Docker

Hello Ground!


OH NO!
NOT AGAIN!

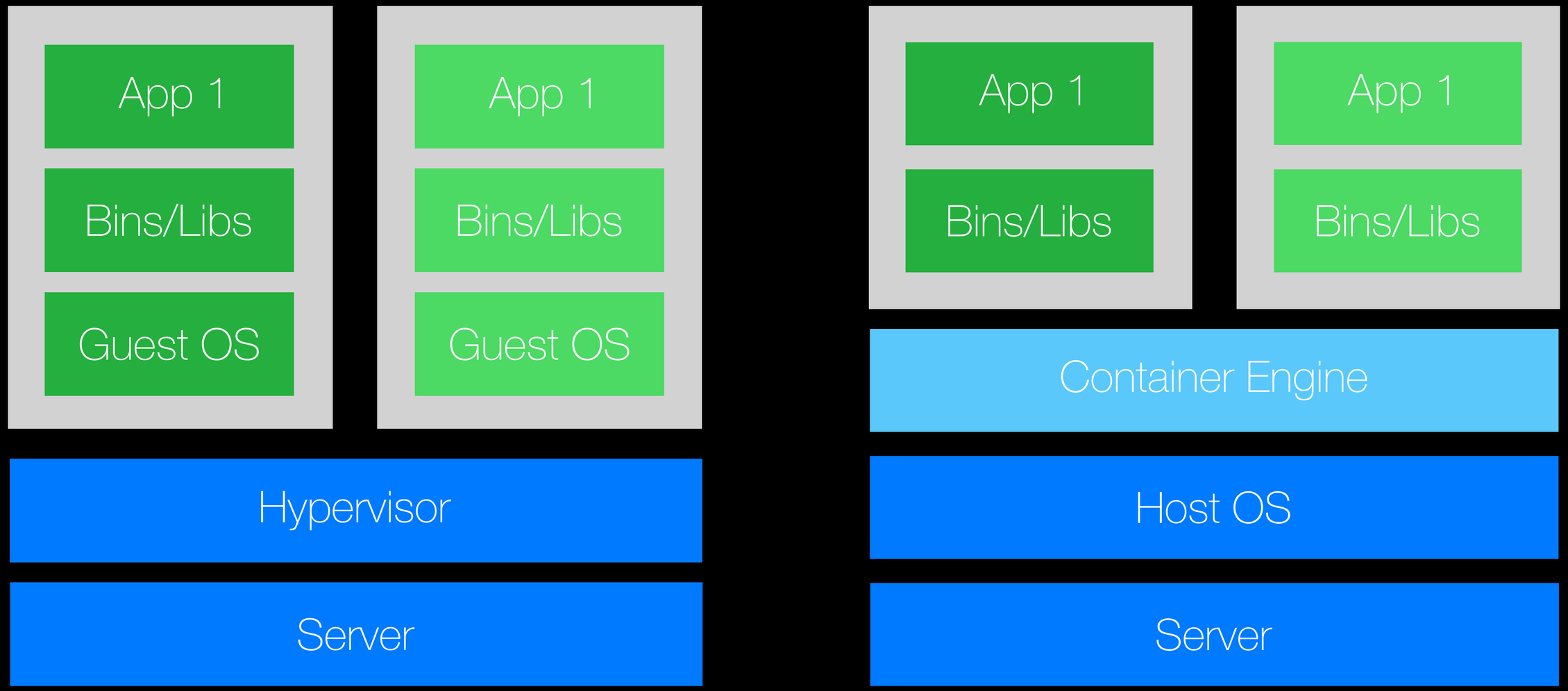
Who Am I

- Started with Docker end of 2013
- Learned a lot by building and maintaining the OpenNMS Docker Images on DockerHub
- Interested in reliability engineering and monitoring
- Full time contributor at **The OpenNMS Group, Inc.**
- Learned a lot at the **Fulda University of Applied Sciences**

Why Containers?

- Changes from a **machine-centric** view to an **application-centric view**
- Resource & Performance Isolation
- Efficiency

VM vs. Containers



Why Docker?

It was the first ecosystem which provided the full package

- Image management
- Resource-, File System-, Network-Isolation
- Change Management
- Sharing
- Process Management
- Service Discovery

There are others

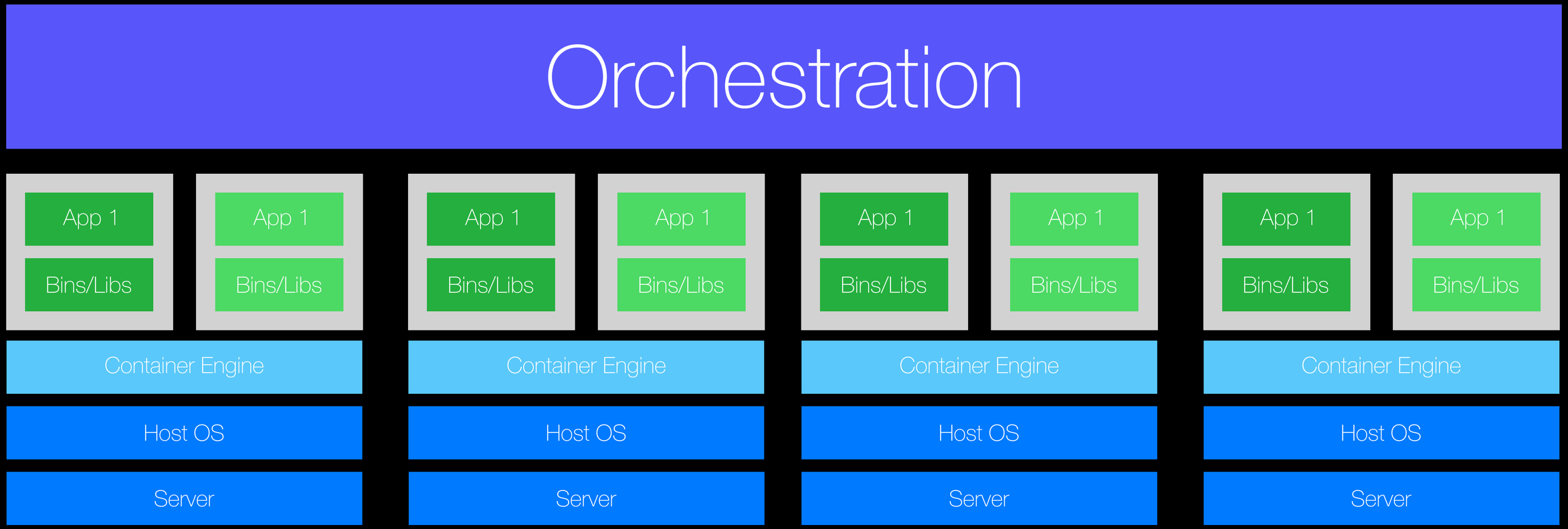
- rkt
- LXC/LXD
- … probably more

You can see a **Container Image**
as a **static linked binary**.

Configuration Patterns

- Injected as **environment** variables
- Injected as **file**
- You can **bake** it in your container
- Secrets -> as **volume mount** or **environment variable**

Container Orchestration



Orchestration

- **Docker Swarm** - Docker Inc., APL 2.0
- **Kubernetes** - Cloud Native Computing Foundation, APL 2.0
- **OpenShift** or **OKD** (the **O**rigin community **D**istribution of **K**ubernetes), APL 2.0
- **Apache Mesos** - Apache Software Foundation, APL 2.0
- **Nomad** - HashiCorp, MPL 2.0

You **can imperatively** change a **running container**. This is an **anti-pattern. Immutable** containers are the **core of everything** you will build using something like Kubernetes.

Migrating a Legacy Java Application to Docker

Getting Started

- Docker for your operating system, works best on Linux
- Docker Compose, declarative way to describe a service stack
- Internet connectivity
- A DockerHub Account
- https://github.com/indigo423/ouce2018

Docker 101

- ENTRYPOINT vs. CMD?!
- Pid 1: Orphanes, Zombies and Signals

Entrypoints in Docker

Demo 1

Java 8 and Docker **aren't** friends out of the box. **Container love** for **Java** is added in **9** and **10**

Java in Docker - Memory

- Docker can set memory and CPU limitations that Java can’t automatically detect
- Limit a container to get only 100MB of memory, Java before 8u131 doesn’t see this limit
- Backported to Java 8u131 onwards

```
-XX:+UnlockExperimentalVMOptions \
-XX:+UseCGroupMemoryLimitForHeap
```

- Java 10+ are the new defaults

Java in Docker - CPU

- JVM will look at the hardware and detect the amount of CPU's
- Docker might not allow you to use all these CPUs
- Not back-ported to Java 8 or Java 9, it was tackled in Java 10 (JDK-8146115)
- `--cpus=".5"` or `--cpuset-cpus="0-3"`

More in detail …

- Java and Docker the Limitations
- Docker and the PID 1 zombie reaping problem
- Is Docker eating Java Lunch

Building Blocks

CentOS 7

OpenJDK

https://github.com/opennms-forge/docker-openjdk

Horizon

https://github.com/opennms-forge/docker-horizon-core-web

Building Blocks

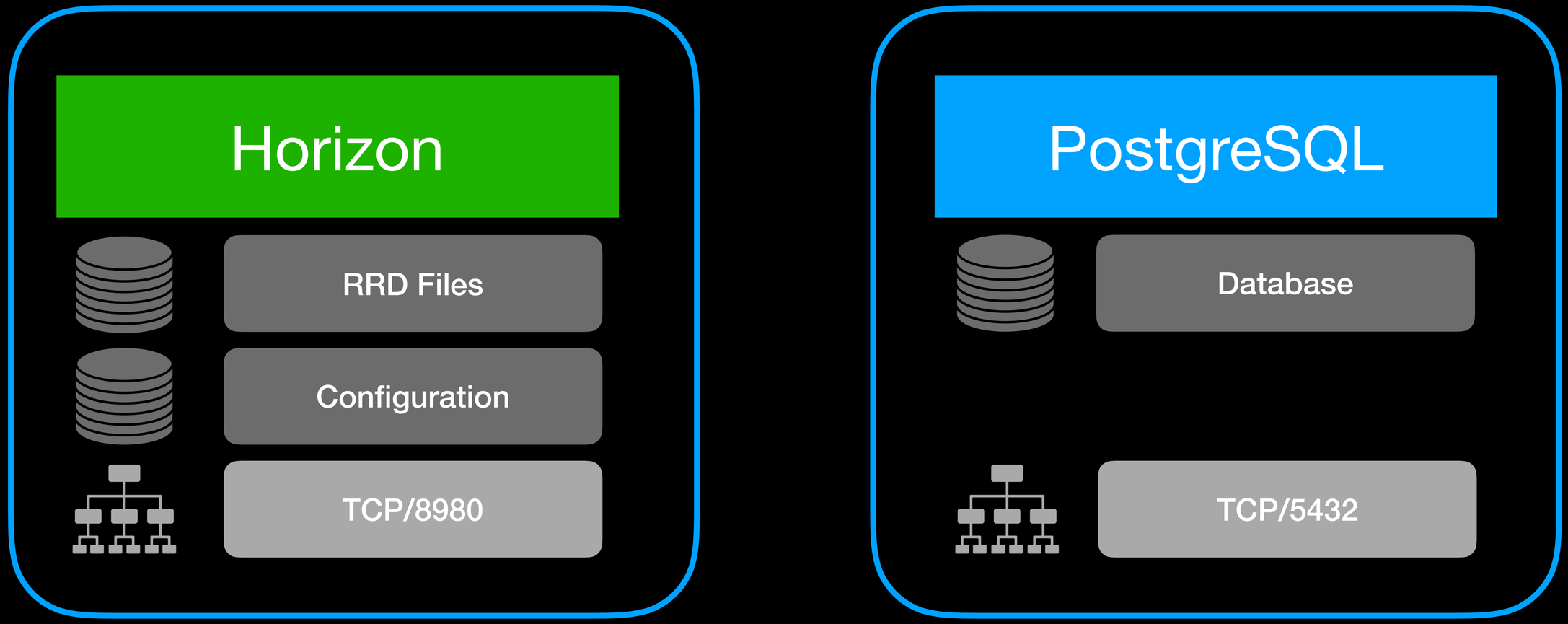
CentOS 7

OpenJDK

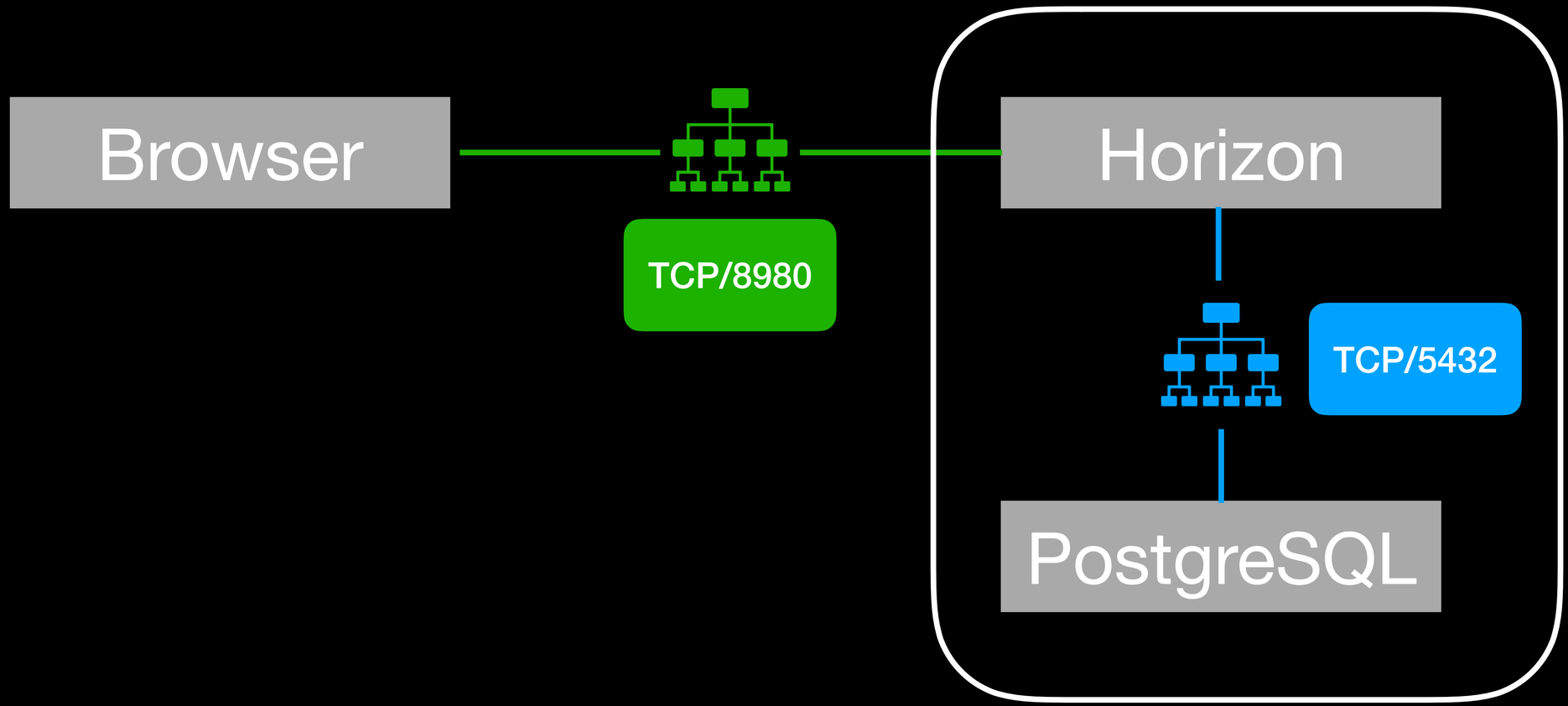
Horizon

https://github.com/opennms-forge/docker-horizon-core-web

Building Blocks



Minimal Setup



Demo 2 - Hello OpenNMS

Important

- `docker-compose stop` vs. `docker-compose down`
- `docker-compose stop` -> Docker persists every file system even a container exits!
- `docker-compose down` -> docker-compose stop && docker-compose rm

Steps to build

- Docker Compose file with two services Postgres and Horizon
- Initialize postgres with a root password
- Configure a database user for OpenNMS and the root password to initialise the database
- Publish port 8980/TCP
- Persist Postgres database, OpenNMS RRD and OpenNMS config

OpenNMS Horizon **Configuration**

Demo 3 - Configs

Different ways

- Start-up configuration in opennms.properties.d
- This is about Runtime configuration
- You have to edit files
- Be aware some configuration files can be changed in the WebUI or from REST, they need to be persisted

Configs changed in the Filesystem

On **startup** we check if there is anything in these directories and **overwrite** whats in **/opt/opennms/etc**

- /opt/opennms-etc-overlay
- /opt/opennms-jetty-webinf-overlay

Upgrade the OpenNMS Horizon

Demo 4 - Upgrade
21.0.3 -> 22.0.3

Demo 4 - Step back
What is your change in 21.0.3

Upgrade a configuration

You have to merge your custom configuration

- /opt/opennms-etc-overlay
- /opt/opennms-jetty-webinf-overlay
- configtester in various versions by mounting your config into a

```
docker run --rm -v \
$(pwd)/your-etc:/opt/opennms/etc \
opennms/horizon-core-web:22.0.3-1 -t -a
```

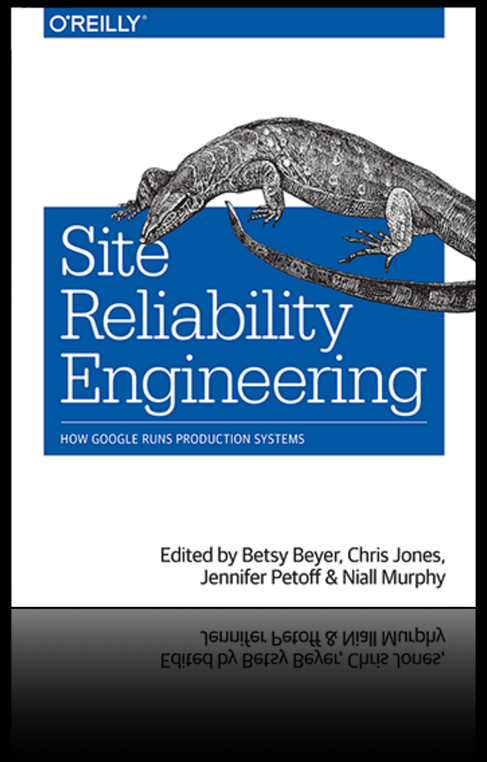
Upgrade a configuration

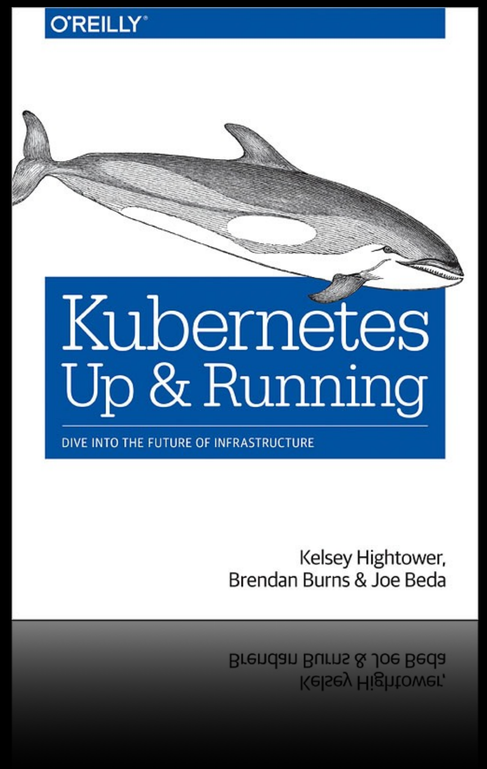
- You can initialise a pristine config from GitHub https://github.com/OpenNMS/opennms-etc-pristine
- You can use the config init from the container image

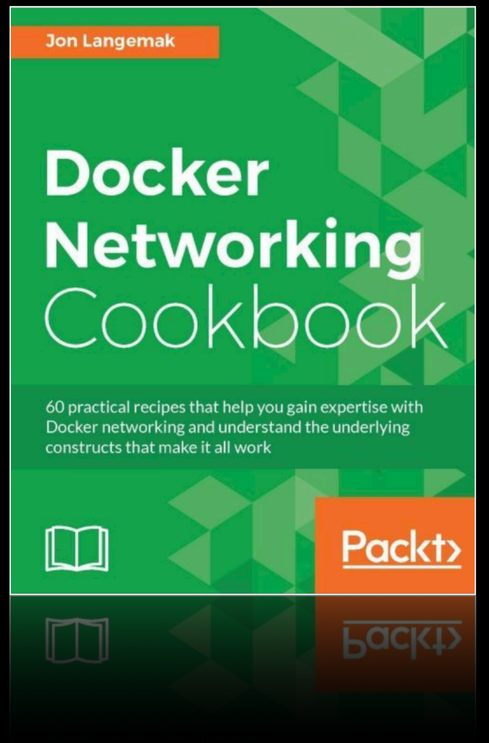
```
diff -rq -EBbw old-cfg new-cfg
```

Docker Horizon Image

- Published on DockerHub: **opennms/horizon-core-web**
- Source code: https://github.com/opennms-forge/docker-horizon-core-web
- Build with CircleCI: https://circleci.com/gh/opennms-forge/docker-horizon-core-web
- **Learn** and **share**
- **Demo: https://github.com/indigo423/ouce2018**


O'REILLY
Site Reliability Engineering
HOW GOOGLE RUNS PRODUCTION SYSTEMS
Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Murphy


O'REILLY
Kubernetes Up & Running
DIVE INTO THE FUTURE OF INFRASTRUCTURE
Kelsey Hightower, Brendan Burns & Joe Beda


Jon Langemak
Docker Networking Cookbook
60 practical recipes that help you gain expertise with Docker networking and understand the underlying constructs that make it all work
Packt>

BACKUP

Logging

Docker output

```
59        logging:
60          driver: "gelf"
61          options:
62            gelf-address: "udp://localhost:12201"
63            tag: "first-logs"
```

OpenNMS Horizon Logs

```
<appenders>
  <Socket name="Graylog_[HOSTNAME]" protocol="udp" host="[HOSTNAME]" port="12201">
    <GelfLayout host="${hostName}" compressionType="GZIP" compressionThreshold="1024">
      <KeyValuePair key="jvm" value="${java:vm}" />
      <KeyValuePair key="application_name" value="opennms" />
    </GelfLayout>
  </Socket>

  <appender-ref ref="RoutingAppender"/>
  <appender-ref ref="Graylog_[HOSTNAME]" />
```

https://wiki.opennms.org/wiki/Sending_OpenNMS_Logs_to_Graylog

Poor Mans Container Service

```
1   [Unit]
2   Description=%i service with docker compose
3   Requires=docker.service
4   After=docker.service
5
6   [Service]
7   WorkingDirectory=/etc/docker/compose/%i
8   ExecStart=/usr/local/bin/docker-compose up
9   ExecStop=/usr/local/bin/docker-compose stop
10
11  [Install]
12  WantedBy=multi-user.target
```

`systemctl [start | stop] docker-compose@<myApplication>`

Containers and Performance

IBM Research Report
An Updated Performance Comparison of Virtual Machines and Linux Containers

"Although containers themselves have almost no overhead, Docker is not without performance gotchas. Docker volumes have noticeably better performance than files stored in AUFS. Docker's NAT also introduces overhead for workloads with high packet rates. These features represent a tradeoff between ease of management and performance and should be considered on a case-by-case basis."