MESOS AND THE STATE OF APPLICATION DEPLOYS

WHO AM I?

- Kory Brown.
- DevOps Engineer From Texas.
- Site Operations Engineer at Fitbit.

WHAT IS THIS TALK ABOUT?

~~MESOS!~~

APPLICATION DEPLOYMENTS!

MORE SPECIFICALLY

- How we, as an industry, used to handle application deployments.
- How we, as an industry, currently handle application deployments.
- How I think we'll handle application deployments in the future.

WAIT, WHAT ABOUT MESOS?

We'll get there.

But to understand the problems Mesos solves for deploys we must first:

- Understand what we do today.
- Understand how we got here.

Mesos, in this case, is an implementation detail.

WHY SHOULD YOU CARE?

- It's super freaking cool
- And you know... DevOps

WHAT IS DEVOPS?

- This is not a talk on DevOps...
- ... but it's important for us to have a common definition.

"DevOps is about recognizing that the backing infrastructure is not separate from your application, but rather a vital part of it."

HOW THINGS USED TO BE

YOU WANT TO DEPLOY

1. Get a Server.

GET A SERVER

- Put in a request.
- A human allocates a server.
- A human installs an operating system.
- A human ensures the networking is correct.
- A human installs all specified dependencies.
- etc, etc, etc

YOU WANT TO DEPLOY

1. ~~Get a Server.~~
2. Deploy your application.

DEPLOY YOUR APPLICATION

- An Ops guy logs into the server.
  - Downloads your application.
  - Installs your application.
  - Configures your application.

DEPLOY YOUR APPLICATION

- The application doesn't start.
  - They call you (probably in the middle of the night).
  - After an hour of troubleshooting, you realize they typoed the config.
  - You hate everything.

YOU WANT TO DEPLOY

1. ~~Get a Server.~~
2. ~~Deploy your application.~~
3. Repeat n times for scale.
    - Each time slightly differently

SOMETHING GOES WRONG

(Hardware failure/OS issues/Maintenance/etc)

God Help You.

SOMETHING GOES WRONG

(Hardware failure/OS issues/Maintenence/etc)

- File a ticket.
- Datacenter tech finds the machine.
- They pull the hard drive.
  - Which is weird, because I said the RAM tested bad.
- Two weeks later the machine is lost in a sea of tickets.
- Everything is terrible.

THIS SUCKED

- Not uncommon to measure turnaround in weeks.
- Little to no automation.
- Incredibly error prone.
- Requires a person at most every step.
- Generally leads to finely crafted artisanal machines.

TL;DR:

Horrifically inefficient, error prone, and time consuming.

HOW THINGS ARE TODAY

YOU WANT TO DEPLOY

1. Spin up a new cloud instance.
2. Run your automation tool of choice (Puppet/Ansible/Chef/etc).
3. Repeat n times.

SOMETHING GOES WRONG

(Hardware failure/OS issues/Maintenance/etc)

Spin up a new instance!

Turn around times are so low, who cares?

Birth of the "Treat servers like Cattle, not pets" thought process.

SUCKS WAY LESS

- Turnaround measured in minutes.
- Almost entirely automated.
- Fairly deterministic.
- Doesn't have to involve people at all!

BUT NOT PERFECT

- Still manage an entire OS for each application. This includes:
  - Updates! (Both OS and any applications/libraries)
  - Backups!
  - Monitoring/Metrics!
  - etc, etc, etc

BUT NOT PERFECT

- Not fully utilizing the available hardware.
  - Unless your application runs constantly at 100% CPU and RAM, you are wasting cycles and money!

BUT NOT PERFECT

TL;DR:

Good, but not great past a certain scale.

ENTER MESOS

WHAT IS MESOS?

- Open source distributed cluster management tool.

WHAT IS MESOS?

- An abstraction layer for computing resources (CPU/RAM/Disk/etc) contained within a pool of servers.

MESOS IS A DISTRIBUTED KERNEL

- A traditional Kernel (like Linux!) provides a set of APIs for interacting with available hardware on a local machine.
- A distributed Kernel (like Mesos!) provides a set of APIs for interacting with available hardware on a pool of servers.

THE DATACENTER OS

- The Kernel itself is a small part of an Operating System.
- Many other components that make it something useful.
- Most relevant for us right now:
  - Init system -- Some process to manage the lifecycle of other processes.
  - Cron -- Some process to run tasks on some specified interval

Mesos implements this functionality with "Frameworks".

WHAT'S A FRAMEWORK?

- A Mesos "Application".
- Composed of:
  - A scheduler, which registers with the Master, and receives resource offers.
  - One or more Executors, which launches tasks on slaves.

MESOS FRAMEWORKS

- Init System:
  - Marathon
  - Aurora
- Cron:
  - Chronos
  - Aurora

A QUICK ASIDE ON FRAMEWORKS

The Frameworks listed are just ones that are relevant in the context of application deploys.

A QUICK ASIDE ON FRAMEWORKS

Mesos is meant to be used as a generic computing cluster manager, which means you can also use, as a framework:

- Jenkins
- Hadoop
- Spark
- Storm
- Kafka
- Many more things probably!

Perhaps more importantly, all of these frameworks can share the same cluster of machines!

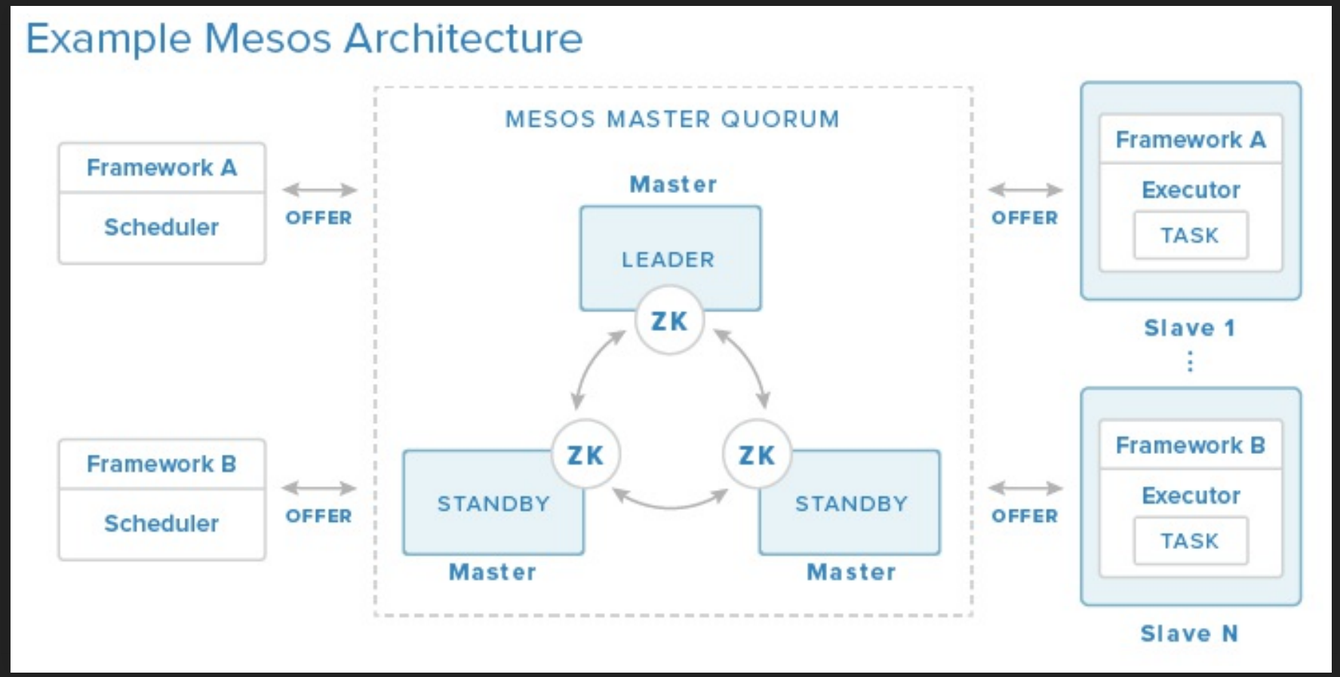
MESOS OFFERS

- A list of an agent node's available resources (CPU/RAM/Disk/etc).
- Two-Tier Scheduling System.
  - Agent Node sends resources to Master node.
  - Master node uses an algorithm called "Dominant Resource Fairness" to fairly pass that resource offer to it's registered Frameworks.

OTHER MESOS RELATED WORDS

- **Master:** Runs on master node, manages agents.
- **Agent:** Runs tasks that belong to frameworks.
  - Previously referred to as a Slave, but this is being changed in newer versions.
- **Task:** A unit of work scheduled by the Framework, and executed on the Agent.
- **ZooKeeper:** Distributed consensus framework.

MESOS ARCHITECTURE



SHOULD MY APP RUN ON MESOS?

STATELESS APPLICATION

- Web App (Rails, Django, Play, etc)
- Jenkins Build Slaves
- Memcached

Yes

STATEFUL APPLICATIONS

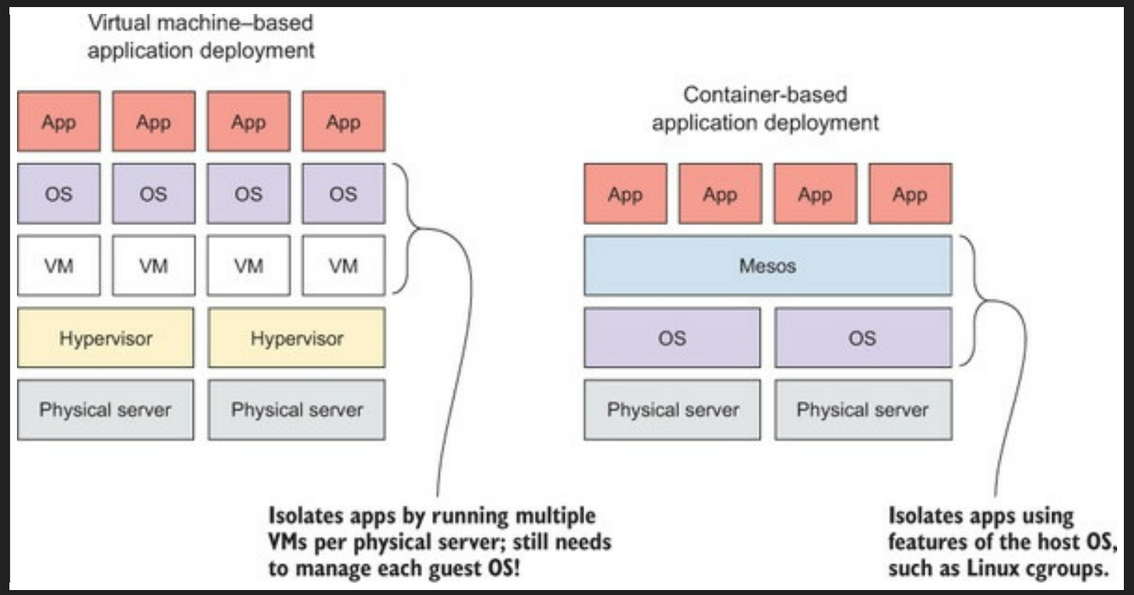
- MySQL
- PostgresSQL
- Jenkins Master

No

DOES IT ADDRESS CURRENT PROBLEMS?

- Managing fewer Operating Systems.
- Better utilization.

MANAGING FEWER OPERATING SYSTEMS



Manage an OS for each physical host, NOT each application.

BETTER UTILIZATION

Instead of one application running per host...

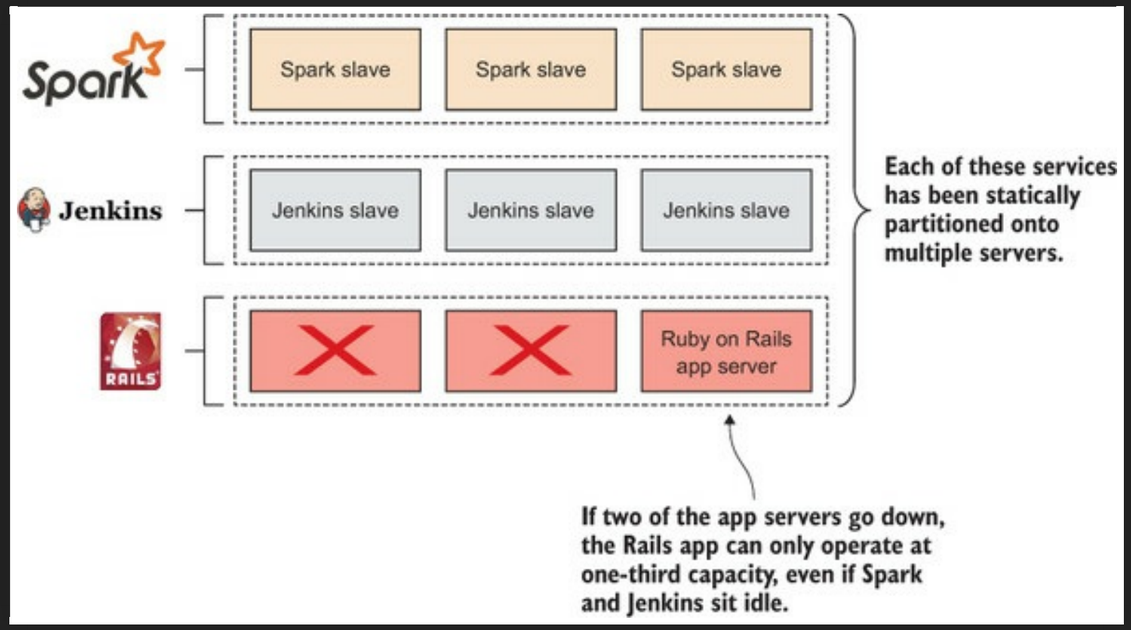
... we bin pack them all together!

Any resources your application isn't using can be shared as needed!

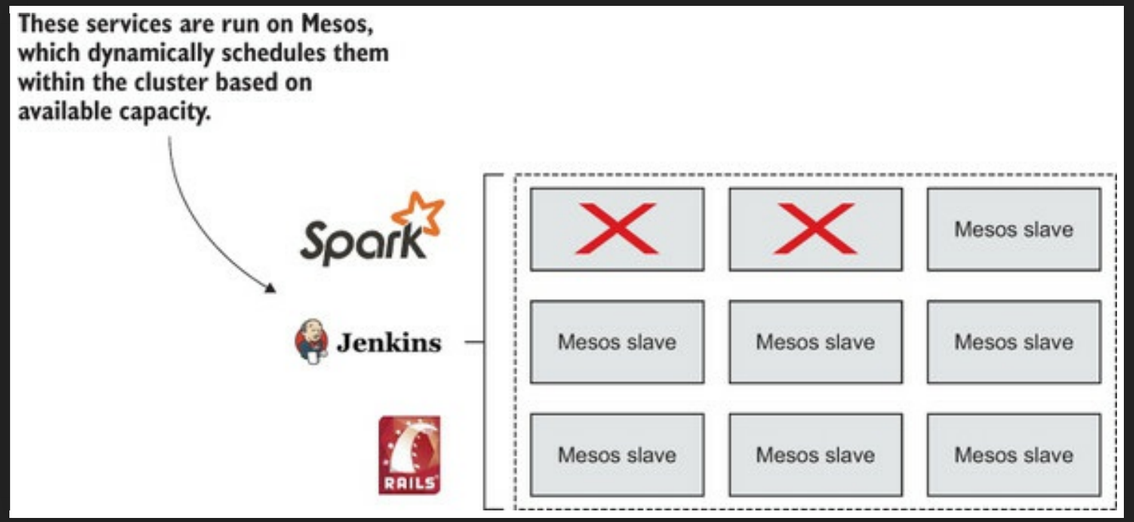
BONUS: REDUNDANCY BAKED IN!

Since we have this big pool of shared resources, if an instance or host dies, we just reschedule it somewhere else.

STATICALLY PROVISIONED CLUSTER



DYNAMICALLY PROVISIONED CLUSTER



DEPLOYING WITH MESOS

YOU WANT TO DEPLOY

1. Write a Job Config for your Framework of Choice.
   - We use Aurora -- it's pretty sweet.
2. Tell Aurora to run it.

SOMETHING GOES WRONG

(Hardware failure/OS issues/Maintenance/etc)

- Aurora reschedules your application to another node.

HARDLY SUCKS AT ALL!

- Turnaround measured in seconds!
- Entirely automated!
- Deterministic!
- Fault Tolerant!

DEVELOPER PERSPECTIVE

TESTING

- Mesos can be your dev, qa, staging, and production environments.
  - Aurora splits permissions and allocations between configured environments.
  - Prioritizes production. Meaning it will preempt jobs running in "lesser" environments.

Becomes easy to test in a Prod like environment because it IS the same environment!

SELF-MANAGEMENT

- No more waiting on Ops for hardware!
- Easily automate-able!
  - Want Heroku styled deploys?
  - *This is how Heroku does it.*

TL;DR

- We've come a long long way.
- Thing's suck way less now.
- Sucking less and less every day.

QUESTIONS?