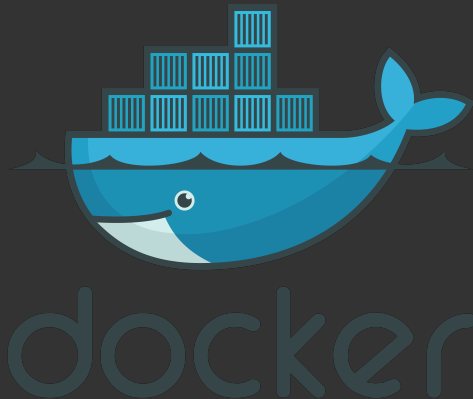


```
>>> >>> Docker
```

```
Name: Henrique Yara (opus-software)
```

```
Date: January 13, 2023
```



**Figure:** Docker logo

>>> Índice

1. Introdução
2. Desenvolvimento e Execução
3. Entrega
4. Arquitetura
5. Docker Objects
6. Volumes
7. Redes
8. Pratico
9. Segurança
10. Maquina host
11. Docker Daemon
12. Links

>>> O que é docker?

Gerenciar a infra-estrutura:

- \* Ambientes "isolados" usando container;

>>> O que é docker?

Gerenciar a infra-estrutura:

- \* Ambientes "isolados" usando container;
- \* Agilizar as entregas das aplicações;

>>> O que é docker?

Gerenciar a infra-estrutura:

- \* Ambientes "isolados" usando container;
- \* Agilizar as entregas das aplicações;
- \* Aplicações seguras;

>>> O que é docker?

Gerenciar a infra-estrutura:

- \* Ambientes "isolados" usando container;
- \* Agilizar as entregas das aplicações;
- \* Aplicações seguras;
- \* Portabilidade;

>>> O que é um container?

- ★ Containers:

- ★ Usa o *kernel* da máquina *host*;

>>> O que é um container?

- ★ Containers:

- ★ Usa o *kernel* da máquina *host*;

- ★ Contém:

- ★ *Softwares*;



>>> O que é um container?

- ★ Containers:

- ★ Usa o *kernel* da máquina *host*;

- ★ Contém:

- ★ *Softwares*;

- ★ *Bibliotecas*;

>>> O que é um container?

★ Containers:

- ★ Usa o *kernel* da máquina *host*;
- ★ Contém:
  - ★ *Softwares*;
  - ★ *Bibliotecas*;
- ★ Leves (Somente o necessário);

>>> O que é um container?

★ Containers:

- ★ Usa o *kernel* da máquina *host*;
- ★ Contém:
  - ★ *Softwares*;
  - ★ *Bibliotecas*;
- ★ Leves (Somente o necessário);
- ★ Execução quase igual à execução nativa;

## >>> Características do container

- ★ Portabilidade:
  - ★ Fácil compartilhar;

## >>> Características do container

- ★ Portabilidade:
  - ★ Fácil compartilhar;
  - ★ Ambiente igual para todos;

## >>> Características do container

- ★ Portabilidade:

- ★ Fácil compartilhar;
- ★ Ambiente igual para todos;
- ★ A mesma execução para todos;

## >>> Características do container

- ★ Portabilidade:
  - ★ Fácil compartilhar;
  - ★ Ambiente igual para todos;
  - ★ A mesma execução para todos;
- ★ Alternativa viável para as máquinas virtuais (hypervisor):
  - ★ Melhor uso da capacidade do seu computador;

## >>> Características do container

- ★ Portabilidade:
  - ★ Fácil compartilhar;
  - ★ Ambiente igual para todos;
  - ★ A mesma execução para todos;
- ★ Alternativa viável para as máquinas virtuais (hypervisor):
  - ★ Melhor uso da capacidade do seu computador;
  - ★ Ambientes grandes e densos;



## >>> Características do container

- ★ Portabilidade:
  - ★ Fácil compartilhar;
  - ★ Ambiente igual para todos;
  - ★ A mesma execução para todos;
- ★ Alternativa viável para as máquinas virtuais (hypervisor):
  - ★ Melhor uso da capacidade do seu computador;
  - ★ Ambientes grandes e densos;
  - ★ Ambientes pequenos ou médios;

## >>> Máquina virtual X Container

### \* Máquina Virtual

- \* Virtualiza a camada de aplicação e a camada do SO;

### \* Container

## >>> Máquina virtual X Container

### ★ Máquina Virtual

- ★ Virtualiza a camada de aplicação e a camada do SO;

### ★ Container

- ★ Virtualiza a camada de aplicação;

## >>> Máquina virtual X Container

### ★ Máquina Virtual

- ★ Virtualiza a camada de aplicação e a camada do SO;
- ★ Mais pesado (Na casa dos Gigabytes);

### ★ Container

- ★ Virtualiza a camada de aplicação;

## >>> Máquina virtual X Container

### ★ Máquina Virtual

- ★ Virtualiza a camada de aplicação e a camada do SO;
- ★ Mais pesado (Na casa dos Gigabytes);

### ★ Container

- ★ Virtualiza a camada de aplicação;
- ★ Mais leve (Na casa dos megabytes);

## >>> Máquina virtual X Container

### ★ Máquina Virtual

- ★ Virtualiza a camada de aplicação e a camada do SO;
- ★ Mais pesado (Na casa dos Gigabytes);
- ★ Demoram mais para inicializar;

### ★ Container

- ★ Virtualiza a camada de aplicação;
- ★ Mais leve (Na casa dos megabytes);

## >>> Máquina virtual X Container

### ★ Máquina Virtual

- ★ Virtualiza a camada de aplicação e a camada do SO;
- ★ Mais pesado (Na casa dos Gigabytes);
- ★ Demoram mais para inicializar;

### ★ Container

- ★ Virtualiza a camada de aplicação;
- ★ Mais leve (Na casa dos megabytes);
- ★ Inicializam rapidamente;

## >>> Máquina virtual X Container

### ★ Máquina Virtual

- ★ Virtualiza a camada de aplicação e a camada do SO;
- ★ Mais pesado (Na casa dos Gigabytes);
- ★ Demoram mais para inicializar;
- ★ Executa qualquer sistema operacional em qualquer Sistema Operacional;

### ★ Container

- ★ Virtualiza a camada de aplicação;
- ★ Mais leve (Na casa dos megabytes);
- ★ Inicializam rapidamente;



## >>> Máquina virtual X Container

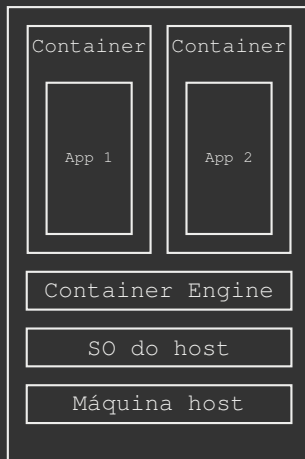
### ★ Máquina Virtual

- ★ Virtualiza a camada de aplicação e a camada do SO;
- ★ Mais pesado (Na casa dos Gigabytes);
- ★ Demoram mais para inicializar;
- ★ Executa qualquer sistema operacional em qualquer Sistema Operacional;

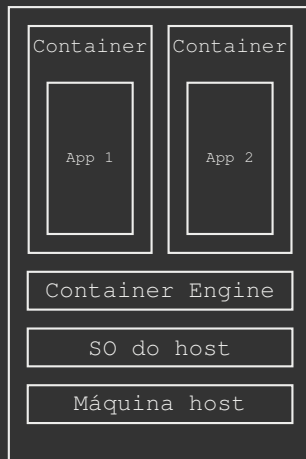
### ★ Container

- ★ Virtualiza a camada de aplicação;
- ★ Mais leve (Na casa dos megabytes);
- ★ Inicializam rapidamente;
- ★ O sistema operacional precisa ser compatível com a máquina principal ou usar uma camada de compatibilidade;

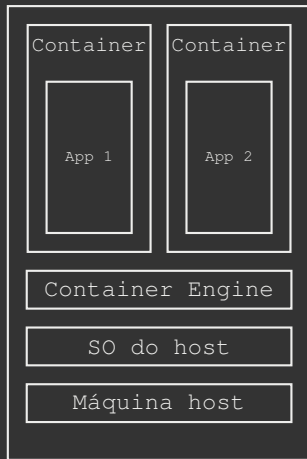
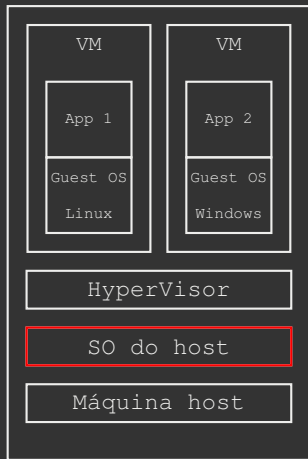
## >>> Máquina virtual X Container (Imagens)



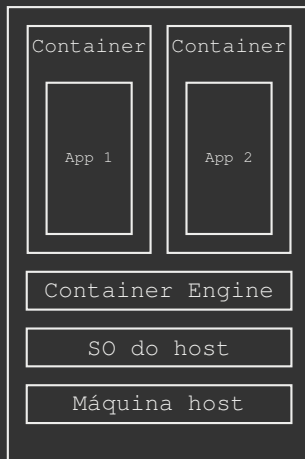
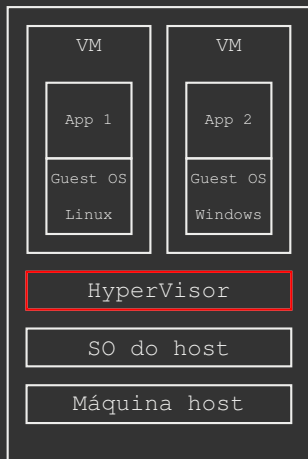
## >>> Máquina virtual X Container (Imagens)



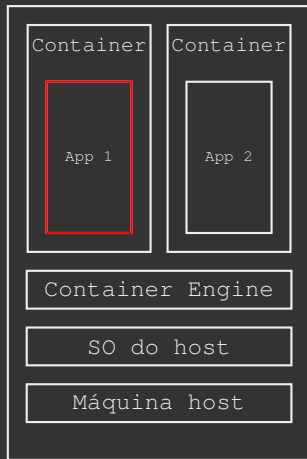
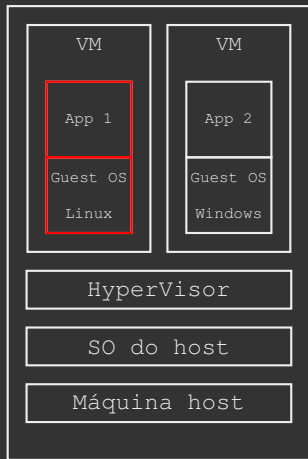
## >>> Máquina virtual X Container (Imagens)



## >>> Máquina virtual X Container (Imagens)



## >>> Máquina virtual X Container (Imagens)



>>> Onde os containers podem ser encontrados?

- \* Repositórios privados:

- \* Repostórios das empresas;

>>> Onde os containers podem ser encontrados?

- \* Repositórios privados:
  - \* Repositórios das empresas;
- \* Repositórios públicos:
  - \* Docker hub;



## >>> Desenvolvimento e Execução - Objetivos

- ★ Ambientes "isolados":

- ★ Não é necessário instalar softwares dependentes na máquina principal;

## >>> Desenvolvimento e Execução - Objetivos

★ Ambientes "isolados":

- ★ Não é necessário instalar softwares dependentes na máquina principal;
- ★ Evita conflitos de versão;

## >>> Desenvolvimento e Execução - Objetivos

### ★ Ambientes "isolados":

- ★ Não é necessário instalar softwares dependentes na máquina principal;
- ★ Evita conflitos de versão;
- ★ Abstrair a parte da infra-estrutura aplicação;

## >>> Desenvolvimento e Execução - Objetivos

### ★ Ambientes "isolados":

- ★ Não é necessário instalar softwares dependentes na máquina principal;
- ★ Evita conflitos de versão;
- ★ Abstrair a parte da infra-estrutura aplicação;
- ★ Toda a infraestrutura pode ser facilmente executada na máquina local;

## >>> Entrega - Objetivos

- \* Ambientes "isolados":

- \* Ambiente de testes = Ambiente de produção;

## >>> Entrega - Objetivos

- \* Ambientes "isolados":

- \* Ambiente de testes = Ambiente de produção;
- \* Entregas de software de forma mais rápida;
  - \* Não é necessário instalar dependências nas máquinas (↑ tempo);

## >>> Entrega - CI/CD

- ★ Docker é ótimo para *continuous delivery* e *continuous integration*:
  - ★ Desenvolvedores fazem mudanças no código localmente;

## >>> Entrega - CI/CD

- \* Docker é ótimo para *continuous delivery* e *continuous integration*:
  - \* Desenvolvedores fazem mudanças no código localmente;
  - \* No ambiente de teste são executados testes manuais e automáticos;



## >>> Entrega - CI/CD

- \* Docker é ótimo para *continuous delivery* e *continuous integration*:
  - \* Desenvolvedores fazem mudanças no código localmente;
  - \* No ambiente de teste são executados testes manuais e automáticos;
  - \* Erros podem ser reproduzidos e arrumados usando docker;

## >>> Entrega - CI/CD

- ★ Docker é ótimo para *continuous delivery* e *continuous integration*:
  - ★ Desenvolvedores fazem mudanças no código localmente;
  - ★ No ambiente de teste são executados testes manuais e automáticos;
  - ★ Erros podem ser reproduzidos e arrumados usando docker;
  - ★ Então a imagem vai para o ambiente de produção;

## >>> Entrega - Produção e escalabilidade

- ★ Responsive deployment and scaling
  - ★ Pode ser executado em:
    - ★ Máquinas físicas;

## >>> Entrega - Produção e escalabilidade

- ★ Responsive deployment and scaling

- ★ Pode ser executado em:

- ★ Máquinas físicas;
    - ★ Máquinas virtuais;

## >>> Entrega - Produção e escalabilidade

- ★ Responsive deployment and scaling

- ★ Pode ser executado em:

- ★ Máquinas físicas;
    - ★ Máquinas virtuais;
    - ★ Data centers;

## >>> Entrega - Produção e escalabilidade

### \* Responsive deployment and scaling

- \* Pode ser executado em:

- \* Máquinas físicas;
- \* Máquinas virtuais;
- \* Data centers;
- \* Nuvem;

## >>> Entrega - Produção e escalabilidade

- \* Responsive deployment and scaling

- \* Pode ser executado em:

- \* Máquinas físicas;
    - \* Máquinas virtuais;
    - \* Data centers;
    - \* Nuvem;
    - \* Misto;

## >>> Entrega - Produção e escalabilidade

### \* Responsive deployment and scaling

#### \* Pode ser executado em:

- \* Máquinas físicas;
- \* Máquinas virtuais;
- \* Data centers;
- \* Nuvem;
- \* Misto;

#### \* Portabilidade → escalar os projetos ou retirar aplicações e serviços (Quase tempo real);



>>> **Ciclo de vida**

- \* Desenvolver: Aplicação e componentes isolados no container;

>>> **Ciclo de vida**

- \* Desenvolver: Aplicação e componentes isolados no container;
- \* Execução: Testar e distribuir containers;

## >>> Ciclo de vida

- \* Desenvolver: Aplicação e componentes isolados no container;
- \* Execução: Testar e distribuir containers;
- \* Entrega: Ambiente de produção usando container ou serviço orquestrado;

## >>> Arquitetura do docker

- \* Arquitetura cliente-servidor:
  - \* Cliente docker se comunica com o servidor do docker;

## >>> Arquitetura do docker

- \* Arquitetura cliente-servidor:
  - \* Cliente docker se comunica com o servidor do docker;
  - \* Comunicação usando sockets UNIX ou pela interface de redes usando uma API REST;

## >>> Arquitetura do docker

### \* Arquitetura cliente-servidor:

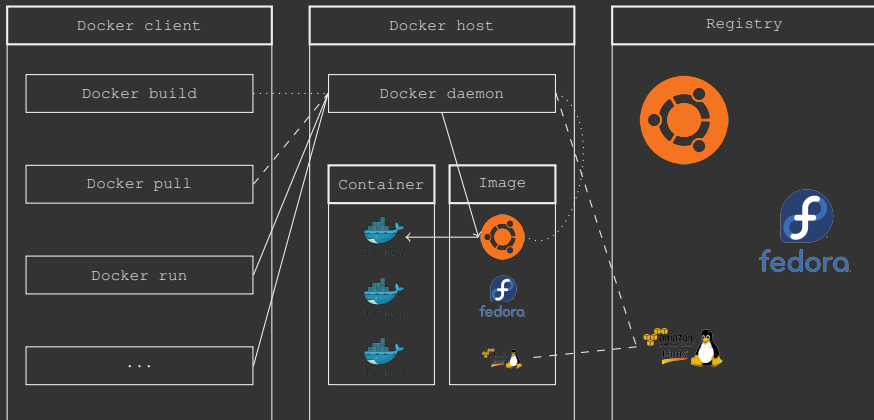
- \* Cliente docker se comunica com o servidor do docker;
- \* Comunicação usando sockets UNIX ou pela interface de redes usando uma API REST;
- \* O servidor é o responsável por criar imagens, rodar imagens e distribuir;

## >>> Arquitetura do docker

### \* Arquitetura cliente-servidor:

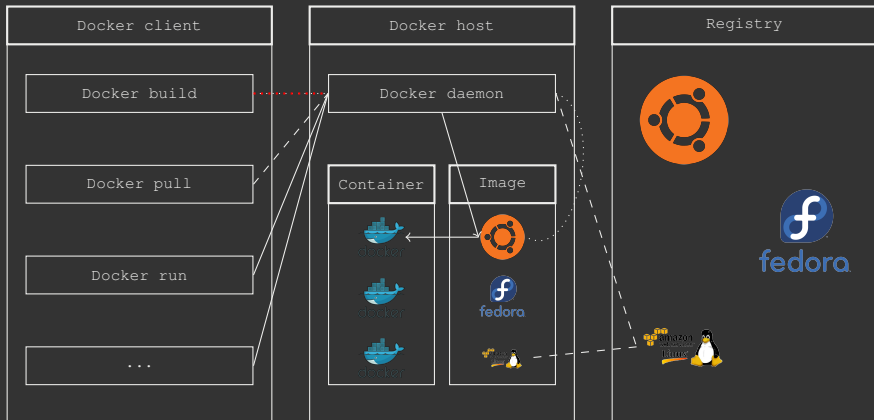
- \* Cliente docker se comunica com o servidor do docker;
- \* Comunicação usando sockets UNIX ou pela interface de redes usando uma API REST;
- \* O servidor é o responsável por criar imagens, rodar imagens e distribuir;
- \* Podem rodar na mesma máquina ou em máquinas diferentes;

## >>> Arquitetura do docker

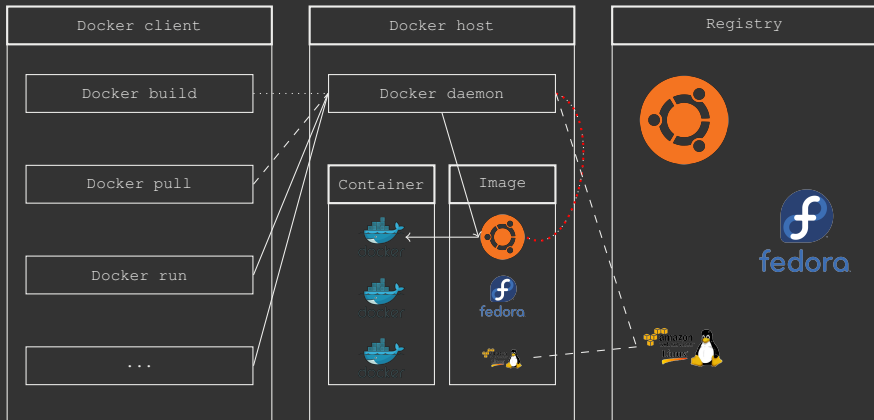




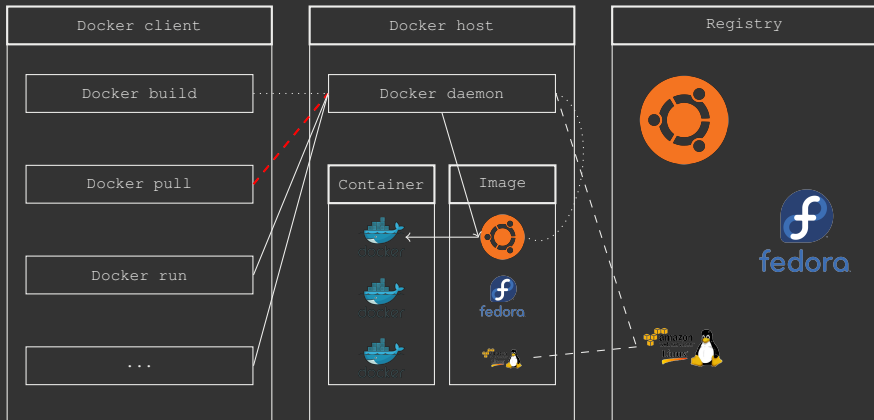
## >>> Arquitetura do docker



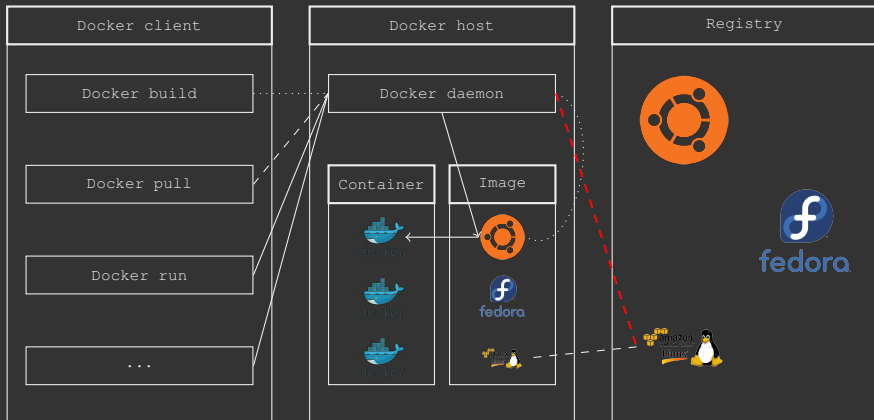
## >>> Arquitetura do docker



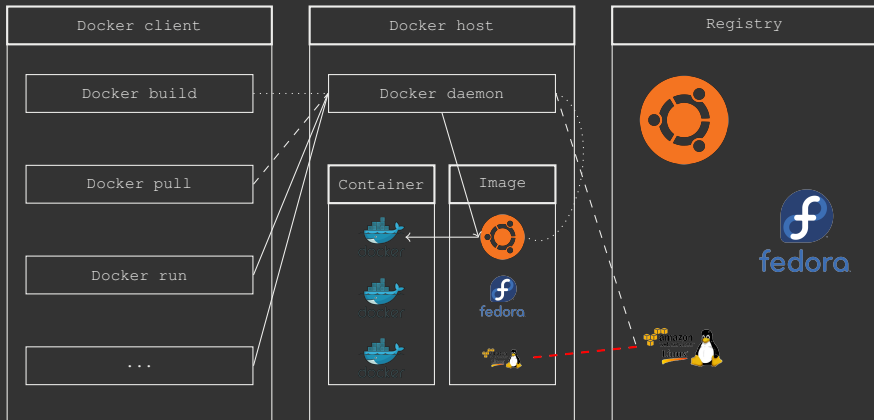
## >>> Arquitetura do docker



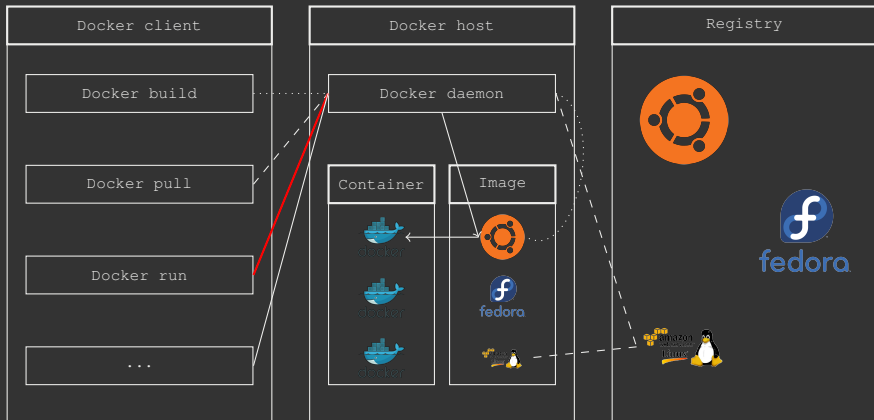
## >>> Arquitetura do docker



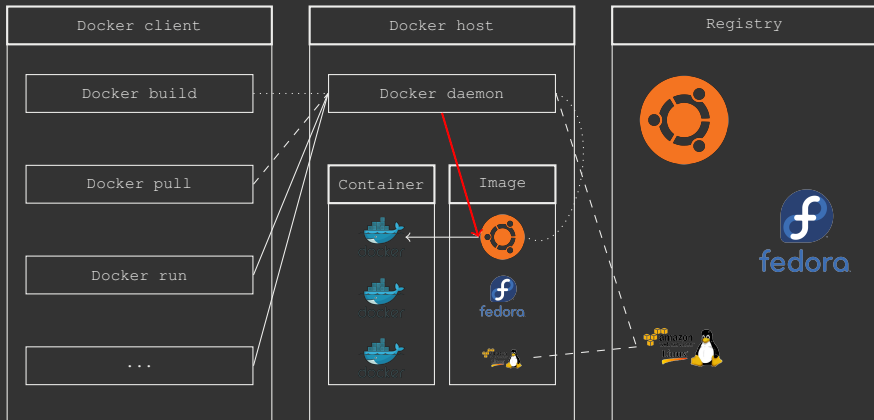
# >>> Arquitetura do docker



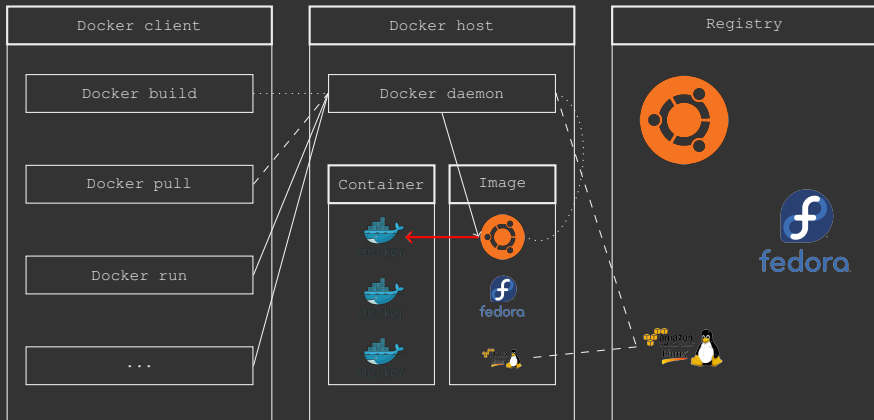
## >>> Arquitetura do docker



## >>> Arquitetura do docker



## >>> Arquitetura do docker





## >>> Docker Objects - Imagem

\* É um **template**. Portanto, só é permitido ler;

## >>> Docker Objects - Imagem

- \* É um **template**. Portanto, só é permitido ler;
- \* Contém instruções para criar um container de *Docker*;

## >>> Docker Objects - Imagem

- \* É um **template**. Portanto, só é permitido ler;
- \* Contém instruções para criar um container de *Docker*;
- \* Uma **Imagem** é composta por camadas;

## >>> Docker Objects - Imagem

- \* É um **template**. Portanto, só é permitido ler;
- \* Contém instruções para criar um container de *Docker*;
- \* Uma **Imagem** é composta por camadas;
- \* Camada inicial → Imagem pequena de Linux;

## >>> Docker Objects - Imagem

- \* É um **template**. Portanto, só é permitido ler;
- \* Contém instruções para criar um container de *Docker*;
- \* Uma **Imagem** é composta por camadas;
- \* Camada inicial → Imagem pequena de Linux;
- \* Imagens existentes → Customizadas para gerar novas imagens;

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.

## >>> Docker Objects - Imagem personalizada

- \* Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- \* *Dockerfile*:
  - \* Arquivo com instruções para construir e executar uma imagem de um container;

## >>> Docker Objects - Imagem personalizada

- \* Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- \* *Dockerfile*:
  - \* Arquivo com instruções para construir e executar uma imagem de um container;
  - \* Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;



## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
WORKDIR /usr...
```

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
RUN pip install fla...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
COPY . .
```

```
RUN pip install fla...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
CMD ["python",...
```

```
COPY . .
```

```
RUN pip install fla...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
CMD ["python",...
```

```
COPY . .
```

```
RUN pip install fla...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
CMD ["python",...
```

```
COPY . .
```

```
RUN pip install fla...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

```
WORKDIR /usr...
```

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
CMD ["python",...
```

```
COPY . .
```

```
RUN pip install fla...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

```
RUN pip install uvi...
```

```
WORKDIR /usr...
```

```
FROM python:3
```



## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
CMD ["python", ...
```

```
COPY . .
```

```
RUN pip install fla...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

```
COPY . .
```

```
RUN pip install uvi...
```

```
WORKDIR /usr...
```

```
FROM python:3
```

## >>> Docker Objects - Imagem personalizada

- ★ Para construir a uma imagem customizada, é preciso construir um *Dockerfile*.
- ★ *Dockerfile*:
  - ★ Arquivo com instruções para construir e executar uma imagem de um container;
  - ★ Cada instrução no arquivo Dockerfile adiciona uma camada nova na imagem;

```
CMD ["python",...  
  
COPY . .  
  
RUN pip install fla...  
  
WORKDIR /usr...  
  
FROM python:3
```

```
ENTRYPOINT u...  
  
COPY . .  
  
RUN pip install uvi...  
  
WORKDIR /usr...  
  
FROM python:3
```

## >>> Docker Objects - Imagem

\* Novas customizações → Novas camadas na Imagem;

## >>> Docker Objects - Imagem

- \* Novas customizações → Novas camadas na Imagem;
- \* Na construção de uma imagem, apenas as camadas que sofreram mudanças vão ser reconstruídas;

## >>> Docker Objects - Imagem

- \* Novas customizações → Novas camadas na Imagem;
- \* Na construção de uma imagem, apenas as camadas que sofreram mudanças vão ser reconstruídas;
- \* Imagens leves, pequenas e rápidas (Comparadas com outras tecnologias);

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;

## >>> Docker Objects - Containers

- ★ Instância de uma imagem;
  - ★ Configurações definidas na inicialização e na imagem;
  - ★ Mudanças no container não são armazenadas;
- ★ Com a API do docker é possível:
  - ★ Criar um container;



## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;
- \* Com a API do docker é possível:
  - \* Criar um container;
  - \* Executar um container;

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;
- \* Com a API do docker é possível:
  - \* Criar um container;
  - \* Executar um container;
  - \* Parar um container;

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;
- \* Com a API do docker é possível:
  - \* Criar um container;
  - \* Executar um container;
  - \* Parar um container;
  - \* Mover um container;

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;
- \* Com a API do docker é possível:
  - \* Criar um container;
  - \* Executar um container;
  - \* Parar um container;
  - \* Mover um container;
  - \* Deletar um container;

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;
- \* Com a API do docker é possível:
  - \* Criar um container;
  - \* Executar um container;
  - \* Parar um container;
  - \* Mover um container;
  - \* Deletar um container;
- \* Gerenciar:
  - \* Redes;

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;
- \* Com a API do docker é possível:
  - \* Criar um container;
  - \* Executar um container;
  - \* Parar um container;
  - \* Mover um container;
  - \* Deletar um container;
- \* Gerenciar:
  - \* Redes;
  - \* Armazenamento;

## >>> Docker Objects - Containers

- \* Instância de uma imagem;
  - \* Configurações definidas na inicialização e na imagem;
  - \* Mudanças no container não são armazenadas;
- \* Com a API do docker é possível:
  - \* Criar um container;
  - \* Executar um container;
  - \* Parar um container;
  - \* Mover um container;
  - \* Deletar um container;
- \* Gerenciar:
  - \* Redes;
  - \* Armazenamento;
  - \* Imagem;

## >>> Docker Objects - Tecnologias

- \* Linguagem **Go**;



## >>> Docker Objects - Tecnologias

- \* Linguagem **Go**;
- \* Muitas funcionalidades do kernel do linux:
  - \* namespaces: Garante que cada um dos containers tenha a capacidade de se isolar em níveis:

## >>> Docker Objects - Tecnologias

- \* Linguagem **Go**;
- \* Muitas funcionalidades do kernel do linux:
  - \* namespaces: Garante que cada um dos containers tenha a capacidade de se isolar em níveis:
    - \* PID: Ter seus próprios PIDs, mas a máquina host pode ver os PIDs do container;
    - \* NET: Ter suas próprias interfaces de redes e portas. Responsável pela comunicação de containers;
    - \* MNT: Garante que um sistema de arquivos montado não consiga acessar outro sistema de arquivos montado por outro mnt;
    - \* IPC: Ter um SystemV IPC isolado, além de uma fila de mensagens POSIX própria;
    - \* UTS: Isolamento do hostname, nome do domínio, versão do SO, etc...;

## >>> Docker Objects - Tecnologias

- \* Linguagem **Go**;
- \* Muitas funcionalidades do kernel do linux:
  - \* namespaces: Garante que cada um dos containers tenha a capacidade de se isolar em níveis:
    - \* PID: Ter seus próprios PIDs, mas a máquina host pode ver os PIDs do container;
    - \* NET: Ter suas próprias interfaces de redes e portas. Responsável pela comunicação de containers;
    - \* MNT: Garante que um sistema de arquivos montado não consiga acessar outro sistema de arquivos montado por outro mnt;
    - \* IPC: Ter um SystemV IPC isolado, além de uma fila de mensagens POSIX própria;
    - \* UTS: Isolamento do hostname, nome do domínio, versão do SO, etc...;
  - \* cgroups: Garantir o controle do consumo de processo;

## >>> Docker Objects - Tecnologias

- \* Linguagem **Go**;
- \* Muitas funcionalidades do kernel do linux:
  - \* namespaces: Garante que cada um dos containers tenha a capacidade de se isolar em níveis:
    - \* PID: Ter seus próprios PIDs, mas a máquina host pode ver os PIDs do container;
    - \* NET: Ter suas próprias interfaces de redes e portas. Responsável pela comunicação de containers;
    - \* MNT: Garante que um sistema de arquivos montado não consiga acessar outro sistema de arquivos montado por outro mnt;
    - \* IPC: Ter um SystemV IPC isolado, além de uma fila de mensagens POSIX própria;
    - \* UTS: Isolamento do hostname, nome do domínio, versão do SO, etc...;
  - \* cgroups: Garantir o controle do consumo de processo;
  - \* Linux Security Modules: Permissões do que eu sou permitido fazer;

## >>> Docker Objects - Container Runtimes

- \* As ferramentas **Container Runtimes** são ferramentas que facilitam a criação de containers de forma isolada e segura.
- \* Open container Initiative runtimes:
  - \* Native Runtimes:
    - \* runC: escrito em go e mantido pelo projeto moby do docker;
    - \* Railcar: escrito em rust, mas foi abandonado;
    - \* Crun: escrito em c, performático e leve;
- \* Container Runtime Interface
  - \* containerd: um runtime de alto nível desenvolvido no projeto moby do docker, por padrão usa o runC por baixo dos panos;
  - \* cri-o: implementação liderada pela Redhat, feita especificamente para o kubernetes;

## >>> Docker Objects - Resumo

- \* Daemon (dockerd): espera por chamads API do docker, e gerencia objetos como imagens, containers, redes e volumes;

## >>> Docker Objects - Resumo

- \* Daemon (dockerd): espera por chamads API do docker, e gerencia objetos como imagens, containers, redes e volumes;
- \* Cliente (docker): interage com o dockerd;
  - \* Comandos como: docker run, o cliente envia comandos para o dockerd;

## >>> Docker Objects - Resumo

- \* Daemon (dockerd): espera por chamads API do docker, e gerencia objetos como imagens, containers, redes e volumes;
- \* Cliente (docker): interage com o dockerd;
  - \* Comandos como: docker run, o cliente envia comandos para o dockerd;
  - \* O cliente docker pode se comunicar com mais de um daemon;



## >>> Docker Objects - Resumo

- \* Daemon (dockerd): espera por chamads API do docker, e gerencia objetos como imagens, containers, redes e volumes;
- \* Cliente (docker): interage com o dockerd;
  - \* Comandos como: docker run, o cliente envia comandos para o dockerd;
  - \* O cliente docker pode se comunicar com mais de um daemon;
- \* Docker desktop: é uma aplicação fácil de ser instalada nos ambientes Mac, Windows ou Linux.

## >>> Docker Objects - Resumo

- \* Daemon (dockerd): espera por chamads API do docker, e gerencia objetos como imagens, containers, redes e volumes;
- \* Cliente (docker): interage com o dockerd;
  - \* Comandos como: docker run, o cliente envia comandos para o dockerd;
  - \* O cliente docker pode se comunicar com mais de um daemon;
- \* Docker desktop: é uma aplicação fácil de ser instalada nos ambientes Mac, Windows ou Linux.
  - \* O docker desktop contém o daemon (dockerd), o docker client (docker), docker compose, docker content trust, kubernetes e credential helper;

## >>> Docker Objects - Resumo

- ★ Docker registries: Guarda as imagens de docker;
  - ★ Docker Hub: é um registro público, por padrão o docker procura por imagens no Docker Hub;

## >>> Docker Objects - Resumo

- ★ Docker registries: Guarda as imagens de docker;
  - ★ Docker Hub: é um registro público, por padrão o docker procura por imagens no Docker Hub;
  - ★ É possível ter até seu registro próprio;

## >>> Docker Objects - Resumo

- ★ Docker registries: Guarda as imagens de docker;
  - ★ Docker Hub: é um registro público, por padrão o docker procura por imagens no Docker Hub;
  - ★ É possível ter até seu registro próprio;
  - ★ Os comandos docker pull, docker run e docker push procuram ou enviam as imagens para o registro configurado na máquina que estão rodando;

## >>> Docker Objects - Resumo

- \* Docker registries: Guarda as imagens de docker;
  - \* Docker Hub: é um registro público, por padrão o docker procura por imagens no Docker Hub;
  - \* É possível ter até seu registro próprio;
  - \* Os comandos docker pull, docker run e docker push procuram ou enviam as imagens para o registro configurado na máquina que estão rodando;
- \* Docker objects: Imagens, containers, redes, volumes, plugins, etc...

## >>> Volumes

- ★ Os volumes são formas de armazenar informações persistentes para uso futuro.

## >>> Volumes

- ★ Os volumes são formas de armazenar informações persistentes para uso futuro.
- ★ Tipos de volumes:
  - ★ Volumes: Os volumes são gerenciados pelo próprio **docker**.



## >>> Volumes

- ★ Os volumes são formas de armazenar informações persistentes para uso futuro.
- ★ Tipos de volumes:
  - ★ Volumes: Os volumes são gerenciados pelo próprio **docker**.
  - ★ Birt mounts: É possível montar diretórios dentro do container, dessa forma o container pode acessar arquivos que existem dentro da máquina host.

## >>> Volumes

- ★ Os volumes são formas de armazenar informações persistentes para uso futuro.
- ★ Tipos de volumes:
  - ★ Volumes: Os volumes são gerenciados pelo próprio **docker**.
  - ★ Birt mounts: É possível montar diretórios dentro do container, dessa forma o container pode acessar arquivos que existem dentro da máquina host.
  - ★ Tmpfs: O **tmpfs** é uma forma de evitar armazenar dados permanenteente, e melhora a performace do container evitando escrever na camada de escrita do container

## >>> Redes - Bridge

- ★ Usar a interface de rede **docker0**;
  - ★ Cada container vai ter um veth (Virtual ethernet interface);

## >>> Redes - Bridge

- ★ Usar a interface de rede **docker0**;
  - ★ Cada container vai ter um veth (Virtual ethernet interface);
  - ★ Cada veth vai estar conectado com o docker0;

## >>> Redes - Bridge

- \* Usar a interface de rede **docker0**;
  - \* Cada container vai ter um veth (Virtual ethernet interface);
  - \* Cada veth vai estar conectado com o docker0;
- \* Todos os containers na rede vão conseguir se comunicar (Ruim);

```
>>> Redes - User-Defined Bridge
```

★ Igual ao Bridge

## >>> Redes - User-Defined Bridge

- ★ Igual ao Bridge
- ★ Mais controle das conexões entre os containers

## >>> Redes - User-Defined Bridge

- ★ Igual ao Bridge
- ★ Mais controle das conexões entre os containers
- ★ Isolados dos containers dentro do bridge padrão



>>> **Redes - Host**

- ★ Compartilha o ip e as portas com a máquina host

>>> **Redes - Host**

- ★ Compartilha o ip e as portas com a máquina host
- ★ É como se fosse uma aplicação rodando no computador

## >>> Redes - Host

- \* Compartilha o ip e as portas com a máquina host
- \* É como se fosse uma aplicação rodando no computador
- \* Acaba não ficando isolado

## >>> Redes - macVLAN

- ★ Feito para aplicações antigas que precisam conectar diretamente na internet física
  - ★ Conseguem um endereço MAC próprio;

## >>> Redes - macVLAN

- ★ Feito para aplicações antigas que precisam conectar diretamente na internet física
  - ★ Conseguem um endereço MAC próprio;
  - ★ Tem seus próprios IPs;

## >>> Redes - macVLAN

- ★ Feito para aplicações antigas que precisam conectar diretamente na internet física
  - ★ Conseguem um endereço MAC próprio;
  - ★ Tem seus próprios IPs;
- ★ Desvantagens:
  - ★ Existe só um endereço MAC;
    - ★ Precisa do modo Promiscuous (Um endereço físico pode ter múltiplos endereços MAC)

## >>> Redes - macVLAN

- ★ Feito para aplicações antigas que precisam conectar diretamente na internet física
  - ★ Conseguem um endereço MAC próprio;
  - ★ Tem seus próprios IPs;
- ★ Desvantagens:
  - ★ Existe só um endereço MAC;
    - ★ Precisa do modo Promiscuous (Um endereço físico pode ter múltiplos endereços MAC)
  - ★ Não existir o DHCP do roteador e o DHCP do docker;

## >>> Pratico - Docker client

```
* docker system info
* docker ps [-a]
* docker container;
    * ls;
    * {un,}pause;
    * stop;
    * start;
    * restart;
    * create;
    * run;
* docker
    {image,volume,network};
* docker build;
```

```
* docker history;
* docker inspect;
* docker logs;
* docker ports;
* docker stats;
* docker top;
```



>>> **Pratico - Docker client**

- \* `docker commit;`
- \* `docker login;`
- \* `docker pull;`
- \* `docker push;`
- \* `docker search;`

## >>> Pratico - Dockerfile

- \* **FROM:** Indica qual imagem vai ser utilizada para ser a base do container;
- \* **MAINTAINER:** Autor da imagem;
- \* **LABEL:** Adiciona metadados (Versão, descrição e fabricante);
- \* **ADD:** Copia arquivos para dentro do container (Pode descomprimir alguns arquivos e pegar arquivos de URL);
- \* **COPY:** Copia arquivos e diretórios para dentro do container;
- \* **CMD:** Executar um comando no início da execução do container. Mas pode definir argumentos padrão para o **ENTRYPOINT**;

## >>> Pratico - Dockerfile

- ★ **ENTRYPOINT:** Assim como o **CMD** executa um comando no início da execução do container, mas é mais difícil sobrescrever. E pode receber argumentos do **CMD**;
- ★ **ENV:** Coloca variáveis de ambiente no container;
- ★ **EXPOSE:** Abre uma porta para a **Rede interna**;
- ★ **RUN:** Executa um comando durante a criação da imagem;
- ★ **USER:** Define qual usuário vai ser utilizado para executar os próximos comandos;
- ★ **WORKDIR:** Define o diretório raiz;
- ★ **VOLUME:** Permite a criação de um ponto de montagem no container;

## >>> OWASP Docker Top 10

- \* D01 - Secure User Mapping;
- \* D02 - Patch Management Strategy;
- \* D03 - Network Segmentation and Firewalling;
- \* D04 - Secure Defaults and Hardening;
- \* D05 - Maintain Security Contexts;
- \* D06 - Protect Secrets;
- \* D07 - Resource Protection;
- \* D08 - Container Image Integrity and Origin;
- \* D09 - Follow Immutable Paradigm;
- \* D10 - Logging;

## >>> Segurança - Ferramentas

- \* Docker scan: Segurança de Imagens (Snyk)
- \* Docker bench security
- \* Guideline para usar docker (CIS)
- \* InSpec (Segurança e Compliance)
- \* Lynis
- \* OWASP Docker-security
- \* Dockscan
- \* Comparações de scanners de vulnerabilidade de containers
- \* 20 de scanners de vulnerabilidade de containers
- \* Lista de scanners de vulnerabilidade de container
- \* SELinux (Security Enchantment Linux): Administrar permissões no Linux
  - \* O Linux usa o DAC (Discretionary Access Control);
  - \* O *SELinux* permite o MAC (Mandatory Access Control);
  - \* Adiciona categorias/rótulos/perfis em todos os objetos contidos no sistema de arquivos;

## >>> Segurança - Máquina host

- ★ Docker-bench-security: Security auditing and benchmarktool for Docker;

## >>> Segurança - Máquina host

- ★ Docker-bench-security: Security auditing and benchmarktool for Docker;
- ★ The Linux Auditing Framework: Linux Auditing Framework

## >>> **Segurança - Máquina host**

- ★ Docker-bench-security: Security auditing and benchmarktool for Docker;
- ★ The Linux Auditing Framework: Linux Auditing Framework
- ★ InSpec: Automated security and compliance Framework



## >>> Segurança - Máquina host

- ★ Docker-bench-security: Security auditing and benchmarktool for Docker;
- ★ The Linux Auditing Framework: Linux Auditing Framework
- ★ InSpec: Automated security and compliance Framework
- ★ Lynis: Security auditing tool for systems based on UNIX like Linux, macOS, BSD and others.  
**in-depht security scan** and runs on systems itself. The primary goal is to test security defenses and provide tips for further system hardening

## >>> Docker Bench Security

- \* *docker-bench-security* vai analisar todo o sistema e dar uma pontuação total para o seu sistema.
- \* O *docker-bench-security* pode ser usado com a flag *-c* e o argumento *host\_configuration*;
  - \* Dessa forma vai ser rodado testes de segurança na máquina local;

## >>> The Linux Audit Framework

★ *auditctl*: Vai fazer logs de alguns programas que estão rodando na máquina host.

- ★ */run/containerd*
- ★ */var/lib/docker*
- ★ */etc/docker*
- ★ */lib/systemd/system/docker.service*
- ★ */lib/systemd/system/docker.socket*
- ★ */etc/default/docker*
- ★ */usr/bin/docker-containerd*
- ★ */usr/bin/docker-runc*
- ★ */usr/bin/containerd*
- ★ */usr/bin/containerd-shim*
- ★ */usr/bin/containerd-shim-runc-v1*
- ★ */usr/bin/containerd-shim-runc-v2*

## >>> The Linux Audit Framework

- \* As regras precisam ser adicionadas dentro do arquivo *audit.rules*;
  - \* O arquivo das regras fica armazenado no */etc/audit/rules.d/audit.rules*;

## >>> The Linux Audit Framework

- \* As regras precisam ser adicionadas dentro do arquivo *audit.rules*;
  - \* O arquivo das regras fica armazenado no */etc/audit/rules.d/audit.rules*;
- \* O comando *aureport* vai ser usado para verificar os logs;

## >>> Logins por SSH

- \* Arquivo `/etc/ssh/sshd_config`

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);

## >>> Logins por SSH

- \* Arquivo */etc/ssh/sshd\_config*

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);
- \* LogLevel: Mudar para VERBOSE;

## >>> Logins por SSH

### \* Arquivo */etc/ssh/sshd\_config*

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);
- \* LogLevel: Mudar para VERBOSE;
- \* LoginGraceTime: Diminuir o tempo limite (O server desconecta o usuário se ele não conseguir fazer o login);



## >>> Logins por SSH

### \* Arquivo */etc/ssh/sshd\_config*

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);
- \* LogLevel: Mudar para VERBOSE;
- \* LoginGraceTime: Diminuir o tempo limite (O server desconecta o usuário se ele não conseguir fazer o login);
- \* PermitRootLogin: Não permitir;

## >>> Logins por SSH

### \* Arquivo `/etc/ssh/sshd_config`

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);
- \* LogLevel: Mudar para VERBOSE;
- \* LoginGraceTime: Diminuir o tempo limite (O server desconecta o usuário se ele não conseguir fazer o login);
- \* PermitRootLogin: Não permitir;
- \* MaxAuthTries: Colocar um limite nas tentativas de autenticação;

## >>> Logins por SSH

### \* Arquivo `/etc/ssh/sshd_config`

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);
- \* LogLevel: Mudar para VERBOSE;
- \* LoginGraceTime: Diminuir o tempo limite (O server desconecta o usuário se ele não conseguir fazer o login);
- \* PermitRootLogin: Não permitir;
- \* MaxAuthTries: Colocar um limite nas tentativas de autenticação;
- \* MaxSessions: Número máximo de sessões concorrentes;

## >>> Logins por SSH

### \* Arquivo `/etc/ssh/sshd_config`

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);
- \* LogLevel: Mudar para VERBOSE;
- \* LoginGraceTime: Diminuir o tempo limite (O server desconecta o usuário se ele não conseguir fazer o login);
- \* PermitRootLogin: Não permitir;
- \* MaxAuthTries: Colocar um limite nas tentativas de autenticação;
- \* MaxSessions: Número máximo de sessões concorrentes;
- \* PasswordAuthentication: Desabilitar login por senha;

## >>> Logins por SSH

### \* Arquivo `/etc/ssh/sshd_config`

- \* Port: Mudar a porta do SSH (Atacante é obrigado a escanear todas as portas);
- \* LogLevel: Mudar para VERBOSE;
- \* LoginGraceTime: Diminuir o tempo limite (O server desconecta o usuário se ele não conseguir fazer o login);
- \* PermitRootLogin: Não permitir;
- \* MaxAuthTries: Colocar um limite nas tentativas de autenticação;
- \* MaxSessions: Número máximo de sessões concorrentes;
- \* PasswordAuthentication: Desabilitar login por senha;
- \* PublicKeyAuthentication: Habilitar (Por padrão já aceita);

## >>> Referencias

- \* Securing Docker Host

## >>> Segurança - Docker Daemon

- \* Controlar acesso do grupo `docker`;

## >>> Segurança - Docker Daemon

- \* Controlar acesso do grupo `docker`;
- \* Criptografia TLS;



## >>> Segurança - Docker Daemon

- \* Controlar acesso do grupo `docker`;
- \* Criptografia TLS;
- \* Implementar Namespaces do usuario;

## >>> Segurança - Docker Daemon

- \* Controlar acesso do grupo `docker`;
- \* Criptografia TLS;
- \* Implementar Namespaces do usuario;
- \* Desabilitar comunicação entre containers na rede default;

## >>> Criptografia TLS

- ★ A criptografia TLS vai ficar entre o cliente do docker e o docker host

## >>> Criptografia TLS

- ★ A criptografia TLS vai ficar entre o cliente do docker e o docker host
- ★ Arrumar o arquivo `/etc/docker/daemon.json`:

## >>> Criptografia TLS

- \* A criptografia TLS vai ficar entre o cliente do docker e o docker host
- \* Arrumar o arquivo `/etc/docker/daemon.json`:
- \* Arrumar o arquivo `/etc/systemd/system/docker.service.d/override.conf`:

## >>> Criptografia TLS

- ★ A criptografia TLS vai ficar entre o cliente do docker e o docker host
- ★ Arrumar o arquivo `/etc/docker/daemon.json`:
- ★ Arrumar o arquivo `/etc/systemd/system/docker.service.d/override.conf`:
  - ★ `-tlsverify`
  - ★ `-tlscert=server-cert.pem`
  - ★ `-tlscacert=ca.pem`
  - ★ `-tlskey=server-key.pem`

## >>> Criptografia TLS

- \* Arrumar variáveis de ambiente no cliente:
  - \* DOCKER\_TLS\_VERIFY=1;
  - \* DOCKER\_CERT\_PATH='path';

## >>> Criptografia TLS

- \* Arrumar variáveis de ambiente no cliente:
  - \* DOCKER\_TLS\_VERIFY=1;
  - \* DOCKER\_CERT\_PATH='path';
- \* Para recarregar:
  - \* systemctl daemon-reload;
  - \* systemctl restart docker;



## >>> Criptografia TLS

- \* Arrumar variáveis de ambiente no cliente:
  - \* DOCKER\_TLS\_VERIFY=1;
  - \* DOCKER\_CERT\_PATH='path';
- \* Para recarregar:
  - \* systemctl daemon-reload;
  - \* systemctl restart docker;
- \* Script for TLS-Authentication

## >>> Criptografia TLS

- \* Arrumar variáveis de ambiente no cliente:
  - \* `DOCKER_TLS_VERIFY=1;`
  - \* `DOCKER_CERT_PATH='path';`
- \* Para recarregar:
  - \* `systemctl daemon-reload;`
  - \* `systemctl restart docker;`
- \* Script for TLS-Authentication
- \* Evitar ataque *Man-in-the-middle*

## >>> User Namespaces

- \* Criar um usuário e grupo nos arquivos /etc/subuid e /etc/subgid;

\* userns-remap="default"

## >>> User Namespaces

- \* Criar um usuário e grupo nos arquivos /etc/subuid e /etc/subgid;
- \* Ou usar o usuário padrão **dockremap**;

\* userns-remap="default"

## >>> User Namespaces

- \* Criar um usuário e grupo nos arquivos /etc/subuid e /etc/subgid;
- \* Ou usar o usuário padrão **dockremap**;

\* userns-remap="default"

## >>> User Namespaces

- \* Criar um usuário e grupo nos arquivos `/etc/subuid` e `/etc/subgid`;
- \* Ou usar o usuário padrão **dockremap**;
- \* Diminuir os danos causados em caso de *container breakout*;

\* `usersns-remap="default"`

## >>> User Namespaces

- \* Criar um usuário e grupo nos arquivos `/etc/subuid` e `/etc/subgid`;
- \* Ou usar o usuário padrão **dockremap**;
- \* Diminuir os danos causados em caso de *container breakout*;
- \* Arrumar o arquivo `/etc/docker/daemon.json`:
- \* Arrumar o arquivo `/etc/systemd/system/docker.service.d/override.conf`:
  - \* `usersns-remap="default"`

## >>> Comunicação entre containers

\* `icc="false"`



## >>> Comunicação entre containers

- ★ `icc="false"`

- ★ Containers dentro da rede default não vão conseguir se comunicar;

## >>> Comunicação entre containers

- \* `icc="false"`

- \* Containers dentro da rede default não vão conseguir se comunicar;

## >>> Comunicação entre containers

- ★ `icc="false"`

- ★ Containers dentro da rede default não vão conseguir se comunicar;

## >>> Comunicação entre containers

- \* Arrumar o arquivo `/etc/docker/daemon.json`:
- \* Arrumar o arquivo `/etc/systemd/system/docker.service.d/override.conf`:
  - \* `icc="false"`
- \* Containers dentro da rede default não vão conseguir se comunicar;

## >>> Links

- ★ Setup docker registry with TLS SSL: <https://www.thegeekstuff.com/2017/01/secure-docker-registry/>
- ★ Generate SSL Key:  
<https://www.thegeekstuff.com/2009/07/linux-apache-mod-ssl-generate-key-csr-crt-file/>
- ★ Docker Registry [https://github.com/marcelogomesrp/curso\\_docker/tree/main/15\\_registry](https://github.com/marcelogomesrp/curso_docker/tree/main/15_registry)
- ★ How to create your own docker registry:  
<https://gabrieltanner.org/blog/docker-registry/>
- ★ Docker docs:  
<https://docs.docker.com/registry/configuration/>
- ★ Docker native basic auth: <https://docs.docker.com/registry/deploying/#native-basic-auth>

## >>> Links

- \* ip addr;
- \* ps ef;
- \* ss tl;
- \* iptable;
- \* SysV;
- \* SystemD;
- \* Passar parametros:
  - \* debian like
    - \* vim /etc/default/docker
  - \* redhatlike
    - \* vim /etc/sysconfig/docker