



# Deploying a Scalable Node.js Application with Docker

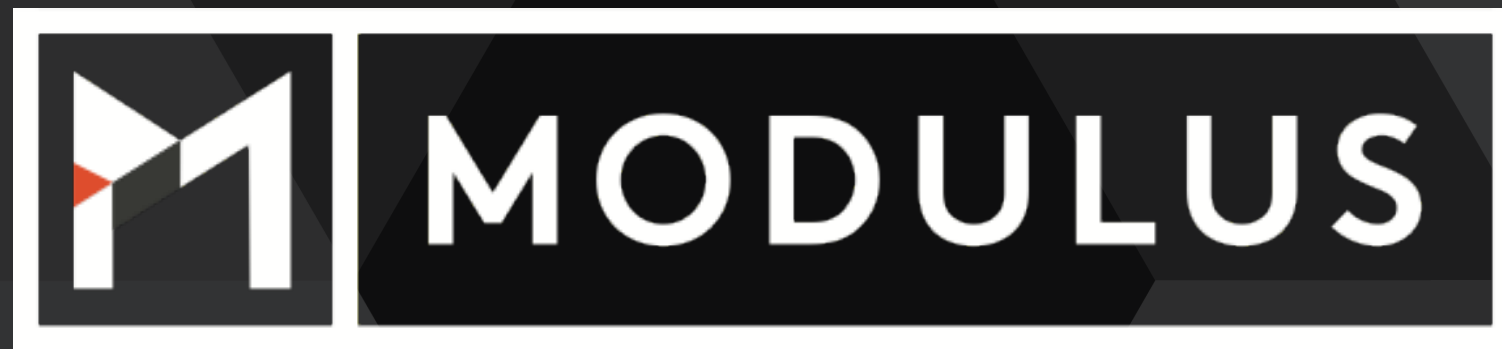
Brandon Cannaday | @TheReddest

**Follow Along!**

Come up and grab a card with a server you can use



# About me



A  **PROGRESS** COMPANY

Brandon Cannaday | @TheReddest  
modulus.io

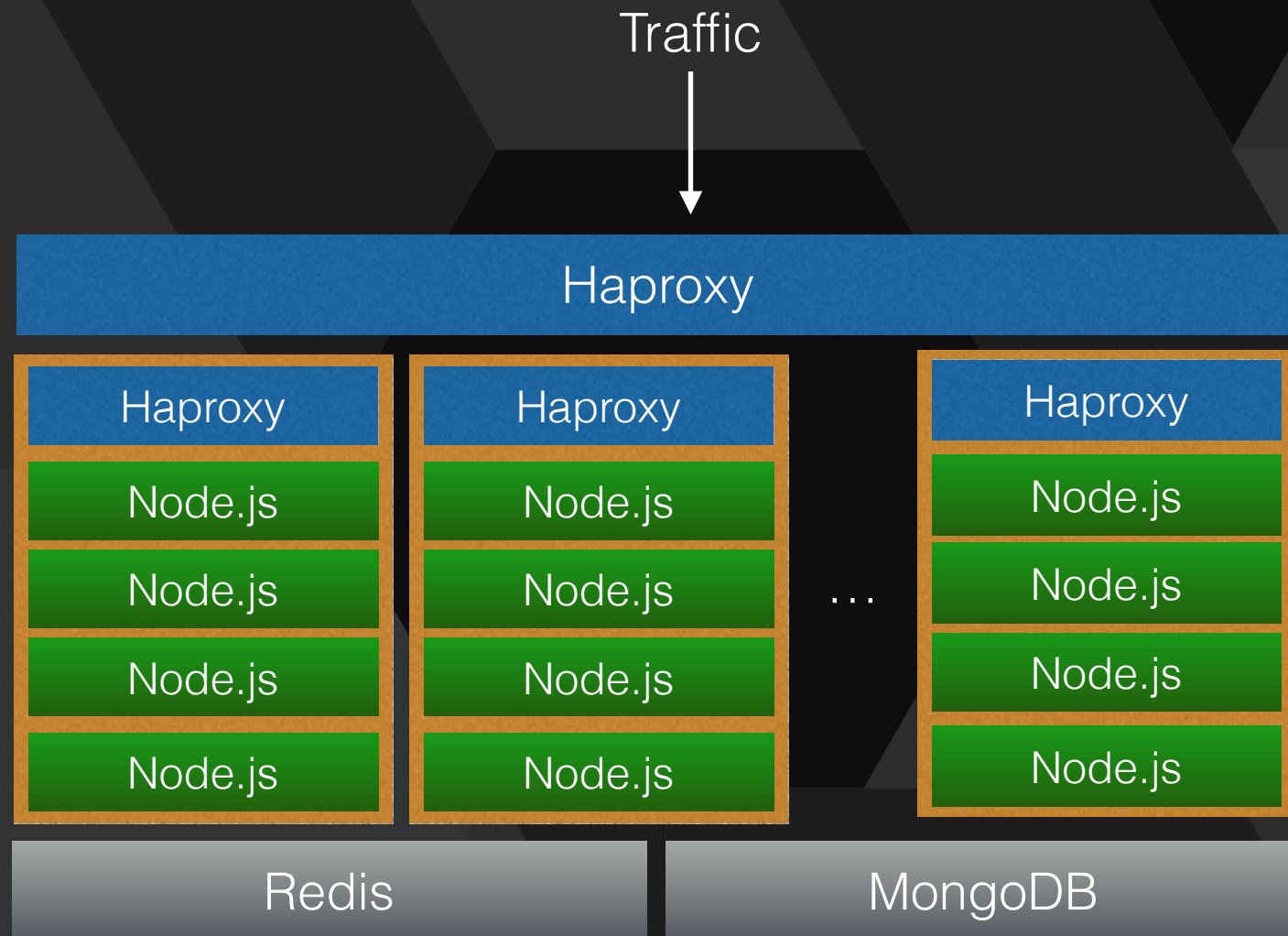
# Follow Along

<https://github.com/InconceivableDuck/Nodevenember>

Test servers provided by:



# What we're building



# Agenda

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

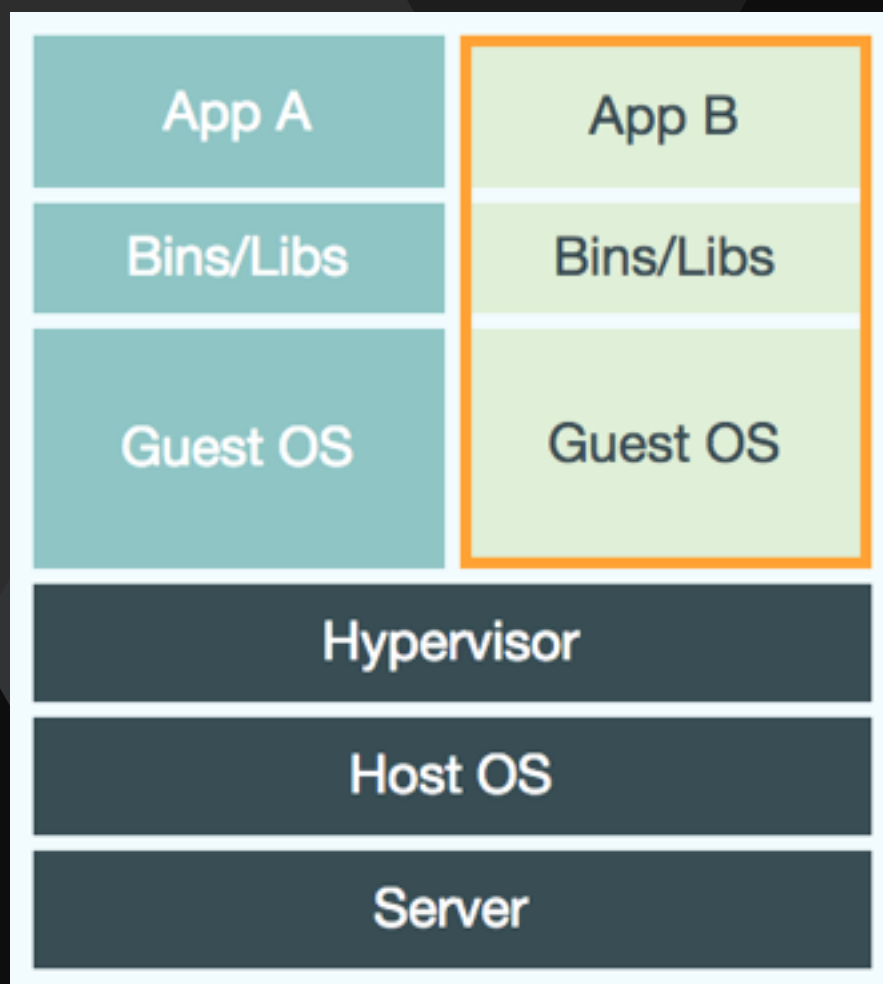
# Docker



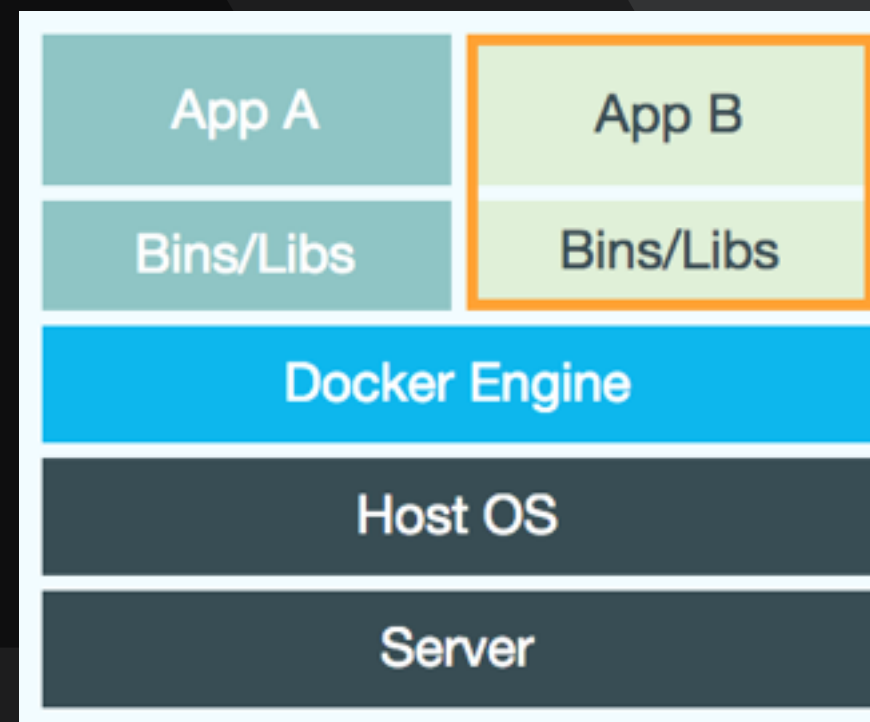
- “Virtualizes” the application layer
  - A bunch of CRUD on top of Linux cgroups
  - Layered filesystem
  - A registry

# 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
```

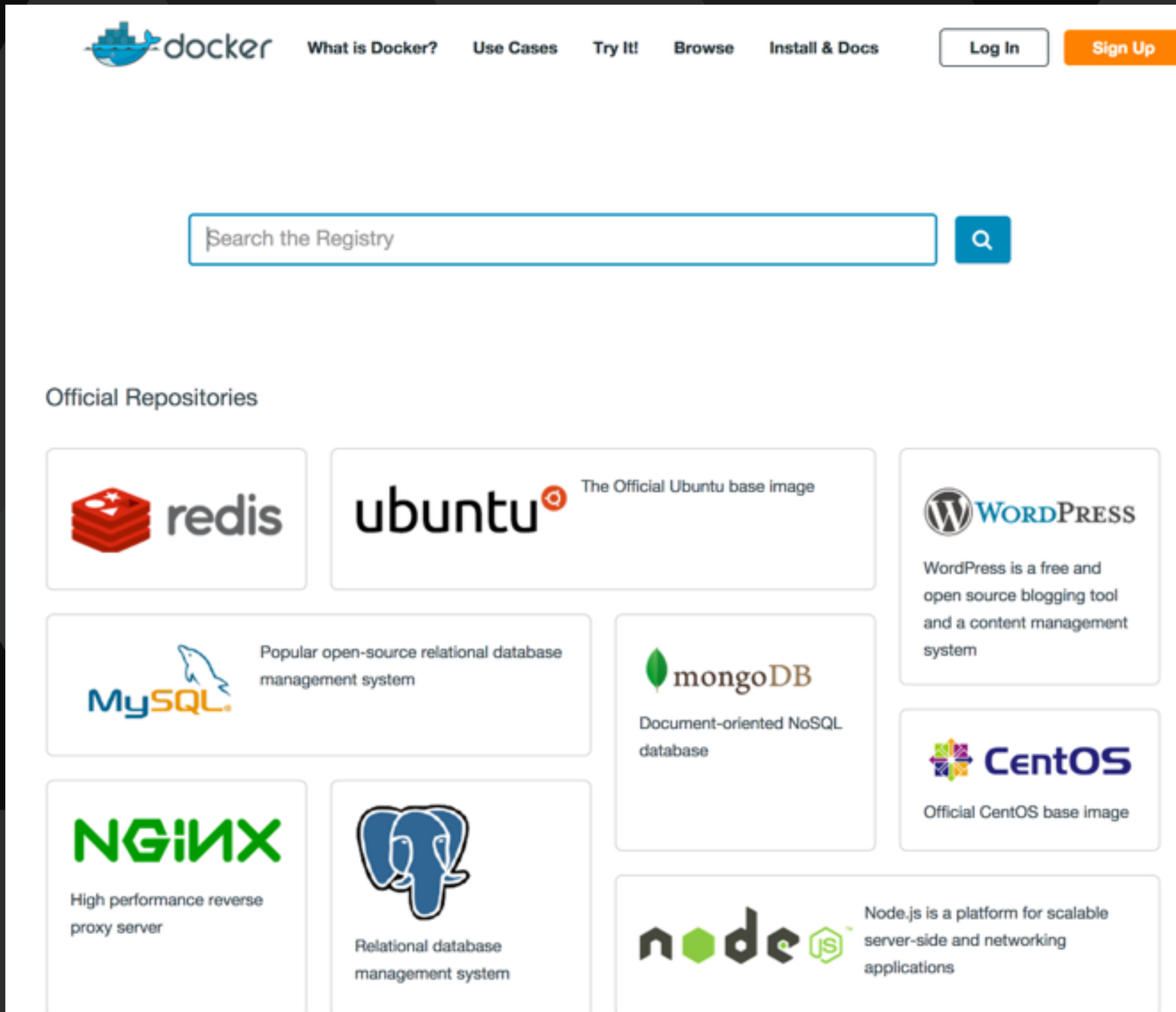
```
echo FROZEN > /cgroup1/child1/freezer.state
```



# Filesytem (aufs)

- AnotherUnionES 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



The screenshot displays the Docker Hub website interface. At the top, the Docker logo is followed by navigation links: "What is Docker?", "Use Cases", "Try It!", "Browse", and "Install & Docs". On the right side of the header, there are "Log In" and "Sign Up" buttons. Below the header is a search bar labeled "Search the Registry" with a magnifying glass icon. The main content area is titled "Official Repositories" and features a grid of repository cards. Each card includes a logo, the repository name, and a brief description.

**Official Repositories**

- redis**: Redis logo and name.
- ubuntu**: The Official Ubuntu base image. Includes the Ubuntu logo and a description: "The Official Ubuntu base image".
- WordPress**: WordPress logo and name. Description: "WordPress is a free and open source blogging tool and a content management system".
- MySQL**: MySQL logo and name. Description: "Popular open-source relational database management system".
- mongoDB**: MongoDB logo and name. Description: "Document-oriented NoSQL database".
- CentOS**: CentOS logo and name. Description: "Official CentOS base image".
- NGINX**: NGINX logo and name. Description: "High performance reverse proxy server".
- PostgreSQL**: PostgreSQL logo (elephant) and name. Description: "Relational database management system".
- node**: Node.js logo and name. Description: "Node.js is a platform for scalable server-side and networking applications".

# 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](https://hub.docker.com)

The screenshot shows the Docker Hub user profile for 'inconceivableduck'. The top navigation bar includes the Docker logo, a search bar, and links for 'Browse Repos', 'Documentation', 'Community', and 'Help'. The user's profile picture and name are on the right. The main content area is divided into several sections:

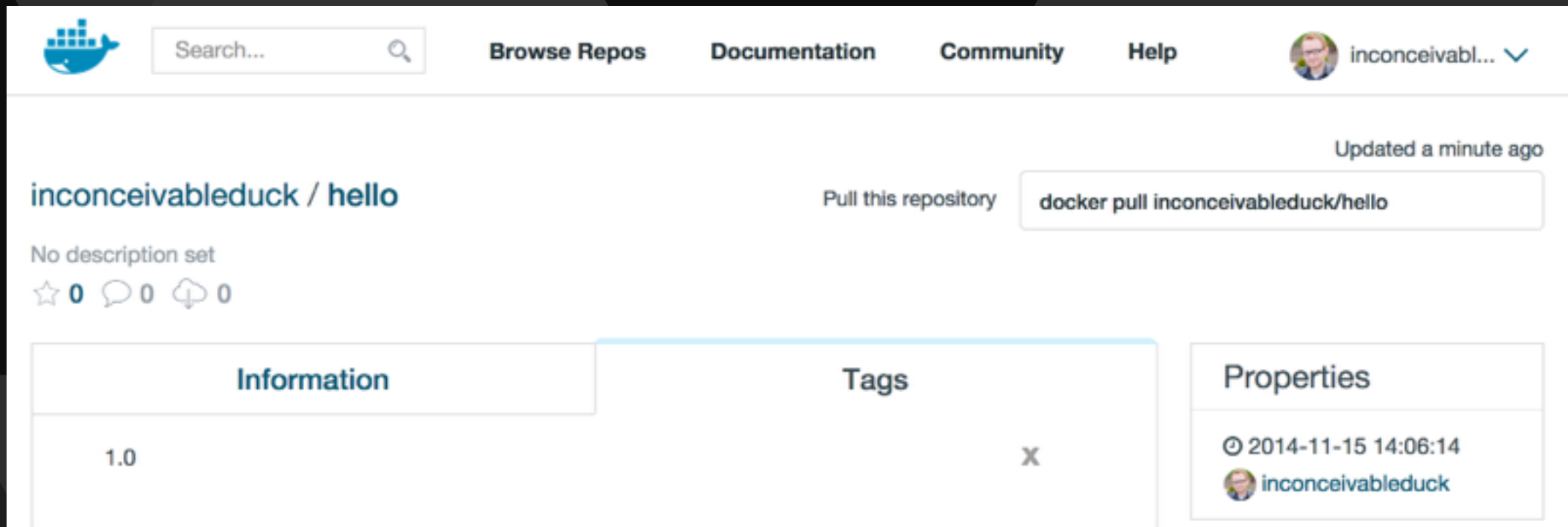
- Left Sidebar:** Contains the user's profile picture and name, a 'Summary' tab (highlighted), 'Repositories', and 'Starred'. Below these are links for 'Manage' and 'Settings'. At the bottom, it shows 'Private Repositories' with a progress bar indicating '(used 0 of 1)' and a 'Buy more!' button.
- Top Right:** A blue button labeled '+ Add Repository'.
- Center:** A section titled 'Your Recently Updated Repositories' showing a repository named 'examples' updated '12 hours ago'. It displays 0 pulls and 0 stars.
- Bottom Left:** A section titled 'Contributed Repositories' with the message 'No contributions... yet!'.
- Bottom Right:** A section titled 'Starred Repositories' with the message 'Browse repositories in the Registry'.
- Activity Feed:** At the bottom, it shows an activity entry: '+ inconceivableduck created the repository' with a link to 'inconceivableduck/examples' and a timestamp of '12 hours ago'.

# Push to Registry

```
$ docker login
```

```
$ docker commit [CONTAINER_ID] username/hello:1.0
```

```
$ docker push username/hello:1.0
```



The screenshot shows the Docker Hub interface for the repository 'inconceivableduck / hello'. The top navigation bar includes the Docker logo, a search bar, and links for 'Browse Repos', 'Documentation', 'Community', and 'Help'. The user 'inconceivable...' is logged in. The repository page shows it was 'Updated a minute ago'. Below the repository name, it states 'No description set' and shows 0 stars, 0 comments, and 0 pulls. A 'Pull this repository' button is present, with the command 'docker pull inconceivableduck/hello' displayed. The 'Tags' tab is active, showing a single tag '1.0' with a cross icon. The 'Properties' tab shows the creation date '© 2014-11-15 14:06:14' and the user 'inconceivableduck'.

Updated a minute ago

inconceivableduck / hello

No description set

☆ 0 💬 0 📦 0

Pull this repository `docker pull inconceivableduck/hello`

Information	Tags	Properties
1.0	x	© 2014-11-15 14:06:14 inconceivableduck



# 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

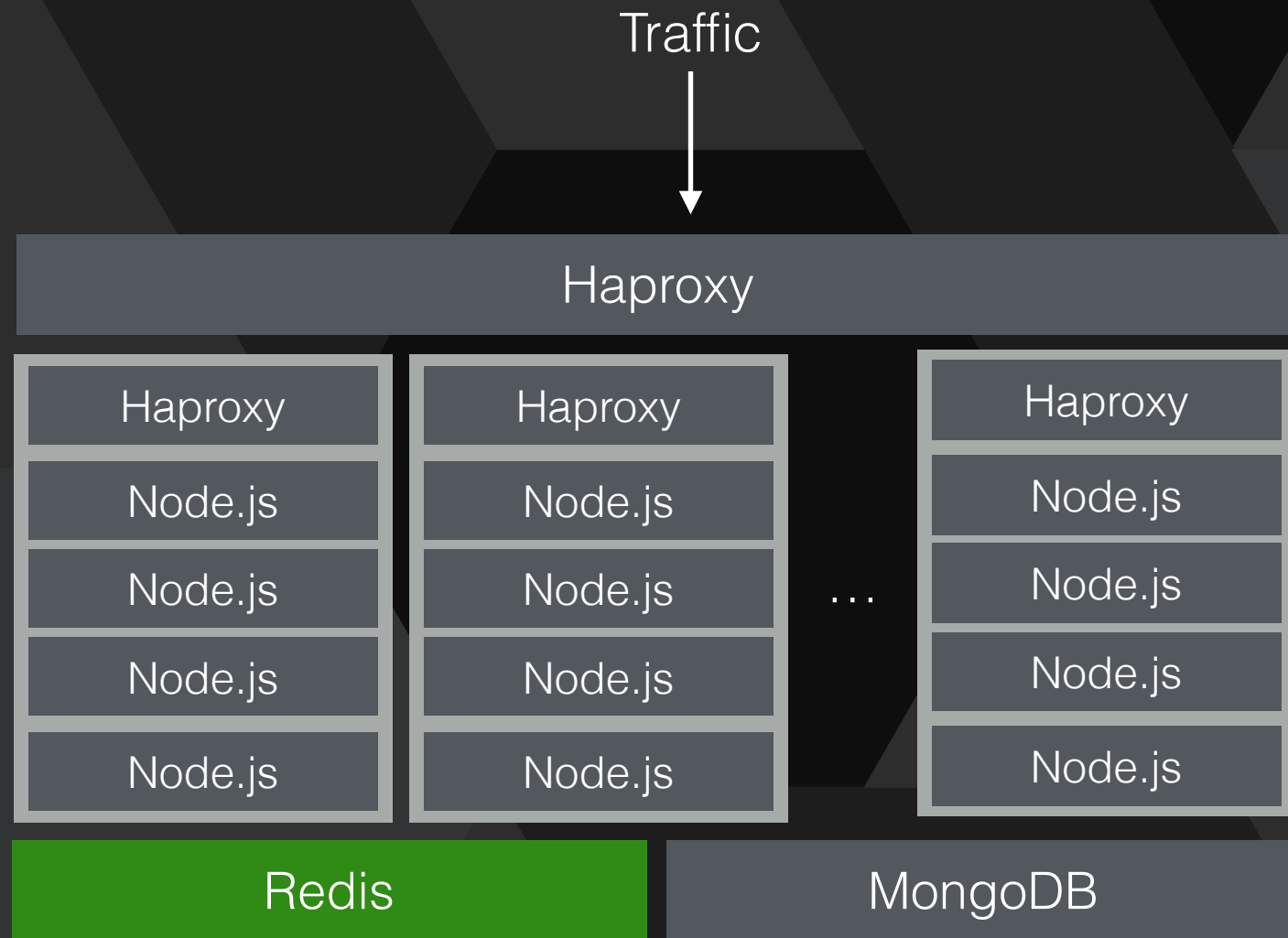
Easy way to automate image creation.

```
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
```


# 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

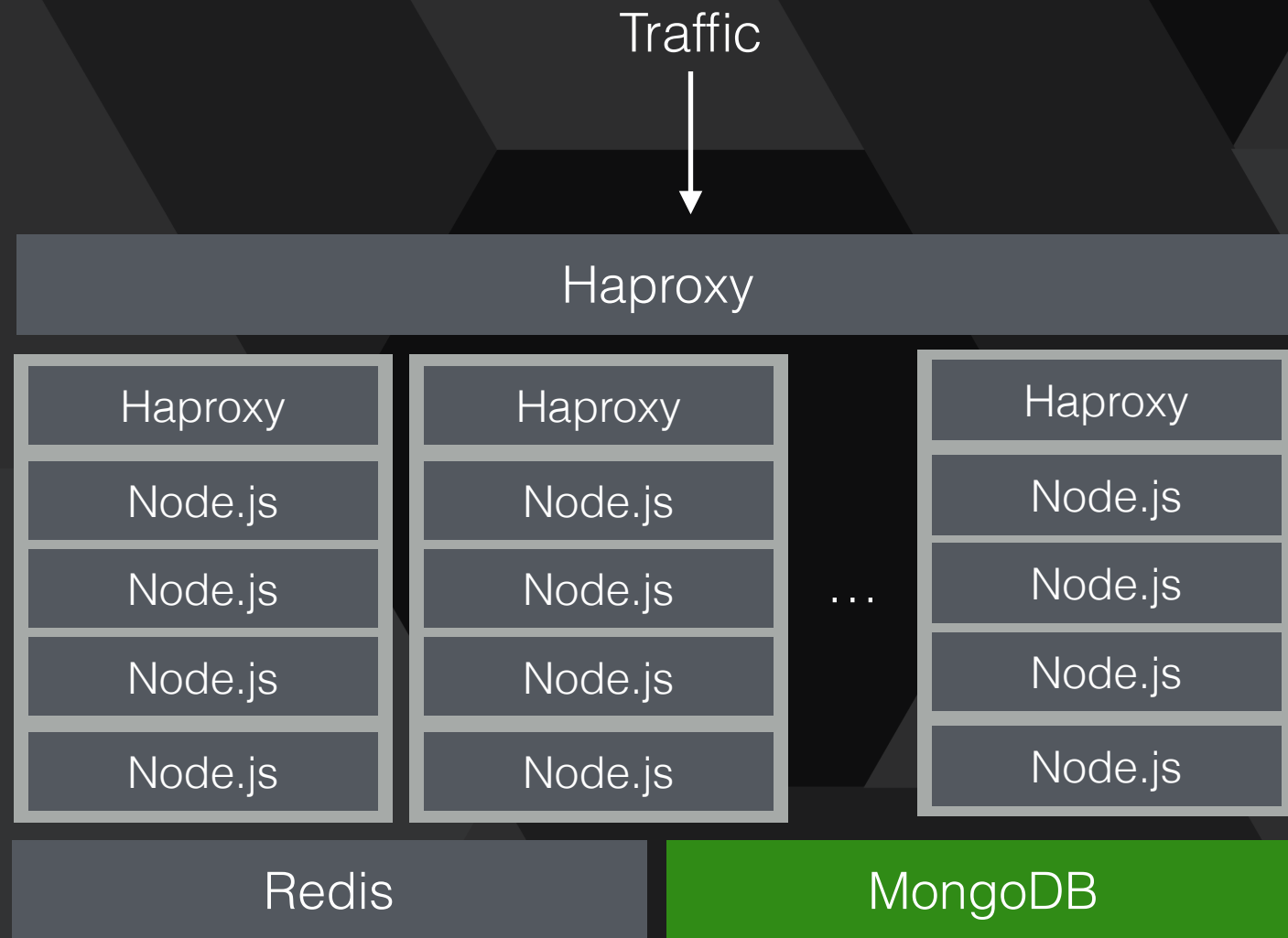
// Install

```
$ 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
```

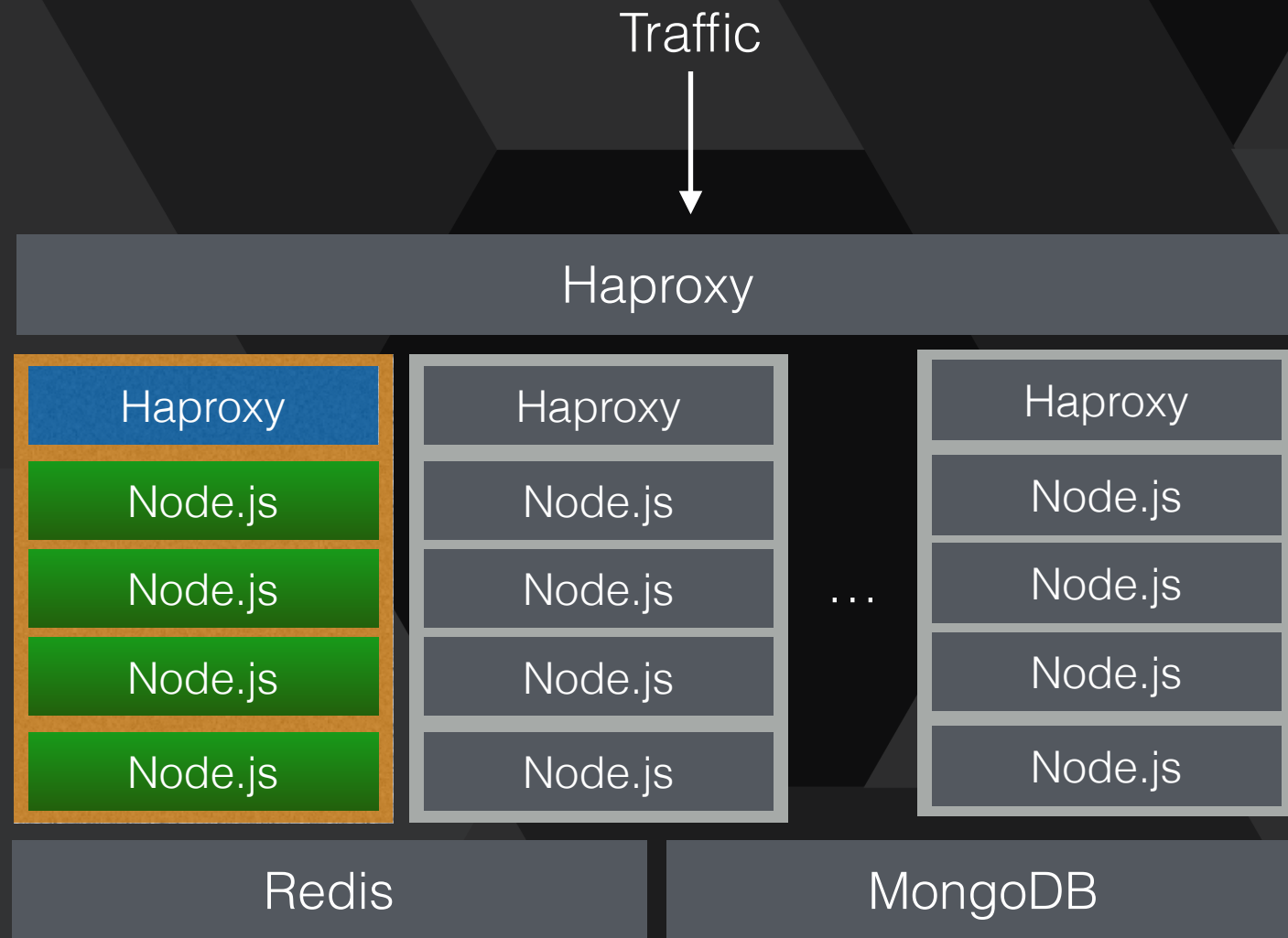
# Run MongoDB

```
$ 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/Nodevenember.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/Nodevenember/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

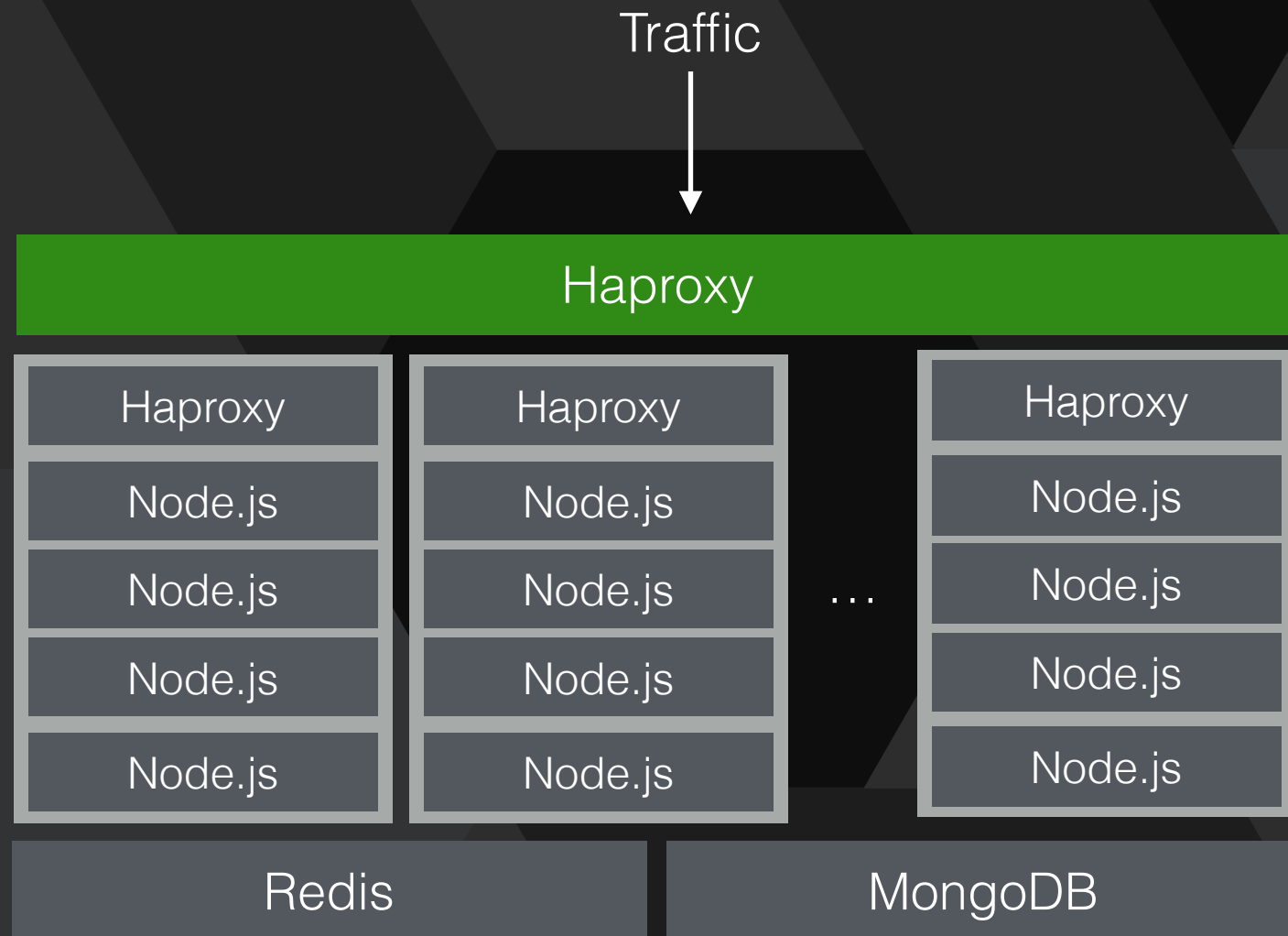
```
$ docker pull username/app:1.0
```

```
$ 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
```

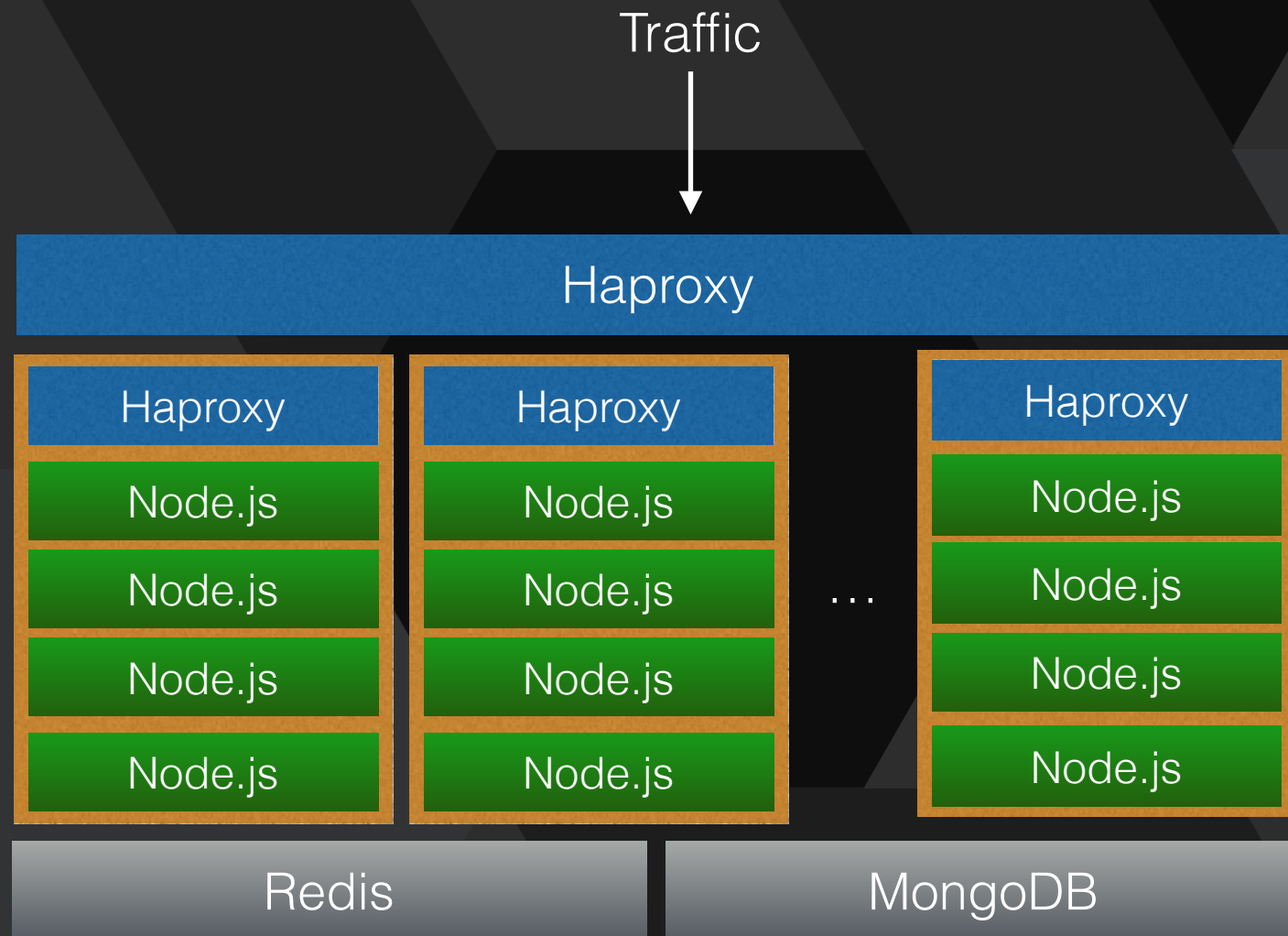
```
    balance roundrobin
```

```
    server app1 appserver.1.mycompany.net:81
```

```
    server app2 appserver.1.mycompany.net:82
```



# The System



# 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