address of a container

```
$ docker inspect --format '{{ .NetworkSettings.IPAddress }}' fervent_sinoussi
172.17.0.6

$ docker attach fervent_sinoussi
root@856aed6a92f1:/# ip addr
// ...
92: eth0@if93: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
    link/ether 02:42:ac:11:00:06 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.6/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:6/64 scope link
       valid_lft forever preferred_lft forever

root@856aed6a92f1:/# cat /etc/hosts
// ...
172.17.0.6    856aed6a92f1
root@856aed6a92f1:/#
```

There are many ways to get the IP address of a container:

1. Use the docker inspect command
2. Use ip addr command from the container's shell
3. Use "cat /etc/hosts" and check the entry for the container

How to get port mappings of a container?

```
$ docker run -d -p5000:5000 registry
c51b984b4d64a05e924c7677f20e8c5c386e8bb53f5de0369337d31f73a7cf7e

$ docker port
c51b984b4d64a05e924c7677f20e8c5c386e8bb53f5de0369337d31f73a7cf7e
5000/tcp -> 0.0.0.0:5000

$ docker run -P -d nginx
de6e17ededc8223c9a5ac9fee4f9493929a22a78fe88c60b643b545078c60648
$ docker port
de6e17ededc8223c9a5ac9fee4f9493929a22a78fe88c60b643b545078c60648
443/tcp -> 0.0.0.0:32768
80/tcp -> 0.0.0.0:32769

$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                           NAMES
de6e17ededc8        nginx               "nginx -g 'daemon off"   21 seconds
ago      Up 20 seconds       0.0.0.0:32769->80/tcp, 0.0.0.0:32768->443/tcp
reverent_wright
```

Three kinds of networks

Default, single-host driver

```
$ docker network ls
NETWORK ID          NAME      DRIVER    SCOPE
a3bb9a40c8e3        bridge    bridge    local
399711fd0635        host      host      local
790ae8b43d9b        none      null      local
```

Docker network commands

Command	Description
docker network connect	Connect a container to a network
docker network create	Create a network
docker network disconnect	Disconnect a container from a network
docker network inspect	Display detailed information on one or more networks
docker network ls	List networks
docker network rm	Remove one or more networks

Bridge network

```
$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "39457e56e7c0d172a239745e10ebf24f9e5046e9bc98f978f4e759f1f4e486c3",
        "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": {
            "19857aa228f7ba76fc10d1e992e9fe49a0d361b5daab3a0a2703267aab862c58": {
                "Name": "furious_mcnulty",
                "EndpointID": "e2a7423a9108fa7bb18ea9893e299e1bceb73fa13c3123c5b0d515790be477d3",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
            "eaf4697b9989666e0c79cce6dc03697c8226aea37157d97bce1d08e250fb3c36": {
                "Name": "cadvisor",
                "EndpointID": "f907c7510aaf7196e6419733b64da689330948ca0f3c88bd4dd41258a4503e42",
                "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.mtu": "1500"
        },
        "Labels": {}
    }
]
```

By default, containers are added to the bridge network. You can see the containers in bridge network here

Pop quiz

You are creating a new container with this command:

```
docker run -d --name myubuntu ubuntu /bin/sh -c "while true; do echo current date and time is: $(date); sleep 10; done”
```

Which network is the “myubuntu” container attached to?

A. Bridge

B. Overlay

C. Custom

D. None

Pop quiz: answer

Bridge network. By default, a newly created container is attached to the bridge network (unless a different network is specified, for example, using the "—network" option with the docker run command).

```
$  docker network inspect bridge
[
    {
        "Name": “bridge",
       // ...
       "Containers": {
            "04579b88a74c981ae854261dffc7ab17328c28bb6fafec0f9c1e9431e77b3b27": {
                "Name": "myubuntu",
                "EndpointID":
"8a0e7a2559eac35eb60a90e85554679de276bd1ba39ff3a4083301d08e9ee384",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
        // ...
        },
      // ...
   }
]
```

Orchestration

Docker swarm provides "clustering capabilities", i.e., be able to treat group of Docker engines into a single virtual Docker engine

Docker Swarm

"swap, plug, and play"

Docker Swarm

```
$ docker swarm init
Swarm initialized: current node (81snul7czu9pg42h3qnm2v5hr) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token
    SWMTKN-1-2tn8fic07uivk29ith095me3r6cro7bkrfnbtv6va3qvf6urew-6olc37m9ljymly3fcb6rig2nu
    192.168.65.2:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the
instructions.
```

Clustering for Docker

"batteries included but swappable"

Docker Security

Docker workbench for security

```
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
```

OR

```
git clone https://github.com/docker/docker-bench-security.git
cd docker-bench-security
docker-compose run --rm docker-bench-security
```

OR

```
git clone https://github.com/docker/docker-bench-security.git
cd docker-bench-security
sh docker-bench-security.sh
```

Source: https://github.com/docker/docker-bench-security

Docker workbench for security

```
$ sh docker-bench-security.sh
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.1.0
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker 1.11 Benchmark:
# https://benchmarks.cisecurity.org/downloads/show-single/index.cfm?file=docker16.110
# ------------------------------------------------------------------------------

[WARN] Some tests might require root to run
[INFO] 1 - Host Configuration
[WARN] 1.1  - Create a separate partition for containers
[PASS] 1.2  - Use an updated Linux Kernel
[WARN] 1.4  - Remove all non-essential services from the host - Network
[WARN]      * Host listening on: 7 ports
[PASS] 1.5  - Keep Docker up to date
[INFO]       * Using 1.12.1 which is current as of 2016-08-16
[INFO]       * Check with your operating system vendor for support and security maintenance for docker
 [INFO] 1.6  - Only allow trusted users to control Docker daemon
[WARN] 1.7  - Failed to inspect: auditctl command not found.
[INFO] 1.8  - Audit Docker files and directories - /var/lib/docker
[INFO]      * Directory not found
[INFO] 1.9  - Audit Docker files and directories - /etc/docker
[INFO]      * Directory not found
...
[INFO] 2 - Docker Daemon Configuration
[WARN] 2.1  - Restrict network traffic between containers
[PASS] 2.2  - Set the logging level
[PASS] 2.3  - Allow Docker to make changes to iptables
[PASS] 2.4  - Do not use insecure registries
[WARN] 2.5  - Do not use the aufs storage driver
[INFO] 2.6  - Configure TLS authentication for Docker daemon
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.7 - Set default ulimit as appropriate
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.8  - Enable user namespace support
[PASS] 2.9  - Confirm default cgroup usage
[PASS] 2.10 - Do not change base device size until needed
[WARN] 2.11 - Use authorization plugin
[WARN] 2.12 - Configure centralized and remote logging
[WARN] 2.13 - Disable operations on legacy registry (v1)
...
```

Docker workbench for security

- Use the free Docker Workbench For Security (https://github.com/docker/docker-bench-security) to check for violations of security best practices

Monitoring Docker

Stats for all running containers

Use “docker stats” command

```
$ docker stats
```

CONTAINER	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
5394e9b78397	0.00%	1.805 MiB / 1.953 GiB	0.09%	648 B / 648 B	389.1 kB / 0 B	2
6b873116f198	0.00%	1.395 MiB / 1.953 GiB	0.07%	4.354 kB / 648 B	0 B / 0 B	2
742e01ef7fd4	0.00%	500 KiB / 1.953 GiB	0.02%	10.68 kB / 648 B	0 B / 0 B	1
c51b984b4d64	0.01%	18.54 MiB / 1.953 GiB	0.93%	12.49 kB / 648 B	16.6 MB / 0 B	7
ce042fe264b2	0.00%	60 KiB / 1.953 GiB	0.00%	14.94 kB / 648 B	0 B / 0 B	1
c8a954a62a19	0.00%	60 KiB / 1.953 GiB	0.00%	17.69 kB / 648 B	0 B / 0 B	1
725d93bb258c	0.00%	64 KiB / 1.953 GiB	0.00%	18.34 kB / 648 B	4.096 kB / 0 B	1
04579b88a74c	0.00%	340 KiB / 1.953 GiB	0.02%	19.08 kB / 648 B	94.21 kB / 0 B	2
99d946d9b4e0	0.00%	104 KiB / 1.953 GiB	0.01%	21.32 kB / 1.026 kB	24.58 kB / 0 B	1
9128bf57e03c	0.00%	288 KiB / 1.953 GiB	0.01%	5.692 kB / 648 B	0 B / 0 B	2

Displays resource utilisation (cpu, memory, etc) details; automatically updated when details change

Stats for a specific Docker

Use “docker stats <<CONTAINTER_ID>>” command

```
$ docker stats sleepy_wescoff
```

```
CONTAINER          CPU %       MEM USAGE / LIMIT   MEM %      NET I/O          BLOCK I/O       PIDS
sleepy_wescoff     0.00%       1.031 MiB / 4 MiB   25.78%     648 B / 648 B    0 B / 0 B       1
```

Printing containers names in stat

```
$ docker stats $(docker ps --format={{.Names}})
```

Monitoring Docker

Use the free cAdvisor tool
(https://github.com/google/cadvisor)

```
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
```

Now open localhost:8080 in your browser

CodeOps

Monitoring Docker

localhost:8080

Monitoring Docker

- **datadog** (https://www.datadoghq.com/)
- **sysdig** (http://www.sysdig.org/)
- **prometheus** (https://prometheus.io/)

Other topics

How do debug on a running container?

Use “docker exec” command

```
$ docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED
STATUS                              PORTS                               NAMES
9128bf57e03c        ubuntu                         "/bin/sh -c 'while tr"   24 minutes ago      Up
24 minutes                                                   lonely_einstein
$ docker exec -ti lonely_einstein /bin/bash
root@9128bf57e03c:/#
```

Can I use GUI instead of command-line?

Use “kitematic” (https://github.com/docker/kitematic)

Different ways to access Docker

Crazy stuff: Docker in Docker!!

Use “docker run --privileged -d docker:dind"

“docker:dind” is the official “Docker in Docker base image”

See: https://github.com/jpetazzo/dind

Myths and Misconceptions

Docker *completely* replaces VMs

Containers AND VMs

Docker is *completely* portable

Build once, run anywhere - but conditions apply!

There are limitations to portability with Docker (depending on what you mean by “portable”).

For example, you can run a Windows Docker container only on Windows and run a Linux Docker container only on Linux (and not vice versa).

“Management says we need Docker, so let’s use it”

Quick reference

Docker commands

```
attach    Attach to a running container
build     Build an image from a Dockerfile
commit    Create a new image from a container's changes
cp        Copy files/folders between a container and the local filesystem
create    Create a new container
deploy    Create and update a stack from a Distributed Application Bundle (DAB)
diff      Inspect changes on a container's filesystem
events    Get real time events from the server
exec      Run a command in a running container
export    Export a container's filesystem as a tar archive
history   Show the history of an image
images    List images
import    Import the contents from a tarball to create a filesystem image
info      Display system-wide information
inspect   Return low-level information on a container, image or task
kill      Kill one or more running container
load      Load an image from a tar archive or STDIN
login     Log in to a Docker registry.
logout    Log out from a Docker registry.
logs      Fetch the logs of a container
network   Manage Docker networks
node      Manage Docker Swarm nodes
pause     Pause all processes within one or more containers
plugin    Manage Docker plugins
```

Docker commands

```
port      List port mappings or a specific mapping for the container
ps        List containers
pull      Pull an image or a repository from a registry
push      Push an image or a repository to a registry
rename    Rename a container
restart   Restart a container
rm        Remove one or more containers
rmi       Remove one or more images
run       Run a command in a new container
save      Save one or more images to a tar archive (streamed to STDOUT by default)
search    Search the Docker Hub for images
service   Manage Docker services
stack     Manage Docker stacks
start     Start one or more stopped containers
stats     Display a live stream of container(s) resource usage statistics
stop      Stop one or more running containers
swarm     Manage Docker Swarm
tag       Tag an image into a repository
top       Display the running processes of a container
unpause   Unpause all processes within one or more containers
update    Update configuration of one or more containers
version   Show the Docker version information
volume    Manage Docker volumes
wait      Block until a container stops, then print its exit code
```

docker Cheat Sheet

REBELLABS by ZEROTURNAROUND

Glossary

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

Image - a read-only layer that is the base 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 -ti --name container_name
    image_name command
```

Create and start container, run command, destroy container

```
docker run --rm -ti image_name command
```

Example filesystem and port mappings

```
docker run -it --rm -p 8080:8080 -v
/path/to/agent.jar:/agent.jar -e
JAVA_OPTS="-javaagent:/agent.jar"
tomcat:8.0.29-jre8
```

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
    - ./myapp.jar:/app/app.jar
mongo: # container name
  image: mongo # image name
```

Create and start containers

```
docker-compose up
```

Interacting with a container

Run a command in the container

```
docker exec -ti container_name command.sh
```

Follow the container logs

```
docker logs -ft container_name
```

Save a running container as an image

```
docker commit -m "commit message" -a "author"
    container_name username/image_name:tag
```

BROUGHT TO YOU BY JRebel

Source: http://zeroturnaround.com/wp-content/uploads/2016/03/Docker-cheat-sheet-by-RebelLabs.png

Where to learn more?

Relevant URLs

- Detailed list of resources: https://github.com/hangyan/docker-resources
- Self-learning courses: https://training.docker.com/
- Detailed documentation: https://docs.docker.com/
- Detailed tutorial (presentation): http://docker.training
- SE-Radio Episode 217: James Turnbull on Docker
- Docker related presentations in parleys.com
- Blog on Docker resource utilisation

http://amzn.com/1491917571

"Docker: Up & Running", Karl Matthias, Sean P. Kane, O'Reilly Media; 1 edition (July 3, 2015)

DOCKER: UP & RUNNING

- Covers how to develop, test, debug, ship, scale, and support with Docker from DevOps perspective
- We liked the useful tips; examples:
  - "Maximize robustness with fast startup and graceful shutdown."
  - "Explicitly declare and isolate dependencies."
  - "Strictly separate build and run stages."