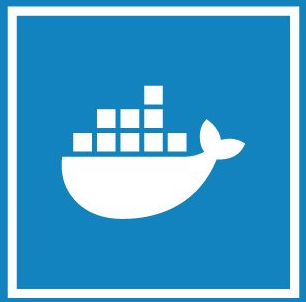
TREINA WEB

WWW.TREINAWEB.COM.BR



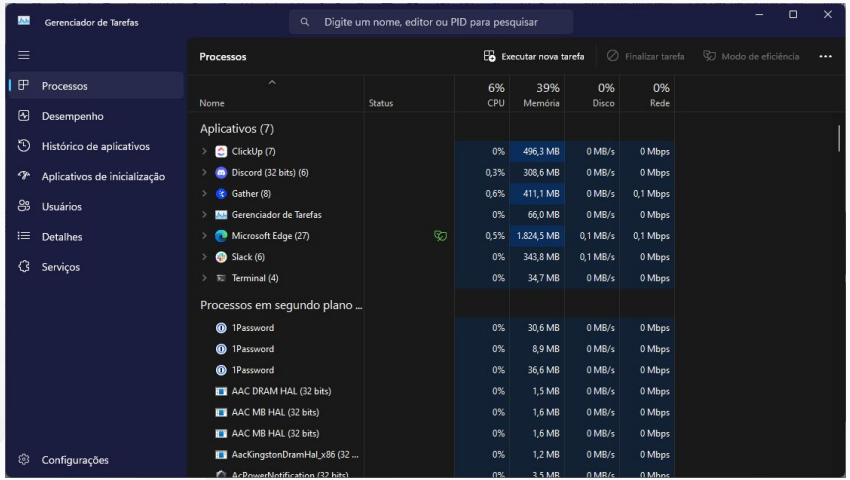
Docker - Containers, Imagens, Volumes e Networks

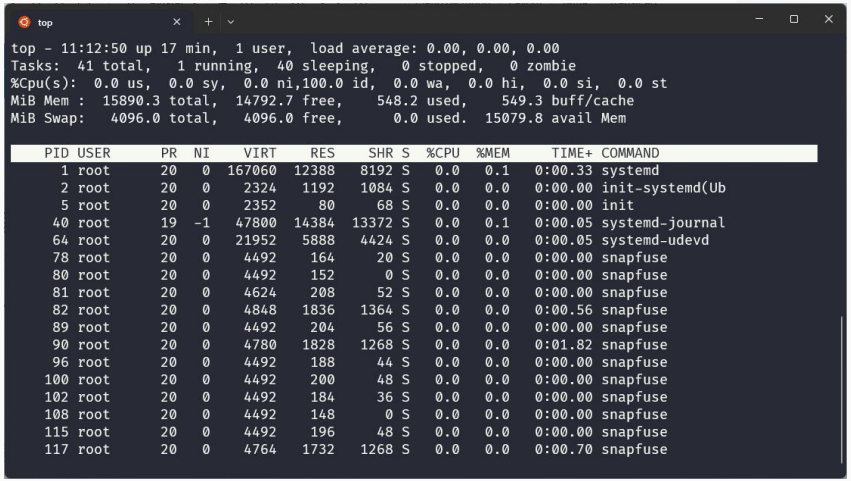
O que são Containers?

- Processos
- Namespaces
- Cgroups

Processos

No contexto de sistemas operacionais, um processo é um programa em execução juntamente com todas as informações necessárias para controlar sua execução. Um processo é a unidade básica de alocação de recursos e gerenciamento de tarefas em um sistema operacional. Cada processo representa uma tarefa ou um programa em execução, e o sistema operacional gerencia múltiplos processos concorrentemente.


Gerenciador de Tarefas
Digite um nome, editor ou PID para pesquisar
Processos
Executar nova tarefa
Finalizar tarefa
Modo de eficiência
Processos
Desempenho
Histórico de aplicativos
Aplicativos de inicialização
Usuários
Detalhes
Serviços
Configurações
Nome
Status
6% CPU
39% Memória
0% Disco
0% Rede
Aplicativos (7)
ClickUp (7) 0% 496,3 MB 0 MB/s 0 Mbps
Discord (32 bits) (6) 0,3% 308,6 MB 0 MB/s 0 Mbps
Gather (8) 0,6% 411,1 MB 0 MB/s 0,1 Mbps
Gerenciador de Tarefas 0% 66,0 MB 0 MB/s 0 Mbps
Microsoft Edge (27) 0,5% 1.824,5 MB 0,1 MB/s 0,1 Mbps
Slack (6) 0% 343,8 MB 0,1 MB/s 0 Mbps
Terminal (4) 0% 34,7 MB 0 MB/s 0 Mbps
Processos em segundo plano ...
1Password 0% 30,6 MB 0 MB/s 0 Mbps
1Password 0% 8,9 MB 0 MB/s 0 Mbps
1Password 0% 36,6 MB 0 MB/s 0 Mbps
AAC DRAM HAL (32 bits) 0% 1,5 MB 0 MB/s 0 Mbps
AAC MB HAL (32 bits) 0% 1,6 MB 0 MB/s 0 Mbps
AAC MB HAL (32 bits) 0% 1,6 MB 0 MB/s 0 Mbps
AacKingstonDramHal_x86 (32 ... 0% 1,2 MB 0 MB/s 0 Mbps


top
top - 11:12:50 up 17 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 41 total, 1 running, 40 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 15890.3 total, 14792.7 free, 548.2 used, 549.3 buff/cache
MiB Swap: 4096.0 total, 4096.0 free, 0.0 used. 15079.8 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 20 0 167060 12388 8192 S 0.0 0.1 0:00.33 systemd
2 root 20 0 2324 1192 1084 S 0.0 0.0 0:00.00 init-systemd(Ub
5 root 20 0 2352 80 68 S 0.0 0.0 0:00.00 init
40 root 19 -1 47800 14384 13372 S 0.0 0.1 0:00.05 systemd-journal
64 root 20 0 21952 5888 4424 S 0.0 0.0 0:00.05 systemd-udevd
78 root 20 0 4492 164 20 S 0.0 0.0 0:00.00 snapfuse
80 root 20 0 4492 152 0 S 0.0 0.0 0:00.00 snapfuse
81 root 20 0 4624 208 52 S 0.0 0.0 0:00.00 snapfuse
82 root 20 0 4848 1836 1364 S 0.0 0.0 0:00.56 snapfuse
89 root 20 0 4492 204 56 S 0.0 0.0 0:00.00 snapfuse
90 root 20 0 4780 1828 1268 S 0.0 0.0 0:01.82 snapfuse
96 root 20 0 4492 188 44 S 0.0 0.0 0:00.00 snapfuse
100 root 20 0 4492 200 48 S 0.0 0.0 0:00.00 snapfuse
102 root 20 0 4492 184 36 S 0.0 0.0 0:00.00 snapfuse
108 root 20 0 4492 148 0 S 0.0 0.0 0:00.00 snapfuse
115 root 20 0 4492 196 48 S 0.0 0.0 0:00.00 snapfuse
117 root 20 0 4764 1732 1268 S 0.0 0.0 0:00.70 snapfuse

Namespaces

Os namespaces são um mecanismo que permite isolar recursos de sistema para processos individuais ou grupos de processos. Eles são uma parte importante da tecnologia de contêineres e são usados para criar ambientes isolados nos quais os processos podem ser executados sem afetar o sistema hospedeiro ou outros processos.

- Namespace ABC
  - Processo 123
  - Processo 456
  - Processo 789
- Namespace DEF
  - Processo 321
  - Processo 654
  - Processo 987

Cgroups

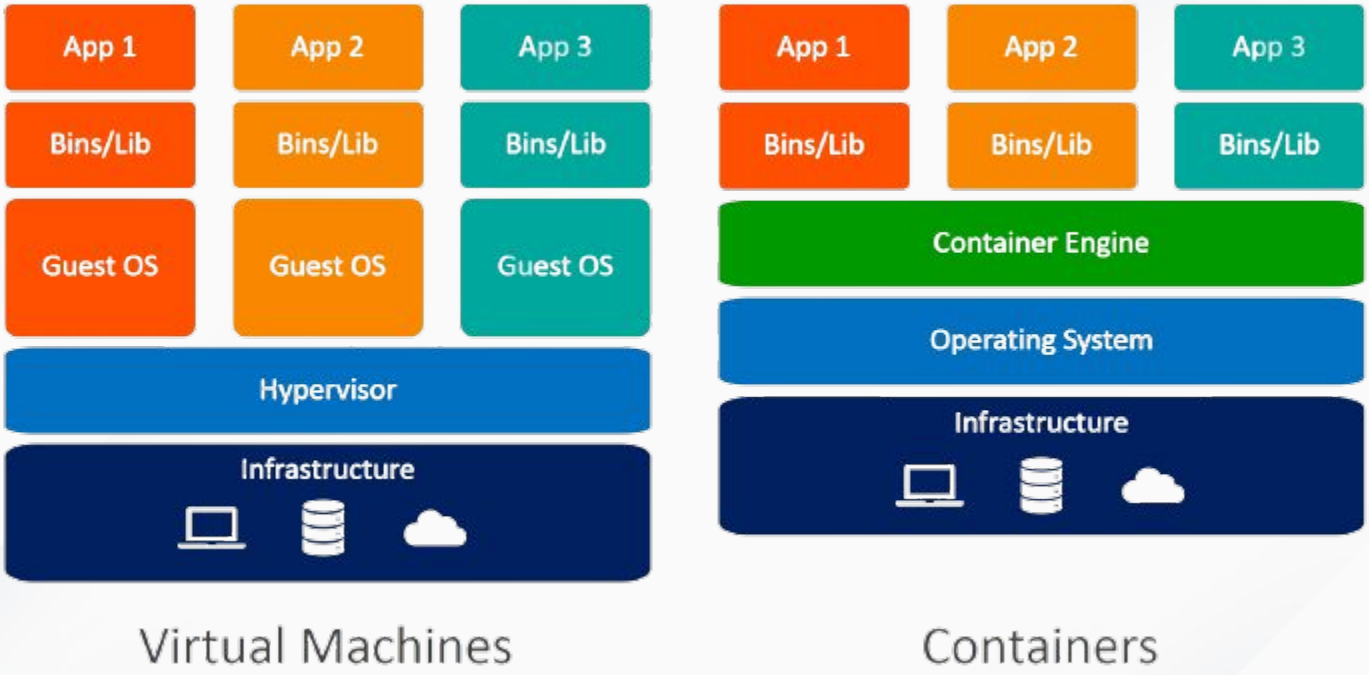
São um recurso em sistemas operacionais Linux que permitem o controle, limitação e monitoramento dos recursos de sistema alocados a processos ou grupos de processos. Eles são usados principalmente para gerenciar o uso de recursos, como CPU, memória, disco e largura de banda da rede, entre outros, para que múltiplos processos possam ser executados de forma eficiente e justa em um sistema compartilhado.

Containers

Um contêiner é um processo isolado em execução em uma máquina host que está isolado de todos os outros processos em execução nessa máquina host. Esse isolamento utiliza namespaces do kernel e cgroups. O Docker torna essas capacidades acessíveis e fáceis de usar. Para resumir, um contêiner:

- É uma instância executável de uma imagem. Você pode criar, iniciar, parar, mover ou excluir um contêiner usando a API, ou CLI do Docker.
- Pode ser executado em máquinas locais, máquinas virtuais ou implantado na nuvem.
- É portátil (e pode ser executado em qualquer sistema operacional).
- Está isolado de outros contêineres e executa seu próprio software, binários, configurações, etc.

Containers != VM



O que são Imagens?

“Um contêiner em execução utiliza um sistema de arquivos isolado. Esse sistema de arquivos isolado é fornecido por uma imagem, e a imagem deve conter tudo o que é necessário para executar um aplicativo - todas as dependências, configurações, scripts, binários, etc. A imagem também contém outras configurações para o contêiner, como variáveis de ambiente, um comando padrão para execução e outros metadados.” – Docker - What is an image?

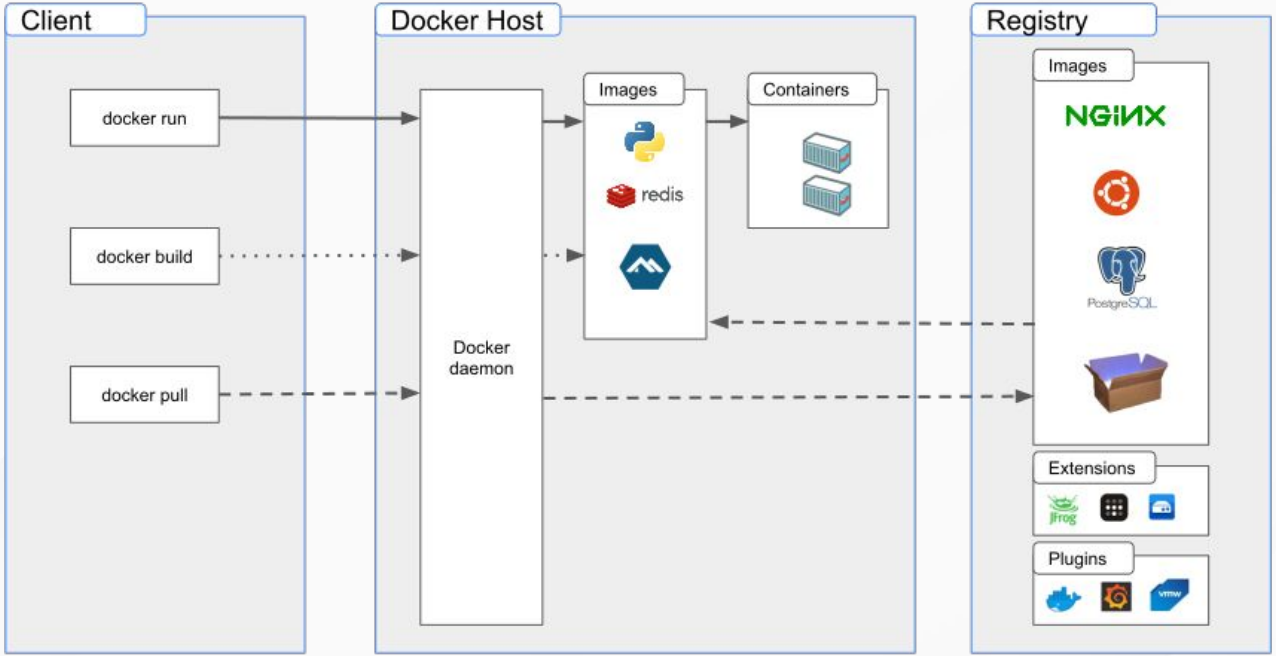
Dockerfile

```
FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000
```

O que é o Docker?

“O Docker é uma plataforma aberta para desenvolver, enviar e executar aplicativos. O Docker permite que você separe seus aplicativos da sua infraestrutura, para que você possa entregar software rapidamente. Com o Docker, você pode gerenciar sua infraestrutura da mesma forma que gerencia seus aplicativos. Ao aproveitar as metodologias do Docker para enviar, testar e implantar código, você pode reduzir significativamente o intervalo entre escrever código e executá-lo em produção.” – Docker - Overview

Arquitetura do Docker



Conceitos importantes

- **Contêineres:** São instâncias em tempo de execução das imagens Docker. Os contêineres são isolados uns dos outros e do sistema host, mas compartilham o kernel do sistema operacional subjacente, o que os torna leves e eficientes.
- **Imutabilidade:** As imagens Docker são imutáveis, o que significa que, uma vez criadas, elas não são alteradas. Se você precisar fazer uma alteração, deve criar uma nova imagem. Isso ajuda a garantir consistência e previsibilidade em ambientes de desenvolvimento, teste e produção.
- **Reutilização:** As imagens Docker podem ser reutilizadas em diferentes ambientes e por diferentes equipes. Isso facilita a colaboração e a implantação de aplicativos em vários ambientes, como desenvolvimento, teste e produção.
- **Registros Docker:** As imagens Docker são armazenadas em registros Docker, que são repositórios online onde as imagens podem ser compartilhadas e distribuídas.
- **Dockerfile:** É um arquivo de configuração usado para definir como uma imagem Docker deve ser construída. Ele lista as instruções para instalar dependências, copiar arquivos, configurar variáveis de ambiente e executar comandos dentro do contêiner.
- **Layering:** As imagens Docker são compostas por camadas (layers) empilhadas umas sobre as outras. Cada instrução no Dockerfile cria uma nova camada. Isso permite que as imagens sejam eficientes em termos de armazenamento, pois camadas compartilhadas entre imagens semelhantes são reutilizadas.