


Estudos DevOps

Criando e administrando repositórios com o GIT e o Gitlab

Antes de dar início a nossa jornada de utilização do git (sistema de controle de versão), precisamos configurar o nosso **git config**.

 O comando "**git config**" é usado para configurar as opções do Git em um determinado nível. Isso pode incluir configurações de usuário, configurações de repositório local e configurações globais do Git.

 **Vamos lá!**

No terminal digite:

```
git config -l --global
```

i O comando **"git config -l --global"** é usado para listar todas as configurações globais do Git em seu sistema. Ele exibe uma lista de pares de chave-valor para cada configuração, incluindo informações como nome de usuário, endereço de e-mail, editor padrão e outros.

Por exemplo, se você digitasse **"git config -l --global"** no terminal, veria algo semelhante a isso:

```
user.name=Seu Nome  
user.email=seuemail@exemplo.com  
core.editor=vim  
color.ui=true
```

Exemplo do meu terminal:

```
fz983@CTS08539127 MINGW64 /  
$ git config -l --global  
user.email=anderson.silva01@external.t-systems.com  
user.name=fy1154
```

Agora, vamos alterar o config para colocar nossos dados no Git. Primeiro, para alterar o usuário, execute o comando:

```
git config --global user.name "Seu Nome"  
git config --local user.email "seuemail@exemplo.com"
```

i O comando `"git config --global user.name"` é usado para definir o nome de usuário global do Git no seu sistema.

i O comando `"git config --local user.email"` é usado para definir o endereço de e-mail do usuário para um repositório Git específico.

Sobre o .gitignore

i O arquivo `".gitignore"` é um arquivo de configuração do Git que especifica arquivos e diretórios que o Git deve ignorar ao rastrear mudanças em um repositório Git. Isso permite que você exclua arquivos temporários, arquivos de compilação, arquivos gerados automaticamente e outros arquivos que não devem ser controlados pelo Git.

Ao adicionar um arquivo ou diretório ao arquivo `".gitignore"`, o Git não rastreará mudanças nesses arquivos e diretórios, o que significa que eles não aparecerão em `"git status"` ou serão incluídos em `"git add"`. Isso ajuda a manter o repositório limpo e organizado, evitando a inclusão de arquivos desnecessários ou sensíveis.

O arquivo `".gitignore"` pode ser adicionado em qualquer diretório do repositório e pode ser especificado para ignorar arquivos ou diretórios com base em padrões de nome de arquivo, expressões regulares ou outras regras de correspondência. É uma prática recomendada usar o arquivo `".gitignore"` em todos os projetos Git para evitar que arquivos indesejados sejam adicionados ao repositório.

Inicializando um novo repositório: git init

O comando `"git init"` é usado para inicializar um novo repositório Git em um diretório existente. Quando você executa esse comando em um diretório, o Git cria uma nova subpasta oculta chamada `".git"` que contém todos os arquivos necessários para o controle de versão.

Para usar o comando `"git init"`, você precisa primeiro navegar até o diretório em que deseja iniciar o repositório Git e executar o comando no terminal. Por exemplo, se você

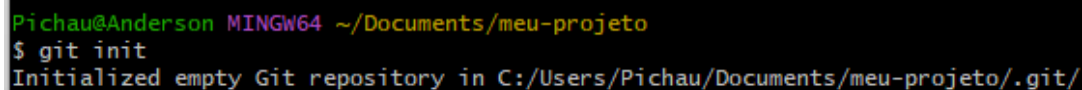
quiser iniciar um repositório Git em um diretório chamado "meu-projeto", você pode fazer o seguinte:

1. Navegue para o diretório "meu-projeto":

```
cd meu-projeto
```

2. Execute o comando "git init":

```
git init
```

A terminal window with a black background and green text. The prompt is 'Pichau@Anderson MINGW64 ~/Documents/meu-projeto'. The command '\$ git init' has been entered, and the output is 'Initialized empty Git repository in C:/Users/Pichau/Documents/meu-projeto/.git/'.

```
Pichau@Anderson MINGW64 ~/Documents/meu-projeto
$ git init
Initialized empty Git repository in C:/Users/Pichau/Documents/meu-projeto/.git/
```

Após executar esse comando, você terá um repositório Git vazio em "meu-projeto" e poderá começar a adicionar arquivos e fazer o controle de versão. Você pode usar outros comandos Git, como "git add" e "git commit", para adicionar arquivos e confirmar as alterações em seu repositório Git.

Clonando um repositório existente: git clone

O comando "git clone" é usado para clonar um repositório Git existente de um servidor remoto para um diretório local. Esse comando é usado com frequência para iniciar um novo projeto ou colaborar em um projeto existente.

Para clonar um repositório Git, você precisa primeiro obter a URL do repositório remoto que deseja clonar. Em seguida, você pode usar o comando "git clone" seguido da URL do repositório para clonar o repositório Git para o seu diretório local.

Para esse tutorial, vou clonar um repositório de teste criado chamado "meuprojeto"

```
git clone git://git.exemplo.com/meuprojeto.git
```

Salvando alterações no repositório: git add e git commit

Depois de criar ou modificar um arquivo em seu diretório de trabalho Git, você precisa adicioná-lo ao repositório e confirmar as alterações. Isso envolve o uso de dois comandos Git principais: "git add" e "git commit".

O comando "git add" é usado para adicionar um arquivo ao índice do Git, que é uma área temporária onde você pode organizar as mudanças antes de confirmá-las. Você pode adicionar um arquivo específico usando o comando "git add <nome_do_arquivo>", ou adicionar todos os arquivos alterados ou novos no diretório atual e seus subdiretórios usando "git add .".

Depois de adicionar os arquivos ao índice, você precisa confirmar as alterações com o comando "git commit". O comando "git commit" cria um registro permanente de suas mudanças no repositório. É importante incluir uma mensagem de commit significativa que descreva as mudanças que você fez. Você pode adicionar uma mensagem de commit usando o comando "git commit -m 'mensagem_do_commit'".

Juntos, esses comandos permitem que você adicione, organize e confirme as mudanças em seu repositório Git. Por exemplo, se você tivesse feito mudanças em um arquivo chamado "meuarquivo.txt" e quisesse confirmá-las em seu repositório Git, você poderia usar os seguintes comandos:

```
git add meuarquivo.txt
git commit -m "Adicionando conteúdo ao arquivo meuarquivo.txt"
```

Praticando!

Realizado o clone do repositório de teste:

<https://gitlab.com/aprendizado6532113/meuprojeto.git>

Vamos inicializar o Git

```
git init
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ git init
Reinitialized existing Git repository in C:/Users/Pichau/Documents/meuprojeto/.git/
```

Agora com o git inicializado, vou criar alguns arquivos, e depois fazer o commit deles.

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ nano index.html

Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ cat index.html
<!DOCTYPE html>
<html>
<body>

<h1>Primeiros passos com o GIT</h1>
<p>Nesse exemplo você viu como utilizar a ferramenta de versionamento GIT! </p>

</body>
</html>

Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ .....
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ cat texto.txt
texto de exemplo

Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ .....
```

Vamos agora salvar o que fizemos com o git add e git commit

```
git add index.html
git commit -m "pagina simples que exibe um título e um paragrafo"
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ git commit -m "pagina simples que exibe um título e um paragrafo"
[main 6584a7e] pagina simples que exibe um título e um paragrafo
1 file changed, 9 insertions(+)
create mode 100644 index.html
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ git commit -m "texto"
[main 429e99e] texto
1 file changed, 1 insertion(+)
create mode 100644 texto.txt
```

Está na hora de subir o que fizemos para o Gitlab.

Para isso vamos rodar o comando git push e o nosso repositório, que no caso é o <https://gitlab.com/aprendizado6532113/meuprojeto.git>

```
git push https://gitlab.com/aprendizado6532113/meuprojeto.git
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ git push https://gitlab.com/aprendizado6532113/meuprojeto.git
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 671 bytes | 671.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
To https://gitlab.com/aprendizado6532113/meuprojeto.git
 0dfe3b3..429e99e  main -> main
```

Lembrando que subimos os arquivos direto para a branch Main.

Arquivos hospedados no Gilab:

The screenshot shows the GitLab web interface for a project named 'meuprojeto'. At the top, there's a header with the project name, ID (44824821), and buttons for notifications, stars (0), and forks (0). Below the header, it shows '3 Commits', '1 Branch', '0 Tags', and '0 Bytes Project Storage'. The main content area displays a commit by 'Anderson B Silva (Ander)' 2 minutes ago, with a commit hash '429e99ee'. Below the commit, there's a section for file navigation with buttons for 'main', 'meuprojeto', and a '+' button. To the right are buttons for 'Find file', 'Web IDE', a download icon, and 'Clone'. Below this, there's a row of buttons for adding various files and configurations: 'README', 'LICENSE', 'CHANGELOG', 'CONTRIBUTING', 'Kubernetes cluster', 'CI/CD', 'Wiki', and 'Integrations'. At the bottom, there's a table listing the files in the repository.

Name	Last commit	Last update
README.md	Initial commit	1 day ago
index.html	pagina simples que exibe um titulo e um paragrafo	3 minutes ago
texto.txt	texto	2 minutes ago

Vamos criar uma nova branch chamada “develop”

```
git branch develop
```



```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ git branch develop
```

Acessando o branch criado

```
git checkout develop
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (main)
$ git checkout develop
Switched to branch 'develop'

Pichau@Anderson MINGW64 ~/Documents/meuprojeto (develop)
$
```

Para testar, vou criar um arquivo e subir ele para a branch “develop”

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (develop)
$ nano arquivo.txt

Pichau@Anderson MINGW64 ~/Documents/meuprojeto (develop)
$ cat arquivo.txt
arquivo para a branch develop
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (develop)
$ git add arquivo.txt
warning: in the working copy of 'arquivo.txt', LF will be replaced by CRLF the next time Git touches it


Pichau@Anderson MINGW64 ~/Documents/meuprojeto (develop)
$ git commit -m "arquivo de teste para a branch develop"
[develop 09bb02a] arquivo de teste para a branch develop
1 file changed, 1 insertion(+)
create mode 100644 arquivo.txt
```

Subindo para o GitLab

```
git push https://gitlab.com/aprendizado6532113/meuprojeto.git develop
```

```
Pichau@Anderson MINGW64 ~/Documents/meuprojeto (develop)
$ git push https://gitlab.com/aprendizado6532113/meuprojeto.git develop
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 394 bytes | 394.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for develop, visit:
remote:   https://gitlab.com/aprendizado6532113/meuprojeto/-/merge_requests/new?merge_request%5Bsource_branch%5D=develop
remote:
To https://gitlab.com/aprendizado6532113/meuprojeto.git
 * [new branch]      develop -> develop
```

Arquivo no Gitlab

 arquivo de teste para a branch develop
Anderson B Silva (Ander) authored 2 minutes ago

09bb02a1

develop meuprojeto +

HistoryFind fileWeb IDEDownloadClone

Name	Last commit	Last update
README.md	Initial commit	1 day ago
arquivo.txt	arquivo de teste para a branch develop	2 minutes ago
index.html	pagina simples que exibe um titulo e um paragrafo	29 minutes ago
texto.txt	texto	28 minutes ago

README.md

meuprojeto

Getting started

To make it easy for you to get started with GitLab, here's a list of recommended next steps.

Colaboração de repositório para repositório: git push

O comando "git push" é usado para enviar suas alterações em um repositório local para um repositório remoto. Este comando é frequentemente usado para colaborar com outros usuários ou para enviar suas alterações a um servidor de hospedagem Git, como GitHub, GitLab ou Bitbucket.

Antes de usar o comando "git push", você precisa garantir que todas as alterações que você deseja enviar para o repositório remoto estejam confirmadas em seu repositório local. Você pode usar o comando "git status" para verificar o status do seu repositório local e garantir que todos os arquivos estejam confirmados e prontos para serem enviados.

Para enviar as alterações do seu repositório local para um repositório remoto, você pode usar o comando "git push". O formato geral do comando é o seguinte:

```
git push <nome_do_repositório_remoto> <nome_do_branch_local>
```

Por exemplo, se você deseja enviar as alterações do branch "main" em seu repositório local para um repositório remoto chamado "origin", você pode usar o seguinte comando:

```
git push origin main
```

Depois de executar este comando, o Git enviará suas alterações para o repositório remoto e elas serão visíveis para outros usuários que acessam o mesmo repositório remoto.git