RED HAT SUMMIT

OPENSHIFT FOR OPERATIONS

Jamie Duncan
@jamieeduncan
Cloud Guy - US Public Sector at Red Hat

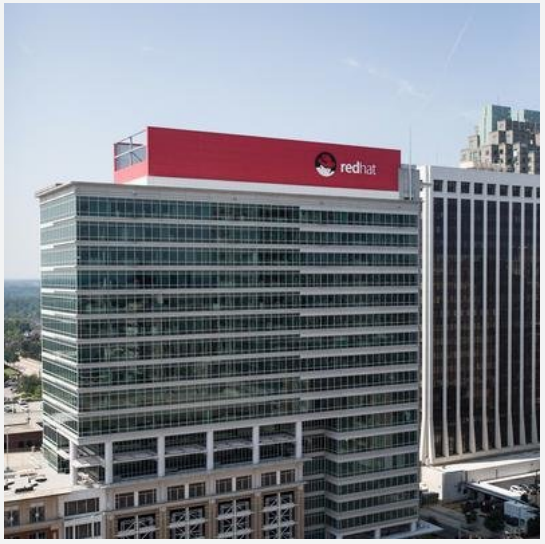
20170504

redhat.

ABOUT JDUNCAN

This is my daughter Elizabeth.
#cutestthingever

I've been at Red Hat just over 5 years



I've worked with every major US Government Agency (I think)

AGENDA

our task & ground rules

containers are taking over the world

deploying an application with OpenShift

what is happening inside OpenShift

...and down into the kernel

investigating networking isolation

investigating PID isolation

conclusions

BEFORE WE GET STARTED

Containers are taking over the world as we know it.

https://www.flickr.com/photos/dpstyles

TODAY'S TASK

OpenShift is the most complete, production-ready container platform out there. While I certainly have a bias, I can back that up with facts, data and reason.

I'll happily share a cold beverage with anyone interested in talking about this after the session.

Today we are going to take a deeper dive at what exactly is going on when you deploy an application in OpenShift.

I am a firm believer that the best way to master a tool is to have a good grasp on how it is fundamentally doing its job. That is our goal today.

Before we get going, there are a few ground rules...

GROUND RULES

This session is a discussion, not a lecture.

This information is accurate, to the best of my ability, through OpenShift 3.5.

This is a nuts and bolts talk. We won't be talking about roadmaps or changelogs.

At the end of the day, it's all about getting some new information and (hopefully) have a few laughs.

GETTING STARTED WITH A NEW APP

We start out by deploying an application in an OpenShift cluster.

```
[jduncan@jduncan ~]$ oc new-app --name=my-app \
> --code=https://github.com/OpenShiftInAction/php-demo-app.git \
> --image-stream=php
```

Raise your hand if this looks familiar.

CONNECTING THE DOTS

OpenShift creates:

a custom container image with your source code and the template

an Image Stream to tie all of your application resources together

a buildconfig to control how your application is built out

a Deployment Config to dictate how your application is deployed and updated

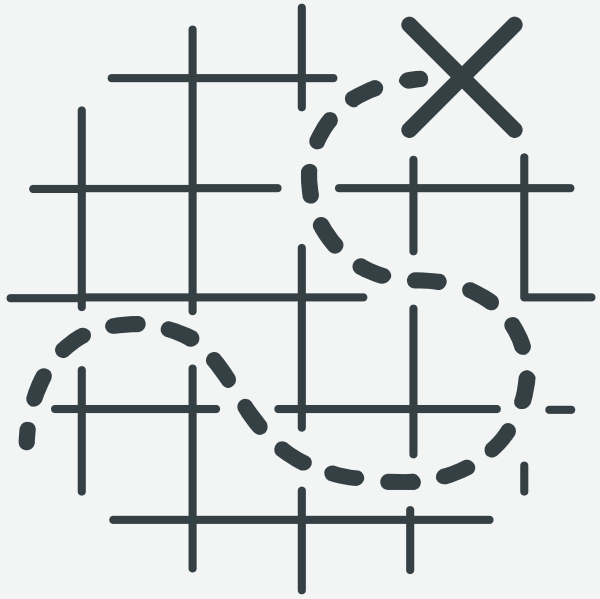
updates a load balancer so traffic from the world can get in and out of your application smoothly

OpenShift tells kubernetes to create:

a Replication Controller to scale your application up and down on demand

a Service to help expose your application to the world

Then kubernetes talks to docker to create containers to put in the new Replication Controller



That moves us down into the kernel...

WHAT THE KERNEL DOES

docker tells Linux to:

take your custom container image and put it in its own mount namespace to isolate the application and its files

create a network interface and isolate it with a unique network namespace

isolate your application's memory resources with an IPC namespace

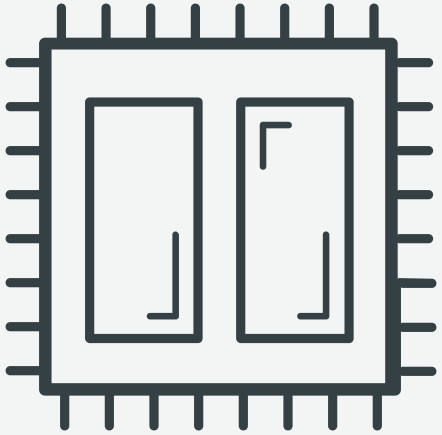
give your container a unique hostname and domainname with a UTS namespace

give your container its own PID counter set with a PID namespace

create control groups to limit the resources your container can consume

create unique SELinux contexts for your new containers

Let's put this in infographic form.



container image
image stream
buildconfig
deploymentconfig
haproxy configuration for routing

RED HAT® OPENSHIFT Container Platform

replication controller
service

kubernetes

mount namespace
network namespace
IPC namespace
UTS namespace
PID namepace

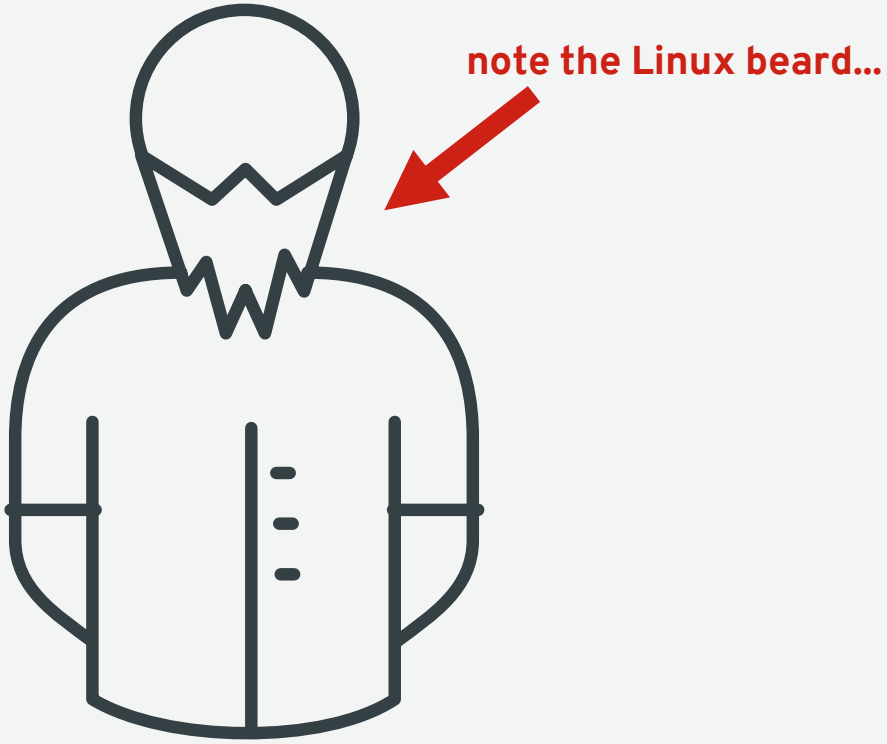
SELinux contexts
Control Groups

RED HAT® ENTERPRISE LINUX®

Also Known As...

A MODERN CONTAINER PLATFORM

FOR AN OPS TEAM...

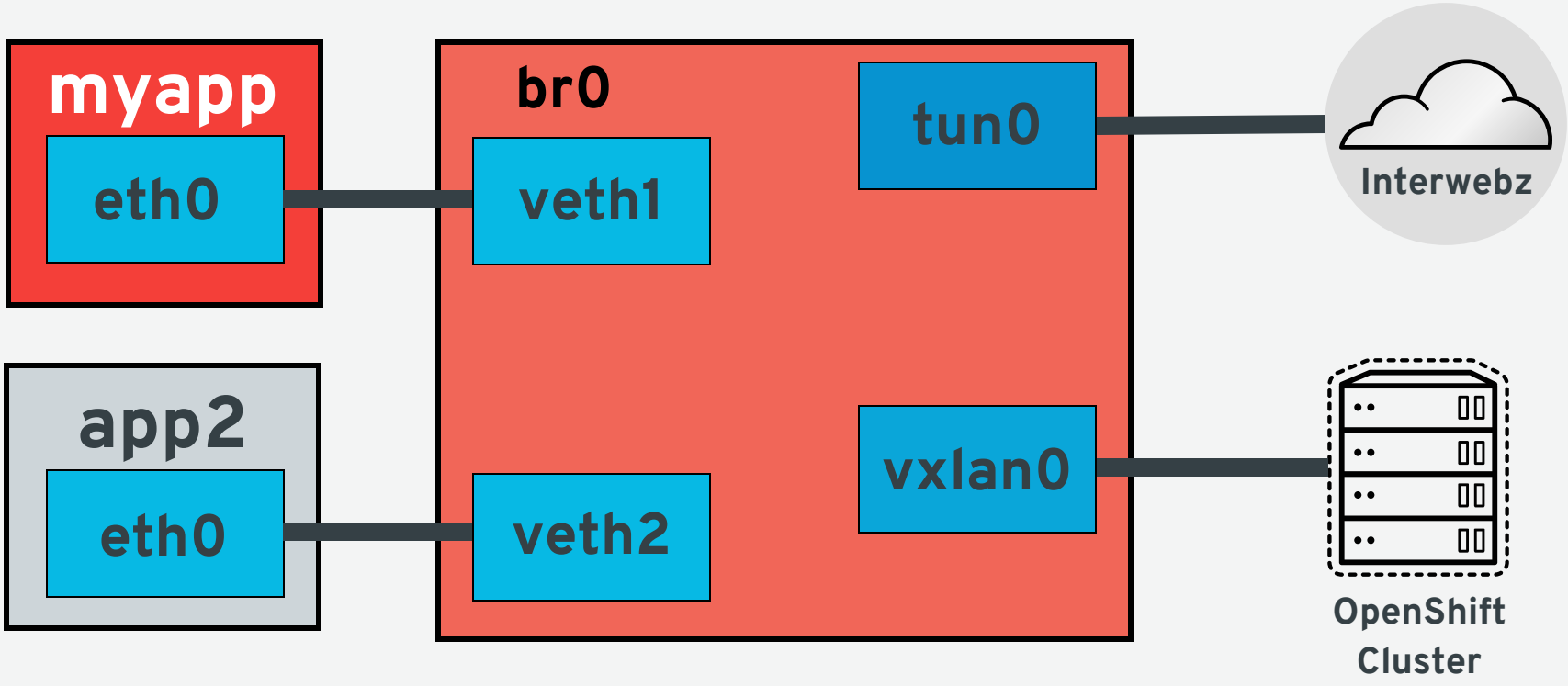


HOW DO YOU MANAGE ALL OF THIS?

Let's start with what everyone always says is the hardest. Networking.

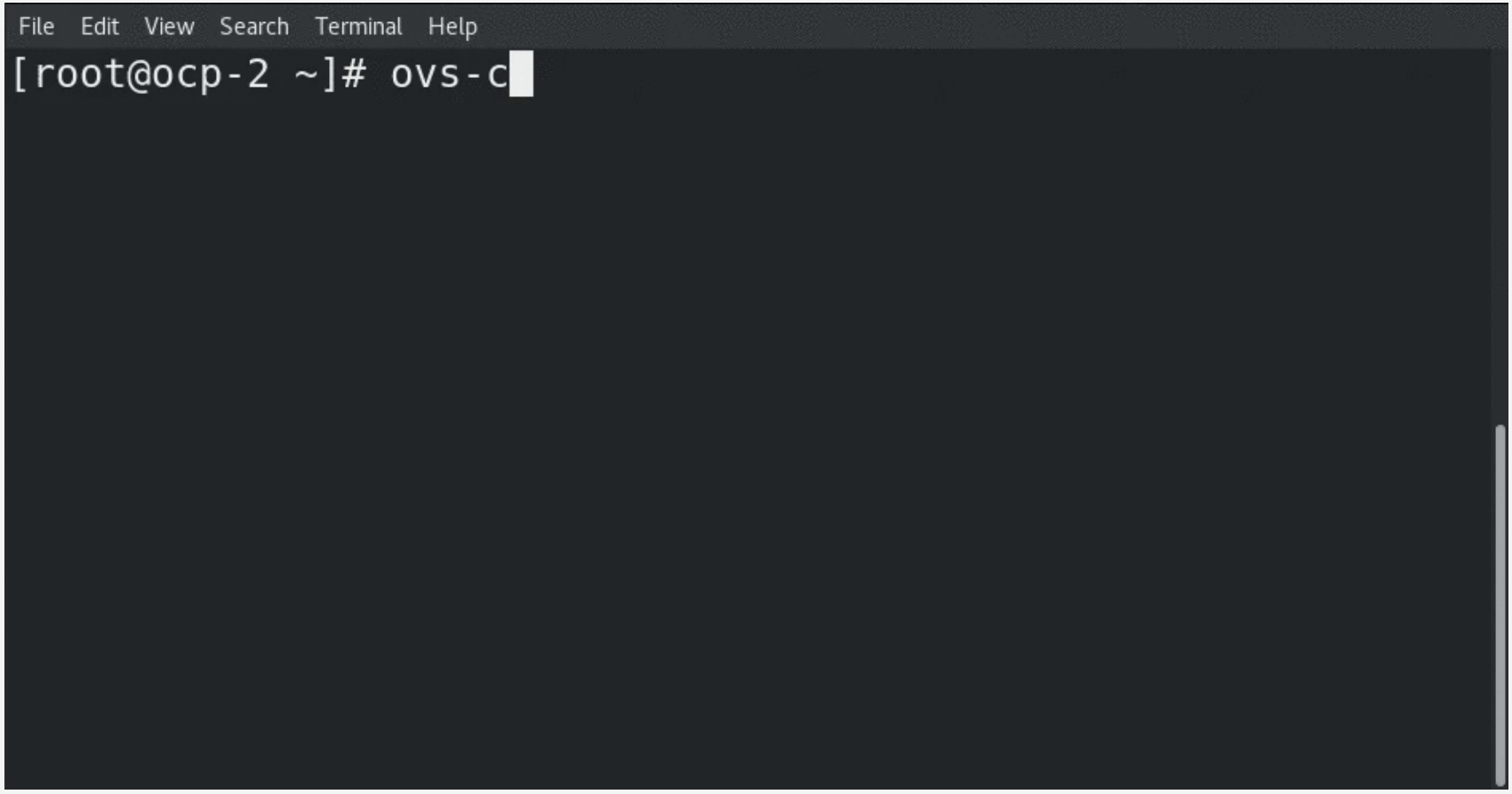
OPENSHIFT NETWORKING

OpenShift uses an SDN solution based on OpenVSwitch technology.
For each container in an application pod:
The eth0 interface is linked to a virtual eth, *veth*, device on the host
The veth0 is added to the ovs bridge, br0
Traffic bound for other OpenShift hosts is routed through the vxlan0 interface
Traffic to the interwebz is routed through the tun0 interface



NETWORKING IT IN ACTION

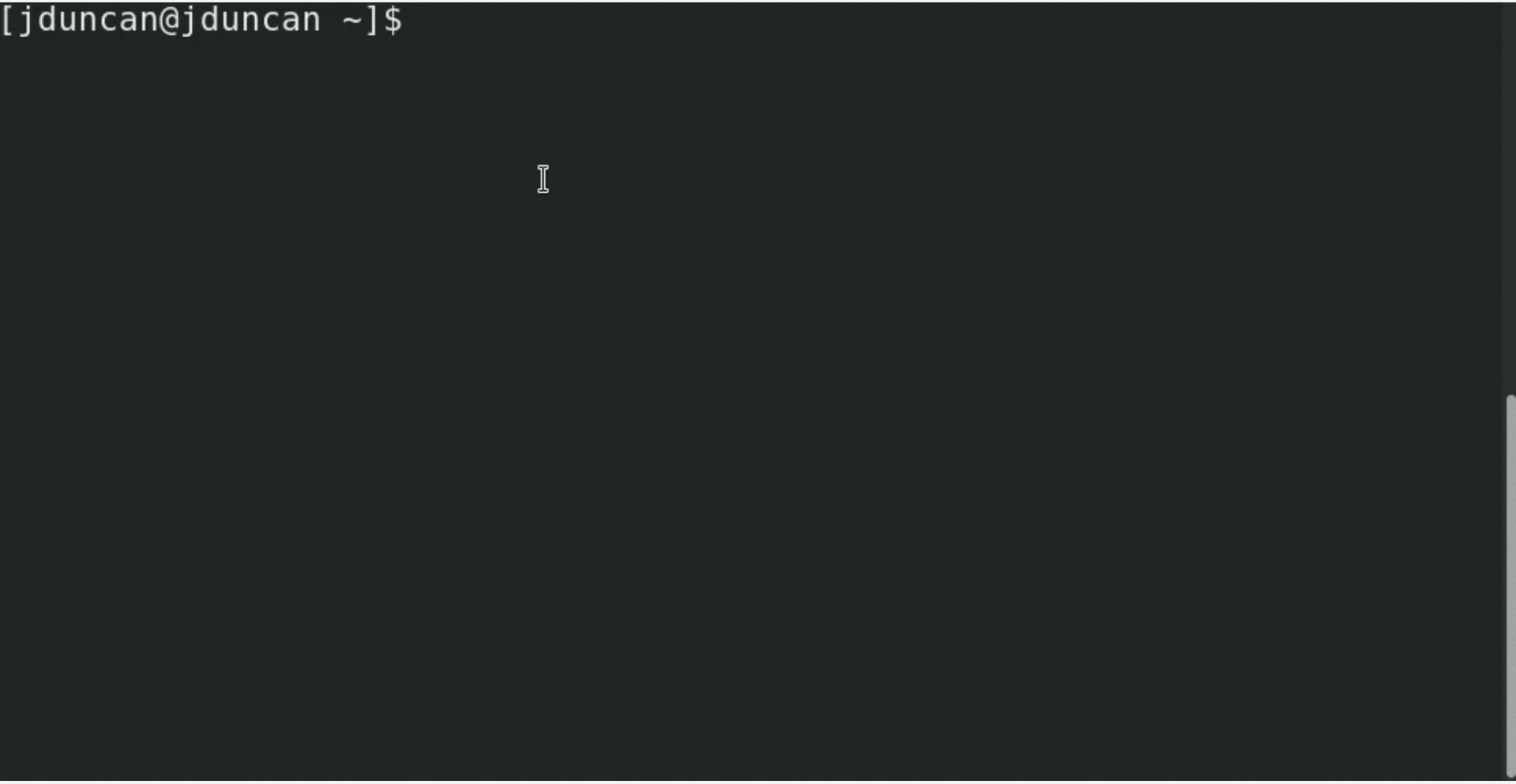
Looking at an OpenShift Node, we can see all of this from the command line.



BUT... how do we know which of those veth interfaces inside the OpenShift SDN bridge is linked to each container?

TRACING NETWORK TRAFFIC

How traffic gets into and out of your container.



The interface inside the container is linked to the veth interface on the host that is hooked into OpenShift-SDN. Voila. Networking.

CONTAINER PIDS ON YOUR HOST

You can see your processes from the host, but how does it look from INSIDE the container? Let's take a look.

PIDS INSIDE YOUR CONTAINER

Your container's PID namespace limits the PIDs that are visible from inside the container.

CONCLUSIONS

OpenShift automates a lot of work to make your applications elastic

For an Operations Team, you have to think about applications differently inside OpenShift to analyze them along with the OpenShift infrastructure.

OpenShift ships with a lot of the tools to get the information you need out of your infrastructure.

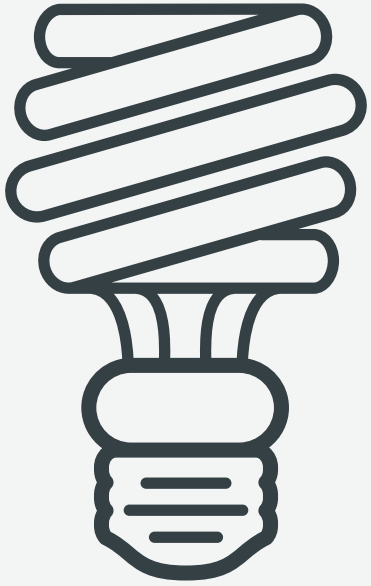
Nowhere in a diagram or a screen capture did I reference voodoo or unicorn blood.

Containers != Magic.

Containers == More efficient process isolation.

Containers == Linux!

Linux == RHEL!



RED HAT SUMMIT

NK YOU

om/+RedHat

facebook.com/redhatinc

company/red-hat

twitter.com/RedHatNews

/user/RedHatVideos

redhat.