

Ignorando arquivos e diretórios

GERANDO COMMIT DE ARQUIVO QUE NÃO DEVERIA SER VERSIONADO

No repositório de João criar uma nova branch gitignore e alterar para ela

```
$ git switch -c gitignore
```

Gerar um novo commit

```
$ echo "entry1" > info.log  
$ git add info.log  
$ git commit -m "add info.log"
```

REALIZANDO ALTERAÇÕES NA ÁREA DE TRABALHO

Alterar o arquivo info.log adicionado no commit anterior

```
$ echo "entry2" >> info.log
```

Criar um novo diretório node_modules

```
$ mkdir node_modules
```

Criar um arquivo logger.js dentro do diretório node_modules

```
$ touch node_modules/logger.js
```

Criar um novo arquivo error.log

```
$ echo "entry1" > error.log
```



Adicionar o arquivo error.log para a área de preparo

```
$ git add error.log
```

Verificar o estado do repositório

```
$ git status
```

ADICIONANDO O ARQUIVO .GITIGNORE

Adicionar o arquivo .gitignore

```
$ echo -e "*.log\nnode_modules/" > .gitignore
```

Nota: a opção -e permite que o carácter de newline (\n) seja interpretado

Nota: o * corresponde a qualquer sequência de caracteres. Ou seja, qualquer arquivo que termina com “.log”. O símbolo * pertence ao padrão glob

Verificar o estado do repositório

```
$ git status
```

Nota: o diretório node_modules e seus arquivos não rastreados (untracked) foram ignorados

Nota: o arquivo info.log não foi ignorado, uma vez que o arquivo está no estado modified

Nota: o arquivo error.log não foi ignorado, uma vez que o arquivo está no estado staged

DESCARTANDO ARQUIVOS QUE NÃO DEVEM ENTRAR NO VERSIONAMENTO

Descartar arquivo error.log da área de preparo

```
$ git restore -staged error.log
```



Verificar o estado do repositório

```
$ git status
```

Nota: o arquivo `error.log` não é mais exibido, uma vez que retornou ao estado `untracked`

Remover o arquivo `info.log` do versionamento

```
$ git rm --cached info.log
```

Nota: o comando `git rm` serve para remover arquivos que já foram adicionados a área de repositório (arquivos que já foram comitados). É adicionado na área de preparo os arquivos removidos com o estado `deleted`

Nota: o uso da opção `--cached` no `rm` mantém os arquivos na área de trabalho (com o estado `untracked`). Caso não fosse utilizado esse parâmetro os arquivos teriam sido excluídos da área de trabalho

Verificar o estado do repositório

```
$ git status
```

Gerar um novo commit

```
$ git commit -m "remove info log"
```

Verificar o estado do repositório

```
$ git status
```

Verificar que o diretório `"node_modules"` existe na área de trabalho

```
$ ls
```



Gerar um commit para o .gitignore

```
$ git add .gitignore  
$ git commit -m "add .gitignore"
```

SINCRONIZANDO REPOSITÓRIOS

Publicar tal branch para o repositório remoto

```
$ git push -u origin gitignore
```

No Bitbucket:

- Criar um pull request da branch gitignore para a branch master
- Aprovar pull request e realizar o merge com a estratégia fast forward

No repositório de João e Maria:

```
$ git switch master  
$ git pull
```



Submódulos

CRIANDO UM NOVO REPOSITÓRIO NO BITBUCKET

Criar repositório "CSV" no Bitbucket

No repositório criado, adicionar um arquivo csv.py diretamente pelo Bitbucket com o seguinte conteúdo:

```
def import():  
    ...
```

Copiar URL do repositório

ADICIONAR UM SUBMÓDULO AO SEU PROJETO GIT

No repositório de João, adicionar o repositório CSV como submódulo

```
$ git submodule add <URL_REPO>
```

Verificar o estado do repositório de João

```
$ git status
```

Nota: é exibido o novo diretório "csv" e um arquivo .gitmodules no estado untracked

Verificar o conteúdo do arquivo .gitmodules

```
$ cat .gitmodules
```

Nota: o arquivo .gitmodules contém a lista de todos os submódulos e suas respectivas URLs

Acessar o diretório csv e verificar o estado do repositório

```
$ cd csv  
$ git status
```



SEÇÃO 9: OUTROS COMANDOS E CONCEITOS

Nota: não há novos arquivos ou alterações pendentes para serem versionadas, uma vez que esse é um repositório independente incorporado dentro do repositório de João

Verificar o histórico

```
$ git hist
```

Nota: o histórico apresentado é da branch master do repositório atual csv

Retornar ao diretório raiz

```
$ cd ..
```

Adicionar o submódulo ao repositório cursogit

```
$ git add .  
$ git commit -m "add csv submodule"
```

Enviar ao repositório remoto

```
$ git push
```

Verificar a atualização no repositório cursogit no Bitbucket



ATUALIZAR UM REPOSITÓRIO QUE FOI ADICIONADO UM SUBMÓDULO

Acessar o repositório de Maria

Atualizar o repositório

```
$ git pull
```

Listar os arquivos do diretório csv

```
$ ls csv/
```

Nota: não há arquivos neste diretório

Para repositórios que contém submódulos é necessário inicializar os submódulos e baixar os arquivos

Inicializar os submódulos

```
$ git submodule init
```

Realizar o download dos arquivos do submódulo

```
$ git submodule update
```

Listar os arquivos do diretório csv novamente

```
$ ls csv/
```

Nota: os arquivos do submódulo foram baixados e os submódulos foram inicializados



CLONAR UM REPOSITÓRIO QUE CONTÉM SUBMÓDULOS

Criar o seguinte diretório na área de trabalho: curso/git/chico

Neste diretório, realizar um clone do repositório cursogit utilizando a opção **--recurse-submodules**

Acessar o repositório

```
$ cd cursogit/
```

Listar todos os arquivos do diretório csv

```
$ ls csv/
```

Nota: os arquivos do submódulo foram baixados e os submódulos foram inicializados pela utilização da opção **--recurse-submodules**

Nota: se a opção acima não for utilizada poderia ser utilizado `git submodule init` e `git submodule update` conforme visto anteriormente



ATUALIZAR UM SUBMÓDULO EM SEU PROJETO GIT

No Bitbucket, atualizar o arquivo csv.py no repositório CSV, incluindo a linha abaixo:

```
def export():  
    ...
```

Acessar o repositório de João

Acessar o diretório csv

```
$ cd csv/
```

Atualizar o repositório CSV

```
$ git pull
```

Verificar o histórico do repositório CSV

```
$ git hist
```

Retornar ao diretório raiz

```
$ cd ..
```

Verificar o estado do repositório

```
$ git status
```

Nota: a alteração do arquivo csv.py baixada através do git pull precisa ser atualizada no repositório cursogit



Gerar um novo commit

```
$ git add csv/  
$ git commit -m "update csv module"
```

Enviar atualização para o repositório remoto

```
$ git push
```

Verificar a atualização no Bitbucket



SUBMÓDULO INTERNALS

Acessar o repositório de João

Verificar todos os arquivos do diretório csv

```
$ ls -a csv/
```

Nota: a opção -a exibe todos os arquivos, incluindo arquivos ocultos

Nota: verifique que em um submódulo .git é um arquivo e não um diretório

Verificar o conteúdo do arquivo .git

```
$ cat csv/.git
```

Nota: o arquivo contém o caminho pro diretório .git dentro do repositório principal

Verificar os arquivos contidos no diretório referenciado no arquivo .git

```
$ ls .git/modules/csv/
```

Nota: este é o diretório .git do submódulo csv

Verificar o histórico

```
$ git hist
```

Copiar a hash do commit “add csv submodule”

Avaliar o commit que adicionou o submódulo (add csv submodule)

```
$ git cat-file -p <hash commit "add csv submodule">
```

Copiar a hash da tree



Avaliar a tree acima

```
$ git cat-file -p <hash tree>
```

Nota: csv não é uma árvore de diretório, trata-se de um commit!

Copiar a hash do commit

Tentar avaliar o commit

```
$ git cat-file -p <commit copiado acima>
```

Nota: não é encontrado esse objeto. Isso ocorre porque esse objeto é do submódulo e não do repositório principal

Acessar repositório csv

```
$ cd csv/
```

Avaliar o commit novamente

```
$ git cat-file -p <commit copiado acima>
```

Nota: trata-se do commit que criou o arquivo csv.py



SSH (Secure Socket Shell)

GERAR UMA CHAVE SSH LOCAL

Acessar o terminal e gerar um par de chaves

```
$ ssh-keygen
```

***Nota:** você pode configurar as opções que aparecem conforme sua preferência ou utilizar o padrão pressionando enter*

Copiar a localização do arquivo que contém a chave ssh pública
(.../.SSH/ID_XXX.PUB)

Mostrar o conteúdo da chave pública

```
$ cat <localização do arquivo copiado acima>
```

Copiar todo o conteúdo do arquivo

CONFIGURAR CHAVE SSH NO BITBUCKET

Acessar sua conta no Bitbucket

No ícone de engrenagem na parte superior direita, clique em Personal Bitbucket settings / SSH keys

Adicione uma nova chave com uma label qualquer e no campo key cole o conteúdo do arquivo da chave ssh pública

CLONAR ALTERAÇÕES VIA SSH

Acessar o repositório no Bitbucket

Na aba Source, clicar em Clone

Alterar o protocolo para SSH

Copiar o comando git clone apresentado



ENVIAR ALTERAÇÕES VIA SSH

Criar um diretório “.../ssh”

Acessar esta pasta via Git Bash

Executar o comando copiado anteriormente

Gerar um novo commit

```
$ echo "function save(){...}" > modules.js  
$ git add modules.js  
$ git commit -m "add save module"
```

Enviar ao repositório remoto

```
$ git push
```



Bare repository

ENTENDENDO SOBRE REPOSITÓRIOS BARE

Criar o seguinte diretório na área de trabalho: curso/git/bare
Acessar esta pasta via linha de comando

Configurar como um repositório git bare

```
$ git init --bare
```

Listar arquivos e diretórios

```
$ ls
```

Nota: o conteúdo do diretório `.git` existente em um repositório non-bare foi adicionado ao repositório

Verificar o estado do repositório

```
$ git status
```

Nota: o Git informa que essa operação é permitada somente para repositórios que possuem uma área de trabalho. Repositórios bare não possuem uma área de trabalho



Está gostando deste curso?

*Compartilhe sua experiência nas redes sociais com a tag
#rsantanatech para que eu possa interagir com a sua postagem.*

**Acompanhe nas redes sociais
e fique por dentro de todos os conteúdos.**

