

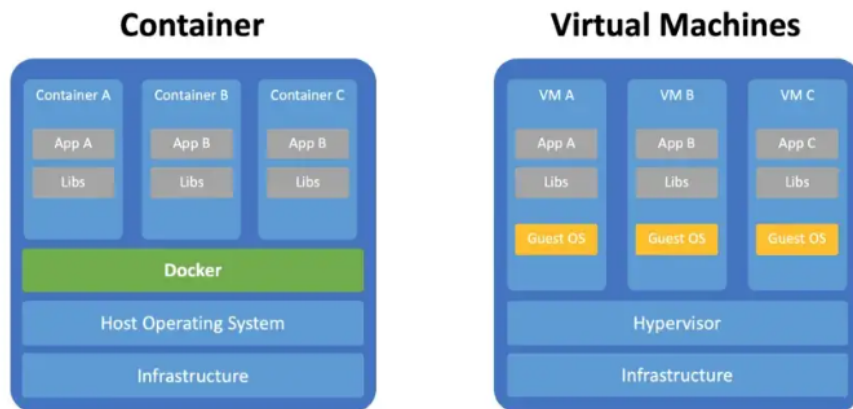
# Automação de Processos com Docker e Airflow

Prof. Dr. Jesmmer Alves\*

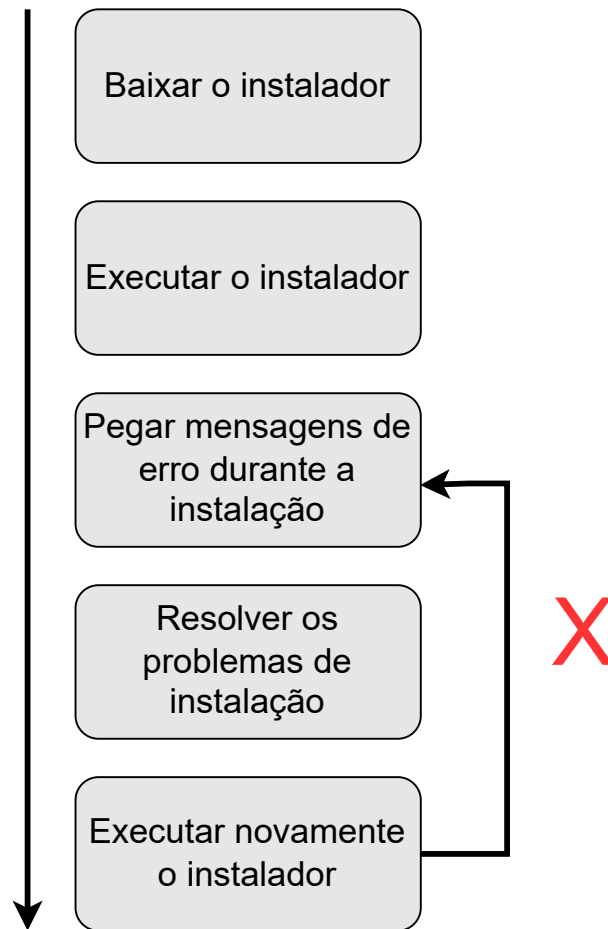
Instituto Federal Goiano Campus Morrinhos  
Curso Bacharelado em Ciência da Computação

# Máquinas Virtuais e Virtualização através de Containers

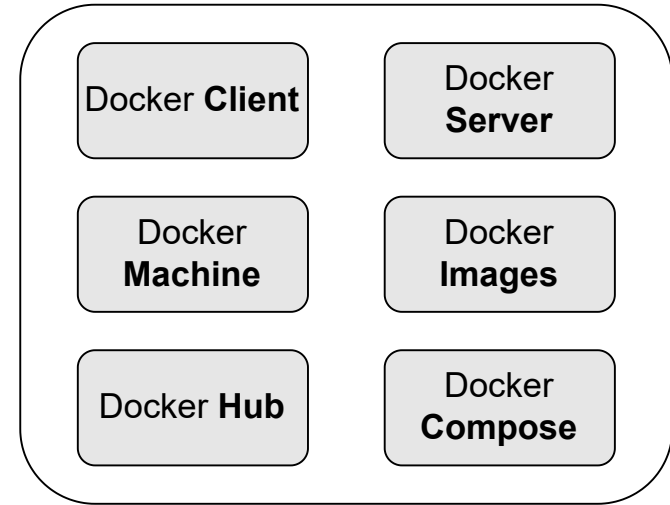
- Container e VM trabalham com virtualização e isolamento de ambientes para promover processamentos independentemente de aplicações
- Virtualização com máquinas virtuais se dá no nível do sistema operacional (hypervisor);
- Containers utilizam os recursos do sistema e processos de kernel para criar os ambientes;
- Uma das principais vantagens do Container é a possibilidade de criar serviços e códigos independentes, que podem ser movidos sem dificuldade entre máquinas e ambientes diferentes sem a perda de dados.



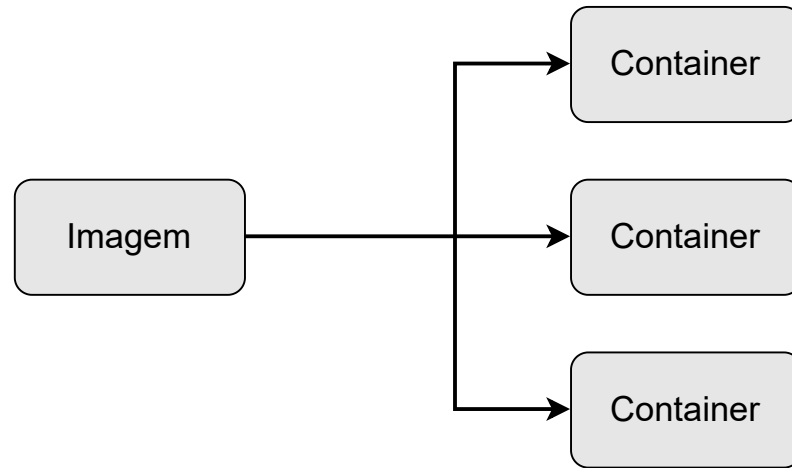
Fácil instalação e manutenção



- **Site oficial**
  - [www.docker.com](http://www.docker.com)
- **Documentação**
  - [docs.docker.com](http://docs.docker.com)
- **Biblioteca de imagens**
  - [hub.docker.com](http://hub.docker.com)



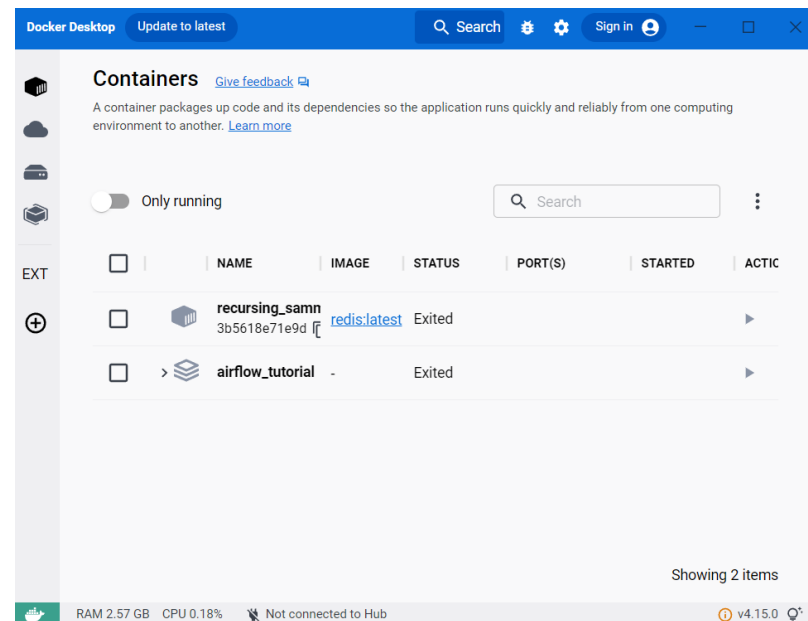
- **Docker** é uma plataforma consistindo de muitas ferramentas que podem ser usadas para simplificar e padronizar o processo de desenvolvimento de software através da utilização de containers.



- **Imagem** é um arquivo com todas as dependências e configurações necessárias para executar um programa;
- **Container** é uma instância de uma imagem.

# Instalação e Teste

1. Acesse docker.com
2. Acesse Get Started
3. Clique em Docker Desktop for Windows
4. Ao instalar o aplicativo, certifique de escolher a opção **usar containers Linux**
5. Acesse a IDE Docker



6. Acesse o visual studio e verifique a instalação

```
docker version
```

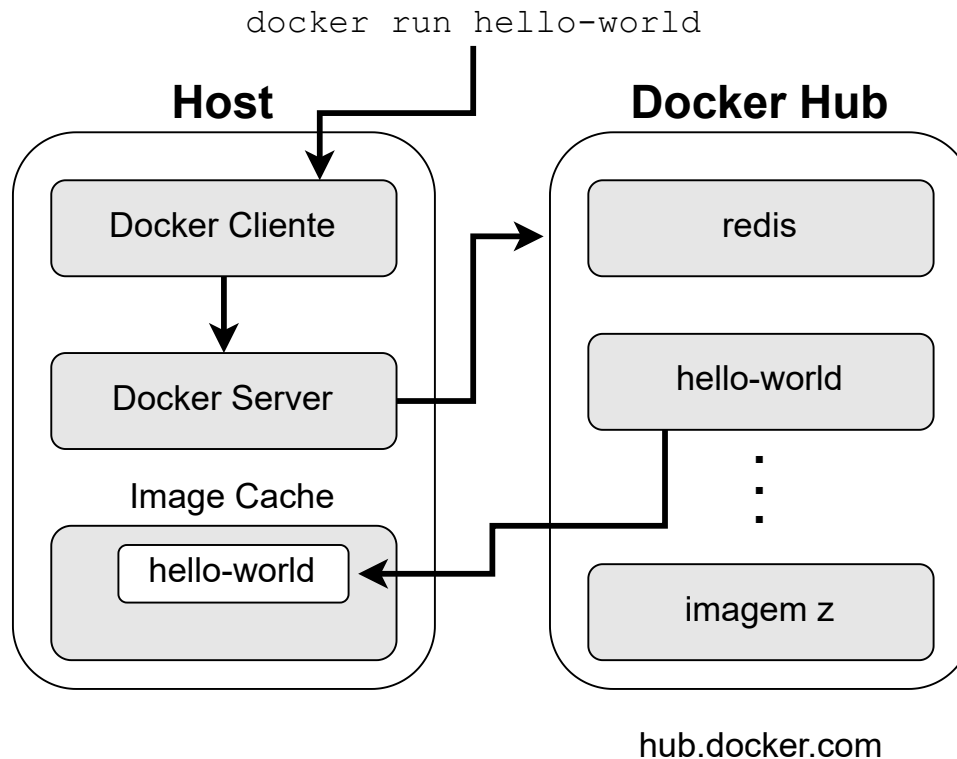
7. Docker hello world

```
docker run hello-world
```

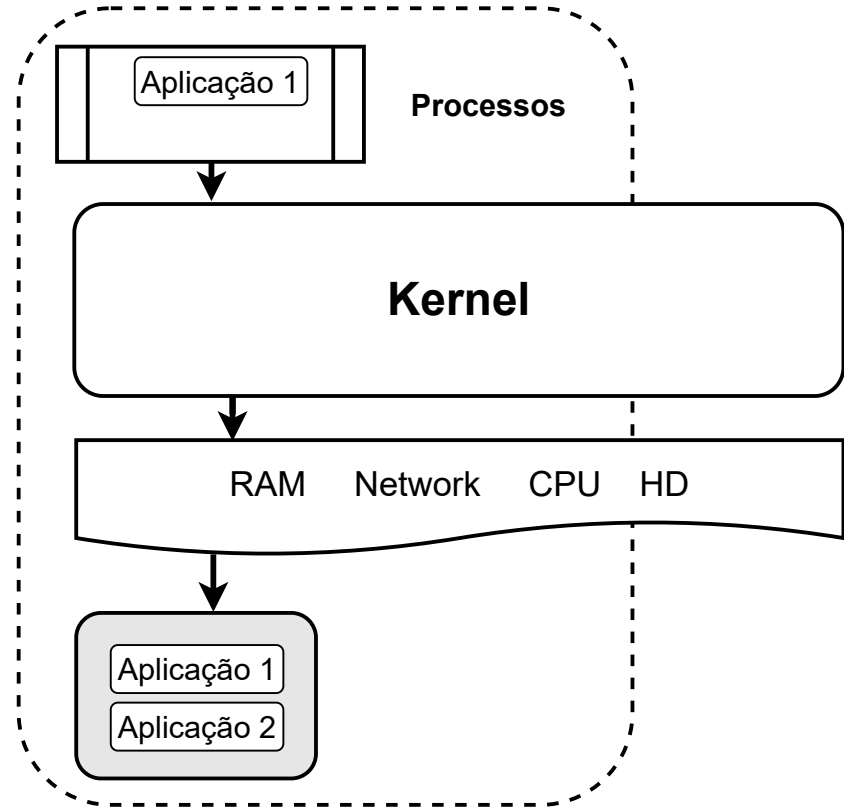
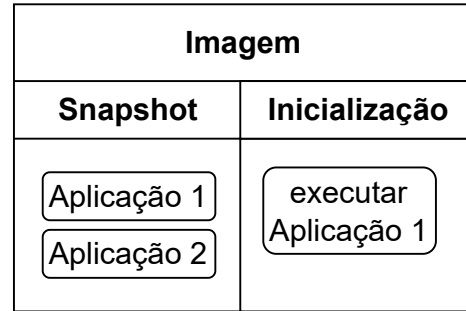
```
PS D:\Optativa BCC> docker version
Built:           Tue Oct 25 18:08:16 2022
OS/Arch:         windows/amd64
Context:         default
Experimental:    true

Server: Docker Desktop 4.15.0 (93002)
Engine:
  Version:       20.10.21
  API version:   1.41 (minimum version 1.12)
```

# Cache e Download de Imagens



# Imagem → Container



- **Namespacing**: isola recursos por processos ou grupo de processos.
- **Control Groups (cgroups)**: limita o montante de recursos usados nos processos.


# Comandos Básicos

## Executar um comando em um container

- docker run imagem comando

```
docker run busybox echo Olá Queridos!
```

```
docker run busybox ls
```



```
docker run -it busybox sh  
touch AulaDocker  
ls
```

verifique que os containers possuem diferentes sistemas de arquivos (use dois terminais)

- docker run = docker create + docker start

```
docker create hello-world
```

```
docker start -a id-container
```

```
PS D:\Optativa BCC> docker run busybox echo Olá Queridos!  
Olá Queridos!  
PS D:\Optativa BCC> docker run busybox ls  
bin  
dev  
etc  
home  
lib  
lib64  
proc  
root  
sys  
tmp  
usr  
var
```

```
PS D:\Optativa BCC> docker create hello-world  
0defe7a75d1fbccdef3bcd5a752d34997f51b8072b3d49  
PS D:\Optativa BCC> docker start -a 0defe7a75d1fbccdef3bcd5a752d34997f51b8072b3d49  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:
```



## Listar e remover containers

- containers rodando na máquina

```
docker ps
```

- containers criados na máquina

```
docker ps --all
```

- containers com status Exited podem ser iniciados com o start

```
docker start id_container
```

- Parar um container

```
docker stop id_container ou docker kill id_container
```

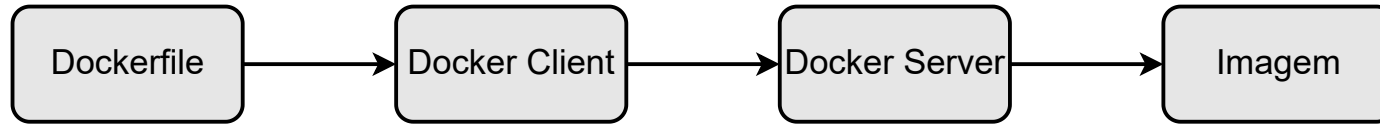
- Remover todos os containers parados

```
docker system prune -a
```

```
D:\Optativa BCC> docker ps
CONTAINER ID   IMAGE          COMMAND
$ NAMES
ad18e9b5a     busybox       "sh"
great_spence
2afb01903f    redis:latest  "docker-entry
tcp         airflow_tutorial-redis-1
D:\Optativa BCC> docker system prune -a
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one c
- all images without at least one contain
- all build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
fe7a75d1fbccdef3bcd5a752d34997f51b8072b3
cac8964cf348cd1c1c4905b751bb8be38abdebe9
e6de66f021b6a94f24b595d875af77edae83e834
e682bfbf2a114f6ffa54114bcb03f9289231c43e2
65f64bee1ccbcf7701077b4284445bd0e6a787c0
0a1e82b866dbcaeff98be8d813bfe50aaa1eae26
```

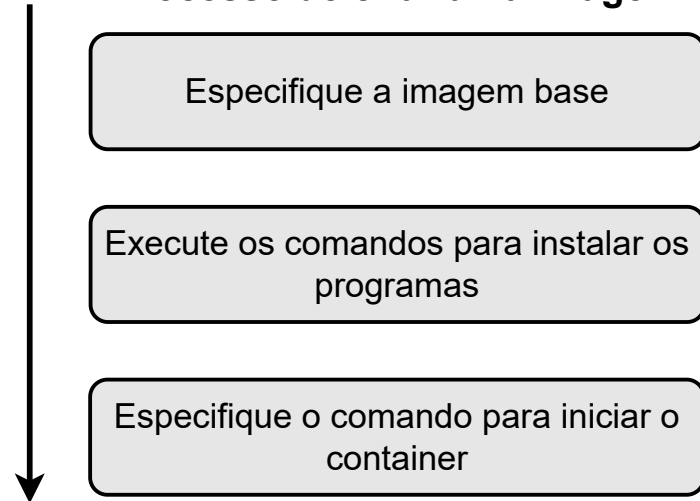
# Criando Imagens Docker



## Dockerfile:

Arquivo com as definições e comandos para montar uma imagem.

## Processo de criar uma imagem



# Exemplo - Imagem com o servidor redis

- Crie a pasta exemplo\_redis
- Abra no visual studio
- Crie o arquivo **Dockerfile**
- No terminal:
  - `docker build .`
  - `docker run id_imagem`
- Execute `docker ps` em outro terminal

## Dockerfile

```
1  # defina imagem base
2  FROM alpine
3
4  # baixe e instale as dependências
5  RUN apk add --update redis
6  RUN apk add --update gcc
7
8  # descreva o que deve ser feito ao iniciar o container
9  CMD ["redis-server"]
```

- **Tagging:** mudar nome e versão
  - `docker build -t jsa/redis: latest .`
  - `docker ps`
- Mudar nome do container
  - `docker ps`
  - `docker rename id_container novo_nome`
  - `docker ps`

## Exemplo 2 - Projeto Node js

- No visual studio, crie a pasta **Projeto Node** com os seguintes arquivos:

### package.json

```
{
  "dependencies": {
    "express": "*"
  },
  "scripts": {
    "start": "node index.js"
  }
}
```

### index.js

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Oieeee! tudo bem?');
});

app.listen(8080, () => {
  console.log('Escutando a porta 8080');
});
```

### Dockerfile

```
# Especificar a imagem base
FROM alpine
#FROM node:alpine

WORKDIR /usr/app
# Instalar as dependências

COPY ./package.json ./
RUN npm install
COPY ./ ./
# Comando padrão
CMD ["npm", "start"]
```

- Compile o arquivo
  - `docker build -t jsa/projeto_node .`
- Execute a imagem com mapeamento de porta
  - `docker run -p 5000:8080 jsa/projeto_node`
- Verifique se funcionou acessando `localhost:5000`
- Liste os arquivos no container
  - `docker ps`
  - `docker exec -it id_imagem sh`
  - `ls`

Apresente uma descrição detalhada do exemplo **Projeto Node**. Esta descrição deve incluir:

- Objetivo do projeto;
- Descrição detalhada dos arquivos com a especificação de cada instrução/comando;
- Descrição detalhada dos comandos (compilar, executar) com a especificação de todos os atributos.

obs.:

- Utilize o conteúdo de [docs.docker.com](https://docs.docker.com) e [hub.docker.com](https://hub.docker.com) para detalhes sobre os comandos e imagens.
- Enviar arquivo com as respostas pelo Moodle até dia 07/02.