Docker & Kubernetes: The Big Picture

Aim of the Game

- Get up to Speed
- Give Directions
- Cover Fundamentals
- Less than 2 hours

The Bad Old Days

Aplicações estão no centro dos negócios

No Apps, No business 🙈

Back in the days, early to mid 2000's:

❌ 1 App per Server (Physical Expensive Servers)

- Ninguém sabia quais os limites físicos a impor nos servidores e portanto investia-se em grandes e rápidos servidores mas que no fundo só se usava cerca de 10-15% da capacidade destes mesmos servidores.

VMware/Hypervisor model

- Permitiu correr várias Apps no mesmo servidor físico
- Physical Server:
  - Apps Running (4 Apps)
  - Every app would need a VM
  - Every VM is a "physical" slice of the Server(Resources)

- 4 installations of Windows &/or Linux, this would steal a lot of resources!
- 4 OS Licenses = More Cost
- Feels like a waste
- Admins
  - Patching, Updates, Anti-Virus....
- Tornou o mundo do IT melhor no entanto ainda possuía algumas lacunas e “se-nãos”
- 2 much potential attack vectors

Containers

- Exemplo: Servidor Físico com 4 Apps
- Instalamos um OS apenas (Linux/Windows Server)
- Criamos 4 containers (cada um destes é um “slice” do OS)
- Dentro destes containers corremos as Apps, uma App por container
- Resolvemos grande parte ou totalidade dos problemas do modelo Hypervisor
- Correm muito rápido

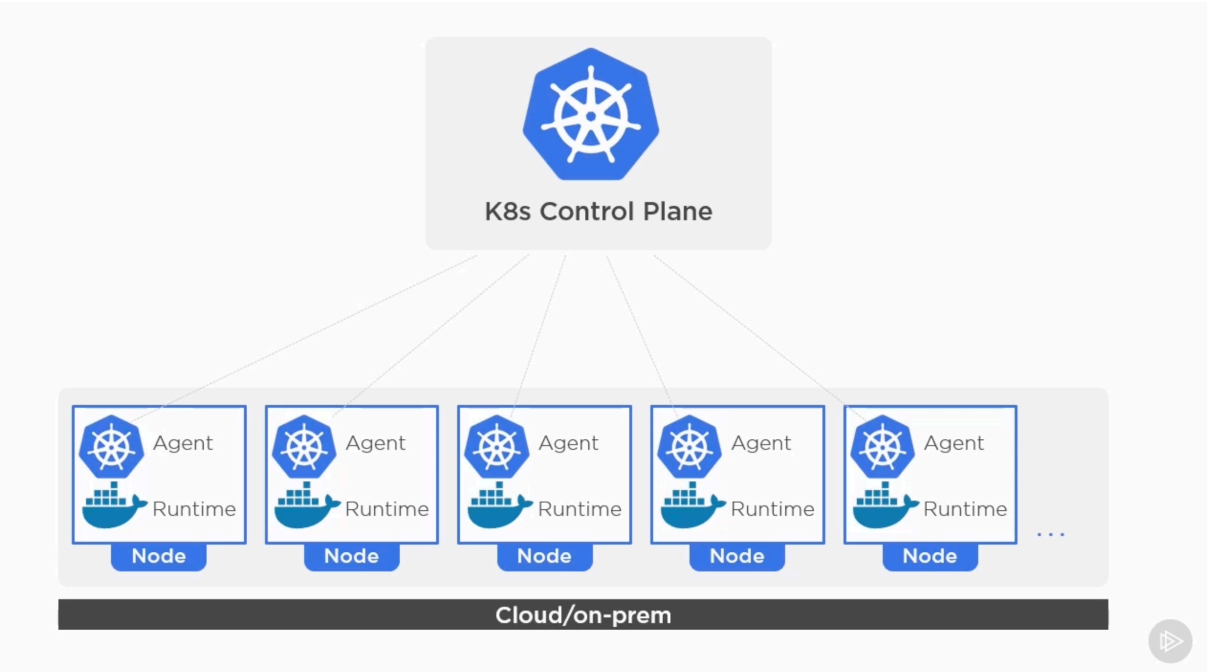
$docker start containername

$docker stop containername

Kubernetes (Google’s Open Source Project)

Scheduling, scaling, healing, updating...

Docker is more start | stop | delete ...



Se por exemplo tivermos containers iguais (replicas) a correr em nós diferentes o Kubernetes faz toda a gestão de scaling quando necessários mais recursos, healing quando algum nó falha etc tudo automaticamente sem mão humana.

Kubernetes manages Docker Nodes!

Podemos migrar as nossas aplicações entre diferentes clouds e on-prem's(data center for ex) com o Kubernetes!

Knowledge & Experience

Use Docker Desktop → Dev Docker and K8s Environment

Play with Docker

Play with Kubernetes

Preparing to Thrive in a Container World

Get ready for change (personal & organizational)

Think of Blockbusters vs Netflix example

Traditional Banks vs Digital Banks (basically software companies)

Suitable Workloads

Containers in: Stateless vs Statefull Apps

Hight level definitions:

Stateful:

- Has to remember stuff (Database App running in a container and using a volume to store data - state)
- When first created its empty, starts to fill up with data, if things pop (node), when restarting everything must comeback with all the data previously stored

Stateless:

- Doesn’t remember stuff
- For example a web server with static content
- If it goes pop, everything its like in day one

Low-hanging Fruit

Cloud Native & Microservices

Slow apps = Slow business

“Hypervisors revolutionized IT”

- Take legacy apps and run them on VM machines!
- Simple migrations
- Life made easy

Containers

- Clouds & Containers made us make better Apps, better for the business and the costumer

- We get to develop apps that are modern, scalable, self-healing, portable (Pain and effort to get to this level)

State & Legacy Apps

Docker and K8s are magic to stateless workloads!

But still are good to stateful Apps!

- State is hard!
- Docker & K8s added features for stateful apps
- Persistance storage
- You can move your legacy apps to containers
    - Directly moved to containers (Lift and shift)

Enterprise and Production Readiness

Docker (dotcloud)

- Community Edition
    - On-cloud/Prem
    - Free
    - Quick release cycle
    - Edge channel (fun/scary stuff)
- Enterprise Edition
    - On-cloud/Prem
    - Costs money
    - Official support
    - Patches
    - Cautious release cycle

    - Stability
    - Extras
        - Enterprise web UI
        - Security Features (Users, groups)
        - AD/LDAP
        - Private registry
        - Policies
        - FIPS
        - Pipelines...

Kubernetes

Came out from Google

Open source like Docker

Everyone is all over it (Big companies and startups)

On Premises

- Build your own

Cloud

- Hosted (canned) options
    - AWS EKS
    - Azure AKS
    - Google GKE
- Build your own

Kubernetes is a massive project and way bigger than Docker!

Stages:

- Alpha, Beta, GA Features
- Alpha:
    - Off by default

        - Early code
        - Uncertain future
        - Not for production
    - Beta:
        - On by default
        - Becoming stable
        - Promising future
        - Some details may change
    - GA:
        - Production ready* (Is still your choise)
        - Solid future
        - Stable:
            - Code
            - Features

Cloud Services

- AWS Elastic Kubernetes Service
- Azure Kubernetes Service
- Google Kubernetes Engine
- Others..

Ecosystem

Networking, ML, Storage, Security, Monitoring, Logging, Automation, Data to elevate your Docker and Kubernetes game!

A lot of companies are offering this solutions right now.

A Word on Orchestration

Analogy

- Team of Soccer, every player has a job, if they alone they weak, if they play together they are stronger.. orchestration.
- Coach's job to orchestrate the team! → Kubernetes
- Team is made of individuals, and everyone has their own job!

Apps Comprise:

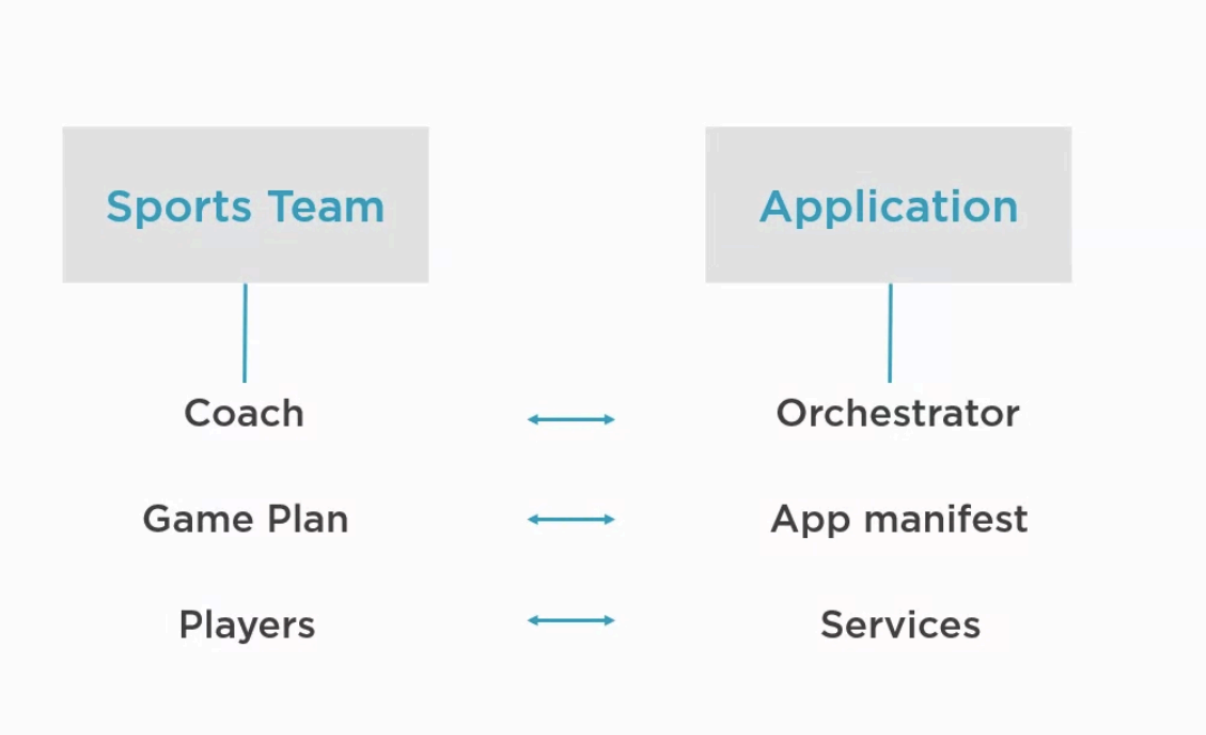
- Multiple parts (services)
- Multiple requirements
- Scale things makes everything more complex!

Game Plan:

- Describes the App

Document Game Plan in version control system

Key to Automation



What Next?

More courses...

- Getting started with Kubernetes → wip
- Getting Started with Docker → done
- Docker Deep Dive → done