BandTec

DIGITAL SCHOOL

Computação em Nuvem

Sistemas Distribuídos

# Containers

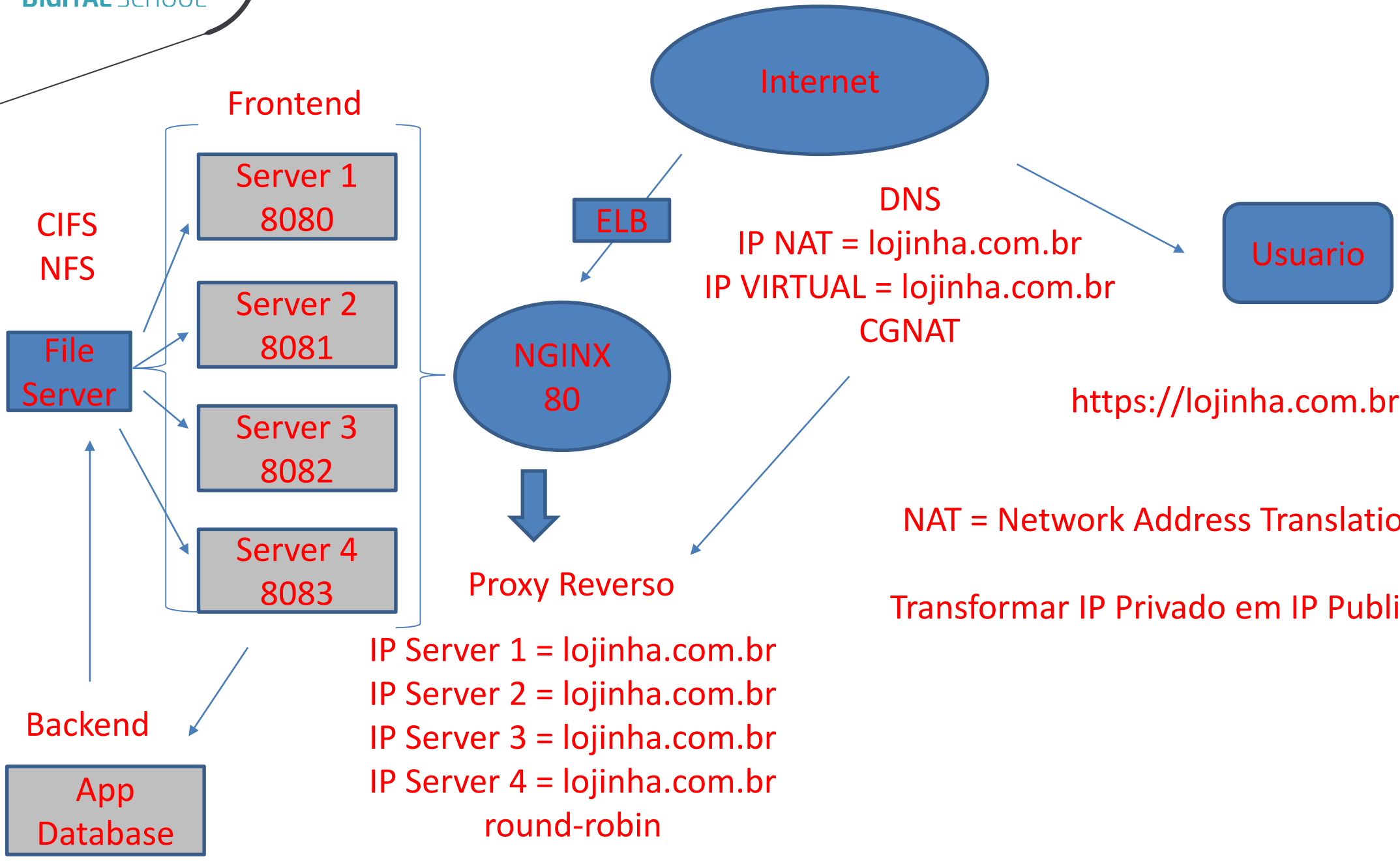
Professor: Rogério Chola

e-mail: rogerio.chola@bandtec.com.br

BandTec DIGITAL SCHOOL

# Agenda do Módulo

- O que é Docker?
  - Docker vs Virtual Machine
  - Origem, Plataformas, Atualidade
- Imagens e Containers
- Volume Mounting, Port Publishing, Linking
- Casos de Uso
- Kubernetes
- Hands-On Workshop


Internet
Frontend
Server 1
8080
Server 2
8081
Server 3
8082
Server 4
8083
CIFS
NFS
File
Server
ELB
NGINX
80
DNS
IP NAT = lojinha.com.br
IP VIRTUAL = lojinha.com.br
CGNAT
Usuario
https://lojinha.com.br
NAT = Network Address Translation
Transformar IP Privado em IP Publico
Proxy Reverso
IP Server 1 = lojinha.com.br
IP Server 2 = lojinha.com.br
IP Server 3 = lojinha.com.br
IP Server 4 = lojinha.com.br
round-robin
Backend
App
Database
Oracle = cores
2 x 16 => 32/2 = 16

Frontend

Server 1
8080

WebServer (Apache)

SO Instalado
Apache e configurar

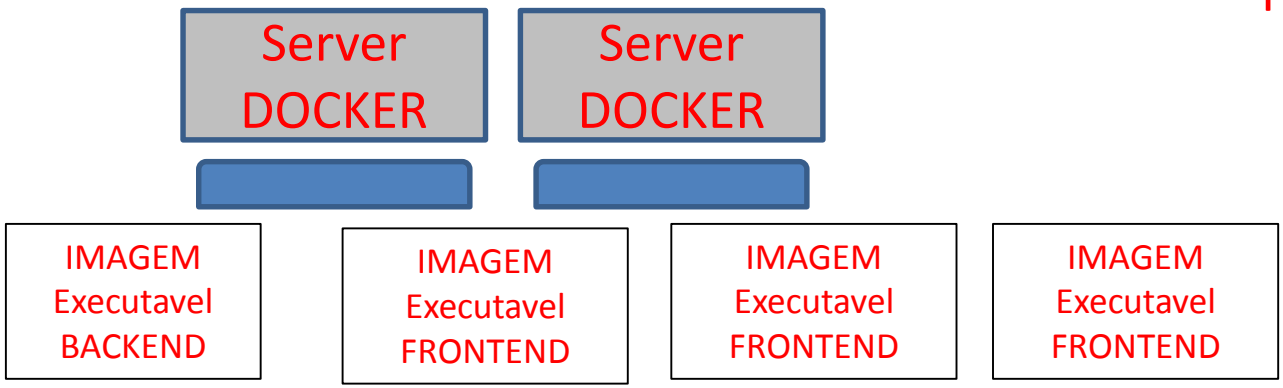
Backend

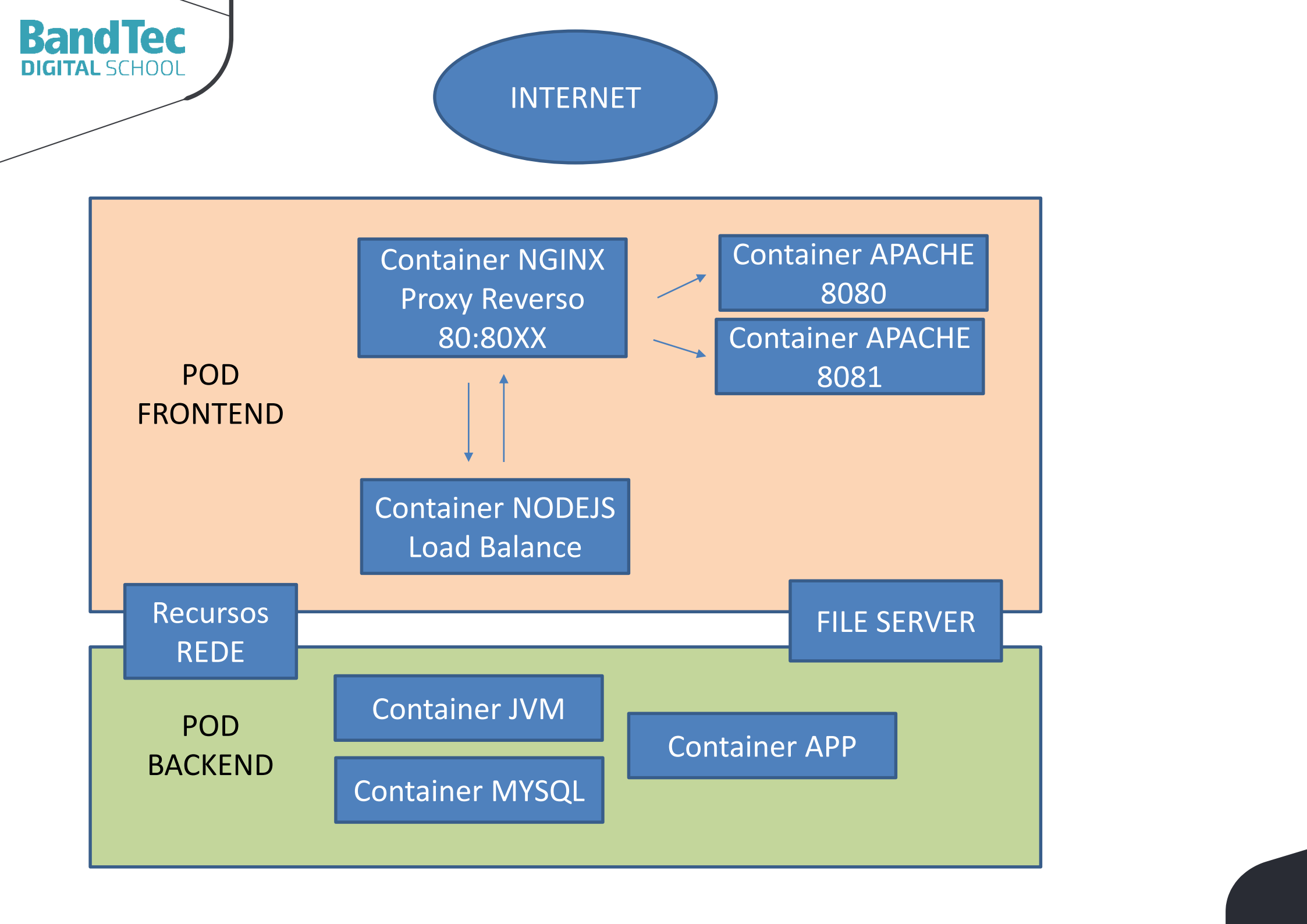
Server 2

SO Instalado
JVM ou JRE
Postgree

Aplicação (JAVA) + Banco de Dados (PostgreeSQL)

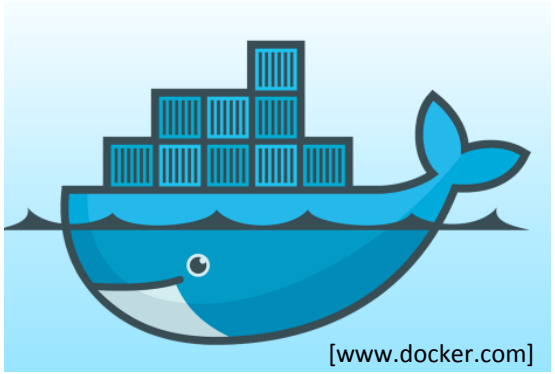
SO Instalado
JVM ou SDK
Postgree




BandTec
DIGITAL SCHOOL
INTERNET
POD
FRONTEND
Container NGINX
Proxy Reverso
80:80XX
Container APACHE
8080
Container APACHE
8081
Container NODEJS
Load Balance
Recursos
REDE
FILE SERVER
POD
BACKEND
Container JVM
Container APP
Container MYSQL

BandTec DIGITAL SCHOOL

# O que é Docker?




[www.docker.com]

*A Wikipedia define Docker como sendo:*

*"An open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of OS-level virtualization on Linux"*

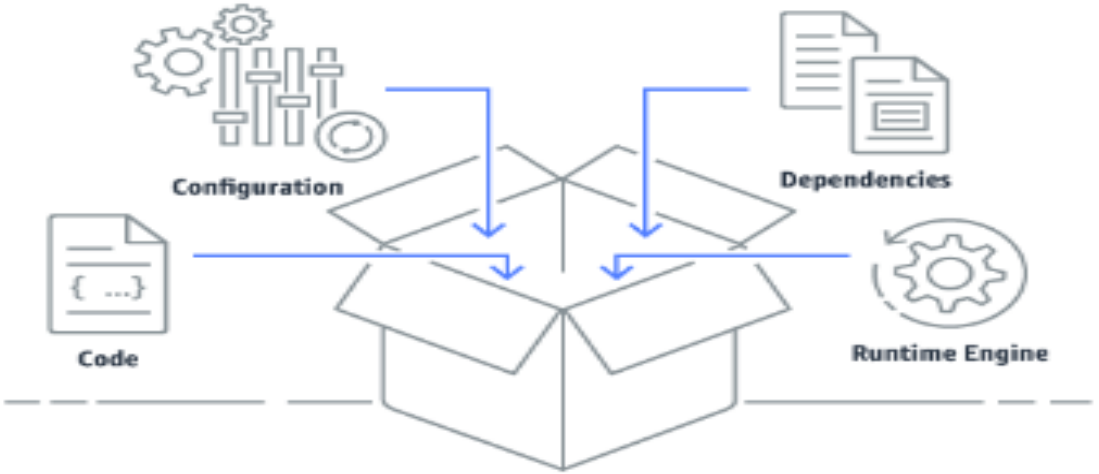
*In simpler words, Docker is a tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called containers) to run on the host operating system i.e. Linux. The key benefit of Docker is that it allows users to package an application with all of its dependencies into a standardized unit for software development. Unlike virtual machines, containers do not have the high overhead and hence enable more efficient usage of the underlying system and resources. Ex.: Ships Containers*

# O que são Container´s

The industry standard today is to use Virtual Machines (VMs) to run software applications. VMs run applications inside a guest Operating System, which runs on virtual hardware powered by the server's host OS.

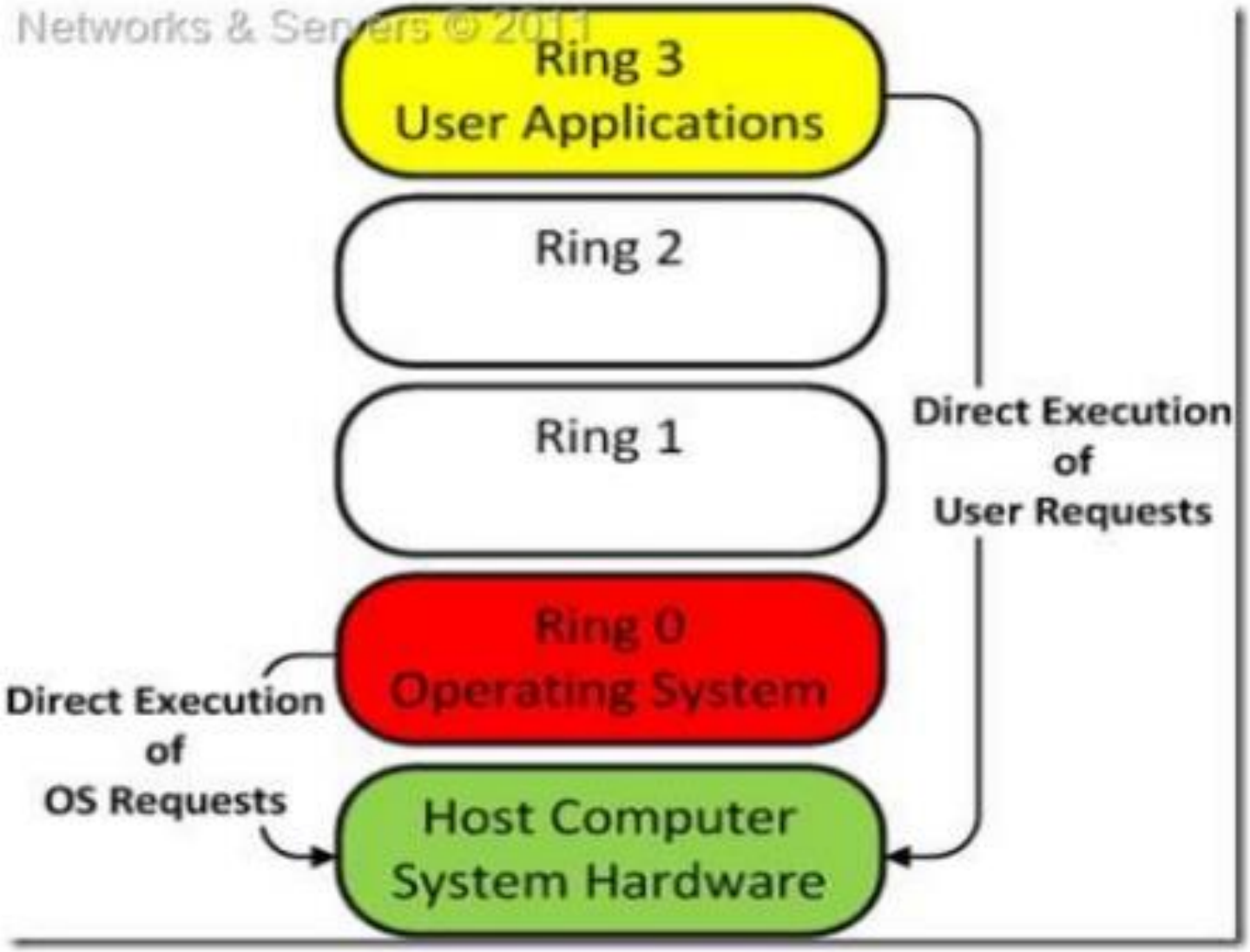
VMs are great at providing full process isolation for applications: there are very few ways a problem in the host operating system can affect the software running in the guest operating system, and vice-versa. But this isolation comes at great cost — the computational overhead spent virtualizing hardware for a guest OS to use is substantial.

Containers take a different approach: by leveraging the low-level mechanics of the host operating system, containers provide most of the isolation of virtual machines at a fraction of the computing power



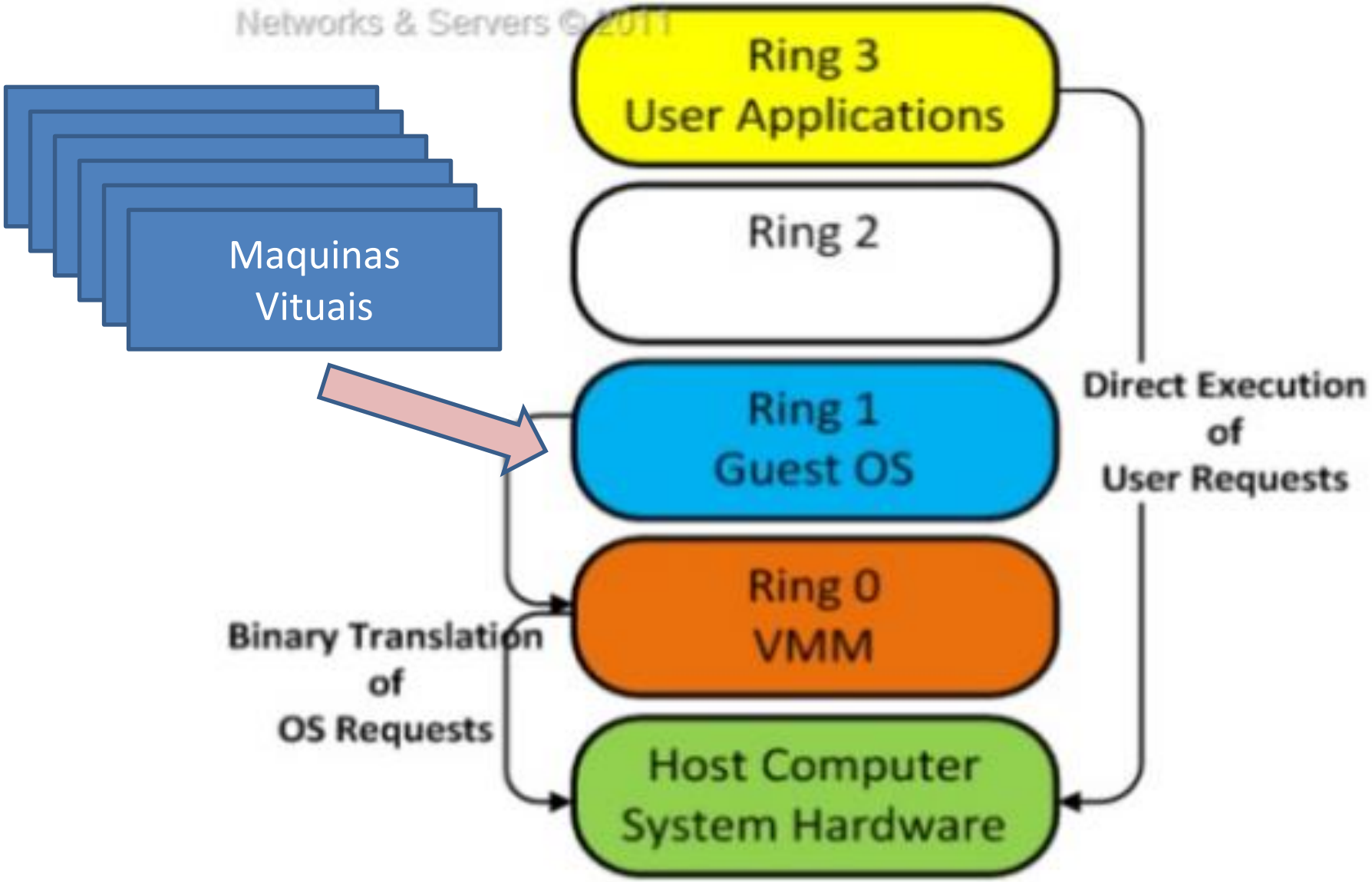
BandTec
DIGITAL SCHOOL

# x86 Architecture Model



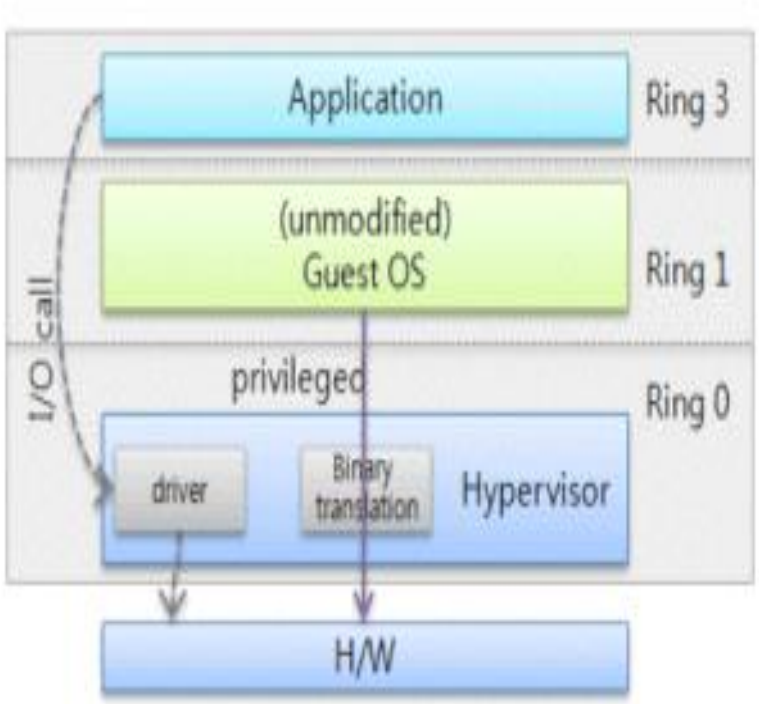
# Virtualized x86 Architecture Model



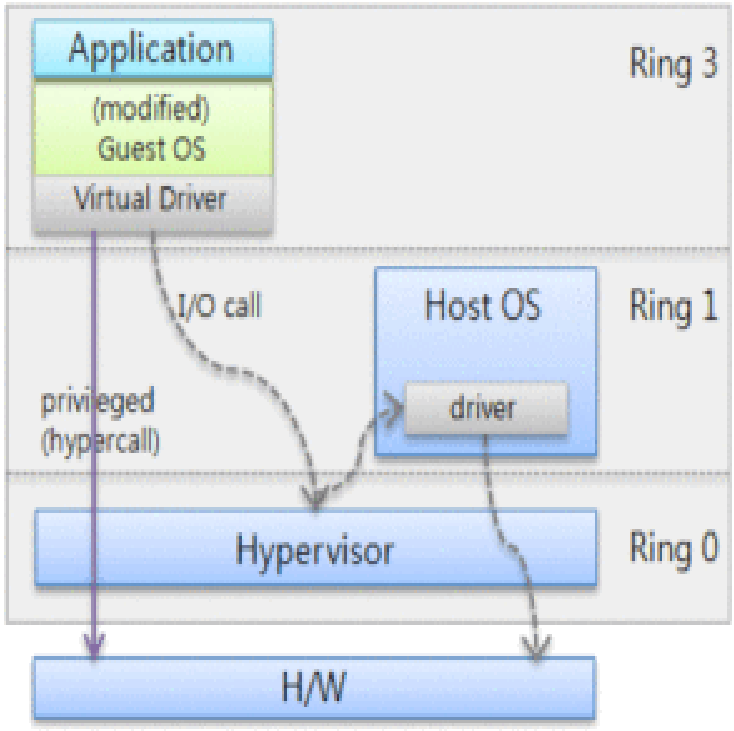


BandTec DIGITAL SCHOOL

# Virtualized x86 Architecture Model

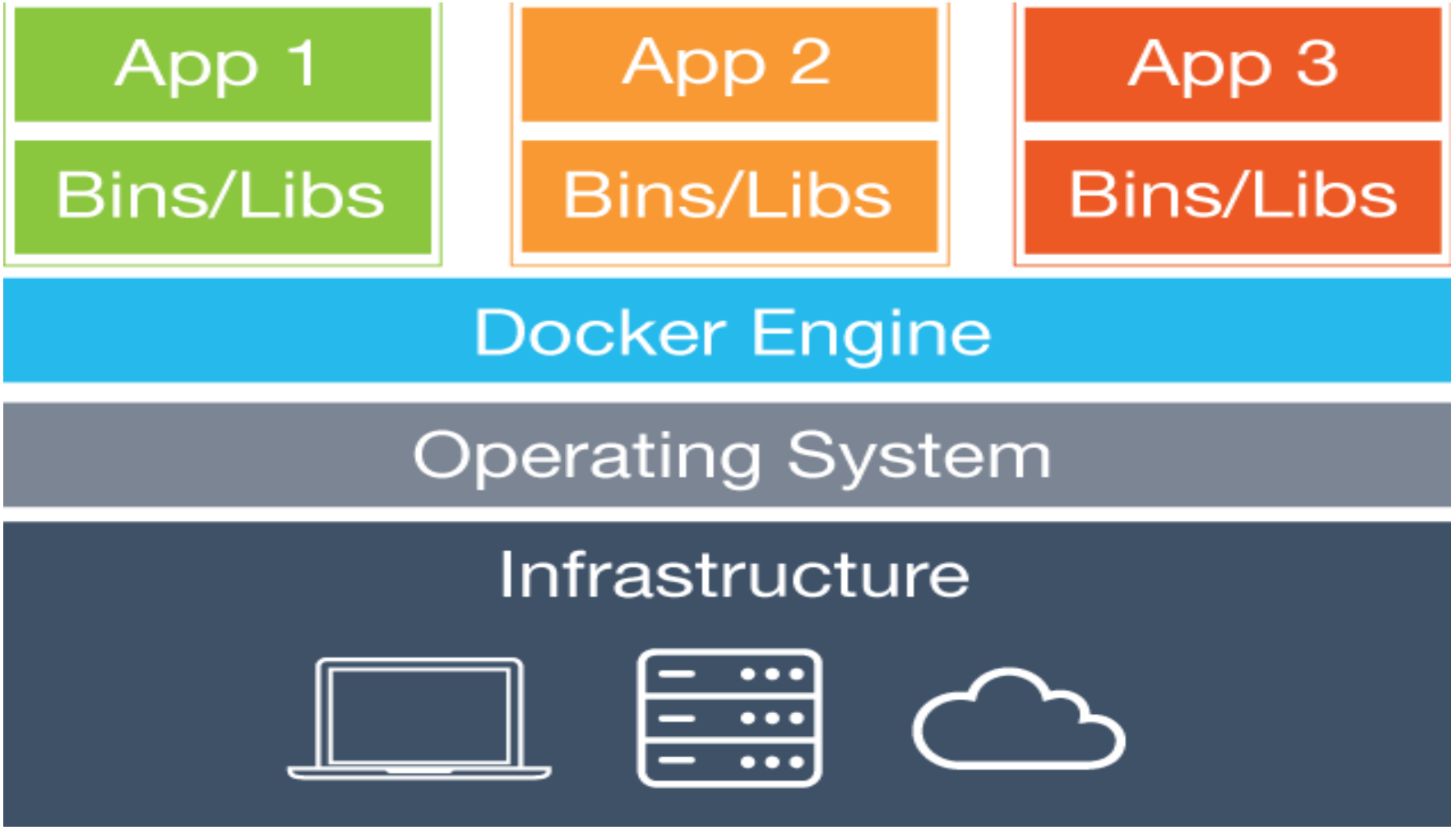



## Full Virtualization
## Monolithic




## Paravirtualization
## Micro-Kernelized

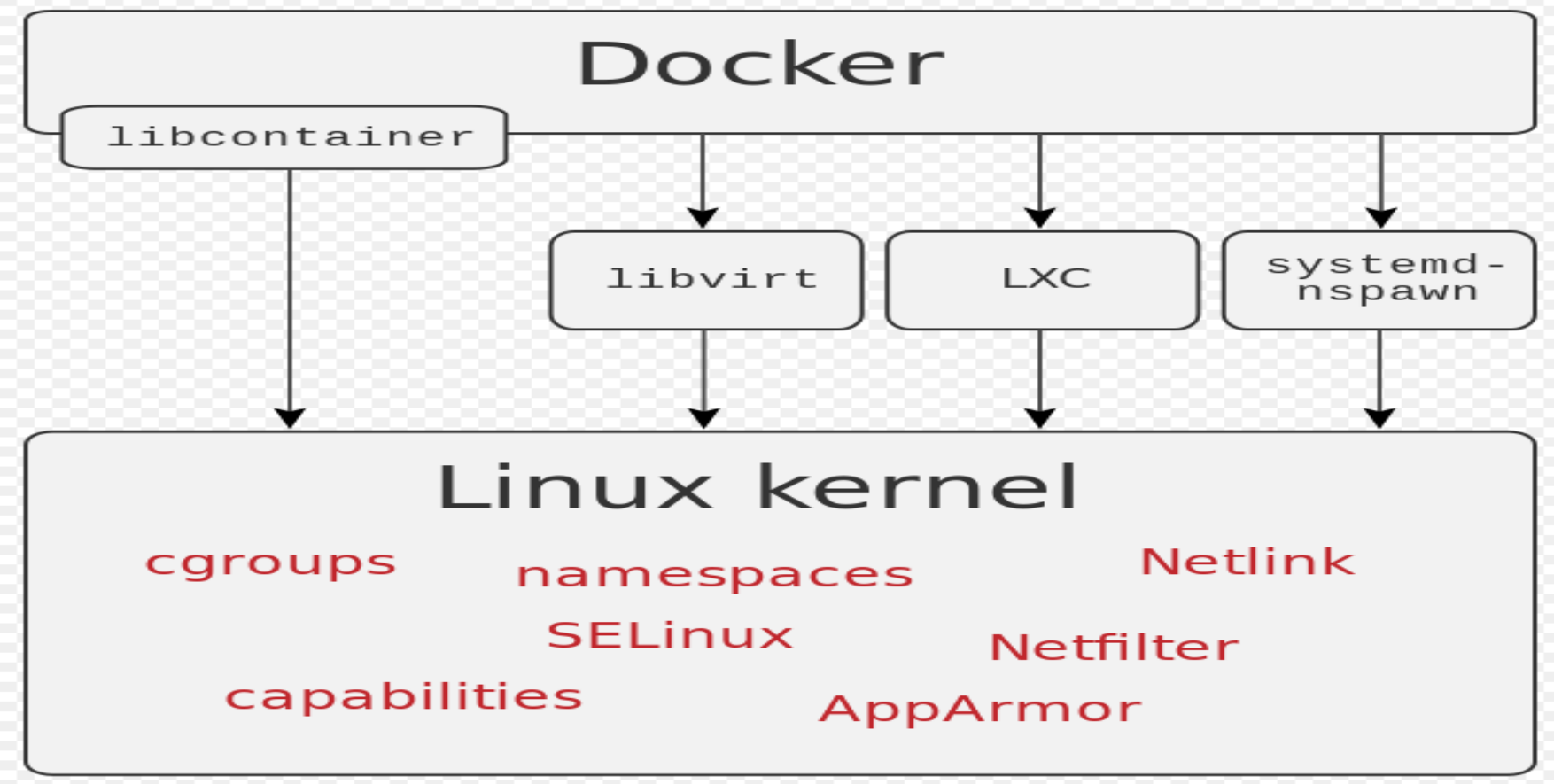
# Docker Engine



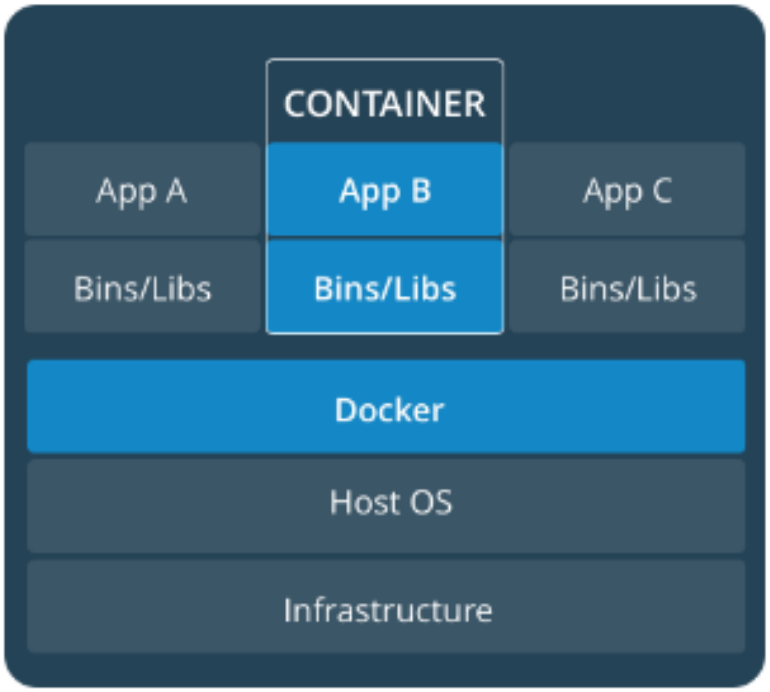
BandTec DIGITAL SCHOOL

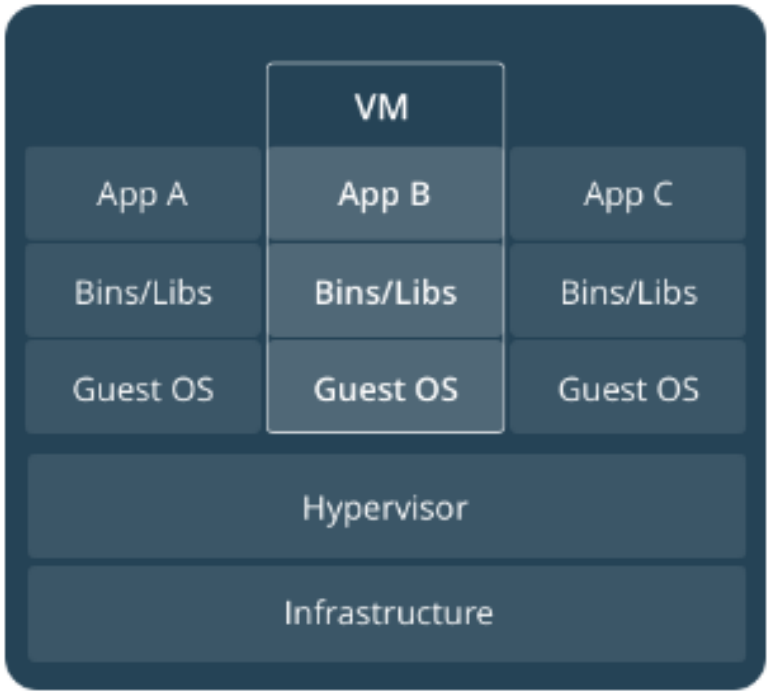
# Docker Technology

- libvirt: Platform Virtualization
- LXC (LinuX Containers): Multiple isolated Linux systems (containers) on a single host
- Layered File System



# Container x Virtual Machine




Each virtualized application includes not only the application, the necessary binaries and libraries but also an entire guest operating system

The Docker Engine container comprises just the application and its dependencies. It run as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VM´s but is much more portable and efficient

Source: https://www.docker.com/whatisdocker/

BandTec DIGITAL SCHOOL

# Container x Virtual Machine

Neste ponto do estudo é fundamental e importante não confundir containers com virtualização. Por exemplo, um servidor físico executando cinco máquinas virtuais teria um hypervisor e cinco sistemas operacionais separados sendo executados sobre ele. Já no caso de um servidor executando aplicações em containers, existiria um único sistema operacional, e cada container compartilhando o kernel do sistema operacional com os demais, mesmo permanecendo individualmente isolados. O impacto que isto traz para um ambiente de TI em termos de custos.; agilidade; escalabilidade; flexibilidade é enorme. Por isto se diz que Containers são o "upgrade" da virtualização

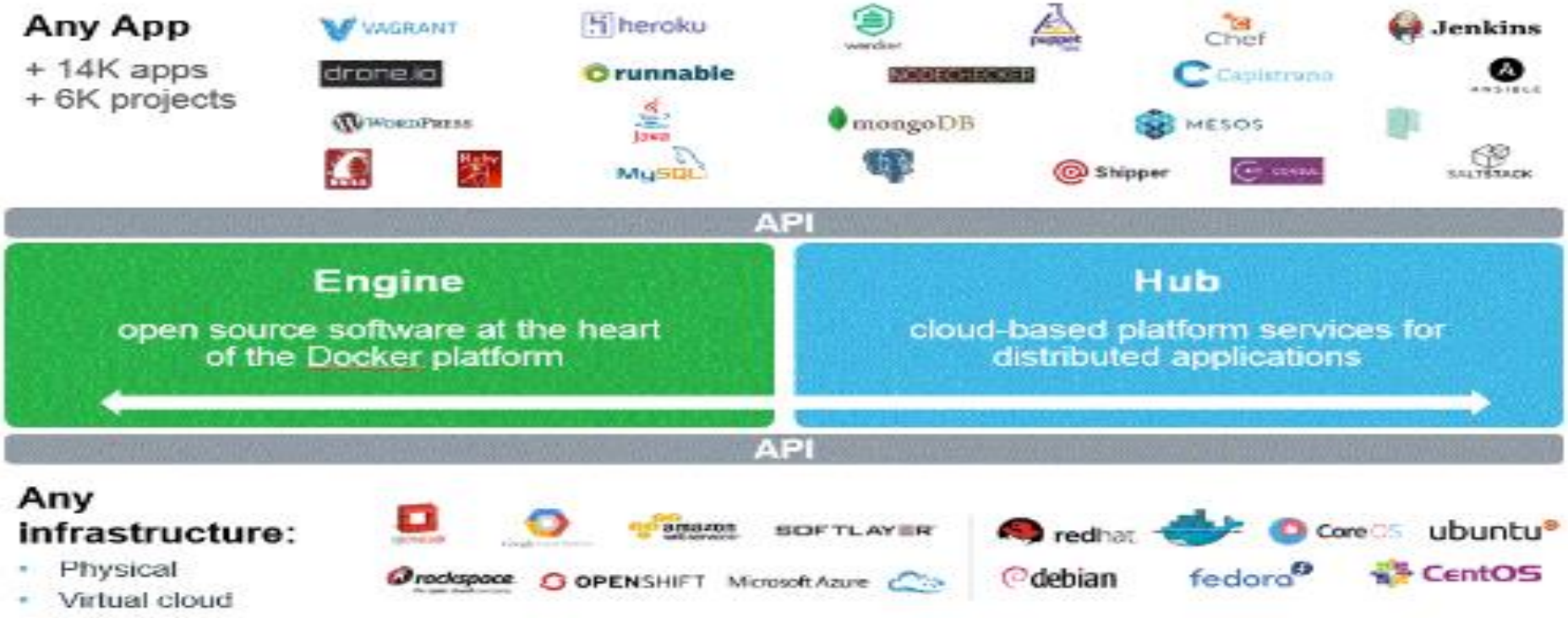
Outro grande benefício é o fato dos containers serem modulares: não há necessidade de executar uma aplicação complexa inteiramente dentro de um único container, sendo perfeitamente possível dividir a aplicação em módulos, por exemplo, separando o banco de dados do front-end. Esse tipo de arquitetura é conhecida como **microsserviço** (microservice) e aplicações que são desenvolvidas nesse modelo são bem mais simples de gerenciar, Por exemplo, como cada modulo conta com interfaces e operações bem definidas, é perfeitamente possível realizar atualizações individuais em cada módulo sem ter que reconstruir a aplicação como um todo

BandTec
DIGITAL SCHOOL

# Docker Hub

Docker started in March 2013, in a few months it had 8,741 commits from more than 460 contributors. 2.75 millions downloads and 14,000 "Dockerized" apps. Docker is supported by major technology and service providers like Canonical, Fedora, Google Cloud Platform, OpenStack, Rackspace and Red Hat.

In 2014 Docker also announced a first release of Docker Hub, providing a cloud based platform service for distributed applications. It provides container image distribution and change management, user and team collaboration, lifecycle workflow automation, and third-party services integration.



BandTec DIGITAL SCHOOL

# Container Chronology

- chroot: limiting filesystem view
- BSD jail (1995): better sandbox, networking, but limited
- Linux-VServer (2001): security
- Solaris Zones (2004)
- OpenVZ (2005) / Parallels
- LXC – Linux Containers (2008)
- Containers in the kernel (2007)

BandTec
DIGITAL SCHOOL

# From JAIL to DOCKER

- LXC: robust
- BSD Jails: well-designed
- lmctfy (Let Me Containerize That For You): Google quality
- OpenVZ: active development
- They have been pretty hard to use!
- DOCKER IS EASY TO USE. EVERYBODY CAN DO IT !

BandTec DIGITAL SCHOOL

# DOCKER is great for...

- Local develop/build/test pipelines - sandbox
- Builds that are “safer” to ship to production
- Testing software in different environments
- CI slave machines
- Creating mini-clusters for development/testing
- Packaging and software delivery – can replace RPMs
- Cloud Hybrid Escalation

BandTec DIGITAL SCHOOL

# Docker History


| | |
|---|---|
| **Original author(s)** | Solomon Hykes |
| **Developer(s)** | Docker, Inc. |
| **Initial release** | 13 March 2013; 4 years ago |
| **Stable release** | 17.03.0-ce[1] / 1 March 2017; 2 months ago |
| **Repository** | https://github.com/docker/docker, https://github.com/docker/docker.git |
| **Written in** | Go[2] |
| **Operating system** | Linux,[a] Windows |
| **Platform** | x86-64, ARM (experimental) with modern Linux kernel, or x86-64 Windows with Hyper-V capabilities |
| **Type** | Operating-system-level virtualization |
| **License** | Apache License 2.0 |
| **Website** | www.docker.com |

Source: Wikipedia

BandTec DIGITAL SCHOOL

# Docker History

Solomon Hykes started Docker in France as an internal project within dotCloud, a platform-as-a-service company,[40] with initial contributions by other dotCloud engineers including Andrea Luzzardi and Francois-Xavier Bourlet.[citation needed] Jeff Lindsay also became involved as an independent collaborator.[citation needed] Docker represents an evolution of dotCloud's proprietary technology, which is itself built on earlier open-source projects such as Cloudlets.

Docker was released as open source in March 2013. On March 13, 2014, with the release of version 0.9, Docker dropped LXC as the default execution environment and replaced it with its own libcontainer library written in the Go programming language. As of October 24, 2015, the project had over 25,600 GitHub stars (making it the 20th most-starred GitHub project), over 6,800 forks, and nearly 1,100 contributors.

A May 2016 analysis showed the following organizations as main contributors to Docker: The Docker team, Cisco, Google, Huawei, IBM, Microsoft, and Red Hat. A January 2017 analysis of LinkedIn profile mentions showed Docker presence grew by 160% in 2016.

- On September 19, 2013, Red Hat and Docker announced a collaboration around Fedora, Red Hat Enterprise Linux, and OpenShift.
- On October 15, 2014, Microsoft announced integration of the Docker engine into the next (2016) Windows Server release, and native support for the Docker client role in Windows. See: https://blogs.msdn.microsoft.com/msgulfcommunity/2015/06/20/what-is-windows-server-containers-and-hyper-v-containers/
- On 10 November 2014, Docker announced a partnership with Stratoscale.
- On December 4, 2014, IBM announced a strategic partnership with Docker that enables Docker to integrate more closely with the IBM Cloud.
- On June 22, 2015, Docker and several other companies announced that they are working on a new vendor and operating-system-independent standard for software containers.
- On June 8, 2016, Microsoft announced that Docker now could be used natively on Windows 10 with Hyper-V Containers, to build, ship and run containers utilizing the Windows Server 2016 Technical Preview 5 Nano Server container OS image.
- On October 4, 2016, Solomon Hykes announced InfraKit as a new self-healing container infrastructure effort for Docker container environments.

Source: Wikipedia

BandTec DIGITAL SCHOOL

# Docker Platform - Editions

## Docker Enterprise Edition

Designed for enterprise development and IT teams who build, ship, and run business critical applications in production at scale. Integrated, certified, and supported to provide enterprises with the most secure container platform in the industry to modernize all applications. Docker EE Advanced comes with enterprise add-ons like UCP and DTR.

## Docker Community Edition

Get started with Docker and experimenting with container-based apps. Docker CE is available on many platforms, from desktop to cloud to server. Build and share containers and automate the development pipeline from a single environment. Choose the Edge channel to get fast access to the latest features, or the Stable channel for more predictability.

# Docker e Kubernetes

No final de 2017, a Docker lançou a compatibilidade entre o Docker e o Kubernetes, e com essa versão do Docker agora é possível ter no mesmo cluster o Swarm (a versão nativa de kubernet do Docker) e o Kubernetes. É só escolher qual o orquestrador deverá ser utilizado no momento da implantação dos serviços.

BandTec DIGITAL SCHOOL

# Docker Platform - Sandbox

## Docker for Mac

A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.

## Docker for Windows

A native Windows application which delivers all Docker tools to your Windows computer.

## Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.

## Docker Cloud

A hosted service for building, testing, and deploying Docker images to your hosts.

## Docker for AWS

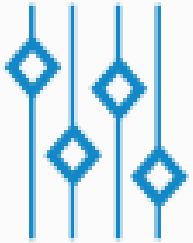
Deploy your Docker apps on AWS.

## Docker for Azure

Deploy your Docker apps on Azure.

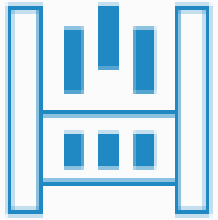
BandTec DIGITAL SCHOOL

# Docker Platform – Add-ons



## Docker Universal Control Plane

(UCP) Manage a cluster of on-premise Docker hosts like a single machine with this enterprise product.

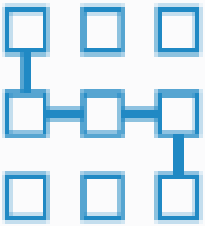


## Docker Trusted Registry

(DTR) An enterprise image storage solution you can install behind a firewall to manage images and access.

BandTec DIGITAL SCHOOL

# Docker Platform – Tools



## Docker Compose

Define application stacks built using multiple containers, services, and swarm configurations.

## Docker Machine

Automate container provisioning on your network or in the cloud. Available for Windows, macOS, or Linux.

BandTec DIGITAL SCHOOL

# Docker – Technology Radar

**Platforms**

## Docker

**ADOPT**

We remain excited about **Docker** as it evolves from a tool to a complex platform of technologies. Development teams love Docker, as the Docker image format makes it easier to achieve parity between development and production, making for reliable deployments. It is a natural fit in a microservices-style application as a packaging mechanism for self-contained services. On the operational front, Docker support in monitoring tools (Sensu, Prometheus, cAdvisor, etc.), orchestration tools (Kubernetes, Marathon, etc.) and deployment-automation tools reflect the growing maturity of the platform and its readiness for production use. A word of caution, though: There is a prevalent view of Docker and Linux containers in general as being "lightweight virtualization," but we would not recommend using Docker as a secure process-isolation mechanism, though we are paying attention to the introduction of user namespaces and seccomp profiles in

Source: http://www.thoughtworks.com/radar/tools/docker

BandTec DIGITAL SCHOOL

# Terminology - IMAGE

Images are persisted snapshot that can be run

- *images:* List all local images
- *run*: Create a container from an image and execute a command in it
- *tag*: Tag an image
- *pull*: Download image from repository
- *rmi*: Delete a local image
- This will also remove intermediate images if no longer used

BandTec DIGITAL SCHOOL

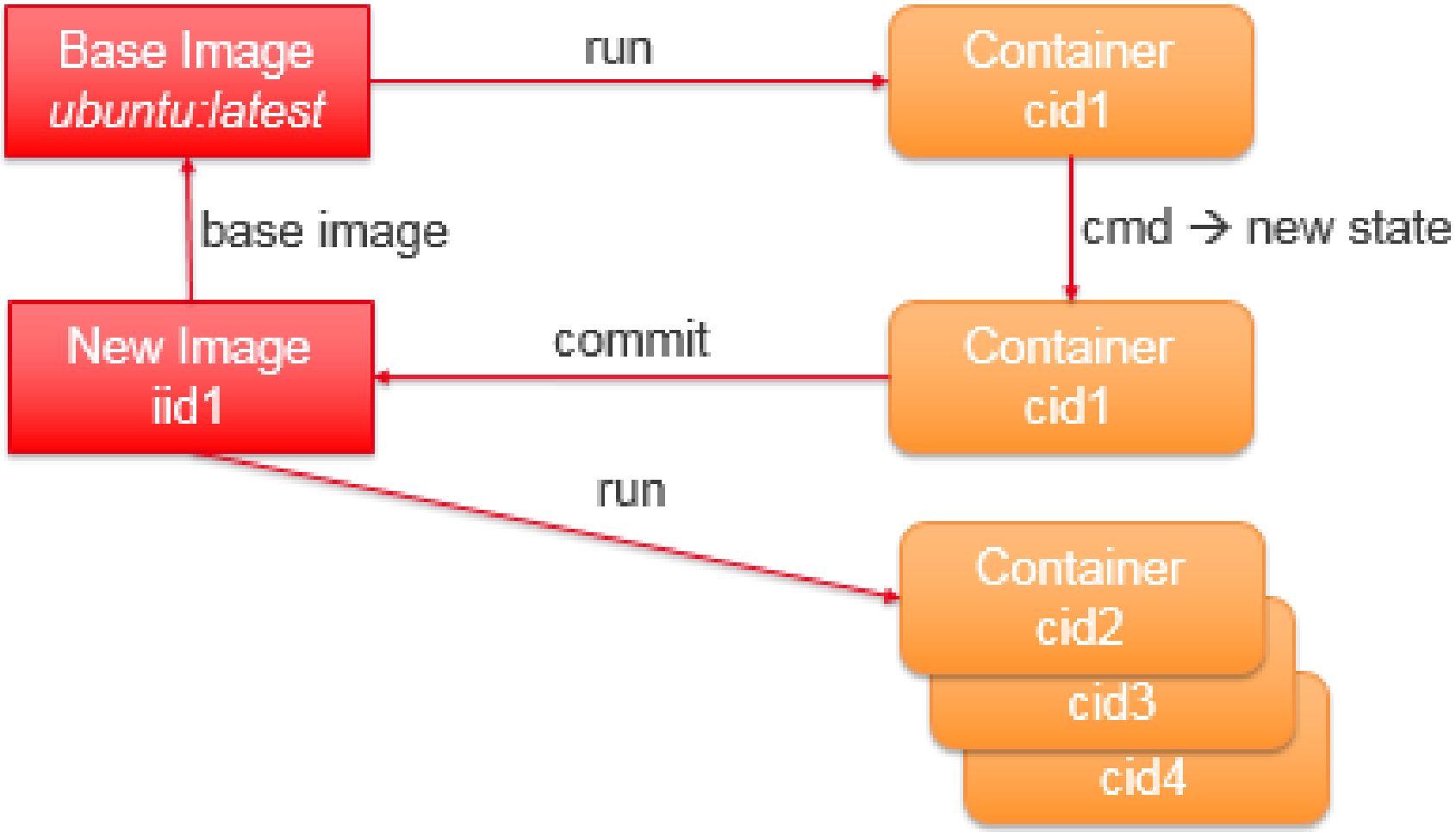
# Terminology - CONTAINER

Containers are runnable instance of an image

- *ps:* List all running containers
- *ps –a*: List all containers (incl. stopped)
- *top*: Display processes of a container
- *start*: Start a stopped container
- *stop*: Stop a running container
- *pause*: Pause all processes within a container
- *rm*: Delete a container
- *commit*: Create an image from a container

BandTec
DIGITAL SCHOOL

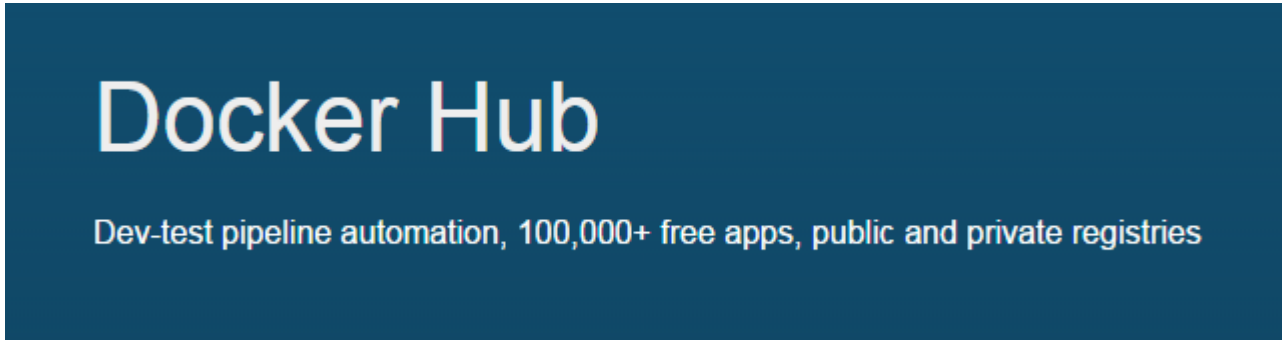
# Image vs. Container



BandTec
DIGITAL SCHOOL

# Docker HUB

https://www.docker.io/account/signup/




Conclusion:

Ship the entire environment instead of just code !

BandTec DIGITAL SCHOOL

# Kubernetes

O Kubernetes, ou K8S (no jargão de TI se pronuncia “keyeights” ou “kube”, é um sistema de cluster e orquestração para containers Docker que suporta outros sistemas de containers, como o Rocket, por exemplo

Em outras palavras, o Kubernetes é uma ferramenta de código aberto usada para orquestrar e gerenciar clusters de containers. Só que aqui o cluster são várias máquinas com um engine de container, que na maioria dos casos é o Docker a engine mais usada para esse trabalho de provisionar os containers nos hosts do cluster. Podemos também afirmar que o Kubernetes é (ou pretende ser) uma ferramenta de orquestração multi-cloud

A palavra Kubernetes tem origem no termo grego “Kuvernetes”, que representa a pessoa que pilota um navio e dai o logotipo ser um leme

BandTec DIGITAL SCHOOL

# Kubernetes

O **Kubernetes** foi criado e desenvolvido pelos engenheiros da Google, uma do pioneiras no desenvolvimento da tecnologia de containers e que executa vários do seus serviços em containers, como Google Docs; Google Play; Google Drive e Gmail etc. A Google chega a realizar mais de 2 bilhões de implantações de contêineres po semana e que são viabilizadas por uma plataforma interna chamada **BORG** antecessor do **Kubernetes** e que serviu como base para seu desenvolvimento

Em 2015 o **Kubernetes** foi doado para a "Cloud Native Computing foundation" Linux Foundations, e se tornou um projeto Open Source. Uma curiosidade sobre Kubernetes é que os sete raios do logotipo fazem referência ao nome original d projeto, "Project Seven of Nine" (Projeto Sete de Nove) que também referencia personagem BORG – Seven of Nine – de Star Trek - Voyager

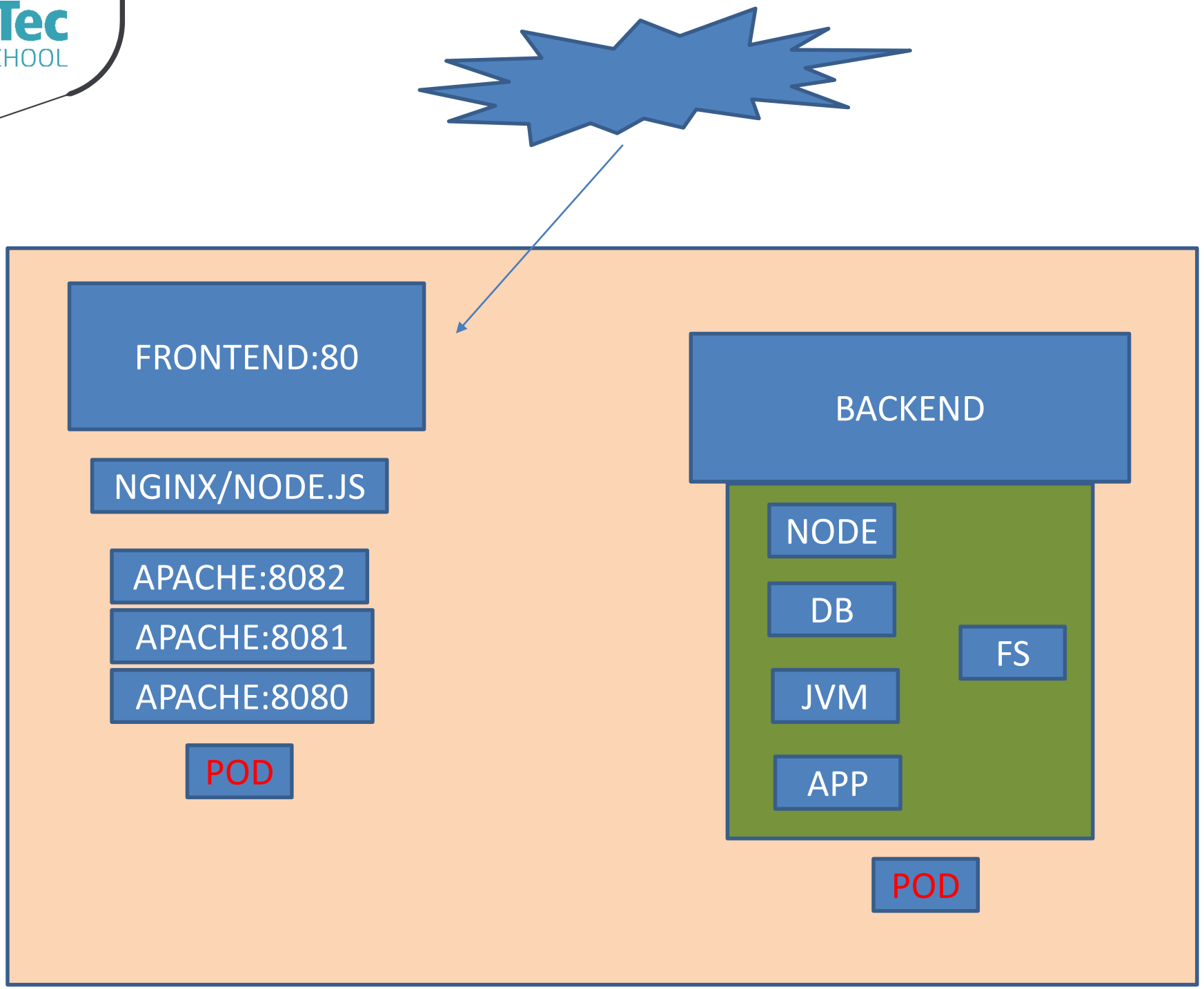
BandTec DIGITAL SCHOOL

# Kubernetes

Resumindo, e de forma simples, podemos dizer que Kubenetes é uma plataforma de código aberto usada para orquestrar e gerenciar clusters de containers, eliminando a maior parte dos processos manuais necessários para implantar e escalar os aplicativos em containers isolados que de outar forma teriam de ser montados por algum processo manual ou automático . Ou seja, quando se é necessário agrupar em clusters os hosts executados nos containers Linux, que podem estar em clouds públicas, privadas ou híbridas, o Kubernetes ajudará a gerenciar esses clusters com facilidade e efetividade

O Kubernetes organiza os containers em grupos chamados de **PODS** e isso permite solucionar boa parte dos problemas relacionados a sua proliferação. Os pods criam uma camada extra de abstração, dessa forma fica bem mais fácil controlar a carga de trabalho, e fornecer serviços necessários ao funcionamento dos containers, como rede e armazenamento

Entendendo o aspecto modular dos contêineres podemos dizer que um ambiente de produção vai incluir vários contêineres espalhados em múltiplos hosts e utilizando poder de orquestração do Kubernetes, é bem mais simples criar e gerenciar serviços de aplicativos abrangendo múltiplos containers, programar como serão usados no cluster, escalá-los e gerenciar a sua integridade ao longo do tempo, sendo também possível a integração com serviços de segurança, rede, armazenamento, monitoramento, medição e outros


FRONTEND:80
NGINX/NODE.JS
APACHE:8082
APACHE:8081
APACHE:8080
POD
BACKEND
NODE
DB
FS
JVM
APP
POD
POD: LOJA VIRTUAL
POD DESE: LOJA VIRTUAL

BandTec DIGITAL SCHOOL

# Funcionalidades

• Orquestrar containers em múltiplos hosts, em clouds públicas, privadas ou híbridas

• Otimizar o uso do hardware, maximizando a disponibilidade de recursos para execução dos aplicativos

• Maior agilidade para escalar aplicativos em containers e recursos relacionados

• Gerenciar e automatizar a maior parte das implantações e atualizações de aplicativos

• Garantir a integridade e auto recuperação dos aplicativos em containers, com posicionamento, reinício, replicação e escalonamento automáticos

BandTec DIGITAL SCHOOL

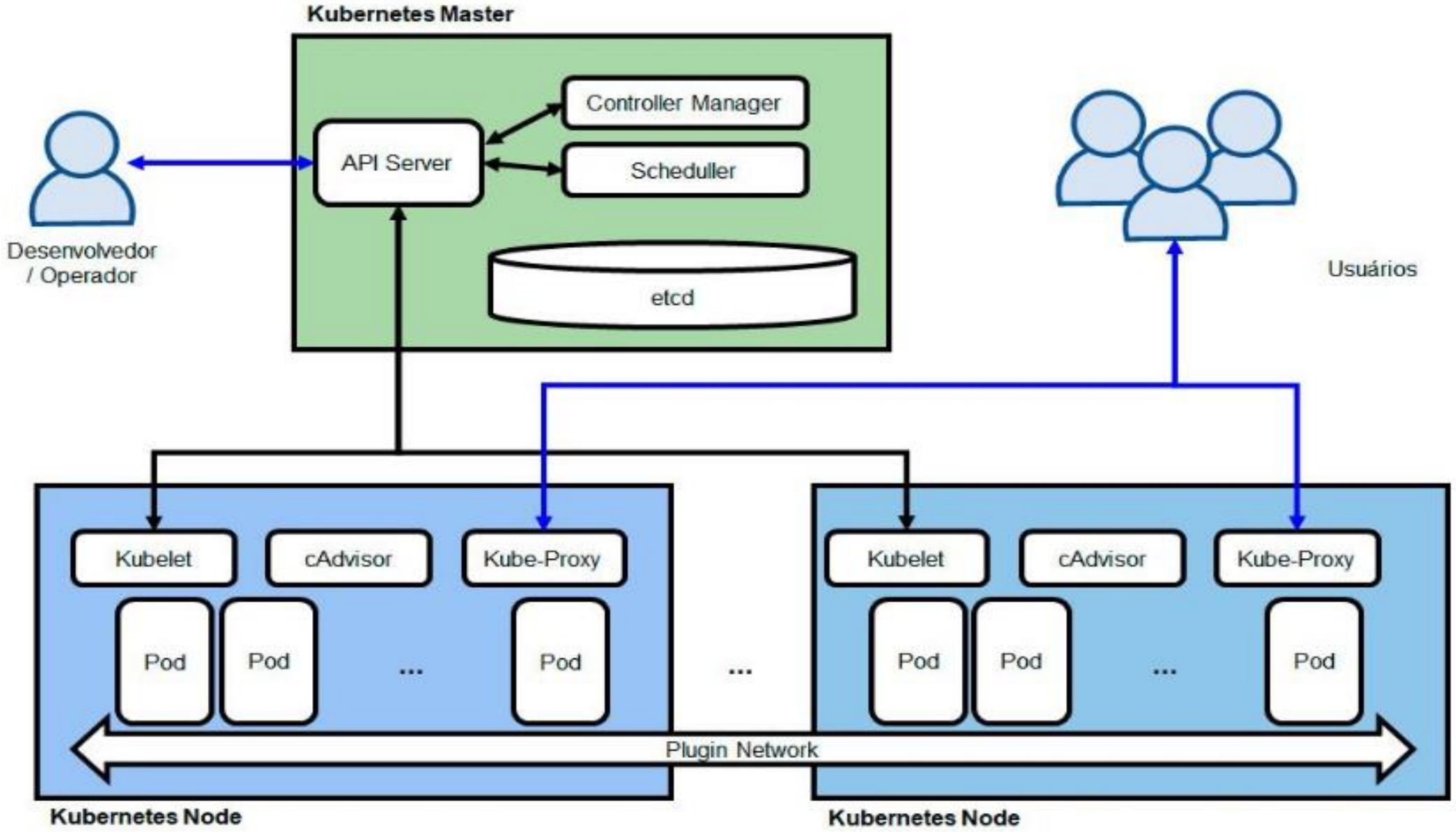
# Arquitetura

A arquitetura do Kubernetes funciona parecido com o conceito do SWARM do DOCKER, onde temos os nós MASTER e os nós WORKER sendo que o conjunto de WORKERS formam um cluster que pode ser orientado a executar um container (no caso um POD que é a menor unidade dentro de um cluster Kubernetes e que pode ser definido como um grupo de containers que são implantados em um único nó de trabalho) de um serviço específico ou vários containers atendendo a vários serviços. Por exemplo. Podemos ter um cluster de containers executando uma solução de WEB Server enquanto outros cluster pode estar executando uma solução de Load Balance e ainda um terceiro cluster de containers executando instâncias de base de dados que ainda podem estar configuradas em cluster!!

Exemplo: imagine um provedor que necessita montar uma infraestrutura de loja virtual com balanceamento de carga e base de dados para atender uma Black Friday. Num cenário clássico com virtualiazação teríamos de ter servidores virtuais rodando Apache para Web Server; outros servidores para Load Balance (ou appliance específico) e outros servidores para montar bases de dados em SQL ou Oracle. Imagine que este provedor deduziu que, para atender a demanda terá de subir 100 VM´s para Web Server 4 para Load Balance e 4 para Base de Dados num total de 108 VM´s, cada uma com seu SO instalado; Licenças; Redes configuradas; Storage; Cluster de SO; etc. Num cenário com containers poderíamos ter 2 ou 4 máquinas bem dimensionadas e dentro delas executar containers de Web Server; Loab Balance e Banco de Dados separados em clusters (Workers) orquestrados pelo Kubernetes

BandTec DIGITAL SCHOOL

# Arquitetura: Diagrama



# Arquitetura: Componentes

**API Server**: O servidor de API é um componente essencial e serve a API do Kubernetes usando JSON sobre HTTP, que fornece a interface interna e externa para o Kubernetes. O servidor de API processa e valida as solicitações REST e atualiza o estado dos objetos da API no etcd, permitindo, assim, que os clientes configurem cargas de trabalho e containers nos nós do Worker

**Controller Manager:** O processo que executa os principais controladores do Kubernetes, como o **DaemonSet Controller** e o **Replication Controller**. Os controladores se comunicam com o servidor de API para criar, atualizar e excluir os recursos que gerenciam como, por exemplo, os **PODS**

**Scheduller**: É o componente que seleciona em qual nó um **POD** não programado (a entidade básica gerenciada pelo Scheduller) é executado, com base na disponibilidade de recursos. O **Scheduler** rastreia o uso de recursos em cada nó para garantir que a carga de trabalho agendada não exceda os recursos disponíveis. Para essa finalidade, o **Scheduller** deve conhecer os requisitos de recursos, a disponibilidade de recursos e outras restrições e diretivas de políticas fornecidas pelo usuário, como requisitos de qualidade de serviço (QoS), localidade de dados e assim por diante. Em essência gerencia o recurso "oferta" com a "demanda" da carga de trabalho (Vide VMware DRS)

BandTec DIGITAL SCHOOL

# Arquitetura: Componentes

**Node**: Um nó é uma máquina controlada no Kubernetes. Um nó pode ser uma máquina virtual ou física, dependendo do cluster. Possui os Serviços necessários para executar os Pods e é gerenciado pelos componentes principais. Os Serviços em um nó incluem **Docker**, **Kubelet** e **Kube-Proxy**

**Pod**: O menor e mais simples objeto do Kubernetes. Um **Pod** representa um conjunto de containers em execução no seu cluster. Pode ser um container rodando nginx, php, apache, etc. É a abstração de containers, pode ter um ou mais containers em um Pod e posso ter vários Pods dentro de cada nó

**ReplicaSet**: A nova geração do **ReplicationController**, o **ReplicaSet** garante que um número especificado de réplicas de **PODs** esteja em execução ao mesmo tempo

**Kubelet**: Um agente que é executado em cada nó no cluster. Isso garante que os containers estejam sendo executados em um **POD**

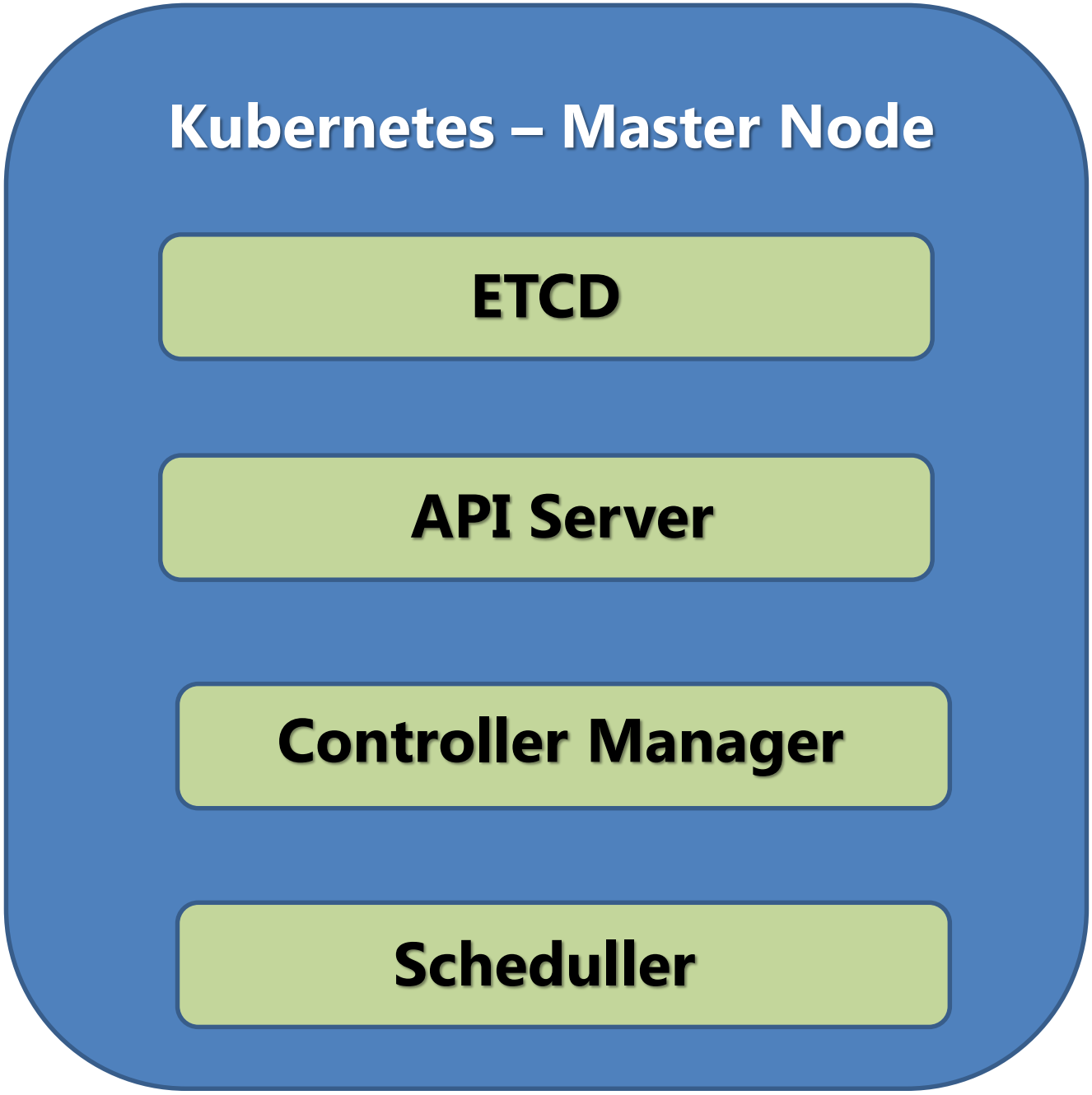
**Kubectl:** Uma ferramenta de linha de comando para se comunicar com um servidor Kubernetes via API

**ETCD**: é uma base de dados de chave valor. Ele armazena os dados de configuração do cluster e o estado do cluster

**cAdvisor:** um container open source que funciona como um agente de análise de uso e performance de todo o ambiente

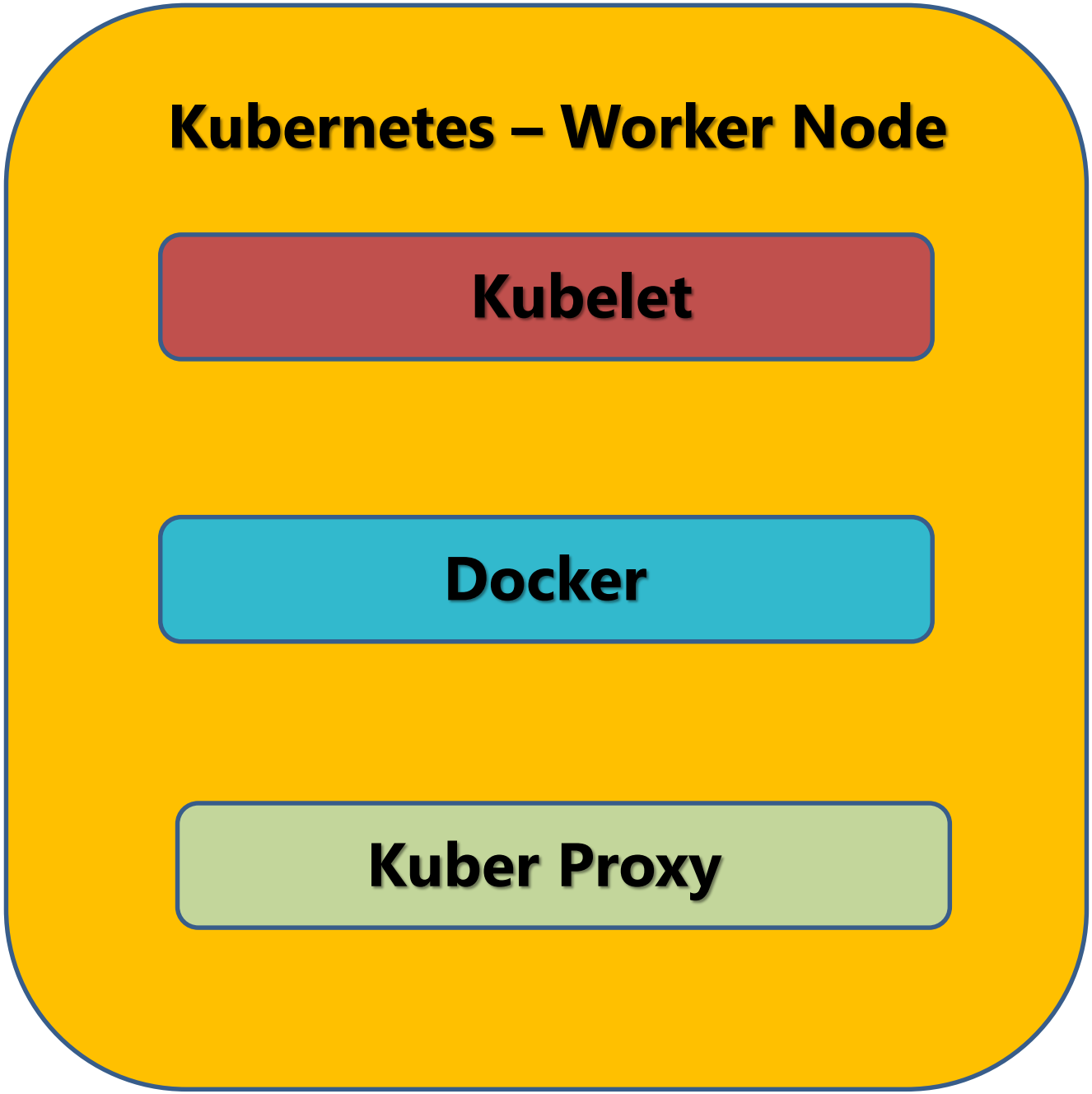
BandTec
DIGITAL SCHOOL

# Arquitetura: Master

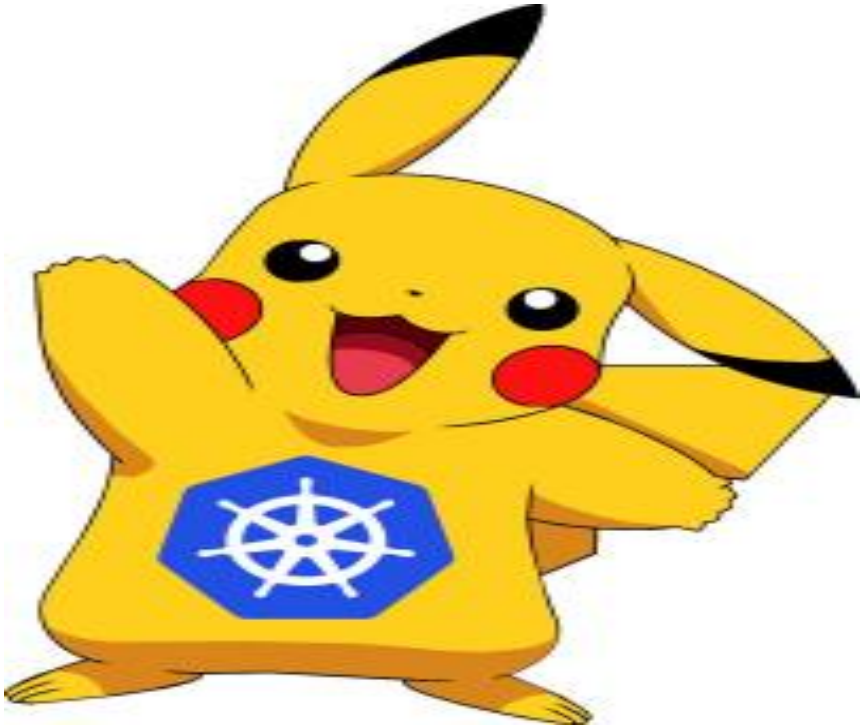


BandTec DIGITAL SCHOOL

# Arquitetura: Worker



# Case de Sucesso: Pokemon GO



Pokémon GO was the largest Kubernetes deployment on Google Container Engine ever. Due to the scale of the cluster and accompanying throughput, a multitude of bugs were identified, fixed and merged into the open source project.

https://cloud.google.com/blog/products/gcp/bringing-pokemon-go-to-life-on-google-cloud

# Hands on !

- VM image of CentOS Linux
- Configuration of Docker repository
- Internet access
- Docker running examples

```
yum update
yum install -y yum-utils
yum-config-manager --add
https://download.docker.com/linux/centos/docker-ce.repo
yum makecache fast
yum -y install docker-ce
systemctl start docker
systemctl enable docker
docker run hello-world
docker run --it ubuntu bash
```

BandTec DIGITAL SCHOOL

# Docker – Hello World

```
[root@matrix-01 ~]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

[root@matrix-01 ~]# _
```

# Docker – Servidor WEB

Exemplo: criando uma imagem de servidor Linux com Apache servindo conteúdo HTML acessível via socket de escuta (listener) na porta 80

Baixando imagem de um apache diretamente do Docker Hub:

```
docker pull httpd
```

Você pode visualizar a imagem que foi baixada:

```
docker image ls
```

Crie um diretório para montar o arquivo Dockerfile para criar o projeto em Docker:

```
mkdir apache
vi Dockerfile
```

Conteúdo do arquivo Dockerfile:

```
FROM httpd:2.4
COPY ./public-html/ /usr/local/apache2/htdocs
```

Salve o arquivo e crie o diretório public-html e crie uma página de teste index.html para ser servida pelo WEB Server

# Docker – Servidor WEB

Agora vamos criar a imagem de container que juntará o arquivo Dockerfile, o conteúdo html e a imagem containerizada do Apache numa mesma imagem. Seria o equivalente a compilar um programa. No final vc terá construído um container de Web Server funcional:

```
docker build -t meu-apache .
```

Agora vamos iniciar o container e instrui-lo para escutar na porta 8080 (de sua máquina local) e redirecionar o tráfego para a porta 80 do servidor Apache containerizado:

```
docker run -dit --name meu-primeiro-container -p 8080:80 meu-apache
```

Com isso o container deve iniciar e abrir um socket de escuta na porta 8080. Você pode checar isso com o comando netstat e com o commando do docker para listar containers em execução:

```
netstat -antp | grep -i :8080
docker ps
```

Após isso, pode abrir um navegador e acessar o endereço lcalhost de sua máquina na porta 8080 ou testar com o curl e a página index.html que você criou deverá abrir

# Docker – Servidor WEB

Agora vamos criar a imagem de container que juntará o arquivo Dockerfile, o conteúdo html e a imagem containerizada do Apache numa mesma imagem. Seria o equivalente a compilar um programa. No final vc terá construído um container de Web Server funcional:

```
docker build -t meu-apache .
```

Agora vamos iniciar o container e instrui-lo para escutar na porta 8080 (de sua máquina local) e redirecionar o tráfego para a porta 80 do servidor Apache containerizado:

```
docker run -dit --name meu-primeiro-container -p 8080:80 meu-apache
```

Com isso o container deve iniciar e abrir um socket de escuta na porta 8080. Você pode checar isso com o comando netstat e com o commando do docker para listar containers em execução:

```
netstat -antp | grep -I :8080
docker ps
```

# Docker – Servidor WEB

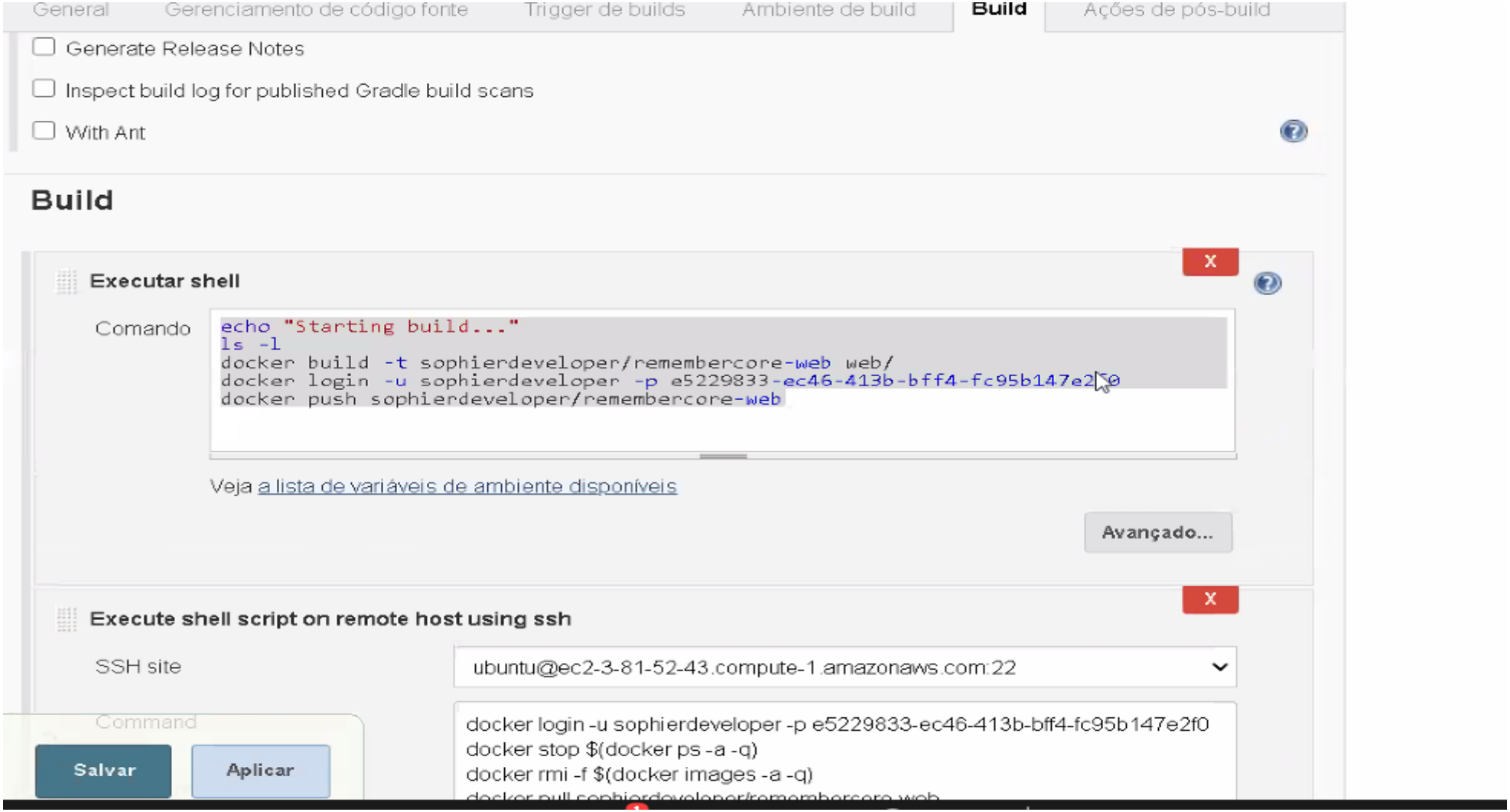
Com isso, se vc abrir um navegador e digitar http://localhost:8080 deve conseguir visualizar a página index.html que criou. Se quiser testar sem navegador pode utilizar o comando curl e abrir a url:

curl http://localhost:8080

yum -y update --nobest

dnf install https://download.docker.com/linux/centos/7/x86_64/stable/Packages/containerd.io-1.2.6-3.3.el7.x86_64.rpm

# Docker – Servidor WEB



BandTec
DIGITAL SCHOOL

rogerio.chola@bandtec.com.br