Intro

Who am I?

- Pablo Maldonado, Ph.D.
- Trainer and consultant, based in Prague, CZ.
- Mostly machine learning stuff, which these days involves more and more cloud.

Who are you?

- Name, what do you do?
- Experience with Docker/Kubernetes?
- 1-2 concrete things you expect out of this training.

Topics

- **Day 1:**
  - Intro, cluster components and `kubectl` command.
  - Pods. Labels and Secrets.
- **Day 2:**
  - Services, deployments and rollouts. Load balancing.
  - Security and storage. CI with Jenkins.

Logistics

- 9:00-12:30 Lecture + Lab.
- 13:30-17:00 Lecture + Lab.
- **Github:**
  - jpmaldonado/np-kubernetes
  - jpmaldonado/orchestrate-with-kubernetes

After the course

- Certificate of completion.
- Course Survey.

DevOps, Docker, Kubernetes

What is DevOps trying to solve? (1/3)

- **Low expectations of (web) projects**:
  - Tacit assumption that all projects will be late, underperforming and a waste of money.
- **Fear of change:** When (and if) the application works, it is vulnerable and should only be touched with care.

What is DevOps trying to solve? (2/3)

- **Risky deployments:** will the code handle real-life production conditions?
- ***It works on my machine!***
- **Silos** in projects, where analysts, developers, QA and sysadmins have different views of the same project.

What is DevOps trying to solve? (3/3)

- **Last-mile problem:** Passing from *development completed* to *in production, stable, making $$*.
- **Building bridges** across teams: DevOps specialists are generalists that are not afraid of infrastructure and configuration, but can also write tests, debug and ship features.

Application design

- Millions of lines of code, hours to build.
- Infrequent releases.
- **Problem:** any small single thing that blocks a deployment, blocks everyone's work.

Modern application design

- Split the functionality of an application into small pieces (with stable APIs).
- Deploy into a light-weight version of virtual machines.
- Automation to coordinate and scale all these many moving parts.

Modern application design

- **Split the functionality of an application into small pieces (with stable APIs).**
- Deploy into a light-weight version of virtual machines.
- Automation to coordinate and scale all these many moving parts.

Microservices architectures

Architectural approach to design applications such that they are:

- Modular.
- Easy to deploy.
- Scale independent.

Monolith App

Microservices App

Modern application design

- Split the functionality of an application into small pieces (with stable APIs).
- **Deploy into a light-weight version of virtual machines.**
- Automation to coordinate and scale all these many moving parts.

Packaging and distributing apps

- Code.
- Dependencies.
- Runtime information.

Typical scenario

Workaround: VMs

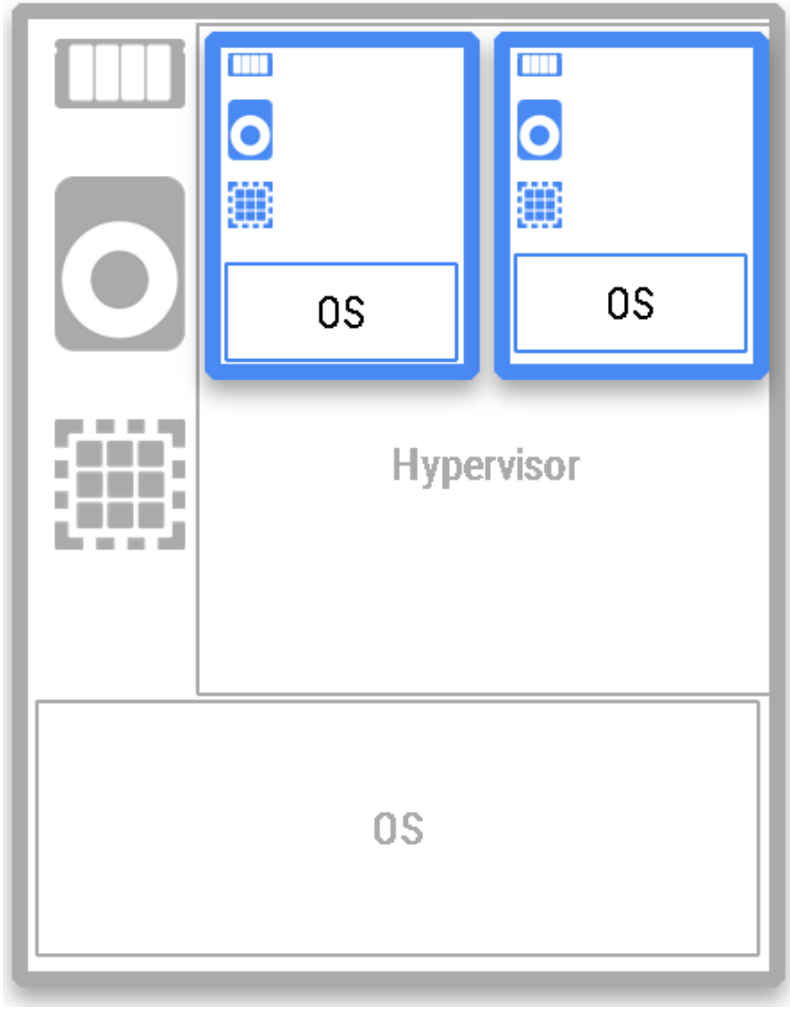
- Good isolation.
- Takes a long time to get the OS started.
- Each VM instance carries its own copy of the OS.
- Moving allocated resources around is not possible.

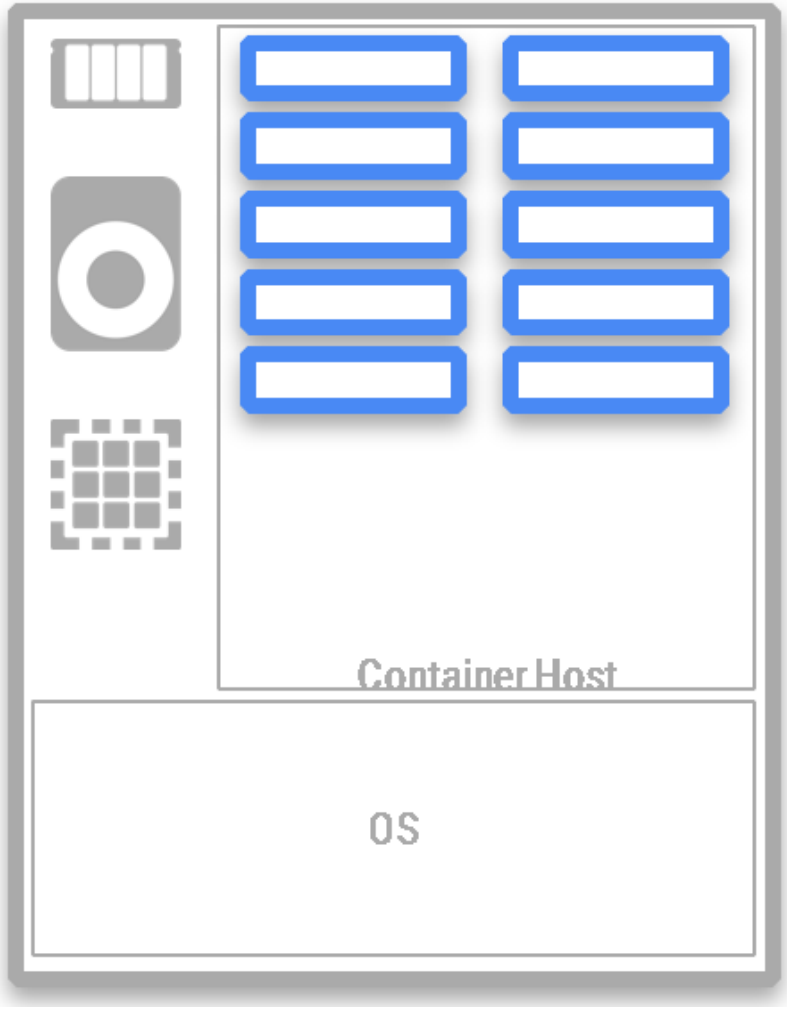
Containers

Containers on the other hand share the same kernel, so that you can share resources as you need them.

- Container images are stateless and contain all dependencies
  - static, portable binaries.
  - constructed from layered filesystems.
- Provide isolation (from each other and from the host)
  - Resources (CPU, RAM, Disk, etc.)
  - Users
  - Filesystem
  - Network

VM's vs Containers





Advantages in terms of dependency management

- No longer care about which environment our containers are running in.
- Versions are not so important either
  - Assuming APIs and functionality remain the same.
- Each part of the stack is bundling its own dependencies.

New scenario

	Dev 1 Laptop	Dev 2 Laptop	QA	Stage	Production
OS					
Frontend					
Services					
Database					
Logs					

But that is just one machine...

- How do you know when containers are alive?
- Or to coordinate between them?
- How about scheduling?
- And security?

Modern application design

- Split the functionality of an application into small pieces (with stable APIs).
- Deploy into a light-weight version of virtual machines.
- **Automation to coordinate and scale all these many moving parts.**

Enter Kubernetes

- *Manage applications, not machines.*
- Open-source container orchestrator.
- A Google product (with all that it entails).

Concepts that define K8s

- Declarative instead of imperative.
  - State your results, let the system figure it out.
- Legacy compatible.
- No grouping (Labels are the only groups).
- No hand-crafted workload.
- Modular: components, interfaces, plugins.
- *Cattle instead of pets*

Pets vs Cattle

- Pets:
    - Have a name.
    - Unique and special.
    - Get personal attention.
- Cattle
    - Has a number.
    - No identity.
    - Treated as a group.

Desired state

- We tell Kubernetes what we want, not what to do.
- **Example:**
  - *Imperative*: Bash script that launches a series of tasks, e.g. create 3x frontend, 2x services, 1x backend.
  - *Declarative*: There should be 3x frontend, 2x services, 1x backend.
- **Advantage:** No need to worry about missing steps/edge cases.
- Sidenote: How is this actually achieved?: Distributed consensus

Mutable vs immutable infrastructure

- Traditionally, computers and software are *mutable* systems, with small incremental updates (e.g. `apt-get update`).
- In immutable systems, an entirely new image is built and the old one destroyed.

Trivia

What is the difference between:

1. Log in to a container, run a command to download your new software, kill the old server, and start the new one.
2. Build a new container image, push it to a container registry, kill the existing container and start a new one.

Solution

- None on the final product!
- How the final product was *reached* is traceable and easier to reconstruct.
- In particular, you can version-control it and roll back if needed.

Online, self-healing system!

- Kubernetes continuously takes action to check that the system is in the specified state.
- If your manifest file says you should have three replicas and a fourth is created by accident, K8s will kill one.
- If you kill one, K8s will create one.

End-to-end demo

- You can follow along!
- Online writeup of the lab