



PUC Minas
Virtual

Introdução ao Docker

Renan Santos Mendes



PUC Minas
Virtual

Introdução

Introdução ao Docker

- Os containers Docker são **unidades isoladas de software** que empacotam todas as dependências e bibliotecas necessárias para executar um aplicativo.
- Fornecem uma maneira **consistente e portátil** de implantar aplicativos em **diferentes ambientes**.
- Permitem isolamento de aplicativos, portabilidade, eficiência de recursos, escalabilidade e facilidade de implantação.



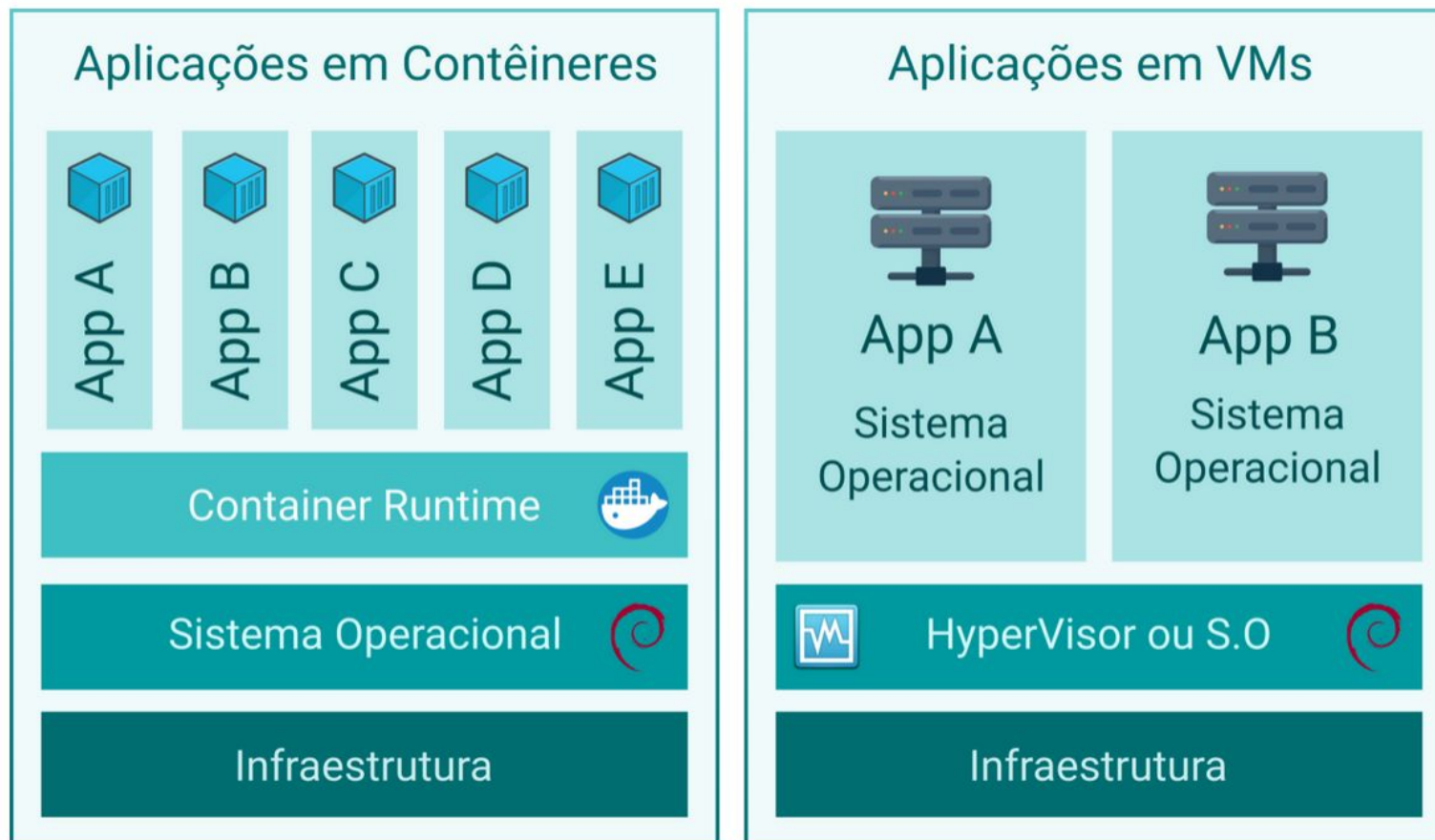
PUC Minas
Virtual

Fundamentos

Fundamentos

- Uma imagem Docker é uma **representação estática de um aplicativo e seu ambiente de execução.**
- Ela contém **todas as dependências e configurações necessárias para executar o aplicativo.**
- As **imagens** são criadas a partir do **Dockerfile.**
- Ao contrário das VMs tradicionais os **containers compartilham o kernel do sistema host.** Isso torna os containers mais leves, eficientes e rápidos de iniciar.

Fundamentos





PUC Minas
Virtual

Imagens e Containers

Imagens e Containers

- As imagens Docker são pacotes autossuficientes que **contêm tudo o que é necessário para executar um aplicativo**.
- As imagens Docker podem ser armazenadas em **repositórios locais** ou em **repositórios remotos**, como o [Docker Hub](#).
- O [Docker Hub](#) é o **registro público oficial** do Docker, onde os desenvolvedores podem encontrar uma ampla variedade de imagens prontas para uso.

Imagens e Containers

- Alguns dos comandos mais comuns incluem:
 - *docker start* para iniciar um contêiner,
 - *docker stop* para pará-lo,
 - *docker restart* para reiniciá-lo e
 - *docker rm* para removê-lo.
- **Interação com contêineres em execução:** inspecionar o estado do aplicativo e até mesmo acessar uma sessão de terminal interativa dentro do contêiner.



PUC Minas
Virtual

Imagens Customizadas

Imagens e Containers

Um arquivo Dockerfile básico tem a seguinte aparência:

```
FROM ubuntu:latest
WORKDIR /app
COPY . /app
RUN apt-get update && apt-get install -y python3
ENV PYTHONPATH=/app
EXPOSE 8000

CMD ["python3", "app.py"]
```




PUC Minas
Virtual

Explicando cada comando

Imagens e Containers

Um arquivo Dockerfile básico tem a seguinte aparência:

Imagem
Base



```
FROM ubuntu:latest
WORKDIR /app
COPY . /app
RUN apt-get update && apt-get install -y python3
ENV PYTHONPATH=/app
EXPOSE 8000

CMD ["python3", "app.py"]
```

Imagens e Containers

Um arquivo Dockerfile básico tem a seguinte aparência:

Diretório
de
trabalho



```
FROM ubuntu:latest
WORKDIR /app
COPY . /app
RUN apt-get update && apt-get install -y python3
ENV PYTHONPATH=/app
EXPOSE 8000

CMD ["python3", "app.py"]
```

Imagens e Containers

Um arquivo Dockerfile básico tem a seguinte aparência:

Cópia dos
arquivos
para o
container

```
FROM ubuntu:latest
WORKDIR /app
COPY . /app
RUN apt-get update && apt-get install -y python3
ENV PYTHONPATH=/app
EXPOSE 8000

CMD ["python3", "app.py"]
```


Imagens e Containers

Um arquivo Dockerfile básico tem a seguinte aparência:

```
FROM ubuntu:latest
WORKDIR /app
COPY . /app
RUN apt-get update && apt-get install -y python3
ENV PYTHONPATH=/app
EXPOSE 8000

CMD ["python3", "app.py"]
```

Instalação de
dependências e
configuração
do ambiente
para aplicativo



Imagens e Containers

Um arquivo Dockerfile básico tem a seguinte aparência:

```
FROM ubuntu:latest
WORKDIR /app
COPY . /app
RUN apt-get update && apt-get install -y python3
ENV PYTHONPATH=/app
EXPOSE 8000
CMD ["python3", "app.py"]
```

Comando de
execução do
aplicativo



Criando e compartilhando uma imagem

Criando e compartilhando uma imagem

- Para criar uma imagem localmente: **docker build nome-da-imagem .**
- Para publicar uma imagem:
 1. Criar uma conta no [docker hub](https://hub.docker.com/)
 2. Dar uma tag para a imagem: **docker tag nome-da-imagem usuario/nome-da-imagem**
 3. Fazer o login: **docker login**
 4. Publicar a imagem: **docker push usuario/nome-da-imagem**



PUC Minas
Virtual