

# **PORTFÓLIO DE PROCESSOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE**

**6ºSEMESTRE**

**DATA:** 24/03/2025

**PROFESSORES:** Romulo de Almeida Neves

**NOME DO ALUNO:** Fernando Henrique Panini

## INTRODUÇÃO

Nos últimos anos, o desenvolvimento de software passou por uma transformação significativa, impulsionada pela adoção de containers e orquestradores como o Kubernetes. Esses avanços tecnológicos não apenas revolucionaram a forma como as aplicações são construídas e implantadas, mas também trouxeram uma série de benefícios que são cruciais para o sucesso no ambiente competitivo atual. Os containers permitem que os desenvolvedores empacotem suas aplicações e todas as suas dependências de maneira leve e portátil, garantindo que funcionem de forma consistente em diferentes ambientes. Por sua vez, o Kubernetes facilita a gestão e a escalabilidade dessas aplicações, automatizando o processo de implantação, monitoramento e manutenção. Juntos, containers e Kubernetes promovem uma abordagem ágil e eficiente, permitindo que equipes de desenvolvimento entreguem software de alta qualidade de forma rápida e confiável. Essa combinação se tornou essencial para empresas que buscam inovação contínua e uma resposta rápida às demandas do mercado.

## OBJETIVO

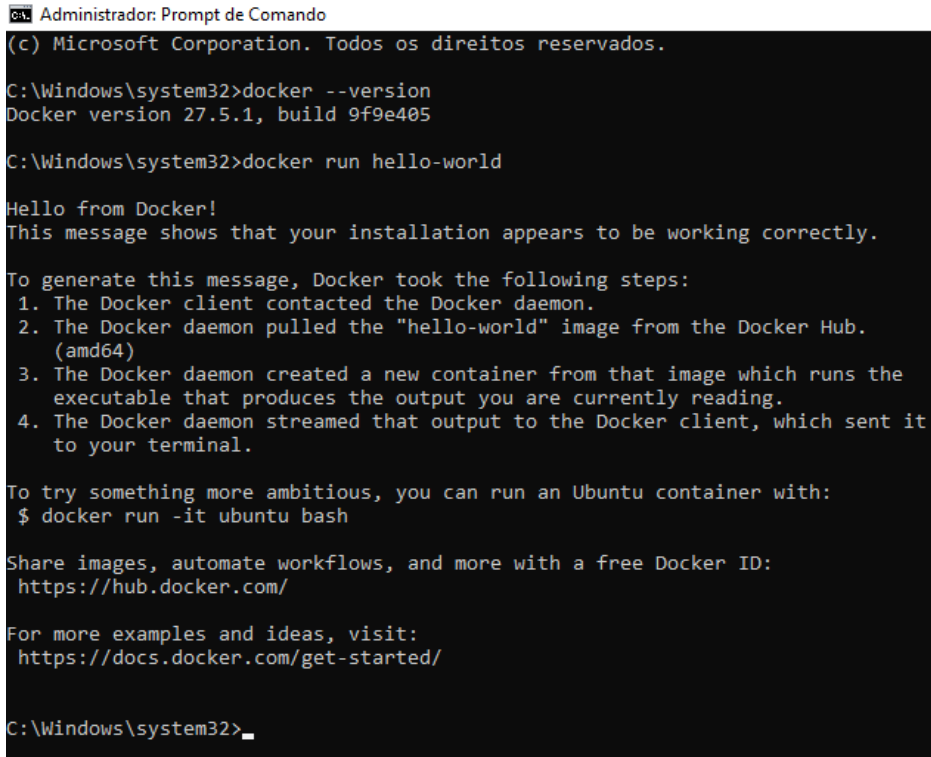
Me capacitar e aprender a utilizar as ferramentas Docker e Kubernetes, proporcionando uma compreensão prática desde a instalação até o gerenciamento de containers e clusters, com ênfase na criação e personalização de imagens Docker, armazenamento no Docker Hub e simulação de ambientes Kubernetes locais utilizando o Minikube.

## ATIVIDADE

Feito a instalação do Docker para Windows, testei através dos comandos no prompt:

```
docker --version
```

```
docker run hello-world
```

A screenshot of a Windows Command Prompt window titled "Administrador: Prompt de Comando". The window shows the output of two Docker commands. First, the command "docker --version" is entered, resulting in the output "Docker version 27.5.1, build 9f9e405". Then, the command "docker run hello-world" is entered, resulting in a multi-line output. It starts with "Hello from Docker!" and "This message shows that your installation appears to be working correctly." followed by a list of four steps explaining how Docker generated the message. It then suggests running an Ubuntu container with the command "\$ docker run -it ubuntu bash". At the bottom, it provides links to Docker Hub and Docker documentation. The prompt "C:\Windows\system32>" is visible at the bottom of the window.

```
Administrador: Prompt de Comando
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Windows\system32>docker --version
Docker version 27.5.1, build 9f9e405

C:\Windows\system32>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.
   (amd64)
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://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

C:\Windows\system32>
```

**Imagens:** Uma imagem é um arquivo que contém tudo o que é necessário para criar um container. Pense nela como uma receita de bolo. Ela inclui o sistema operacional, as bibliotecas e as dependências que o aplicativo precisa para funcionar. Quando você cria um container, está basicamente "cozinhando" a partir dessa receita.

**Containers:** Um container é uma instância em execução de uma imagem. Ele é como o bolo que você assou a partir da receita. Os containers são leves e isolados, o que significa que eles podem rodar em qualquer lugar que tenha o software necessário para gerenciá-los, sem se preocupar com o que está instalado no sistema operacional subjacente.

**Volumes:** Volumes são usados para armazenar dados persistentes que os containers podem usar. Imagine que você está fazendo um bolo e precisa guardar a cobertura na geladeira. Os volumes permitem que você armazene dados fora do container, de modo que, mesmo que o container seja destruído ou recriado, os dados ainda estejam disponíveis. Isso é especialmente útil para bancos de dados e outros aplicativos que precisam manter informações entre as execuções.

Baixando imagem Nginx e rodando no servidor:

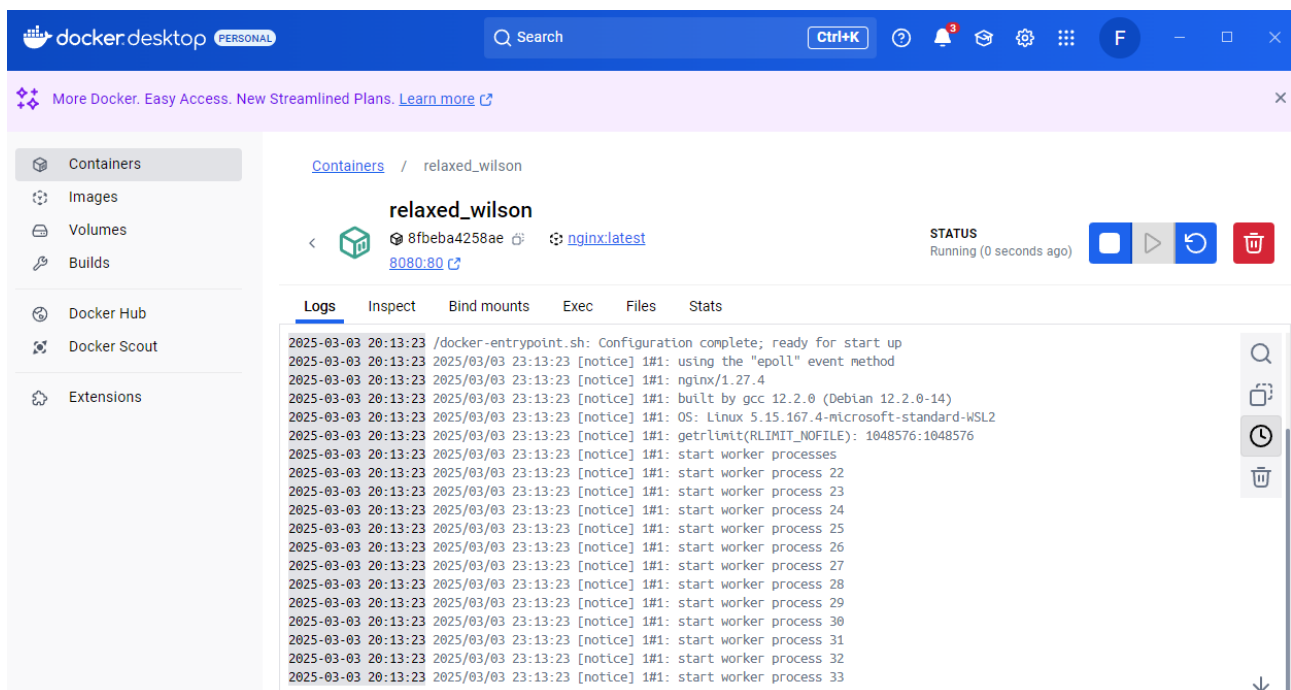
```
docker pull nginx
```

```
docker run -d -p 8080:80 nginx
```

```
Administrador: Prompt de Comando
C:\Windows\system32>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
417c4bccf534: Download complete
e7e0ca015e55: Download complete
373fe654e984: Download complete
5eaa34f5b9c2: Download complete
97f5c0f51d43: Download complete
c22eb46e871a: Download complete
6e909acdb790: Download complete
Digest: sha256:124b44bfc9ccd1f3cedf4b592d4d1e8bddb78b51ec2ed5056c52d3692baebc19
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

C:\Windows\system32>docker run -d -p 8080:80 nginx
4ecf337f5f4e525139321dc98f523ce2e6c02913405989903d501e061e1b5a05

C:\Windows\system32>
```



## Welcome to nginx!

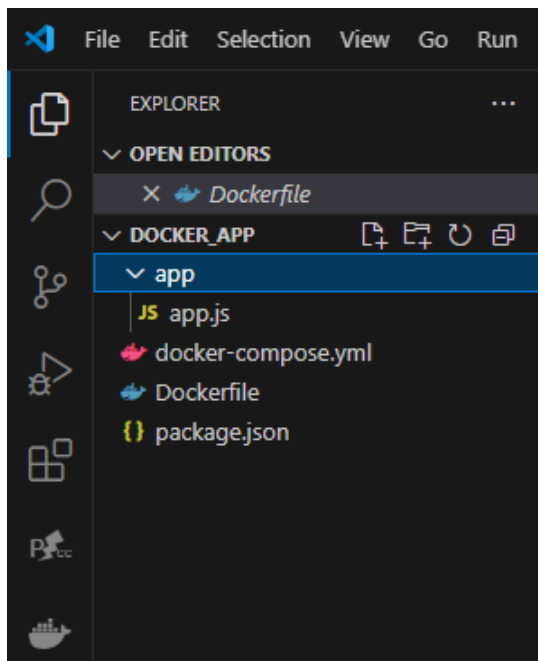
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

Thank you for using nginx.

## Criação de uma imagem personalizada

Arquivos Criados:



Arquivo app.js:

```
JS app.js
app > JS app.js > ...
1  const express = require("express");
2  const app = express();
3  const port = process.env.PORT || 3000;
4
5  app.listen(port, () => {
6    console.log(`Servidor rodando na porta ${port}`);
7  });
8
9  app.get("/", (req, res, next) => {
10    res.json(
11      `Seja bem vindo!!`
12    );
13  });
```

Arquivo docker-compose.yml:

```
docker-compose.yml X
docker-compose.yml
1  version: "3"
   Run All Services
2  services:
   Run Service
3    node:
4      build: .
5      command: "npm run start"
6      working_dir: /home/node/app
7      environment:
8        - NODE_ENV=production
9      expose:
10       - "3000"
11     ports:
12       - "3000:3000"
13
```

Arquivo Dockerfile:

```
Dockerfile X
Dockerfile > ...
1  FROM node:18-alpine
2  RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app
3  WORKDIR /home/node/app
4  COPY package*.json ./
5  RUN npm install
6  COPY --chown=node:node . .
7  EXPOSE 3000
8  CMD [ "node", "app.js" ]
```

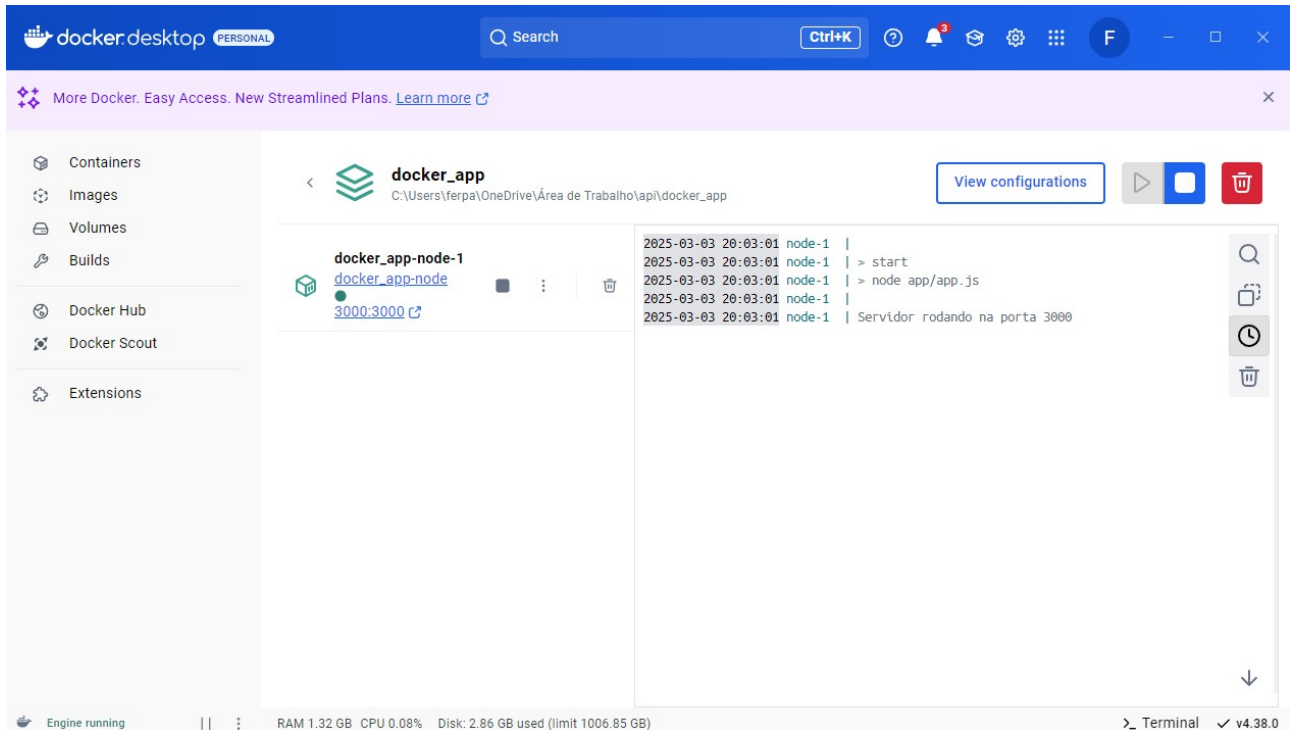
Arquivo package.json:

```
package.json X
package.json > ...
1  {
2    "name": "api",
3    "description": "API para mostrar o uso do Docker",
4    "main": "app/app.js",
   Debug
5    "scripts": {
6      "start": "node app/app.js"
7    },
8    "dependencies": {
9      "express": "^4.18.1"
10   }
11 }
```

Comando para construir a Imagem Criada e Rodar no Servidor:

```
docker buildx build -t app_docker .
```

```
docker run -d -p 3000:3000 app_docker
```





## Publicação da Imagem no Docker Hub:

```
docker tag app_docker ferpan85/app_docker:1.0.0
```

```
docker push ferpan85/app_docker:1.0.0
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POLYGLOT NOTEBOOK


Run 'docker image COMMAND --help' for more information on a command.
PS C:\Users\ferpa\OneDrive\Área de Trabalho\api\docker_app> docker image ls
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
docker_app-node     latest       1f872602cccf   4 hours ago   196MB
ferpan85/docker-node_app  1.0.0       d3a1ca20a028   4 hours ago   196MB
api_docker-node     latest       d3a1ca20a028   4 hours ago   196MB
api-nodejs-docker-main-node latest       b1037f423878   4 hours ago   196MB
ferpan85/app_docker 1.0.0       b2b9520f820c   4 hours ago   196MB
app_docker          latest       b2b9520f820c   4 hours ago   196MB
my-node-app         latest       f57ec5937787   5 hours ago   1.35GB
nginx               latest       9d6b58feebd2   3 weeks ago   279MB
hello-world         latest       bfb0cc14f13    5 weeks ago   20.4kB
PS C:\Users\ferpa\OneDrive\Área de Trabalho\api\docker_app> docker push ferpan85/app_docker:1.0.0
The push refers to repository [docker.io/ferpan85/app_docker]
3d8c59f7308d: Pushed
a5cf150cb374: Pushed
f46e519824fb: Pushed
61ad8a9fa7fe: Pushed
f18232174bc9: Pushed
e495e1787dd7: Pushed
4f4fb700ef54: Pushed
272844528fb6: Pushed
8c4ac176bab5: Pushed
cbb7771b4159: Pushed
1.0.0: digest: sha256:b2b9520f820c0d0a15bc0d82eb3986dfc7c5e84d9967abe3ac971e04294750ec size: 856
PS C:\Users\ferpa\OneDrive\Área de Trabalho\api\docker_app> 
```

hub.docker.com/repository/docker/ferpan85/app\_docker/tags/1.0.0/sha256-b394803088f4db3ec9be528229545345b13a616e90ee2bc1f6a115573ee1b7f8?tab=layers

New Introducing our new CEO Don Johnson - Read More →

dockerhub Explore Repositories Organizations Usage Search Docker Hub

ferpan85 / Repositories / app\_docker / Tags / 1.0.0



**ferpan85/app\_docker:1.0.0**

Delete Tag

INDEX DIGEST sha256:b2b9520f820c0d0a15bc0d82eb3986dfc7c5e84d9967abe3ac971e04294750ec

OS/ARCH linux/amd64

COMPRESSED SIZE 45.06 MB

LAST PUSHED 5 minutes by ferpan85

TYPE Image

MANIFEST DIGEST sha256:b394803088f4db3ec9be528229545345b13a616e90ee2bc1f6a115573ee1b7f8

Image Layers Vulnerabilities

Image Layers

1	ADD alpine-minirootfs-3.21.3-x86_64.tar.gz / # buildkit	3.47 MB
2	CMD ["/bin/sh"]	0 B
3	ENV NODE_VERSION=18.20.7	0 B
4	RUN /bin/sh -c addgroup -g	38.16 MB
5	ENV YARN_VERSION=1.22.22	0 B
6	RUN /bin/sh -c apk add	1.2 MB
7	COPY docker-entrypoint.sh /usr/local/bin/ # buildkit	444 B
8	ENTRYPOINT ["docker-entrypoint.sh"]	0 B
9	CMD ["node"]	0 B
10	RUN /bin/sh -c mkdir -p	167 B

Command

```
ADD alpine-minirootfs-3.21.3-x86_64.tar.gz / # buildkit
```

## Minikube e Kubectl:

Prints:

```
C:\Windows\system32>minikube start --driver=docker
* minikube v1.35.0 on Microsoft Windows 10 Pro 10.0.19045.5487 Build 19045.5487
* Using the docker driver based on user configuration
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.46 ...
* Creating docker container (CPUs=2, Memory=3900MB) ...
* Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Windows\system32>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

C:\Windows\system32>kubectl get nodes
NAME        STATUS    ROLES    AGE   VERSION
minikube    Ready     control-plane   25m   v1.32.0

C:\Windows\system32>kubectl run meu-nginx --image=nginx --port=80
pod/meu-nginx created

C:\Windows\system32>kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
meu-nginx   1/1     Running   0           107s

C:\Windows\system32>kubectl create deployment web --image=nginx
deployment.apps/web created


C:\Windows\system32>kubectl scale deployment web --replicas=3
deployment.apps/web scaled

C:\Windows\system32>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
meu-nginx                           1/1     Running   0           8m49s
web-65d846d465-4sp5j                 1/1     Running   0           4m47s
web-65d846d465-bd566                 1/1     Running   0           66s
web-65d846d465-kvg4b                 1/1     Running   0           66s

C:\Windows\system32>kubectl logs -f meu-nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/03/04 23:04:05 [notice] 1#1: using the "epoll" event method
2025/03/04 23:04:05 [notice] 1#1: nginx/1.27.4
2025/03/04 23:04:05 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/03/04 23:04:05 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/03/04 23:04:05 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/03/04 23:04:05 [notice] 1#1: start worker processes
2025/03/04 23:04:05 [notice] 1#1: start worker process 29
2025/03/04 23:04:05 [notice] 1#1: start worker process 30
2025/03/04 23:04:05 [notice] 1#1: start worker process 31
2025/03/04 23:04:05 [notice] 1#1: start worker process 32
2025/03/04 23:04:05 [notice] 1#1: start worker process 33
2025/03/04 23:04:05 [notice] 1#1: start worker process 34
2025/03/04 23:04:05 [notice] 1#1: start worker process 35
2025/03/04 23:04:05 [notice] 1#1: start worker process 36
2025/03/04 23:04:05 [notice] 1#1: start worker process 37
2025/03/04 23:04:05 [notice] 1#1: start worker process 38
2025/03/04 23:04:05 [notice] 1#1: start worker process 39
2025/03/04 23:04:05 [notice] 1#1: start worker process 40
```

```
C:\Windows\system32>kubectl describe pod meu-nginx
Name:          meu-nginx
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Tue, 04 Mar 2025 20:38:40 -0300
Labels:        run=meu-nginx
Annotations:   <none>
Status:        Running
IP:            10.244.0.3
IPs:
  IP: 10.244.0.3
Containers:
  meu-nginx:
    Container ID:  docker://a18ba2f685d6dc58f02858e2e0adf5f6b1b20983f7af22410c8b9cf51468d62
    Image:         nginx
    Image ID:      docker-pullable://nginx@sha256:9d6b58feebd2dbd3c56ab585333d627cc6e281011cfd6050fa4bcf2072c9496
    Port:         80/TCP
    Host Port:    0/TCP
    State:        Running
      Started:    Tue, 04 Mar 2025 20:38:51 -0300
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6d4fk (ro)
Conditions:
  Type              Status
  PodReadyToStartContainers  True
  Initialized         True
  Ready               True
  ContainersReady      True
  PodScheduled        True
Volumes:
  kube-api-access-6d4fk:
    Type:              Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:      kube-root-ca.crt
    ConfigMapOptional:  <nil>
    DownwardAPI:        true
QoS Class:           BestEffort
Node-Selectors:      <none>
Tolerations:         node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                     node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason      Age   From              Message
  ----    -
  Normal  Scheduled   110s  default-scheduler  Successfully assigned default/meu-nginx to minikube
  Normal  Pulling     110s  kubelet            Pulling image "nginx"
  Normal  Pulled      102s  kubelet            Successfully pulled image "nginx" in 7.475s (7.475s including waiting). Image size: 191998640 bytes.
  Normal  Created     99s   kubelet            Created container: meu-nginx
  Normal  Started     99s   kubelet            Started container meu-nginx
```

## Arquivo deployment.yaml:

 \*deployment.yaml - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: meu-nginx
  name: meu-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: meu-nginx
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: meu-nginx
    spec:
      containers:
      - image: k8s-minikube
        name: k8s-minikube
        ports:
        - containerPort: 80
        resources: {}
status: {}
```

## CONCLUSÃO

A experiência prática, desde a instalação até o gerenciamento de containers e clusters, foi fundamental para aprofundar meu entendimento. Aprendi a criar e personalizar imagens Docker, além de armazená-las no Docker Hub, o que ampliou minha habilidade em gerenciar aplicações de forma eficiente. A simulação de ambientes Kubernetes locais com o Minikube também foi uma etapa valiosa, permitindo-me experimentar e aplicar conceitos em um ambiente controlado. Essa jornada não apenas fortaleceu meu conhecimento técnico, mas também me preparou para enfrentar desafios reais no desenvolvimento e na operação de aplicações em ambientes de contêineres.