

# Mini curso de DevOps

Feito por

**Lucas Elias Baccan**

<https://code.lucasbaccan.com.br/off/germantech>



# Table of contents:

- German Tech Sistemas
  - Servidor
- Linux
  - Sumário
  - Introdução
  - Distribuições Linux
  - Manipulação de arquivos e diretórios
    - ls - Lista os arquivos e pastas do diretório atual
    - cd - Muda o diretório atual
    - pwd - Mostra o diretório atual
    - mkdir - Cria um diretório
    - touch - Cria um arquivo
    - cat - Mostra o conteúdo de um arquivo
    - cp - Copia um arquivo ou diretório
    - mv - Move um arquivo ou diretório / Renomeia um arquivo ou diretório
    - rm - Remove um arquivo ou diretório
  - Comandos do sistema
    - history - Mostra o histórico de comandos
    - clear - Limpa a tela do terminal
    - whoami - Mostra o nome do usuário atual
    - which - Mostra o caminho de um comando
    - sudo - Executa um comando como administrador
    - chmod - Alterar permissões
    - chown - Alterar dono
    - df - Exibir espaço em disco
    - du - Exibir tamanho de arquivo ou diretório
    - find - Buscar arquivo(s)
  - Manipulação de processos
    - ps - Exibir processos
    - top - Exibir processos em tempo real
    - kill - Finalizar processo
  - Gerenciamento de serviços
    - systemctl - Gerenciador de serviços do systemd
    - service - Gerenciador de serviços do init
- Jenkins

- Instalação
- Configuração
- Plugins
- Configurações gerais
- Estrutura
  - Jobs
  - Build
  - Artifacts
- Configurações recomendadas
  - Credenciais
  - Controle de acesso
  - Configuração global de ferramentas
- PostgreSQL
  - Instalação
  - Configuração
    - Alterando o arquivo pg\_hba.conf
    - Alterando o arquivo postgresql.conf
    - Criando um usuário
- Docker
  - Sumário
  - Instalação
  - Diferença entre máquina virtual e container
  - Imagens
  - Containers
  - Volumes
  - Comandos
    - docker run
    - docker start
    - docker stop
    - docker ps
    - docker rm
    - docker rmi
    - docker exec
    - docker logs
    - docker pull
  - Exemplos
    - Dockerfile

- Docker Compose
- Comandos
  - docker-compose up
  - docker-compose start
  - docker-compose stop
  - docker-compose restart
- Exemplo
  - docker-compose.yml
- AWS EC2
  - Instâncias EC2
  - Tipos de instâncias EC2
  - Volumes EC2
    - Snapshots de volumes EC2
  - Network Interfaces EC2
    - Security Groups EC2
  - Regiões EC2
  - Extra
- AWS S3
  - Bucket
  - Objeto
  - Região
- AWS Route 53
  - Domínio
- arara.gtech.site
  - Como clonar
  - Requisitos
  - Containers
    - postgres\_db
    - postgres\_cron
    - pgadmin (opcional)
    - grafana
    - prometheus
    - postgres\_exporter
    - node\_exporter (opcional)
  - Como instalar/iniciar o servidor

# German Tech Sistemas

! DOWNLOAD DO PDF

[Clique aqui para baixar](#)

Com a reestruturação do servidor de banco de dados `arara.gtech.site`, novas tecnologias foram implementadas, e o treinamento de novos colaboradores é necessário. Este tutorial tem como objetivo auxiliar no aprendizado das novas tecnologias.

O objetivo é passar pelos seguintes tópicos:

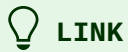
- Linux
- Jenkins
- Postgres
- Docker
- AWS EC2
- AWS S3
- AWS Route53
- `arara.gtech.site`

## Servidor

Para o novo servidor funcionar como esperado, é necessário que o servidor tenha os seguintes requisitos:

- Sistema Operacional Linux
- Git
- Docker
- Docker Compose

# Linux



LINK

LINK: <https://www.linux.org/>

## Sumário

- Introdução
- Manipulação de arquivos e diretórios
- Comandos do sistema
- Manipulação de processos

## Introdução

Quando falamos de servidores, a grande maioria roda Linux. Por isso, é importante saber como funciona o sistema operacional e seus comandos.

Uma forma fácil de aprender é através do `copy.sh`, um emulador de Linux que roda no navegador, então você pode testar os comandos sem precisar instalar nada.

Além disso, utilize **Cheatsheets** para consultar os comandos mais utilizados, aqui abaixo temos alguns exemplos:

- Lista 1
- Lista 2
- Lista 3

### IMPORTANTE

Abaixo você pode encontrar alguns comandos básicos para começar a utilizar o Linux, nos exemplos você pode ver o comando e em que pasta ele foi executado.

Pasta em que o comando foi executado

Comando que foi executado



#### DICA

Os comandos abaixo são um pequeno resumo dos comandos mais utilizados, para mais informações, busque o manual do comando.

```
comando --help
```

Exemplo:

```
ls --help
```

## Distribuições Linux

Existem diversas distribuições Linux, cada uma com suas características e peculiaridades. Algumas das mais populares são:

- Ubuntu
- Debian
- Fedora
- Arch Linux
- Manjaro
- CentOS
- Red Hat

Cada uma delas tem seu próprio gerenciador de pacotes, que é um programa que gerencia a instalação, remoção e atualização de programas. Alguns exemplos são:

- apt

- `yum`
- `pacman`
- `dnf`
- `zypper`

Então antes de trabalhar com um servidor Linux, é importante saber qual distribuição está sendo utilizada e qual é o gerenciador de pacotes e em que versão está instalado.

## Manipulação de arquivos e diretórios

Vamos ver alguns comandos para começar a mexer no Linux, começando pela manipulação de arquivos e diretórios.

O sistema de arquivos do Linux segue uma hierarquia, onde o diretório raiz é representado por `/`, e todos os outros diretórios são filhos dele. Todos os arquivos e diretórios são organizados em uma árvore, onde cada nó é um diretório e cada folha é um arquivo. Vamos ver alguns comandos para manipular os arquivos e diretórios.

### `ls` - Lista os arquivos e pastas do diretório atual

Você pode usar o comando `ls` para listar os arquivos e pastas do diretório atual. Por exemplo, se você estiver no diretório `/home/usuario`, ao executar o comando `ls` irá listar todos os arquivos e pastas do diretório `/home/usuario`.

```
/home/usuario
```

```
ls
```

Mas você pode listar os arquivos e pastas de outros diretórios, basta passar o caminho do diretório como parâmetro do comando `ls`. Por exemplo, se você quiser listar os arquivos e pastas do diretório `/home/usuario/Documentos`, basta executar o comando `ls`



`/home/usuario/Documentos`, mesmo que você não esteja no diretório `/home/usuario/Documentos`.

```
/home
```

```
ls /home/usuario/Documentos
```

### **i** ARGUMENTOS

- `-a` - Lista todos os arquivos e pastas, inclusive os que começam com `.` (ponto).
- `-l` - Lista os arquivos e pastas com mais detalhes.
- `-h` - Lista os arquivos e pastas com o tamanho em formato legível.

Exemplo:

```
ls -lah
```

## **cd** - Muda o diretório atual

Você pode usar o comando `cd` para mudar o diretório atual. Por exemplo, se você estiver no diretório `/home/usuario` e quiser mudar para o diretório `/home/usuario/Documentos`, basta executar o comando `cd Documentos` ou `cd /home/usuario/Documentos`.

```
/home/usuario
```

```
cd Documentos
```

Essa é a forma mais simples de usar o comando `cd`, utilizando os caminhos relativos. Mas você pode usar caminhos absolutos, como no exemplo abaixo:

```
/home
```

```
cd /home/usuario/Documentos
```

## **pwd** - Mostra o diretório atual

Você pode usar o comando `pwd` para mostrar o diretório atual. Por exemplo, se você estiver no diretório `/home/usuario/Documentos`, ao executar o comando `pwd` irá mostrar o caminho `/home/usuario/Documentos`.

```
pwd
```

## **mkdir** - Cria um diretório

Você pode usar o comando `mkdir` para criar um diretório. Por exemplo, se você estiver no diretório `/home/usuario/Documentos` e quiser criar um diretório chamado `Projetos`, basta executar o comando `mkdir Projetos`.

```
mkdir Projetos
```

## **touch** - Cria um arquivo

Você pode usar o comando `touch` para criar um arquivo. Por exemplo, se você estiver no diretório `/home/usuario/Documentos` e quiser criar um arquivo chamado `README.md`, basta executar o comando `touch README.md`.

```
touch README.md
```

## **cat** - Mostra o conteúdo de um arquivo

Você pode usar o comando `cat` para mostrar o conteúdo de um arquivo. Por exemplo, se você estiver no diretório `/home/usuario/Documentos` e quiser mostrar o conteúdo do arquivo `README.md`, basta executar o comando `cat README.md`.

```
cat README.md
```

Esse comando é muito útil para mostrar o conteúdo de arquivos sem abrir um editor de texto.

## **cp** - Cópia um arquivo ou diretório

Você pode usar o comando `cp` para copiar um arquivo ou diretório. Por exemplo, se você estiver no diretório `/home/usuario/Documentos` e quiser copiar o arquivo `README.md` para o diretório `Projetos`, basta executar o comando `cp README.md Projetos`.

```
cp README.md Projetos
```

Caso você queira copiar um diretório, basta adicionar o argumento `-r` (recursivo).

```
cp -r Projetos Projetos2
```

## **mv** - Move um arquivo ou diretório / Renomeia um arquivo ou diretório

Você pode usar o comando `mv` para mover um arquivo ou diretório. Por exemplo, se você estiver no diretório `/home/usuario/Documentos` e quiser mover o arquivo `README.md` para o diretório `Projetos`, basta executar o comando `mv README.md Projetos`.

```
mv README.md Projetos
```

Caso você queira mover um diretório, basta adicionar o argumento `-r` (recursivo).

```
mv -r Projetos Projetos2
```

## **rm** - Remove um arquivo ou diretório

Você pode usar o comando `rm` para remover um arquivo ou diretório. Por exemplo, se você estiver no diretório `/home/usuario/Documentos` e quiser remover o arquivo `README.md`, basta executar o comando `rm README.md`.

```
rm README.md
```

Para remover um diretório, você precisa passar o parâmetro `-r` para o comando `rm`. Por exemplo, se você estiver no diretório `/home/usuario/Documentos` e quiser remover o diretório `Projetos`, basta executar o comando `rm -r Projetos`.

```
rm -r Projetos
```

### ATENÇÃO

Você pode utilizar `rm -rf` para remover um diretório e todos os seus arquivos e subdiretórios, mas tome cuidado ao utilizar esse comando, pois ele não pede confirmação.

## Comandos do sistema

Agora que você já conhece os comandos básicos, vamos ver alguns comandos do sistema.

## **history** - Mostra o histórico de comandos

Você pode usar o comando `history` para mostrar o histórico de comandos. Por exemplo, se você executar o comando `history`, irá mostrar todos os comandos que você executou no terminal.

```
history
```

## `clear` - Limpa a tela do terminal

Você pode usar o comando `clear` para limpar a tela do terminal. Por exemplo, se você executar o comando `clear`, irá limpar a tela do terminal.

```
clear
```

### 💡 DICA

Você pode usar o atalho `Ctrl + L` para limpar a tela do terminal.

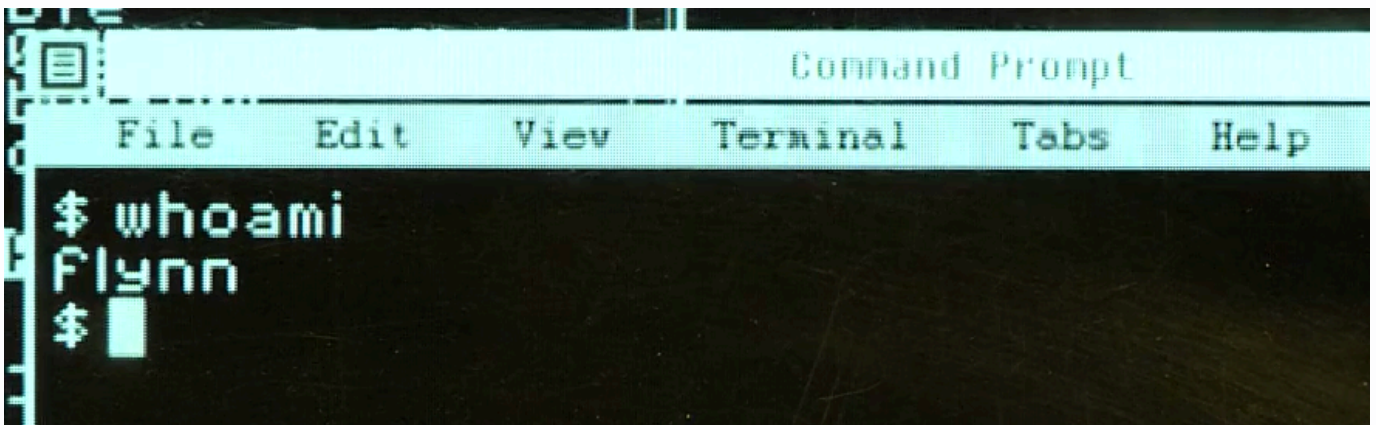
## `whoami` - Mostra o nome do usuário atual

Você pode usar o comando `whoami` para mostrar o nome do usuário atual. Por exemplo, se você executar o comando `whoami`, irá mostrar o nome do usuário atual.

```
whoami
```

### 📘 NOTA

Esse comando aparece no filme Tron: Legacy, quando o personagem Flynn está no terminal do computador do Clu. O personagem Flynn digita o comando `whoami` e o computador do Clu responde com o nome do usuário atual (`Flynn`).



## **which** - Mostra o caminho de um comando

Você pode usar o comando `which` para mostrar o caminho de um comando. Por exemplo, se você executar o comando `which ls`, irá mostrar o caminho do comando `ls`.

```
which ls
```

Caso o não seja retornado nenhum resultado, significa que o comando não está instalado no sistema ou que o comando não está no `PATH`.

## **sudo** - Executa um comando como administrador

Você pode usar o comando `sudo` para executar um comando como administrador. Por exemplo, se você quiser instalar um pacote, basta executar o comando `sudo apt install pacote`.

Geralmente, o comando `sudo` é utilizado para dar privilégios de administrador para um usuário comum executar comandos que necessitam de privilégios de administrador.

```
sudo apt install pacote
```

## **chmod** - Alterar permissões

O comando `chmod` é usado para alterar as permissões de um arquivo ou diretório. Para alterar as permissões de um arquivo, basta passar o caminho do arquivo como parâmetro.

As permissões são representadas por três números, sendo o primeiro número para o dono do arquivo, o segundo número para o grupo do arquivo e o terceiro número para os outros usuários. Cada número representa as permissões de leitura, escrita e execução.

As permissões são números que vão de 0 a 7. O número 0 representa que o usuário não tem permissão de leitura, escrita e execução. O número 1

representa que o usuário tem permissão de execução. O número 2 representa que o usuário tem permissão de escrita. O número 3 representa que o usuário tem permissão de escrita e execução. O número 4 representa que o usuário tem permissão de leitura. O número 5 representa que o usuário tem permissão de leitura e execução. O número 6 representa que o usuário tem permissão de leitura e escrita. O número 7 representa que o usuário tem permissão de leitura, escrita e execução.

Em resumo:

- 0 = Sem permissão
- 1 = **Execução** (x)
- 2 = **Escrita** (w)
- 3 = Escrita e execução (2 + 1) (w + x)
- 4 = **Leitura** (r)
- 5 = Leitura e execução (4 + 1) (r + x)
- 6 = Leitura e escrita (4 + 2) (r + w)
- 7 = Leitura, escrita e execução (4 + 2 + 1) (r + w + x)

```
chmod 777 /home/root/file.txt
```

Na duvida, utilize sites para converter as permissões.

- <https://chmod-calculator.com/>
- <https://www.easyunitconverter.com/chmod-calculator>

## chown - Alterar dono

O comando `chown` é usado para alterar o dono de um arquivo ou diretório. Para alterar o dono de um arquivo, basta passar o caminho do arquivo como parâmetro.

```
chown root /home/root/file.txt
```

Além disso, é possível alterar o dono e o grupo de um arquivo ou diretório, basta adicionar o grupo após o dono.

```
chown root:root /home/root/file.txt
```

## df - Exibir espaço em disco

O comando `df` é usado para exibir o espaço em disco. Para exibir o espaço em disco ou um diretório, basta passar o caminho do diretório como parâmetro.

```
df
```

```
df /home/root
```

## du - Exibir tamanho de arquivo ou diretório

O comando `du` é usado para exibir o tamanho de um arquivo ou diretório.

```
du
```

Sem parâmetros, o comando `du` exibe o tamanho de todos os arquivos e diretórios do diretório atual. Por isso, é recomendado passar o caminho do diretório como parâmetro além da opção `-h` para exibir o tamanho em formato legível e `--max-depth=1` para exibir o tamanho de apenas um nível de diretórios.

```
du -h --max-depth=1 / | sort -h
```



### DICA

Existem outros pacotes que podem ser instalados para exibir o tamanho de arquivos e diretórios de forma mais amigável, como o `ncdu` ou o `dust`



## **find** - Buscar arquivo(s)

O comando **find** é usado para buscar um arquivo ou diretório. Para buscar um arquivo, basta passar o caminho do arquivo como parâmetro.

```
find / --name file.txt
```

# Manipulação de processos

## **ps** - Exibir processos

O comando **ps** é usado para exibir os processos. Para exibir os processos, basta passar o caminho do arquivo como parâmetro.

```
ps
```

## **top** - Exibir processos em tempo real

O comando **top** é usado para exibir os processos em tempo real. Para exibir os processos em tempo real, basta passar o caminho do arquivo como parâmetro.

```
top
```

Em alguns casos, é necessário passar a opção **-u** para exibir os processos de um usuário específico.

```
top -u root
```



### DICA

Existem outras alternativas para exibir os processos, **htop**, ele é um pacote instalado a parte, mas tem uma interface mais agradável.

## `kill` - Finalizar processo

O comando `kill` é usado para finalizar um processo. Para finalizar um processo, basta passar o PID do processo como parâmetro.

```
kill 1234
```

Existem algumas opções para finalizar um processo, como `-9` que finaliza o processo imediatamente.

```
kill -9 1234
```

## Gerenciamento de serviços

### AVISO

Os comandos abaixo são específicos para o sistema operacional Linux, verifique qual o gerenciador de serviços do seu sistema operacional.

Os principais gerenciadores de serviços são:

- `systemd`
- `init`

## `systemctl` - Gerenciador de serviços do `systemd`

O comando `systemctl` é usado para gerenciar os serviços do `systemd`.

Para listar os serviços, basta passar a opção `list-units` como parâmetro.

```
systemctl list-units
```

Para iniciar um serviço, basta passar a opção `start` e o nome do serviço como parâmetro.

```
systemctl start ssh
```

Para parar um serviço, basta passar a opção `stop` e o nome do serviço como parâmetro.

```
systemctl stop ssh
```

Para reiniciar um serviço, basta passar a opção `restart` e o nome do serviço como parâmetro.

```
systemctl restart ssh
```

Para fazer com que um serviço inicie automaticamente, basta passar a opção `enable` e o nome do serviço como parâmetro.

```
systemctl enable ssh
```

Para desabilitar um serviço, basta passar a opção `disable` e o nome do serviço como parâmetro.

```
systemctl disable ssh
```

## `service` - Gerenciador de serviços do `init`

O comando `service` é usado para gerenciar os serviços do `init`.

Para listar os serviços, basta passar a opção `--status-all` como parâmetro.

```
service --status-all  
# or  
/etc/init.d --status-all
```

Para iniciar um serviço, basta passar o nome do serviço como parâmetro.

```
service ssh start  
# or  
/etc/init.d ssh start
```

Para parar um serviço, basta passar o nome do serviço como parâmetro.

```
service ssh stop  
# or  
/etc/init.d ssh stop
```

Para reiniciar um serviço, basta passar o nome do serviço como parâmetro.

```
service ssh restart  
# or  
/etc/init.d ssh restart
```

Para fazer com que um serviço inicie automaticamente, basta passar a opção `--level` e o nome do serviço como parâmetro.

```
service ssh --level 2345  
# or  
/etc/init.d ssh --level 2345
```

# Jenkins



LINK

LINK: <https://www.jenkins.io/>

O Jenkins é uma ferramenta de automação de tarefas, que pode ser utilizada para automatizar o processo de build e deploy de aplicações. O Jenkins é uma ferramenta open source, que pode ser instalada em qualquer servidor, e que pode ser utilizada para automatizar qualquer processo que possa ser executado em um servidor.

## Instalação

Para instalar o Jenkins, basta acessar o **site oficial** e seguir as instruções de instalação para o seu sistema operacional.

## Configuração

Para configurar o Jenkins, basta acessar o endereço do servidor onde o Jenkins foi instalado, e seguir as instruções de configuração.

No primeiro acesso, o Jenkins solicitará a criação de um usuário administrador, que será utilizado para acessar o sistema. Sua senha inicial será gerada automaticamente, e será solicitada no primeiro acesso. A senha é salva em um arquivo chamado `initialAdminPassword` dentro da pasta `secrets` do Jenkins.

## Plugins

O Jenkins possui uma grande quantidade de plugins, que podem ser utilizados para automatizar tarefas específicas. Para instalar um plugin, basta acessar a página de gerenciamento de plugins, e pesquisar pelo

plugin desejado. Para instalar um plugin, basta clicar no botão `Install without restart`.

#### ATENÇÃO

Os plugins ajudam muito no dia a dia, mas com o tempo, isso pode impactar no servidor, tanto em performance, quanto na manutenção. Por isso, é importante avaliar se o plugin é realmente necessário, e se ele não pode ser substituído por uma configuração simples.

## Configurações gerais

Para o melhor funcionamento do Jenkins, deve ser verificado as configurações gerais do sistema. Esses ajustes tem que ser feitos atendendo as necessidades de cada empresa.

## Estrutura

Segue abaixo os principais itens da estrutura do Jenkins:

### Jobs

O Jenkins é composto por jobs, que são as tarefas que serão executadas. Para criar um job, basta clicar no botão `New Item`, e preencher as informações do job.

Ao clicar em novo, selecione o tipo de job que deseja criar e seguir com as configurações que vão ser solicitadas. Os tipos de jobs mais comuns são:

#### Pipeline (Recomendado)

O Jenkins possui um tipo de job chamado `Pipeline`, que permite a criação de jobs que executam scripts. É a forma mais comum de se utilizar o Jenkins, e é a forma mais utilizada para automatizar o processo de build e deploy de aplicações. No projeto, geralmente é utilizado o Jenkinsfile, que é um

arquivo de configuração do Jenkins, que é versionado no repositório do projeto, e que é utilizado para configurar o job.

## Freestyle

O Jenkins também possui um tipo de job chamado **Freestyle**, que permite a criação de jobs que executam scripts. É a forma mais simples e de forma visual, o que pode limitar a configuração do job.

## Outros tipos de jobs

Dependendo dos plugins instalados, o Jenkins pode ter outros tipos de jobs, como por exemplo, o tipo **Maven**, que permite a criação de jobs que executam o Maven.

## Build

Cada job possui um número de build, que é um número sequencial que é incrementado a cada execução do job. O número da build é utilizado para identificar cada execução do job.

Os builds são as execuções do job, e tem as informações de quem, quando e o que foi executado. Você pode configurar quantos builds serão mantidos no histórico, e também se os builds serão mantidos mesmo que o job seja excluído.

## Artifacts

Os artifacts são os arquivos gerados durante a execução do job. Os artifacts são salvos no disco do servidor, e podem ser acessados através do Jenkins, ou diretamente no servidor. Então quando um job é executado, você especifica quais arquivos serão salvos como artifacts, senão nenhum arquivo será salvo.

# Configurações recomendadas

## Credenciais

As credenciais são as informações de acesso a outros sistemas, como por exemplo, o acesso ao repositório de código, ou o acesso ao servidor de deploy. As credenciais são utilizadas para acessar esses sistemas, e são armazenadas no Jenkins.

Para criar uma credencial, basta acessar a página de gerenciamento de credenciais, e clicar no botão `Add Credentials`.

Isso é uma boa prática, pois evita que as credenciais fiquem expostas no código além de facilitar a manutenção e padronização.

## Controle de acesso

O Jenkins possui um controle de acesso, que permite definir quem pode acessar o sistema, e quais jobs podem ser executados por cada usuário. Para configurar o controle de acesso, basta acessar a página de gerenciamento de usuários, e clicar no botão `Add User`.

## Configuração global de ferramentas

Em alguns casos você vai precisar instalar ferramentas no servidor do Jenkins, e depois ir em `Global Tool Configuration` e configurar o caminho para a ferramenta. Por exemplo, se você precisa instalar o Maven no servidor, você vai precisar configurar o caminho para o Maven. Isso torna o processo de configuração mais fácil, pois você não precisa configurar o caminho para a ferramenta em cada job.



# PostgreSQL



LINK

LINK: <https://www.postgresql.org/>

PostgreSQL é um sistema de gerenciamento de banco de dados objeto relacional (SGBD), desenvolvido como projeto de código aberto. Ele suporta a maioria dos recursos SQL e oferece recursos adicionais como tabelas espaciais e JSON.

## Instalação

Cada sistema operacional possui uma forma diferente de instalar o PostgreSQL, para instalar no Windows, basta acessar o site <https://www.postgresql.org/download/windows/> e baixar o instalador.

## Configuração

Quando é feita a instalação do PostgreSQL, é recomendado que seja modificado os arquivos de configuração, para que o banco de dados seja acessado de qualquer lugar.



ATENÇÃO

Não é recomendado deixar o banco de dados aberto para qualquer IP, apenas para o IP da sua máquina ou da sua rede.

## Alterando o arquivo `pg_hba.conf`

O arquivo `pg_hba.conf` é responsável por definir quem pode acessar o banco de dados, por padrão, o arquivo é criado com o seguinte conteúdo:

```
pg_hba.conf
```

```
# TYPE DATABASE USER ADDRESS METHOD
host all all 0.0.0.0/0 trust
```

## Alterando o arquivo postgresql.conf

O arquivo **postgresql.conf** é responsável por definir as configurações do banco de dados, por padrão, o arquivo é criado com o seguinte conteúdo:

```
postgresql.conf
```

```
# - Connection Settings -
listen_addresses = '*' # what IP address(es) to listen on;
port = 5432 # (change requires restart)
```

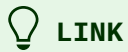
## Criando um usuário

Para criar um usuário, basta acessar o terminal do PostgreSQL e executar o comando abaixo:

```
CREATE USER usuario WITH PASSWORD 'senha';
```

Caso preferir, é possível utilizar o pgAdmin para criar o usuário, basta acessar o servidor e clicar com o botão direito em **Login/Group Roles** e selecionar a opção **Create > Login/Group Role**.

# Docker



LINK

LINK: <https://www.docker.com/>

## Sumário

- Instalação
- Diferença entre máquina virtual e container
- Imagens
- Containers
- Volumes
- Comandos
- Docker Compose

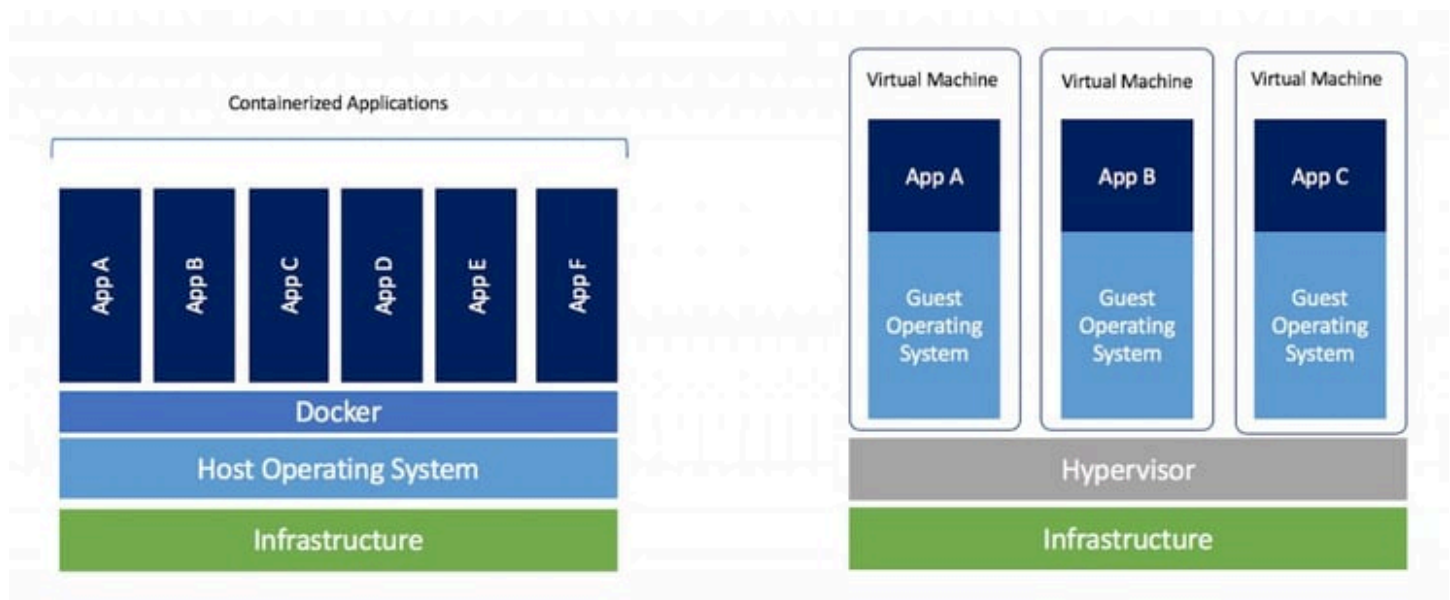
O Docker é uma ferramenta que permite a criação e o gerenciamento de containers, que são ambientes isolados que possuem suas próprias bibliotecas, arquivos de configuração e sistema operacional. Esses containers são utilizados para executar aplicações de forma isolada, sem a necessidade de instalar o software em uma máquina.

## Instalação

Cada sistema operacional possui uma forma diferente de instalar o Docker, para instalar no Windows, basta acessar o site <https://docs.docker.com/docker-for-windows/install/> e baixar o instalador.

## Diferença entre máquina virtual e container

Abaixo é possível ver a diferença entre uma máquina virtual e um container:



Fonte: <https://www.sdxcentral.com/cloud/containers/definitions/containers-vs-vms/>

Enquanto uma máquina virtual possui um sistema operacional completo, um container compartilha o sistema operacional da máquina hospedeira, isso faz com que o container seja mais leve e rápido. Além disso, o container é isolado da máquina hospedeira, o que faz com que ele não tenha acesso aos arquivos da máquina hospedeira, a menos que seja especificado.

## Imagens

As imagens são arquivos que contêm o sistema operacional e as bibliotecas necessárias para executar uma aplicação, isso quer dizer que não é necessário instalar o sistema operacional e as bibliotecas em uma máquina, basta baixar a imagem e executar o container.

As imagens podem ser salvas em um repositório, que é um local onde as imagens são armazenadas, o Docker possui um repositório público chamado Docker Hub, que pode ser acessado em <https://hub.docker.com/>.

## Containers

Os containers são instâncias de uma imagem, eles são criados a partir de uma imagem e possuem um sistema operacional completo, bibliotecas e arquivos de configuração. Os containers são isolados da máquina hospedeira, o que faz com que eles não tenham acesso aos arquivos da máquina hospedeira, a menos que seja especificado.

Por padrão os container rodam e morrem quando o script é finalizado, para que o container fique rodando é necessário especificar que ele deve ficar rodando.

Você pode ter vários containers rodando ao mesmo tempo, cada um com uma aplicação diferente ou com a mesma aplicação.

## Volumes

Por padrão os arquivos criados dentro de um container são perdidos quando o container é finalizado, para que os arquivos criados dentro do container sejam mantidos é necessário criar um volume, que é um local onde os arquivos criados dentro do container são armazenados.

Podemos criar dois tipos de volumes:

- **Volumes locais:** você vai fazer o mapeamento de um diretório da máquina hospedeira para um diretório do container, assim os arquivos criados dentro do container serão armazenados no diretório da máquina hospedeira e vice-versa.
- **Volumes do Docker:** o Docker vai criar um diretório na máquina hospedeira e fazer o mapeamento com o diretório do container, assim os arquivos criados dentro do container serão armazenados no diretório do Docker e vice-versa. Essa forma você deixa com que o Docker salve os arquivos em um local específico, sem precisar especificar o diretório.

## Comandos

Os comandos do Docker são executados no terminal, para executar um comando é necessário digitar `docker` seguido do comando desejado.

## docker run

O comando `docker run` é utilizado para criar um container a partir de uma imagem, ele possui as seguintes opções:

- `-d`: Executa o container em background, ou seja, o container fica rodando mesmo após o script ser finalizado.
- `-p`: Mapeia uma porta do container para uma porta da máquina hospedeira, por exemplo, se você mapear a porta 80 do container para a porta 8080 da máquina hospedeira, ao acessar a porta 8080 da máquina hospedeira, você estará acessando a porta 80 do container.
- `-v`: Mapeia um diretório do container para um diretório da máquina hospedeira, por exemplo, se você mapear o diretório `/var/www` do container para o diretório `C:\www` da máquina hospedeira, ao acessar o diretório `C:\www` da máquina hospedeira, você estará acessando o diretório `/var/www` do container.
- `--name`: Define o nome do container.
- `--rm`: Remove o container após o script ser finalizado.

## docker start

O comando `docker start` é utilizado para iniciar um container que está parado, ele possui as seguintes opções:

- `-a`: Exibe o log do container.
- `-i`: Permite interagir com o container.

## docker stop

O comando `docker stop` é utilizado para parar um container que está rodando, ele possui as seguintes opções:

- `-t`: Tempo para parar o container, por padrão é 10 segundos.

## docker ps

O comando `docker ps` é utilizado para listar os containers que estão rodando, ele possui as seguintes opções:

- `-a`: Lista todos os containers, inclusive os que estão parados.
- `-q`: Lista apenas o ID dos containers.

## docker rm

O comando `docker rm` é utilizado para remover um container, ele possui as seguintes opções:

- `-f`: Remove o container mesmo que ele esteja rodando.

## docker rmi

O comando `docker rmi` é utilizado para remover uma imagem, ele possui as seguintes opções:

- `-f`: Remove a imagem mesmo que ela esteja sendo utilizada por algum container.

## docker exec

O comando `docker exec` é utilizado para executar um comando em um container que está rodando, ele possui as seguintes opções:

- `-it`: Permite interagir com o container.

## docker logs

O comando `docker logs` é utilizado para exibir o log de um container, ele possui as seguintes opções:

- `-f`: Exibe o log em tempo real.

## docker pull

O comando `docker pull` é utilizado para baixar uma imagem do repositório.

### ! OBSERVAÇÃO

Por padrão quando você executa o comando `docker run` ele baixa a imagem do repositório caso ela não exista na máquina.

## Exemplos

Para criar um container a partir de uma imagem, execute o comando `docker run` seguido do nome da imagem, por exemplo, para criar um container a partir da imagem `nginx`, execute o comando `docker run nginx`.

Se não existir nenhuma imagem com o nome `nginx` na máquina, o Docker irá baixar a imagem do repositório. Por padrão, as imagens são baixadas do `hub.docker.com`, na versão `latest`. Caso queira especificar a versão da imagem, basta adicionar o número da versão após o nome da imagem, por exemplo, para baixar a imagem `nginx` na versão `1.15.8`, execute o comando `docker run nginx:1.15.8`.

## Dockerfile

O Dockerfile é um arquivo de configuração que possui as instruções para criar uma imagem, ele é utilizado para automatizar a criação de imagens.

```
FROM ubuntu:18.04
RUN apt-get update && apt-get install -y nginx
CMD ["nginx", "-g", "daemon off;"]
```

## Docker Compose

O Docker Compose é uma ferramenta que permite criar e gerenciar vários containers ao mesmo tempo, ele é utilizado para criar ambientes de desenvolvimento, testes e produção.

O Docker Compose é executado através de um arquivo de configuração chamado `docker-compose.yml`, que possui as configurações dos containers.



# Comandos

Os comandos do Docker Compose são executados no terminal, para executar um comando é necessário digitar `docker-compose` seguido do comando desejado.

## `docker-compose up`

O comando `docker-compose up` é utilizado para criar e iniciar os containers, ele possui as seguintes opções:

- `-d`: Executa os containers em background, ou seja, os containers ficam rodando mesmo após o script ser finalizado.
- `--build`: Recria os containers, ou seja, ele remove os containers e as imagens e cria novamente.

## `docker-compose start`

O comando `docker-compose start` é utilizado para iniciar os containers que estão parados.

## `docker-compose stop`

O comando `docker-compose stop` é utilizado para parar os containers que estão rodando, ele possui as seguintes opções:

- `-t`: Tempo para parar os containers, por padrão é 10 segundos.

## `docker-compose restart`

O comando `docker-compose restart` é utilizado para reiniciar os containers.

# Exemplo

## `docker-compose.yml`

Para criar um ambiente de desenvolvimento com o Docker Compose, crie um arquivo chamado `docker-compose.yml` com o seguinte conteúdo:

`docker-compose.yml`

```
version: '3.7'

services:
  web:
    image: nginx:1.15.8
    ports:
      - 8080:80
    volumes:
      - ./www:/var/www
```

# AWS EC2

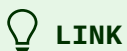


LINK

LINK: <https://aws.amazon.com/pt/ec2/>

Elastic Compute Cloud (EC2) é um serviço web que fornece capacidade de computação na nuvem de forma segura e escalável. É um serviço de computação em nuvem que permite que você aumente ou diminua a capacidade de computação de acordo com a demanda. O EC2 permite que você tenha controle total sobre a configuração de hardware do servidor, permitindo que você instale e configure o software que desejar, configure grupos de segurança e crie imagens de servidor que contenham o software e as configurações que você deseja.

Uma *alternativa* ao EC2 é o Lightsail, que é um serviço de computação em nuvem da Amazon que fornece servidores virtuais pré-configurados e gerenciados. O Lightsail é uma alternativa mais simples e mais barata ao EC2, mas não oferece a mesma flexibilidade e controle que o EC2.



LINK

LINK: <https://aws.amazon.com/pt/lightsail/>



ATENÇÃO

- O Lightsail não é uma alternativa ao EC2, mas sim uma alternativa mais simples e mais barata ao EC2.
- Não é possível migrar um servidor EC2 para o Lightsail e vice-versa.
- Enquanto o Lightsail tem um preço fixo, o EC2 tem um preço variável dependendo da quantidade de recursos que você está usando.

## Instâncias EC2

As instâncias EC2 são máquinas virtuais (VMs) que você pode usar para executar aplicativos na nuvem. Cada instância EC2 inclui um sistema operacional (SO), aplicativos de software pré-instalados, uma quantidade de armazenamento e uma quantidade de memória que você pode especificar. Você pode usar uma instância EC2 para executar aplicativos de banco de dados, aplicativos web, aplicativos de servidor de arquivos e muito mais.

## Tipos de instâncias EC2

Existem vários tipos de instâncias EC2 disponíveis, cada um com diferentes recursos e preços. Cada instancia EC2 tem um tipo de processador, uma quantidade de memória e limites de banda de rede. Recentemente, começou a ser possível escolher tipo de instancias que rodam em processadores ARM, que são mais baratos e mais eficientes do que os processadores Intel x86, mas sua aplicação tem que ser compatível com processadores ARM.

## Volumes EC2

Os volumes EC2 são discos de armazenamento que você pode anexar a uma instância EC2. Os volumes EC2 podem ser usados para armazenar dados de aplicativos e sistemas operacionais, ou para servir como discos de inicialização para suas instâncias. Os volumes EC2 podem ser anexados a uma instância EC2 quando a instância é iniciada, ou podem ser anexados a uma instância EC2 que já está em execução. Os volumes EC2 podem ser anexados a várias instâncias EC2 ao mesmo tempo.

### ⓘ CUSTO

O custo de um volume é cobrado por hora e por GB alocado, mesmo que o volume não esteja sendo usado ou a maquina esteja desligada.

## Snapshots de volumes EC2

Os snapshots de volumes EC2 são cópias de segurança de volumes EC2. Você pode criar um snapshot de um volume EC2 a qualquer momento, e os snapshots

de volume EC2 são criados incrementalmente, o que significa que apenas as alterações desde o último snapshot são armazenadas. Os snapshots de volume EC2 são armazenados em S3 e podem ser copiados para outras regiões da AWS.

## Network Interfaces EC2

As instâncias EC2 possuem uma ou mais interfaces de rede, que são usadas para se comunicar com a internet e com outras instâncias EC2. As interfaces de rede EC2 podem ser configuradas para usar endereços IP públicos ou privados, e podem ser configuradas para usar endereços IP estáticos ou dinâmicos.

## Security Groups EC2

Os security groups EC2 são grupos de regras de firewall que podem ser aplicadas a uma ou mais interfaces de rede EC2. Os security groups EC2 podem ser usados para permitir ou bloquear o tráfego de entrada e saída para uma instância EC2. Os security groups EC2 podem ser aplicados a uma instância EC2 quando a instância é iniciada, ou podem ser aplicados a uma instância EC2 que já está em execução.

## Regiões EC2

As regiões EC2 são locais geográficos onde você pode executar suas instâncias EC2. Cada região EC2 é composta por pelo menos uma zona de disponibilidade. As regiões EC2 são usadas para fornecer redundância e disponibilidade, pois as instâncias EC2 em uma região EC2 podem falhar, mas as instâncias EC2 em outras regiões EC2 não serão afetadas.

## Extra

Os serviços da AWS são extremamente flexíveis e configuráveis, e você pode criar uma infraestrutura de nuvem altamente personalizada para atender às suas necessidades. O que foi apresentado aqui é apenas uma introdução.



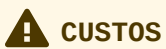
# AWS S3



LINK

LINK: <https://aws.amazon.com/pt/s3/>

Simple Storage Service (S3) é um serviço de armazenamento de objetos na nuvem da Amazon Web Services (AWS). O S3 armazena qualquer tipo de arquivo, desde imagens e vídeos até documentos e dados de aplicativos. O S3 é um serviço altamente escalável, seguro e confiável, projetado para fazer com que os desenvolvedores de aplicativos e as empresas armazenem e recuperem qualquer quantidade de dados de qualquer lugar na web.



CUSTOS

Você não paga nada para enviar os dados para o S3, mas você paga por armazenamento e transferência de dados. Para mais informações, consulte <https://aws.amazon.com/pt/s3/pricing/>.

## Bucket

Um bucket é um espaço de armazenamento na AWS. É um container para armazenar objetos. Cada bucket é único e possui um nome exclusivo. O nome do bucket é usado para identificar o bucket em todas as solicitações de API e na URL do bucket. O nome do bucket deve ser exclusivo em todo o AWS.

## Objeto

Um objeto é um arquivo que você armazena no S3. Um objeto consiste em dados e metadados. Os dados são o conteúdo do objeto. Os metadados são pares de nome-valor que descrevem os dados. Os metadados não afetam o conteúdo do objeto. Os metadados são armazenados separadamente do conteúdo do objeto.

# Região

A região é uma localização geográfica que contém pelo menos um data center. Cada região é isolada geograficamente da outra. As regiões são independentes entre si, mas compartilham a mesma infraestrutura global. Por exemplo, se você armazenar um objeto em uma região, você pode acessá-lo de qualquer outra região.



# AWS Route 53



LINK

**LINK:** <https://aws.amazon.com/pt/route53/>

Route 53 é um serviço de DNS da Amazon Web Services (AWS). O Route 53 é um serviço altamente escalável, seguro e confiável, projetado para fazer com que os desenvolvedores de aplicativos e as empresas gerenciem o tráfego de domínio da web e a resolução de nomes para seus recursos de aplicativos.

## Domínio

Você pode registrar um domínio com o Route 53 ou transferir um domínio existente para o Route 53. Em ambos os casos você pode gerenciar domínios e subdomínios, gerenciando toda a parte de DNS do domínio.

# arara.gtech.site

## PROJETO

Projeto privado de Lucas Baccan:

<https://github.com/lucasbaccan/arara.gtech.site>

## Como clonar

```
git clone https://github.com/lucasbaccan/arara.gtech.site.git
```

## Requisitos

Certifique-se que o servidor tenha os seguintes programas instalados:

- **Git**
- **Docker**
- **Docker-Compose**

## Containers

Esse projeto sobe alguns containers para o funcionamento do banco de dados. São eles:

- **PostgreSQL**
- **PgAdmin** (opcional)
- **Grafana**
- **Prometheus**
- **Postgres Exporter**
- **Node Exporter**

A imagem do PostgreSQL é a oficial do Docker Hub, na versão `postgres:11.16`, o que significa que a versão do Postgres é a 11.16 e o SO é Debian. Já o PgAdmin é a imagem `dpage/pgadmin4:6.13`, que é a versão 6.13 do PgAdmin4, que está rodando em um SO Alpine.

## postgres\_db

Esse container é o banco de dados em si, onde todos os bancos de dados vão rodar. Todos os dados são salvos em um volume, que é o `postgres-db-data:/var/lib/postgresql/data`, então caso o container seja removido, os dados não serão perdidos. Para configurar, deve ser editado o arquivo `env.sh`, que contém as seguintes variáveis:

- `POSTGRES_USER`: usuário do banco de dados
- `POSTGRES_PASSWORD`: senha do usuário do banco de dados
- `POSTGRES_HOST`: host do banco de dados (deve ser `postgres_db`)
- `POSTGRES_PORT`: porta do banco de dados (deve ser `5432`)

## postgres\_cron

Esse container utiliza a mesma imagem do `postgres_db`, mas é utilizado para rodar scripts que precisam dos binários do postgres, mas que não precisa ter o banco rodando. Então a rotina de backup, por exemplo, é rodada nesse container. Todos os dados são salvos em um volume do tipo *bind*, que é o `./backups/:/backups/`, que faz com que os arquivos gerados pelo container sejam salvos na pasta `backups` do host. Para configurar, deve ser editado o arquivo `env.sh`, que contém as seguintes variáveis:

- `POSTGRES_USER`: usuário do banco de dados
- `POSTGRES_PASSWORD`: senha do usuário do banco de dados
- `POSTGRES_HOST`: host do banco de dados (deve ser `postgres_db`)
- `POSTGRES_PORT`: porta do banco de dados (deve ser `5432`)
- `S3_BUCKET_BACKUPS`: bucket do S3 onde os backups serão salvos
- `DIRETORIO_SAIDA_BACKUPS`: diretório onde os backups serão salvos no container
- `SSH_PASSWORD`: senha do usuário `root` do container `postgres_cron`

## Rotinas

Todas as rotinas estão dentro da pasta `crontab` sendo o arquivo `crontab` o arquivo de configuração do cron. As rotinas são copiadas para o container, basta colocar em `crontab/jobs`.

- `/crontab/jobs/backup_todos_bancos.sh`: faz o backup de todos os bancos de dados do postgres.
  - Pode receber um parâmetro `true` ou `false`, que indica se o backup deve ser enviado para o S3 ou não. Por padrão não é enviado.
- `/crontab/jobs/drop_loop.sh`: verifica as conexões indevidas no banco de dados e mata elas.

## pgadmin (opcional)

Este container é opcional, e serve para facilitar a visualização dos bancos de dados. Ele é acessado através do endereço `http://IP-SERVIDOR:8080`, e o usuário e senha padrão são `admin@admin.com` e `admin`, respectivamente.

## grafana

Esse container é o responsável por exibir os gráficos de métricas do Prometheus. Ele é acessado através do endereço `http://IP-SERVIDOR:3000`, e o usuário e senha padrão são `admin` e `admin`, respectivamente. No primeiro login da para alterar a senha.

## prometheus

Esse container é o responsável por coletar as métricas dos bancos de dados e dos containers. O padrão seria exportar a porta 9090, mas ela só é visível para os containers.

## postgres\_exporter

Esse container é o responsável por exportar as métricas do banco de dados para o Prometheus. O padrão seria exportar a porta 9187, mas ela só é visível para os containers.

## node\_exporter (opcional)

Esse container é o responsável por exportar as métricas do servidor para o Prometheus. O padrão seria exportar a porta 9100, mas ela só é visível para os containers.

## Como instalar/iniciar o servidor

Após o clone do projeto, vai existir alguns arquivos `.sh`, basta executar eles:

- Configuração
  - `server_install.sh`: script para instalar o servidor, é resultado dos comandos utilizados na criação do servidor utilizando **AWS Linux**.
  - `env.sh`: arquivo de configuração do ambiente do servidor.
  - `ssh_key_generate.sh`: script para gerar a chave SSH para o container `postgres_cron`.
  - `criar_ambiente.sh`: script para criar o ambiente do servidor, utilizado na primeira vez.
- Status do servidor
  - `start.sh`: script para iniciar o servidor (pode passar o parâmetro `true` para ver o log).
  - `stop.sh`: script para parar o servidor.
  - `restart.sh`: script para reiniciar o servidor.
- Restaurar backup
  - `restore.sh`: script para restaurar os backups do S3.
  - `restore_stop.sh`: script para parar o script de restauração.
  - `restore_continue.sh`: script para continuar o script de restauração.
- Outros
  - `convert_unix.sh`: script para converter os arquivos da pasta atual para o formato UNIX, caso estejam no formato Windows. Requer o

pacote `dos2unix` instalado.