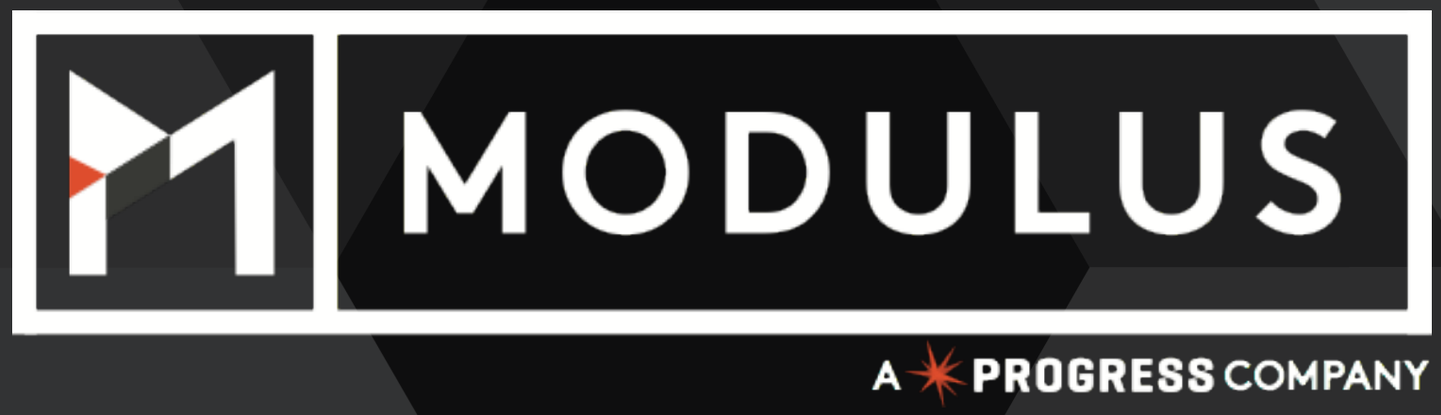
Deploying a Scalable Node.js Application with Docker

Brandon Cannaday | @TheReddest

Follow Along!
Come up and grab a card with a server you can use

About me



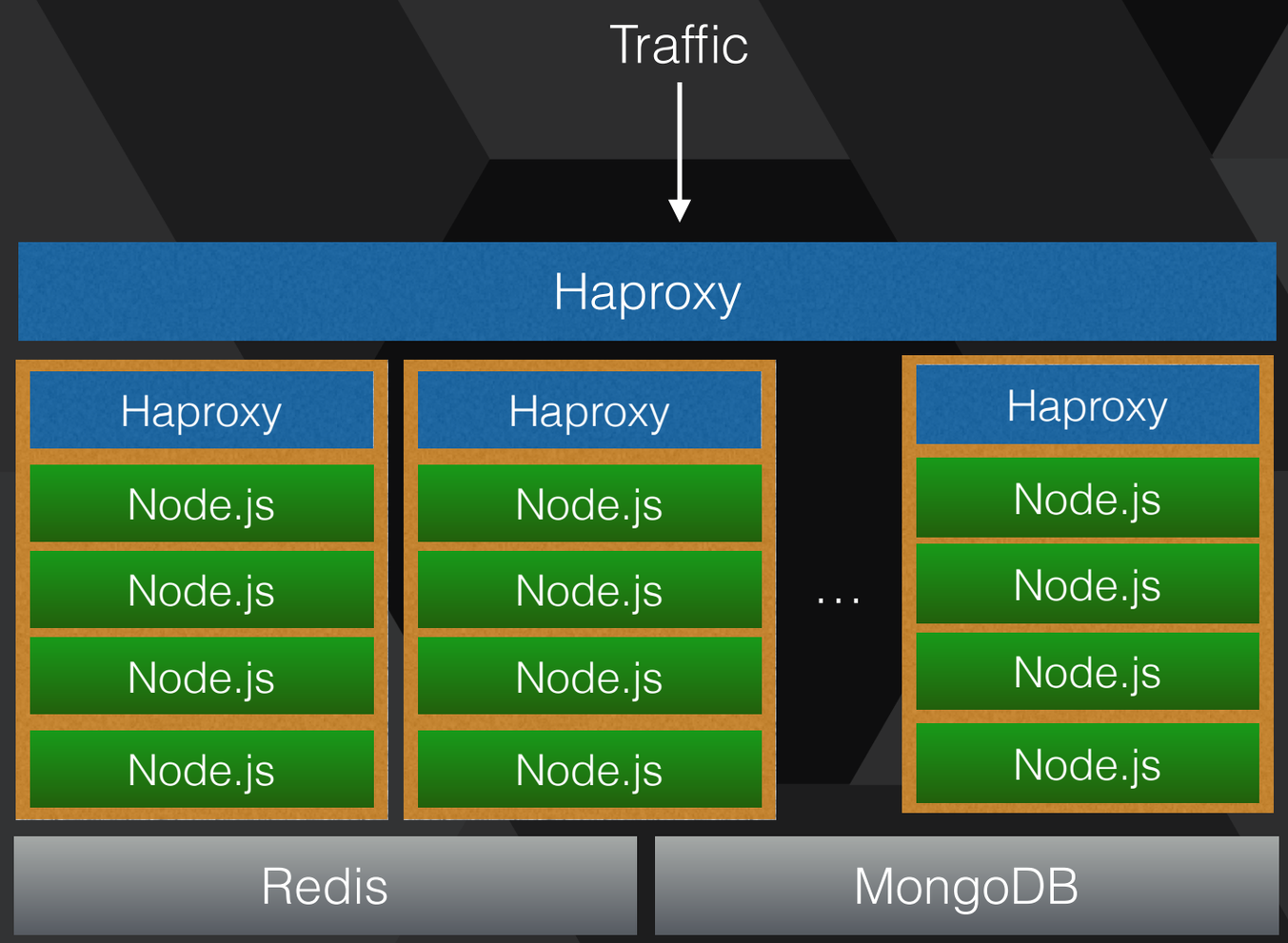
Brandon Cannaday | @TheReddest
modulus.io

Follow Along

https://github.com/InconceivableDuck/Nodevember

Test servers provided by:

What we're building



Agenda

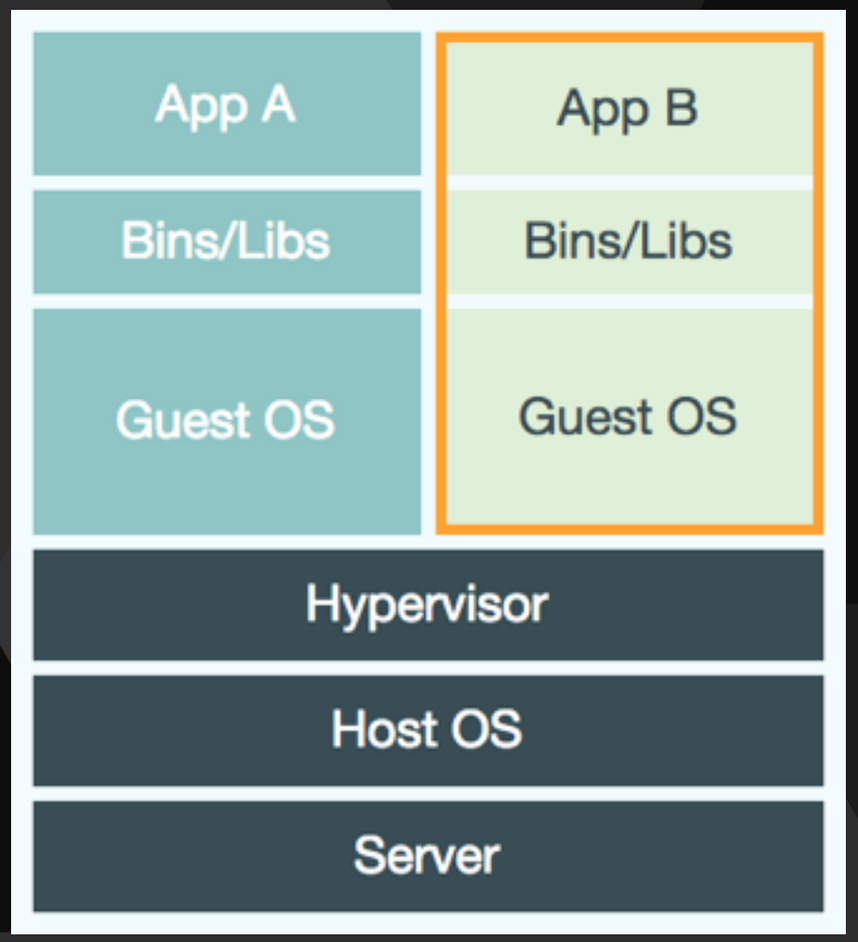
- Docker basics
- The Docker Registry
- The Dockerfile
- Dockerizing Redis / Mongo
- Dockerizing Node.js
- Dockerizing Haproxy

Docker

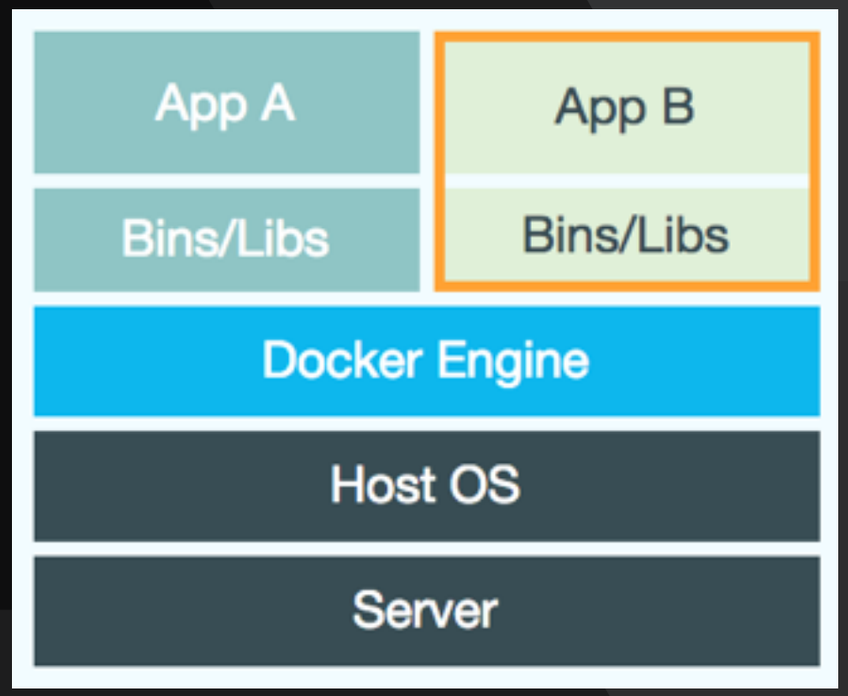
- “Virtualizes” the application layer
  - A bunch of CRUD on top of Linux cgroups
  - Layered filesystem
  - A registry

Docker vs. VM

Virtual Machine



Docker



cgroups

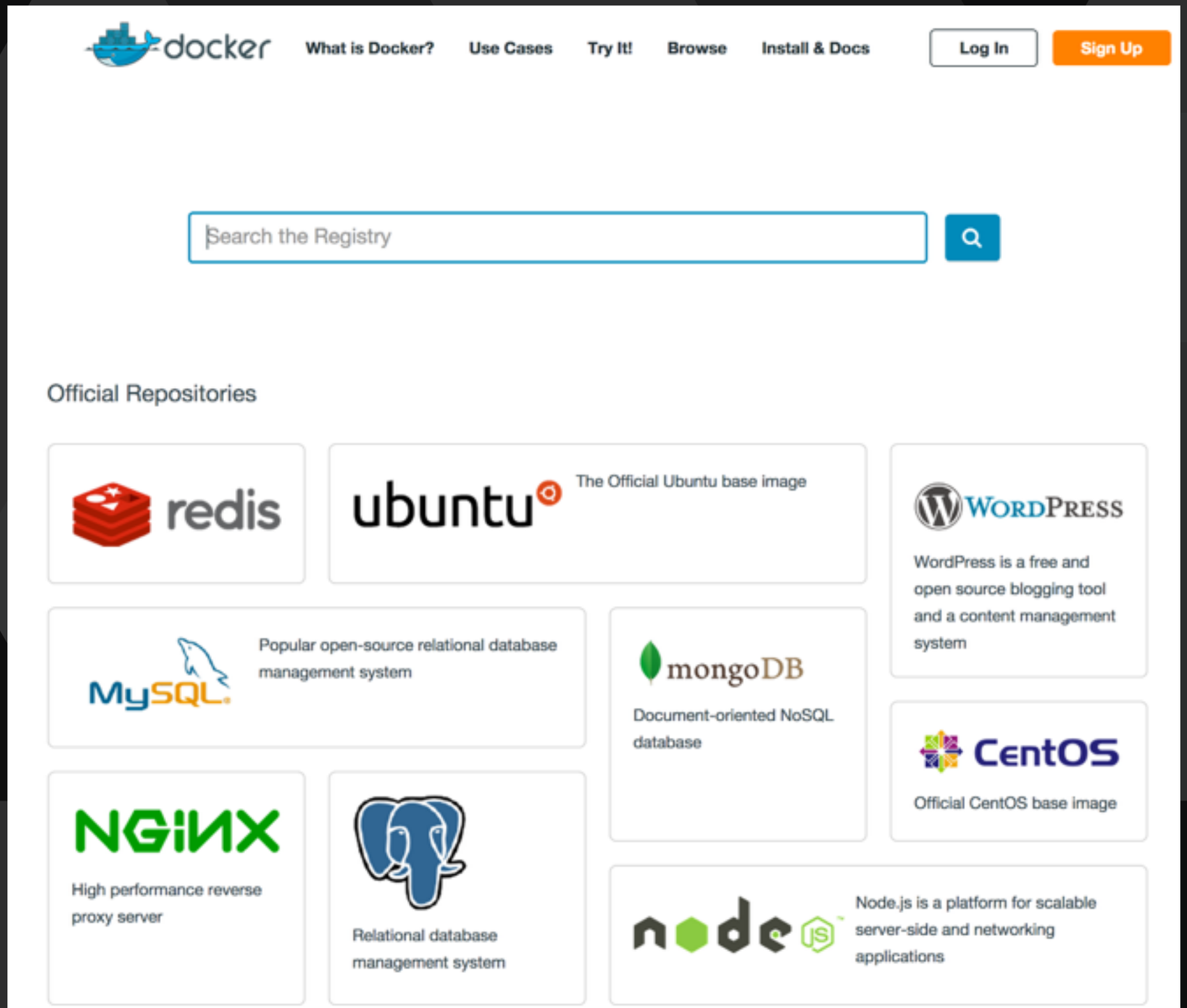
```
mount -t cgroup -o devices,memory,freezer cgroup /cgroup1

mkdir /cgroup1/child1

sleep 100

echo $! > /cgroup1/child1/cgroup.procs

echo FROZEN > /cgroup1/child1/freezer.state
```

https://help.ubuntu.com/lts/serverguide/cgroups-fs.html

Filesytem (aufs)

- AnotherUnionFS or Advanced Multi Layered Unification FileSystem
- Each change committed to layer, like version control
- Layers are merged and presented as a single filesystem
- Provides rollback, history, etc.

hub.docker.com



Installing Docker

```
$ sudo apt-get install linux-image-extra-`uname -r`

$ curl -sSL https://get.docker.com/ubuntu/ | sudo sh

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

Run Docker

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

The image to run

The process to run inside that image

Do Something

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

# mkdir -p /opt/acme

# echo “Hello!” > /opt/acme/hello.txt

# exit
```

Persisting Changes

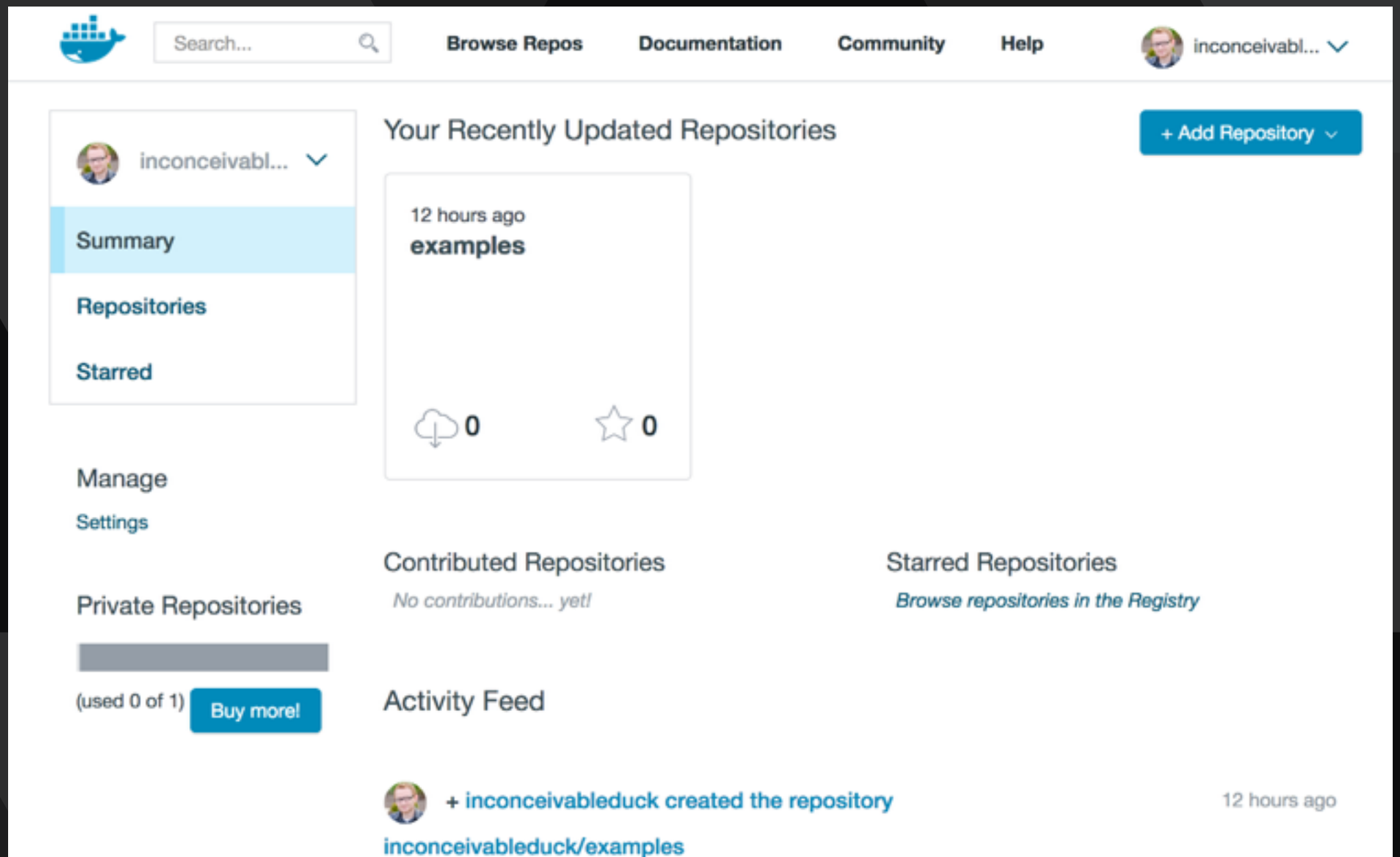
All of your changes are made to a new AUFS layer. If not committed, they are lost.

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

// Make some changes
```

```
$ docker ps

$ docker commit [CONTAINER_ID] hello:1.0

$ docker run -i -t hello:1.0 /bin/bash
```

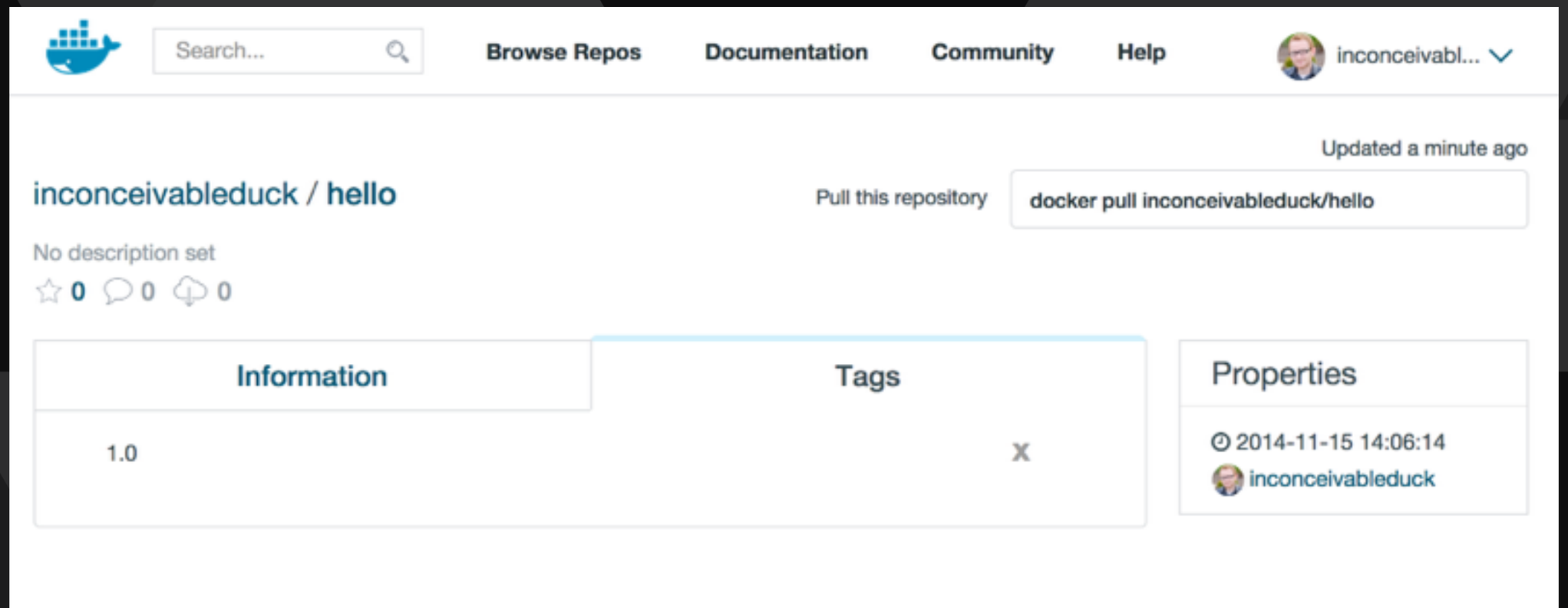
The Registry

Persist changes to the registry for easy portability: hub.docker.com



Push to Registry

```
$ docker login

$ docker commit [CONTAINER_ID] username/hello:1.0

$ docker push username/hello:1.0
```



Pull from Registry

```
$ docker login

$ docker pull username/hello:1.0

$ docker images

$ docker run -i -t username/hello:1.0 /bin/bash

# cat /opt/acme/hello.txt
```

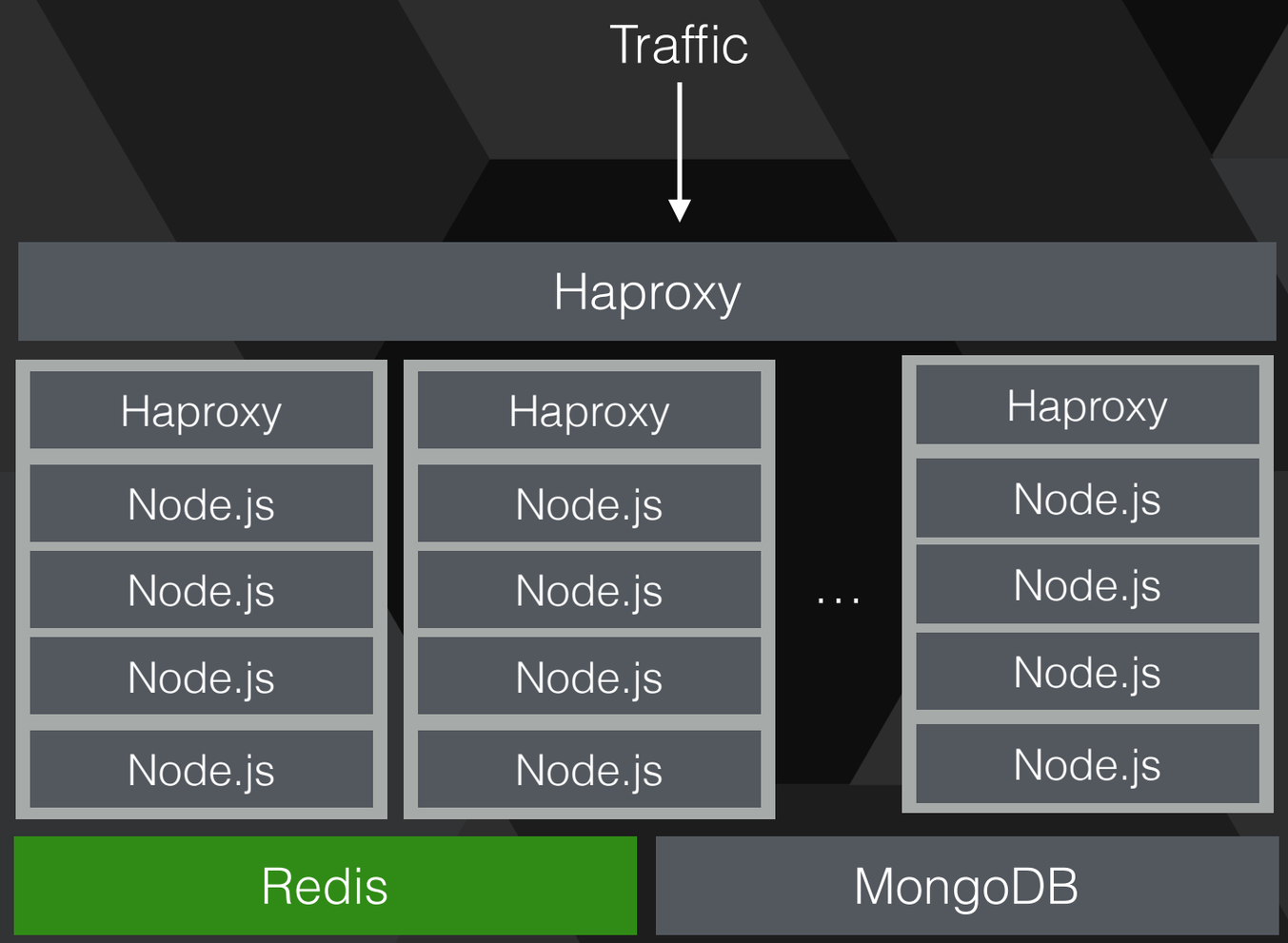
The Dockerfile

Easy way to automate image creation.

```
FROM ubuntu:14.04
RUN mkdir -p /opt/acme
RUN echo “Hello” > /opt/acme/hello.txt
CMD /bin/bash
```

```
$ docker build -t inconceivableduck/hello:1.0 .
$ docker run -i -t inconceivableduck/hello:1.0
```

Dockerize Redis



Dockerize Redis

```
FROM ubuntu:14.04
RUN apt-get update -y && apt-get install -y wget build-essential
RUN apt-get install -y supervisor
RUN cd /opt && wget http://download.redis.io/releases/redis-2.8.17.tar.gz
RUN cd /opt && tar -xvzf redis-2.8.17.tar.gz
RUN cd /opt/redis-2.8.17 && make
RUN mkdir -p /var/log/supervisor
RUN mkdir -p /data
ADD supervisor.conf /etc/supervisor/conf.d/supervisor.conf
ADD redis.conf /opt/redis-2.8.17/redis.conf
EXPOSE 6379
CMD "/usr/bin/supervisord"
```

supervisor.conf

```
[supervisord]
nodaemon=true

[program:redis]
command=/opt/redis-2.8.17/src/redis-server /opt/redis-2.8.17/redis.conf
```

http://supervisord.org

Running Redis

```
$ mkdir -p /mnt/data/redis
$ docker run -d -v /mnt/data:/data -p 6379:6379 redis
```

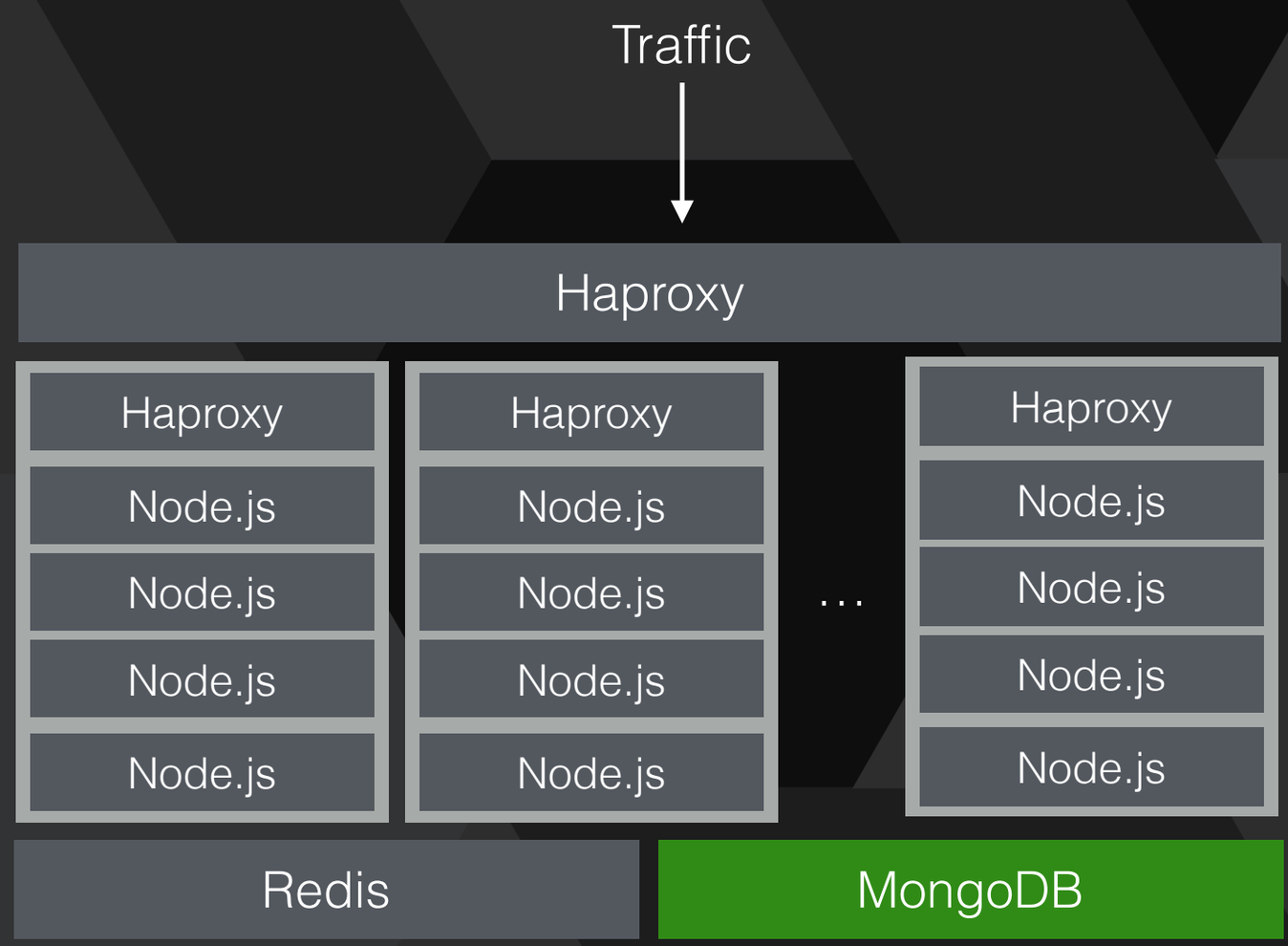
Mount /mnt/data on the host
to /data inside the container

Publish the container's port
6379 to the host's port 6379

docker-bash

```
// Install

$ curl --fail -L -O https://github.com/phusion/baseimage-
docker/archive/master.tar.gz && \

tar xzf master.tar.gz && \
sudo ./baseimage-docker-master/install-tools.sh

// Use
$ docker ps
$ docker-bash [CONTAINER_ID]
# supervisorctl
```

Dockerize MongoDB



Dockerize MongoDB

```
FROM ubuntu:14.04
RUN apt-get update -y && apt-get install -y wget
RUN apt-get install -y supervisor
RUN mkdir -p /var/log/supervisor
RUN mkdir -p /data
RUN mkdir -p /logs
RUN cd /opt && wget -nv http://fastdl.mongodb.org/linux/mongodb-
linux-x86_64-2.6.5.tgz
RUN cd /opt && tar -xvzf mongodb-linux-x86_64-2.6.5.tgz
ADD mongo.conf /opt/mongodb-linux-x86_64-2.6.5/mongo.conf
ADD supervisor.conf /etc/supervisor/conf.d/supervisor.conf
EXPOSE 27017
CMD "/usr/bin/supervisord"
```

supervisor.conf

```
[supervisord]
nodaemon=true

[program:mongo]
command=/opt/mongodb-linux-x86_64-2.6.5/bin/mongod --config
  /opt/mongodb-linux-x86_64-2.6.5/mongo.conf
```

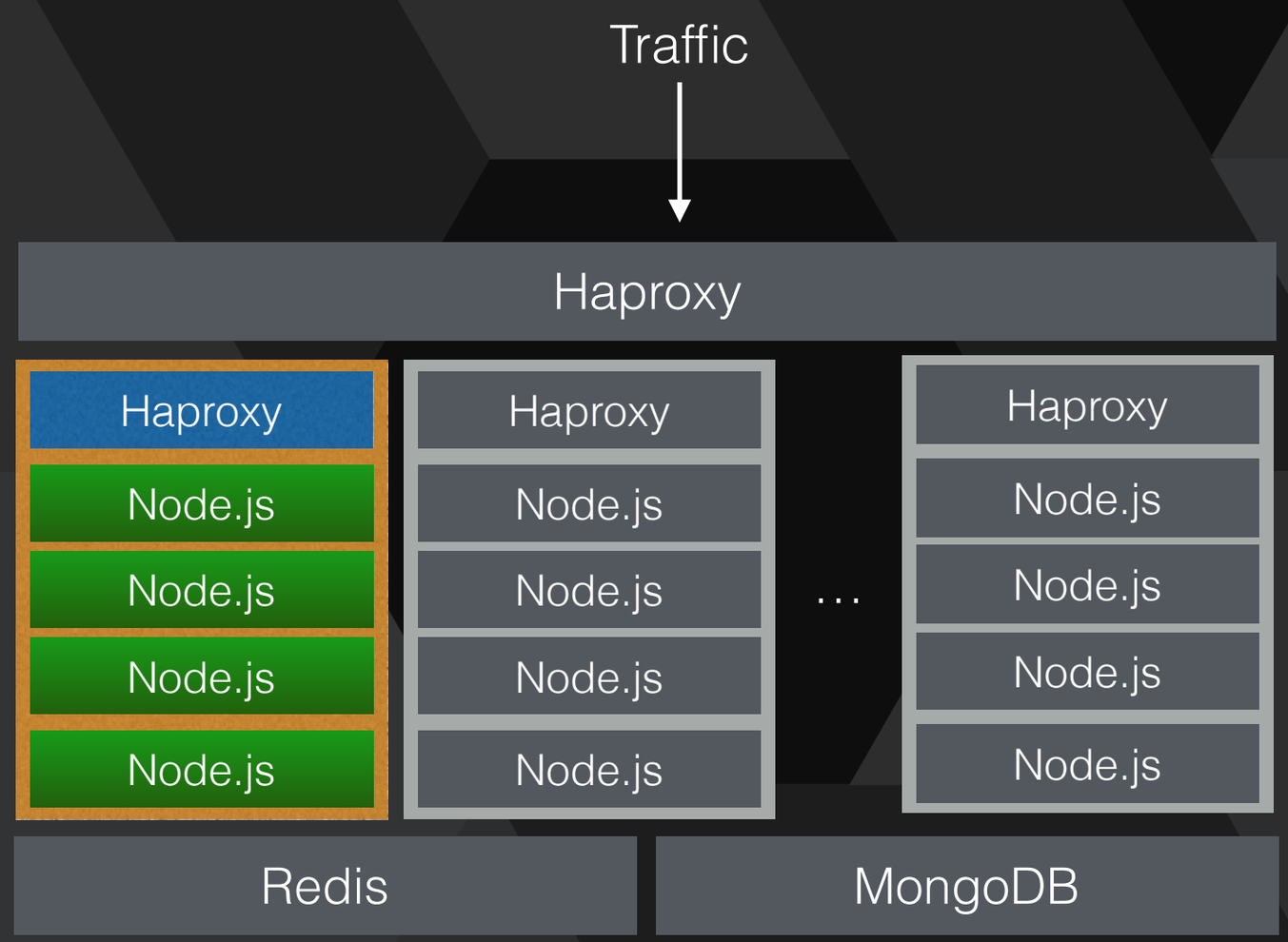
mongo.conf

```
logpath = /logs/mongo.log
logappend = true
dbpath = /data/mongo
smallfiles = true
```

Run MongoDB

```
$ mkdir -p /mnt/data/mongo
$ mkdir -p /mnt/logs
$ docker run -d -v /mnt/data:/data -v /mnt/logs:/logs -p 27017:27017 mongo
```

Dockerize Node.js



Dockerize Node.js

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('My port is ' + process.env.PORT);
}).listen(process.env.PORT);
console.log('Server running on port ' + process.env.PORT);
```

Dockerfile

```
FROM ubuntu:14.04
RUN apt-get update -y && apt-get install -y curl wget git supervisor build-essential
RUN mkdir -p /var/log/supervisor
RUN mkdir -p /logs
RUN cd /opt && git clone https://github.com/InconceivableDuck/Nodevember.git
RUN curl https://raw.githubusercontent.com/isaacs/nave/master/nave.sh > /opt/nave.sh
RUN bash /opt/nave.sh usemain 0.10.33
RUN cd /opt && wget http://www.haproxy.org/download/1.5/src/haproxy-1.5.3.tar.gz
RUN cd /opt && tar xzf haproxy-1.5.3.tar.gz
RUN cd /opt/haproxy-1.5.3 && make TARGET=linux2628 && make install
ADD haproxy.cfg /opt/haproxy-1.5.3/haproxy.cfg
ADD supervisor.conf /etc/supervisor/conf.d/supervisor.conf
EXPOSE 80 8081 8082 8083 8084
CMD "/usr/bin/supervisord"
```

supervisor.conf

```
[supervisord]
nodaemon=true

[program:app1]
command=node /opt/Nodevember/app/index.js
environment=PORT="8081"
stdout_logfile=/logs/app1.log
stdout_logfile_maxbytes=1GB
redirect_stderr=true

...

[program:haproxy]
command=haproxy -f /opt/haproxy-1.5.3/haproxy.cfg
stdout_logfile=/logs/haproxy.log
stdout_logfile_maxbytes=1GB
redirect_stderr=true
```

haproxy.cfg

```
...
listen app *:80
    mode http
    balance roundrobin
    server app1 127.0.0.1:8081
    server app2 127.0.0.1:8082
    server app3 127.0.0.1:8083
    server app4 127.0.0.1:8084
```

Run Node.js App

```
$ mkdir -p /mnt/logs
$ docker run -d -v /mnt/logs:/logs -p 80:80 app
```

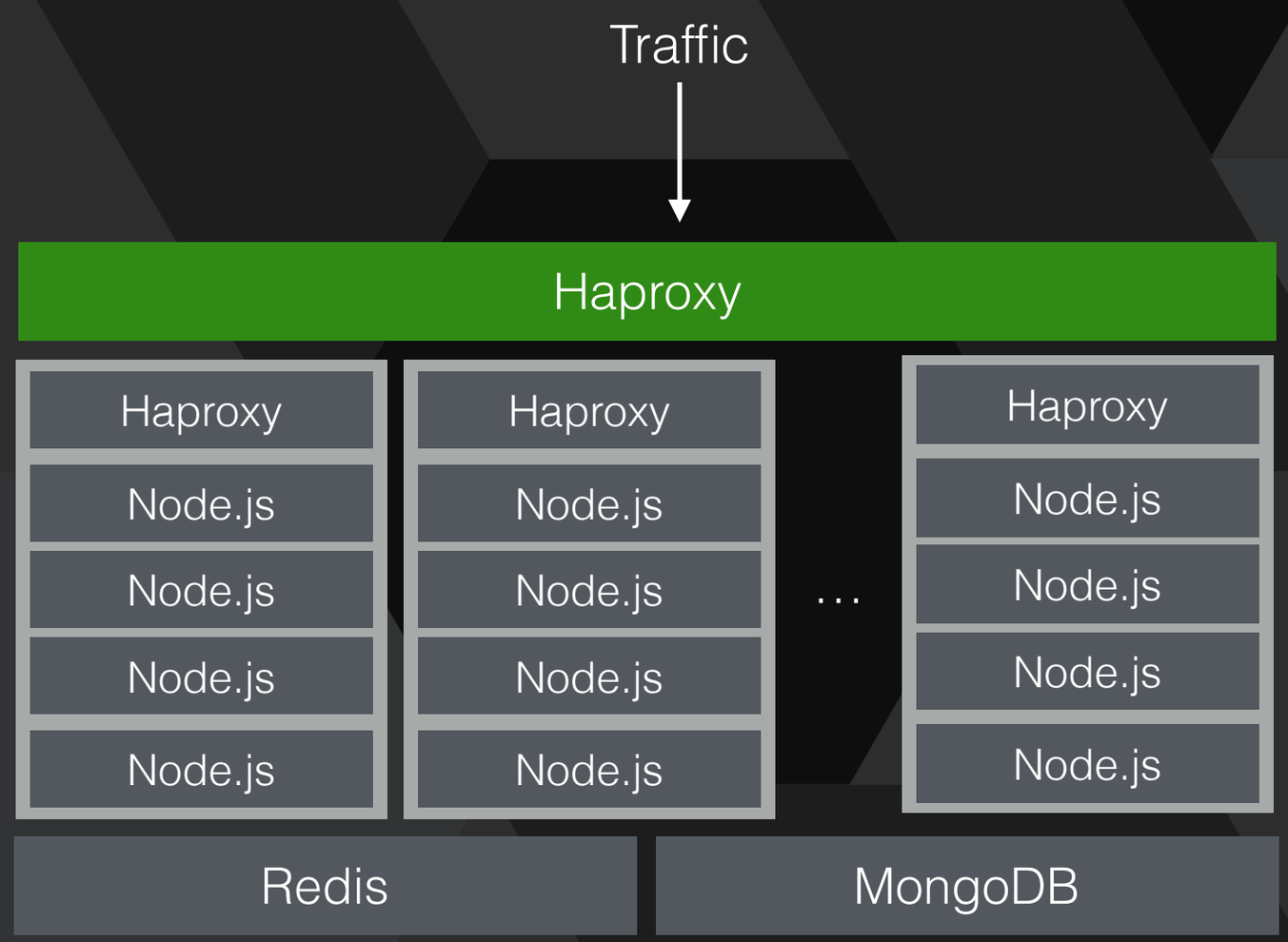
Load Balancing

```
$ docker pull username/app:1.0
$ docker run -d -v /mnt/logs:/logs -p 80:80 app:1.0
```

Load Balancing

```
$ docker run -d -v /mnt/logs:/logs -p 81:80 app
$ docker run -d -v /mnt/logs:/logs -p 82:80 app
$ docker run -d -v /mnt/logs:/logs -p 83:80 app
$ docker run -d -v /mnt/logs:/logs -p 84:80 app
```

Scaling Node.js



Dockerfile

```
FROM ubuntu:14.04
RUN apt-get update -y && apt-get install -y curl wget supervisor build-essential
RUN mkdir -p /var/log/supervisor
RUN mkdir -p /logs
RUN cd /opt && wget http://www.haproxy.org/download/1.5/src/haproxy-1.5.3.tar.gz
RUN cd /opt && tar xzf haproxy-1.5.3.tar.gz
RUN cd /opt/haproxy-1.5.3 && make TARGET=linux2628 && make install
ADD haproxy.cfg /opt/haproxy-1.5.3/haproxy.cfg
ADD supervisor.conf /etc/supervisor/conf.d/supervisor.conf
EXPOSE 80
CMD "/usr/bin/supervisord"
```

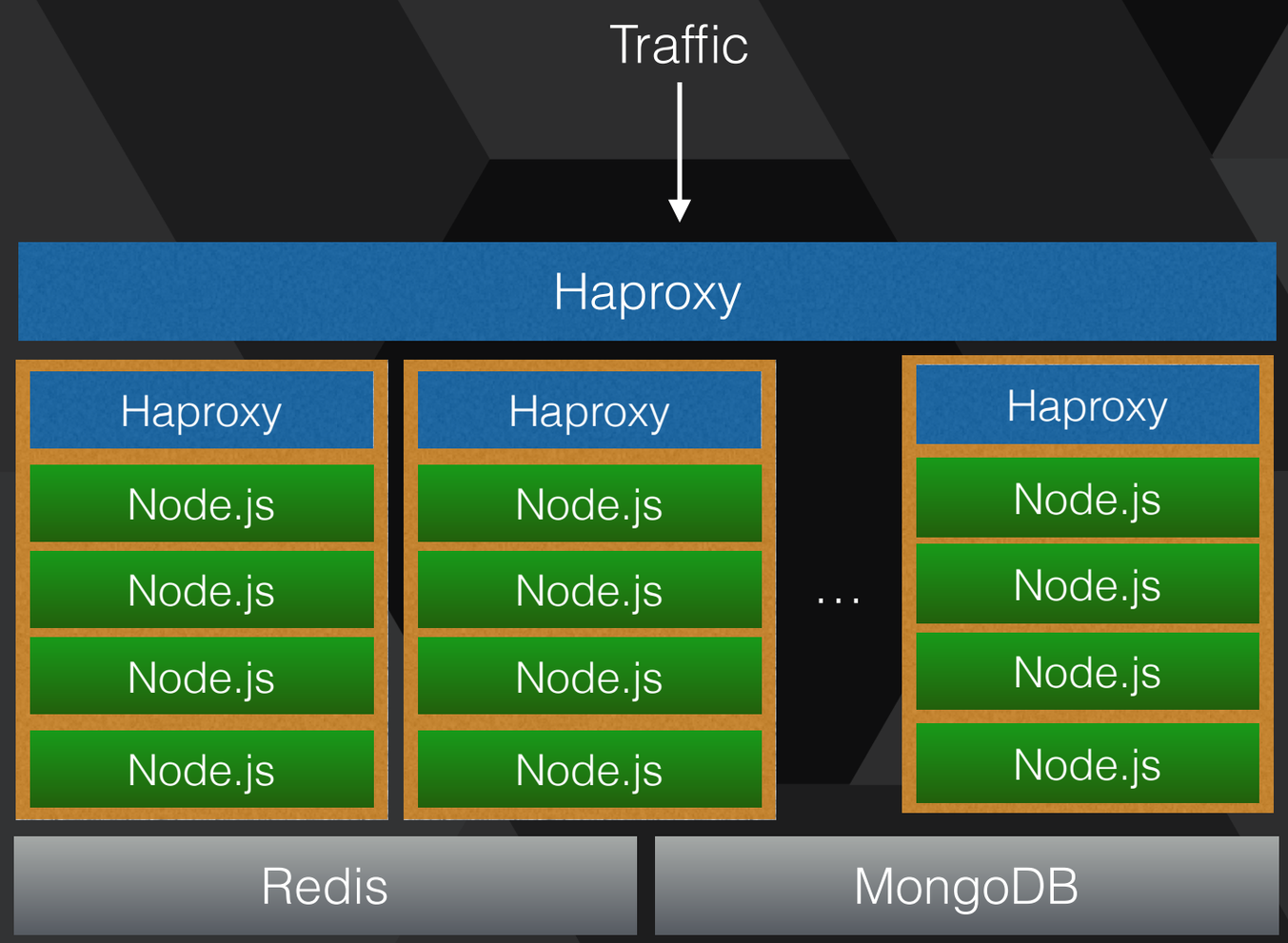
supervisor.conf

```
[supervisord]
nodaemon=true

[program:haproxy]
command=haproxy -f /opt/haproxy-1.5.3/haproxy.cfg
stdout_logfile=/logs/haproxy.log
stdout_logfile_maxbytes=1GB
redirect_stderr=true
```

haproxy.cfg

```
...
listen app *:80
    mode http
    balance roundrobin
    server app1 appserver.1.mycompany.net:81
    server app2 appserver.1.mycompany.net:82
```

The System


Traffic
Haproxy
Haproxy
Node.js
Node.js
Node.js
Node.js
Haproxy
Node.js
Node.js
Node.js
Node.js
...
Haproxy
Node.js
Node.js
Node.js
Node.js
Redis
MongoDB

The System

```
# Redis
docker run -d -v /mnt/data:/data -p 6379:6379 redis

# Mongo
docker run -d -v /mnt/data:/data -v /mnt/logs:/logs -p 27017:27017 mongo

# App servers
docker run -d -v /mnt/logs:/logs -p 81:80 app
docker run -d -v /mnt/logs:/logs -p 82:80 app
docker run -d -v /mnt/logs:/logs -p 83:80 app
docker run -d -v /mnt/logs:/logs -p 84:80 app

# Load Balancer
docker run -d -v /mnt/logs:/logs -p 80:80 haproxy
```

Common Gotchas

- Containers go away when the process exits
  - If you run /bin/bash, attach, and then type “exit”, the container is killed
- Data is gone when containers exit. Keep important stuff on the host filesystem
- Keep Dockerfiles short
  - Move the bulk of work into a shell script and execute it
- Use Dockerfiles
  - Avoid making changes directly to the container

Thanks

Brandon Cannaday | @TheReddest
modulus.io