AARON BROWN

# DOCKER & INTEGRATION TESTING

# ME

*aaronbbrown* - Site Reliability Engineer @ GitHub

*@aaronbbrown777*

I sometimes write things: http://blog.9minutesnooze.com

# WHO ARE YOU?

# OVERVIEW

- What is Docker?
- How can I gain confidence in deployments by using Docker for integration tests

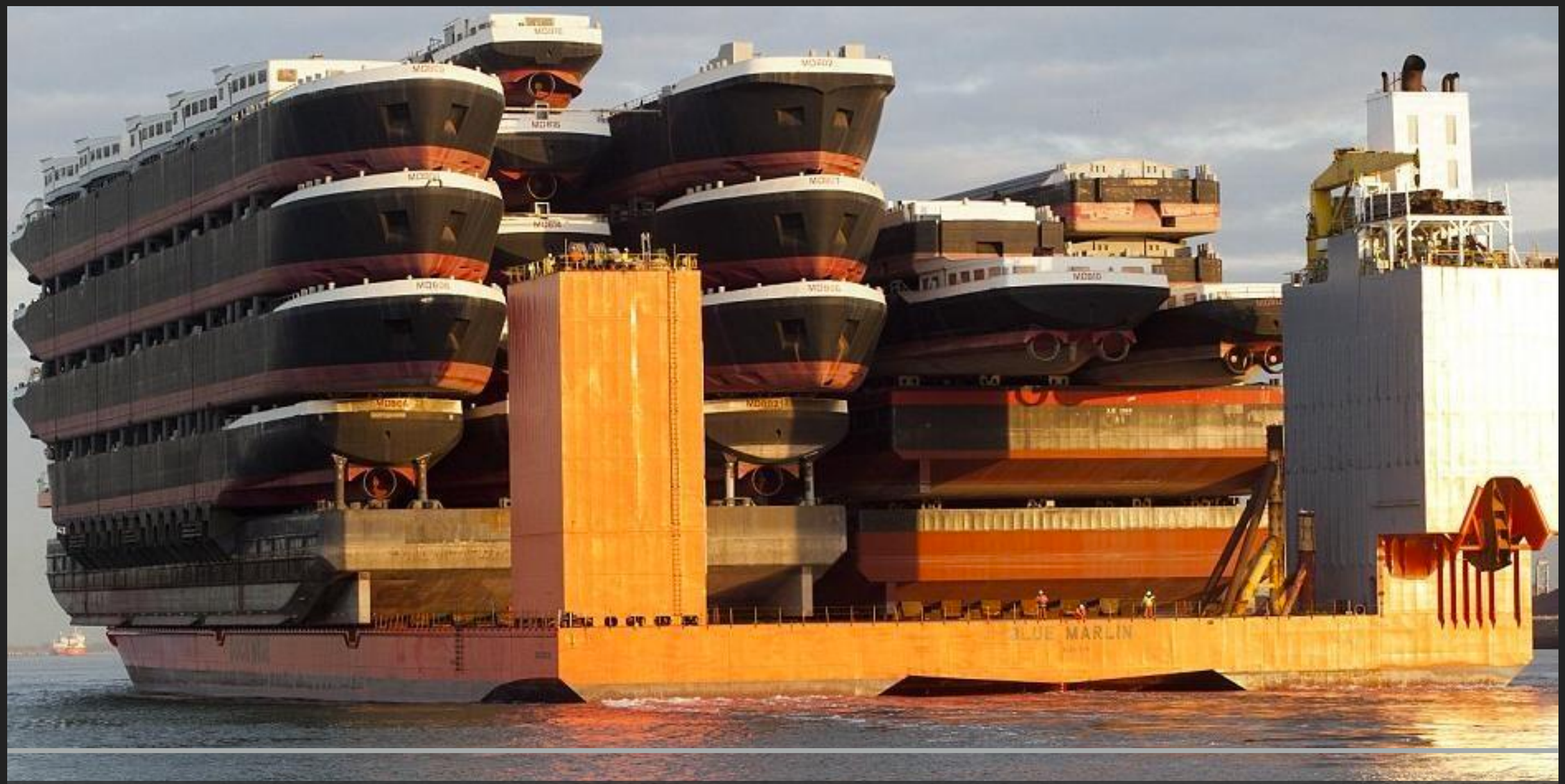
# WHAT IS DOCKER?

- Allows processes to run inside a light-weight “container”
- Virtual Machine-like isolation
- Near bare-metal performance (on Linux)
- built on top of a bunch of Linux technologies like LXC, namespaces, and cgroups
- Easily portable between systems
- **Run services without installing dependencies on host**

# WHAT ABOUT OS X & WINDOWS?

- Even though it's Linux, it works on Mac and Windows
- Docker for Mac and Docker for Windows transparently run a Linux VM for you
- Performance penalty for the indirection

# SHIPPING CONTAINERS




# SHIP SHIPPING SHIP SHIPPING SHIPPING SHIPS

# FIVE THINGS TO UNDERSTAND

- Image - a blueprint for containers
- Container - a running copy of the image
- TCP & UDP ports can be published to the host
- Network can be shared between containers
- Files can be shared between the host and container with volumes

# PULLING AN IMAGE FROM DOCKER HUB

```
$ docker pull debian:jessie

jessie: Pulling from library/debian
693502eb7dfb: Pull complete
Digest:
sha256:52af198afd8c264f1035206ca66a5c48e602afb32d
c912ebf9e9478134601ec4
Status: Downloaded newer image for debian:jessie
```

# RUNNING A CONTAINER INTERACTIVELY

```
$ docker run --rm -it debian:jessie bash

root@9fb5f7a744e2:/# apt-get update
root@9fb5f7a744e2:/# apt-get install -y curl
root@9fb5f7a744e2:/# curl http://icanhazip.com
74.76.253.167

root@9fb5f7a744e2:/etc# ps -ef
UID   PID  PPID  C STIME TTY          TIME CMD
root    1     0  0 23:25 ?    00:00:00 bash
root    7     1  0 23:26 ?    00:00:00 ps -ef
```

# BUILD A NEW IMAGE WITH A DOCKERFILE

```
# Dockerfile
FROM debian:jessie

RUN apt-get update && \
    apt-get install -y curl; \
    apt-get clean

ENTRYPOINT [“curl”]
```

# BUILD AND RUN THE IMAGE

```
$ docker build -t sharatoga-curl .
…

# `curl -s icanhazip.com` INSIDE the container
$ docker run --rm sharatoga-curl -s icanhazip.com
74.76.253.167
```

# PUBLISH A PORT TO THE HOST

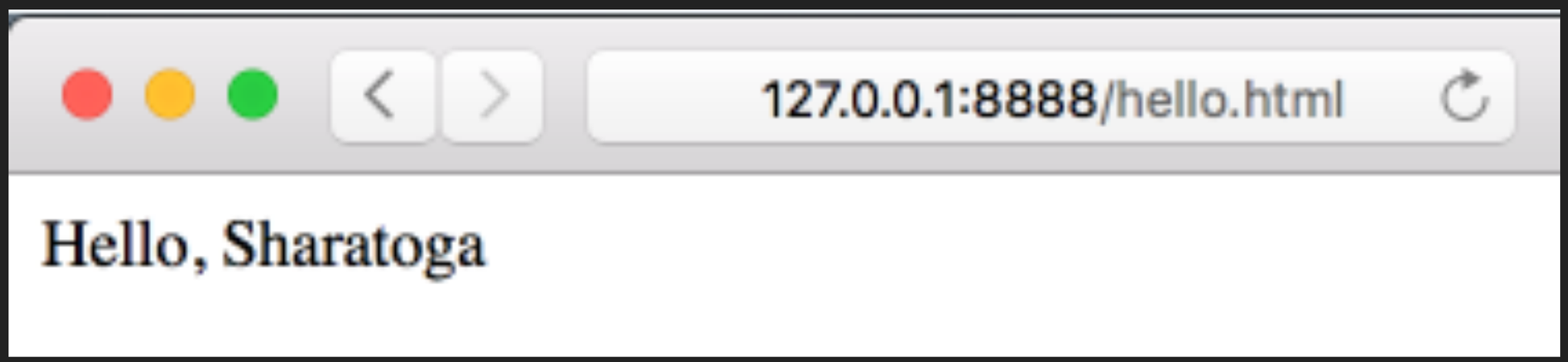
```
$ docker run -p 8888:80 -d \
  --name sharatoga-nginx nginx:1.11
```

# LINK CONTAINERS TOGETHER

```
$ docker run --rm \
    --link sharatoga-nginx \
    sharatoga-curl -sI http://sharatoga-nginx
HTTP/1.1 200 OK
Server: nginx/1.11.10
…
```

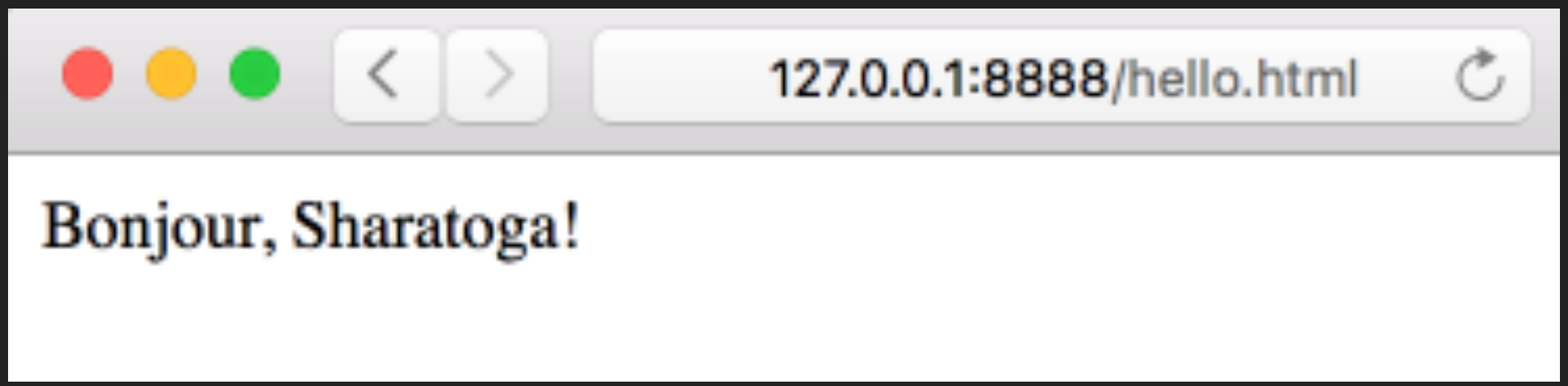
# CUSTOMIZE CONTAINER WITH VOLUMES

```
$ echo 'Hello, Sharatoga' > hello.html

$ docker run -d \
    -v `pwd`:/usr/share/nginx/html:ro \
    -p 8888:80 \
--name sharatoga-nginx nginx:1.11
```



# CHANGE FILES LIVE

```
$ echo 'Bonjour, Sharatoga!' > hello.html
```



# DOCKER-COMPOSE

```
version: '3'
services:
  nginx:
    image: nginx:1.11
    volumes:
      - ../custom-nginx:/usr/share/nginx/html:ro
  curl:
    build: ../curl
    entrypoint:
      - bash
      - -c
      - |-
        while : ; do
          curl -s http://nginx/hello.html;
          sleep 1;
        done
```

# LET'S START THIS THING

```
$ docker-compose up
Building curl
…
Creating compose_curl_1
Creating compose_nginx_1
Attaching to compose_curl_1, compose_nginx_1

curl_1   | Bonjour, Sharatoga!
nginx_1  | 172.18.0.2 - - [19/Mar/2017:18:33:39
+0000] "GET /hello.html HTTP/1.1" 200 17 "-" "curl/
7.38.0" "-"
```

BUT AARON, I CAN DO ALL THAT WITH VIRTUAL MACHINES!

Naysayers

# STARTUP IS WICKED FAST

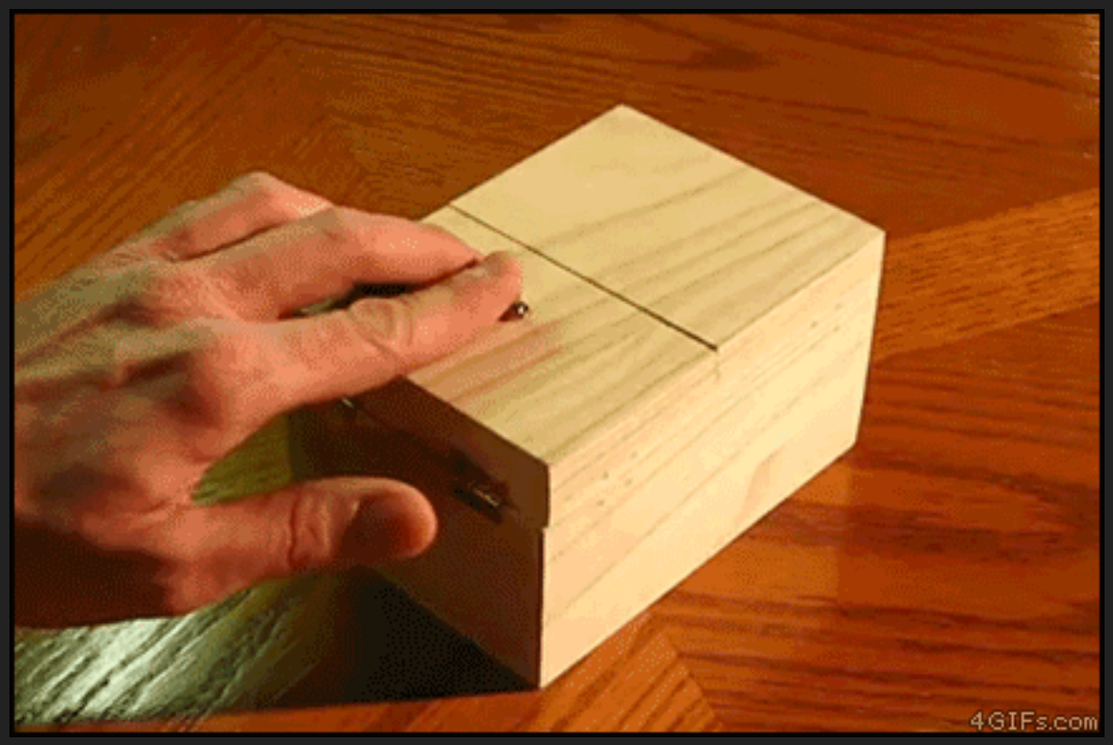
```
$ time docker run --rm debian:jessie true
real0m0.980s

# Vagrant w/ debian/jessie64 box
$ time vagrant up
real0m35.359s
```

# LOW OVERHEAD

# STORY TIME

# CIRCUIT BREAKER

- GitHub has over 100 haproxy instances load balancing traffic
- rolling out new backend configuration
- Requirement to disable new backends nearly instantly in case of error



# HAPROXY AGENT CHECK

- haproxy feature “agent-check”
- haproxy sends arbitrary string over TCP to a backend agent
- agent responds with up, down (or a few other states)
- haproxy sets backend state

# WROTE A THING

- agent-checker implements haproxy agent-check backend
- stores responses in YAML

```
# agent-checker.yaml
checks:
- key: feature1
  response: up
- key: feature2
  response: down
```

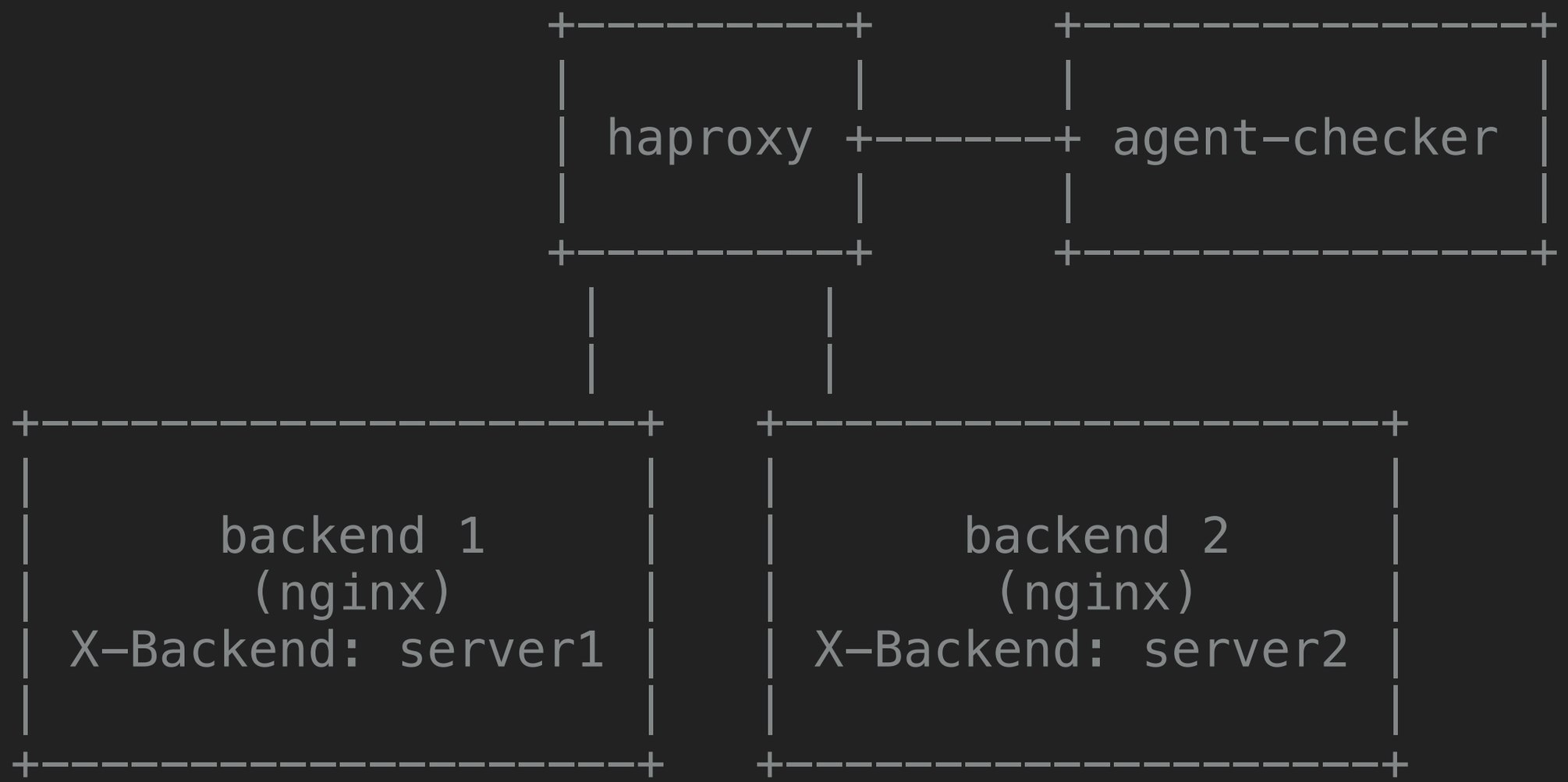
# AGENT-CHECKER

```
$ echo “feature1” | nc 127.0.0.1 3333
up

$ echo “feature2” | nc 127.0.0.1 3333
down
```

# SIMPLEST KEY VALUE STORE, EVER

# 2 UNIT TESTS, 0 INTEGRATION TESTS

# A BUNCH OF COMPONENTS



# DOCKER-COMPOSE TO THE RESCUE

```
  haproxy:
    ports:
      - "80"
    volumes:
      - ./haproxy:/usr/local/etc/haproxy
  server1:
    image: nginx:1.10-alpine
    volumes:
      - ./server1/conf.d:/etc/nginx/conf.d
  server2:
    image: nginx:1.10-alpine
    volumes:
      - ./server2/conf.d:/etc/nginx/conf.d
  agentchecker:
    build: ../../
    volumes:
      - ./agent-checker/config.yaml:/etc/agent-checker.yaml:ro
```

# AN INTEGRATION TEST

```
checks:
- key: server1
  response: down
- key: server2
  response: up
```

```
#!/usr/bin/env bash

URI=$(docker-compose port haproxy)
for _ in {1..10}; do
  curl -sI "$URI" | grep -q 'X-Backend: server2’

  # if we get something other than server2, fail
  if [ $? -ne 0 ]; then
    echo “Test failed”; exit 1
  fi
done
```

# CONFIDENCE & DEPLOYMENT VELOCITY

- Write tests we want to be confident the system works holistically
- Run on your dev machine or in CI **without installing any dependencies**
- Greater confidence means you can deploy faster

# QUESTIONS?