Docker 101

Docker is not magic
it is an abstraction.

What is Docker?

- Linux Control Groups (cgroups)
- Linux namespaces
- Union file system
- Docker CLI
- Docker API

Linux namespaces allow for isolation of system resources.

PID namespace
Networking namespace
Mount namespace

PID Namespaces

PID namespaces isolate the process ID number space, meaning that processes in different PID namespaces can have the same PID ~manpages

Practical Example

```
$ sudo unshare --fork --pid --mount-proc bash
root@dev:# ps aux
```

Network Namespaces

A network namespace is logically another copy of the network stack, with its own routes, firewall rules, and network devices. ~manpages

Practical Example

Make a network namespace:

```
sudo ip netns add test1
ip netns list
```

Execute command in network namespace:

```
sudo ip netns exec test1 ip a
```

Delete the network namespace:

```
sudo ip netns delete test1
```

Mount Namespaces

A mount namespace is the set of filesystem mounts that are visible to a process. ~ *manpages*

Practical Example

Start bash in a namespace:

```
sudo unshare -m /bin/bash
```

Create a new tempory directory:

```
secret_dir=`mktemp -d -tmpdir=/tmp`
```

Mount the directory:

```
mount -n -o size=1m -t tmpfs tmpfs $secret_dir
grep /tmp /proc/mounts <- note the directory listed in the output
```

Add something to the directory:

```
cd /tmp/tmp.XXXXXXXXX
touch iamsecret
touch iamalsosecret
ls -lFa
```

In another terminal….

```
sudo su
cd /tmp/tmp.XXXXXXXXX
ls -lFa
```

Linux Control Groups

Control cgroups, usually referred to as cgroups, are a Linux kernel feature which allow processes to be organized into hierarchical groups whose usage of various types of resources can then be limited and monitored. ~ *manpages*

blkio	Limits on IO
cpu	Cpu Scheduling
cpuset	Assigns cpu on multicore systems
devices	Controls access to devices
memory	Memory limits

Practical Example

Cgroups are managed in the filesystem:

```
mkdir /sys/fs/cgroup/memory/mygroup
ls -la /sys/fs/cgroup/memory/mygroup
```

Set memory limit of 10MB for this group:

```
echo 10000000 > /sys/fs/cgroup/memory/mygroup/memory.limit_in_bytes

echo 0 > /sys/fs/cgroup/memory/mygroup/memory.swappiness
```

Make something memory hungry:

```
cat <<EOT >> crash.py
f = open("/dev/urandom", "r")
data = ""

i=0
while True:
    data += f.read(1000000) # 1mb
    i += 1
    print "%dmb" % (i*10,)
EOT

python crash.py
```

Add current PID to the cgroup we just made:

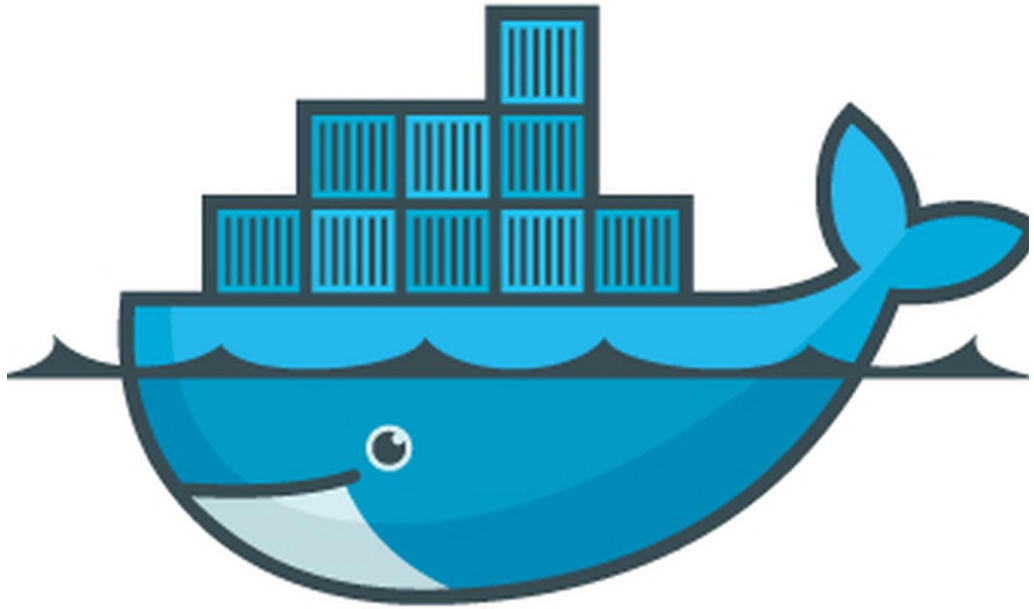
```
echo $$ > /sys/fs/cgroup/memory/mygroup/tasks

python crash.py
```

That is complicated!

I wish there was a tool to make all of this easy.

That is complicated!

I wish there was a tool to make all of this easy.



Union File System

Union file systems, or UnionFS, are file systems that operate by creating layers, making them very lightweight and fast. Docker Engine uses UnionFS to provide the building blocks for containers. ~*docs.docker.com*

Practical Example

Pull images:

```
docker pull alpine:latest
docker pull colek42/cassandra-lucene
```

Export to tar:

```
docker image save colek42/cassandra-lucene > cassandra.tar
docker image save alpine:latest > alpine.tar
```

Dockerfile

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

~docs.docker.com

Practical Example

Pull repo:

```
go get github.com/colek42/helloworld
cd $GOPATH/src/github.com/colek42/helloworld
```

Build binary:

```
CGO_ENABLED=0 GOOS=linux go build -o main .
```

Dockerfile:

```
cat <<EOT >> Dockerfile

FROM scratch
COPY main /
CMD ["/main"]

EOT
```

Build image:

```
docker build --no-cache -t colek42/helloworld:latest .
```

Run Container:

```
docker run -p 8383:8383 colek42/helloworld:latest
```

- docker-compose
- Kubernetes