

# BOOTCAMP

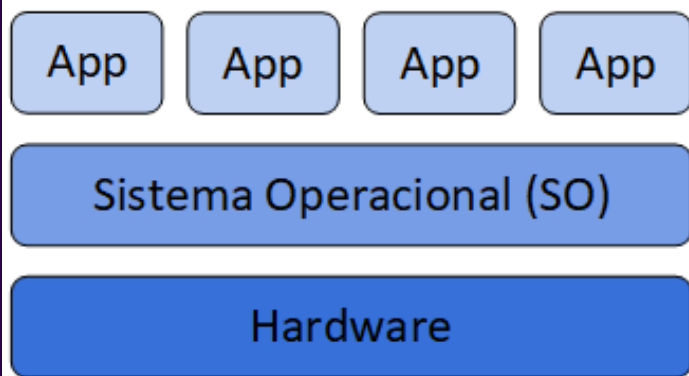
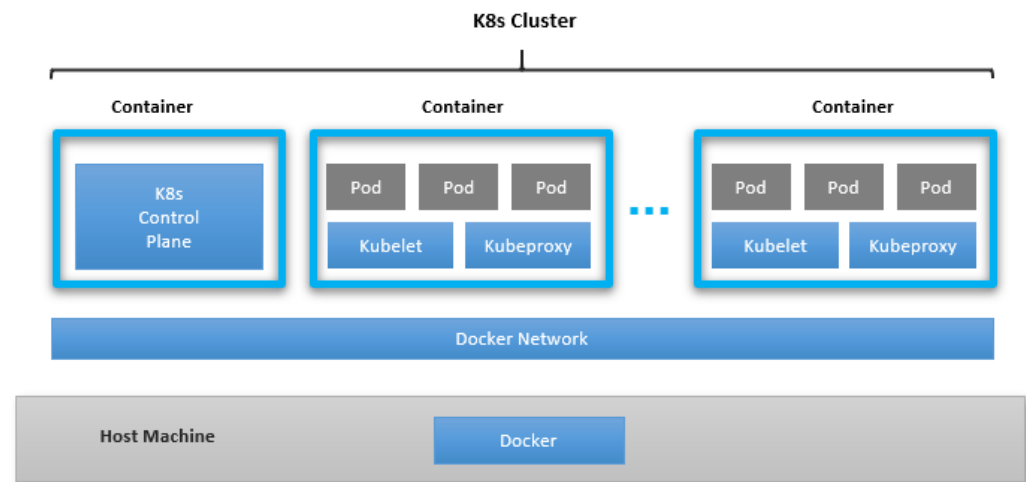
APPLICATION NODE.JS

DEPLOY APPLICATION WITH YARN

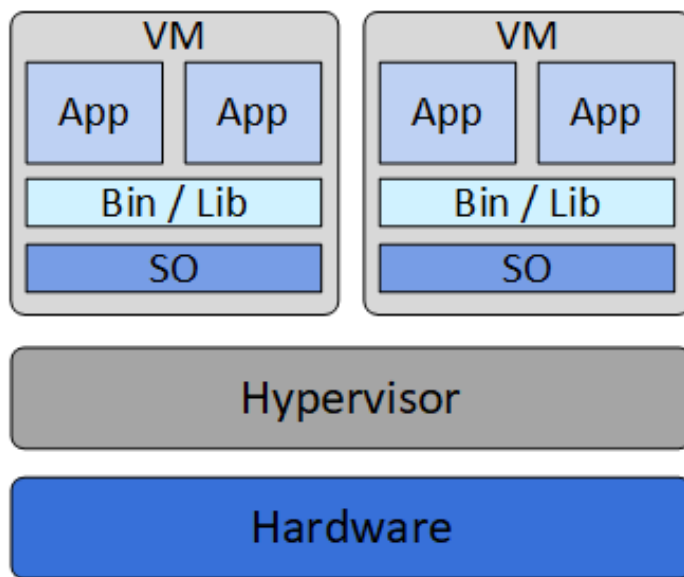
DOCKER

K8S COM MINIKUBE E AWS EKS

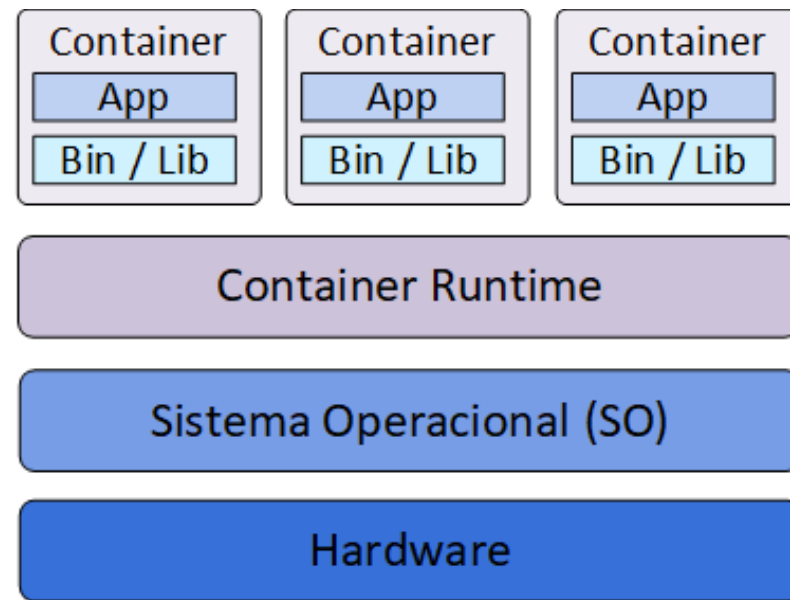
TERRAFORM



**Era Tradicional**



**Era Virtualização**



**Era do Containerização**

Available commands in your project:

```
#yarn develop
-Start Strapi in watch mode.
(Changes in Strapi project files will trigger
a server restart)
```

```
#yarn start
-Start Strapi without watch mode.
```

```
#yarn build
-Build Strapi admin panel.
```

```
#yarn strapi
-Display all available commands.
```

You can start by doing:

```
#cd my-folder-project
#yarn develop
```

# APLICAÇÃO LOCAL (TRADICIONAL)

O Strapi é um Serviço de Gerenciamento de Conteúdo (Content Management Service, ou CMS, em inglês) open-source e headless. Ele permite ao usuário criar APIs de alta qualidade em Javascript, tudo através de uma interface gráfica do usuário, simples e direta.

Vamos instalar a aplicação localmente usando o Yarn. O Yarn é um gerenciador de pacotes que trouxe mais funcionalidades e vantagens para programadores. Com uma estrutura já conhecida e utilizada por ferramentas renomadas, como o NPM, essa aplicação tem se destacado por sua simplicidade e segurança.

install chocolatey <https://chocolatey.org/install>

```
choco install nodejs-lts
choco install yarn
```

Strapi <https://docs.strapi.io/dev-docs/installation/cli>

```
yarn create strapi-app my-folder-project --quickstart
cd my-folder-project
yarn develop
```

Se quiser adicionar uma documentação Swagger na API:

```
yarn add @strapi/plugin-documentation
```



# DOCKER

## Dockerfile

O Dockerfile nada mais é do que um meio que utilizamos para criar nossas próprias imagens. Ele serve como a receita para construir um container, permitindo definir um ambiente personalizado e próprio para meu projeto pessoal ou empresarial.

## Dockerhub

O Docker Hub é um repositório público de imagens de containers, onde diversas empresas e pessoas podem publicar imagens pré-compiladas de soluções.  
<https://hub.docker.com/>

## Comandos

```
docker image list ou ls
docker image pull nginx
docker container run nginx
docker container run -p 80:80 nginx
docker container run -d -p 80:80 nginx (executar backgroud)
docker container run --name teste -d -p 80:80 nginx
docker stop CONTAINER_ID
docker container rm CONTAINER_ID
docker image rm nginx
```

## Imagem

```
docker login
docker tag nginx:latest contadockerhub/nginx:latest
docker push contadockerhub/nginx:latest
```

<https://docs.docker.com/engine/reference/commandline/>

# DOCKERFILE

```
FROM node:18-alpine

# Installing libvips-dev for sharp Compatibility

RUN apk update && apk add --no-cache build-base gcc autoconf
    automake zlib-dev libpng-dev nasm bash vips-dev

ARG NODE_ENV=development

ENV NODE_ENV=${NODE_ENV}

WORKDIR /opt/

COPY ./package.json ./yarn.lock ./

ENV PATH /opt/node_modules/.bin:$PATH

RUN yarn config set network-timeout 600000 -g && yarn install

WORKDIR /opt/app

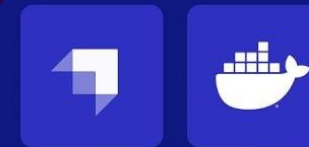
COPY ./ .

RUN yarn build

EXPOSE 1337

CMD ["yarn", "develop"]
```

```
".dockerignore"
.tmp/
.cache/
.git/
build/
node_modules/
data/
```

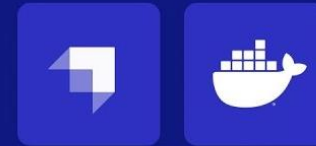


## Using Strapi v4 with Docker

# VAMOS CONSUMIR A API?

- Vamos criar um novo content-type em collection types chamado “Cadastro” com os campos Nome, email e CPF (numérico simples sem validação).
- Crie um token para autenticação da aplicação.
- Em Content Manager criaremos um cadastro pelo form do CMS.  
Se preferir e conseguir faça via curl ou postman. (não precisa ser com dados reais para evitar exposição de dados pessoais)
- Exemplo de comando para listar o cadastro:  

```
curl -X GET "http://ip:1337/api/cadastros" -H "Content-Type: application/json" -H "accept: application/json" -H "Authorization: Bearer a04b27f78d9474932283ff8b66d821fa5c51384a02d9e3077ea06893d97509f64dcb4b5ef3f269128716be28bbe5399b14510a74799ba04b455e66d7f5a55effd587bd42767ddc3b577e01c8bb9d7bebd9485f56546c06ae5ae979f8156b57486289d51232cafd13c33d4edc09dc3737165af266288247acdfa699983f1cc2e9"
```



**Using Strapi v4  
with Docker**

# BUILD IMAGE AND DEPLOY



## Docker build

Crie o Dockerfile no repositório do strapi incluindo o dockerignore.

```
docker build -t contadockerhub/strapi:latest .
```



## Docker push

Vamos enviar a imagem para o repositório

```
docker push contadockerhub/strapi:latest
```



## K8S Minikube

Cluster K8s local  
Deployments yaml file



## EKS AWS

Cluster K8s AWS  
With terraform

# DEPLOYMENTS

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: strapi
spec:
  selector:
    matchLabels:
      app: strapi
  template:
    metadata:
      labels:
        app: strapi
    spec:
      containers:
        - name: strapi
          image: rodrigordavila/strapi-k8s:latest
          ports:
            - containerPort: 1337
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: strapi
spec:
  selector:
    app: strapi
  ports:
    - port: 80
      targetPort: 1337
  type: LoadBalance
```



# MINIKUBE K8S



## Install

```
choco install minikube  
minikube start  
minikube addons enable ingress  
minikube tunnel  
minikube addons disable ingress  
minikube delete --all
```



## Deploy

```
Kubectl get all  
Kubectl get nodes  
kubectl apply -f deployments.yaml  
kubectl expose deployment strapi-k8s --port 1337
```

<https://minikube.sigs.k8s.io/docs/start/>

# OBRIGADO

[rodrigordavila@gmail.com](mailto:rodrigordavila@gmail.com)

<https://www.linkedin.com/in/rodrigordavila/>