Principais Comandos

# Obtendo ajuda

Caso você precise de ajuda usando o Git, existem algumas formas de se obter ajuda

$ git help

$ git --version

$ git help <verb>

$ git <verb> --help

# Sua Identidade e Seu Editor

A primeira coisa que você deve fazer quando instalar o Git é definir o seu nome de usuário e endereço de e-mail. Isso é importante porque todos os commits no Git utilizam essas informações, e está imutavelmente anexado nos commits que você realiza:

Por padrão, Git usa o editor padrão do sistema, que é geralmente Vi ou Vim. Caso você queira utilizar um editor diferente, tal como o Sublime, Bracks, Dreamwever, você pode executar o comando:

$ git config --global user.name "adammacias"

$ git config --global user.email eu@adammacias.com

$ git config --global core.editor "'C:\Program Files\Sublime Text 3\sublime\_text.exe' -multiInst -nosession"

$ git config user.name

$ git config user.email

$ git config core.editor

$ git config –list

# Obtendo um Repositório Git

Você pode obter um projeto Git utilizando duas formas principais.

A primeira faz uso de um projeto ou diretório existente e o importa para o Git. A segunda clona um repositório Git existente a partir de outro servidor.

Inicializando um Repositório em um Diretório Existente

Caso você esteja iniciando o monitoramento de um projeto existente com Git, você precisa ir para o diretório do projeto e digitar

$ git init

Isso cria um novo subdiretório chamado .git que contem todos os arquivos necessários de seu repositório

Clonando um Repositório Existente

Caso você queira copiar um repositório Git já existente — por exemplo, um projeto que você queira contribuir — o comando necessário é git clone.

$ git clone [url] [diretory]

# Verificando o Status de Seus Arquivos

Arquivos monitorados são arquivos que estavam no último commit;

Arquivos não monitorados e modificados são todo o restante; (Arquivos novos ou deletados)

$ git status

# On branch master

nothing to commit, working directory clean

Isso significa que você tem um diretório de trabalho limpo — em outras palavras, não existem arquivos monitorados e modificados. Git também não encontrou qualquer arquivo não monitorado, caso contrário eles seriam listados aqui. Por fim, o comando lhe mostra em qual branch você se encontra.

working diretory (diretorio de trabalho)

staging area (encenação/pre-commmits)

## Monitorando Novos Arquivos

#### Criar arquivo a.html

> a.html

git add -A

git add .

#### Selecionando Arquivos Modificados

> b.html

echo b > b.html

git add -A

# Fazendo Commit de Suas Mudanças

Agora que a sua área de seleção está do jeito que você quer, você pode fazer o commit de suas mudanças. — qualquer arquivo que você criou ou modificou que você não tenha rodado o comando git add desde que editou — não fará parte deste commit. Estes arquivos permanecerão como arquivos modificados em seu disco.

git commit -m "Story 182: Fix benchmarks for speed"

# Pulando a Área de Seleção

Se você quiser pular a área de seleção, o Git provê um atalho simples. Informar a opção -a ao comando git commit faz com que o Git selecione automaticamente cada arquivo que está sendo monitorado antes de realizar o commit, permitindo que você pule a parte do git add:

git commit -a -m "Story 182: Fix benchmarks for speed"

# Visualizando Suas Mudanças Selecionadas e Não Selecionadas

Se o comando git status for muito vago — você quer saber exatamente o que você alterou, não apenas quais arquivos foram alterados — você pode utilizar o comando git diff.

git diff

Este comando compara o que está no seu diretório de trabalho com o que está na sua área de seleção (staging). O resultado te mostra as mudanças que você fez que ainda não foram selecionadas.

Se você quer ver o que selecionou que irá no seu próximo commit, pode utilizar git diff --cached.

# Fazendo Commit de Suas Mudanças

## Ignorando Arquivos

Muitas vezes, você terá uma classe de arquivos que não quer que o Git automaticamente adicione ou mostre como arquivos não monitorados. Normalmente estes arquivos são gerados automaticamente como arquivos de log ou produzidos pelo seu sistema de build. Nestes casos, você pode criar um arquivo contendo uma lista de padrões a serem checados chamado .gitignore.

> .gitignore

# Visualizando o Histórico de Commits

Depois que você tiver criado vários commits, ou se clonou um repositório com um histórico de commits existente, você provavelmente vai querer ver o que aconteceu. A ferramenta mais básica e poderosa para fazer isso é o comando git log.

git log

git log --oneline --decorate --all --graph (Traz info de onde vc esta (HEAD) e dos branchs, all mostra os branchs acima, mostra o grafico graph )

## Desfazendo Coisas

Modificando Seu Último Commit

Por exemplo, se você fez um commit e esqueceu de adicionar na área de seleção as modificações de um arquivo que gostaria de ter adicionado nesse commit, você pode fazer algo como isso:

$ git commit -m 'initial commit'

$ git add forgotten\_file

$ git commit --amend

Depois desses três comandos você obterá um único commit — o segundo commit substitui os resultados do primeiro.

## Tirando um arquivo da área de seleção

git reset HEAD [namefile]

git reset HEAD .

# CRIANDO BRANCH

git branch [name]

git checkout [name]

git checkout -b [name]

// Mostrar ls arquivos comparação com branchs

# DANDO MERGE

SINCRONIZAR REPOSITORIOS

Manter repositório do Github forkado sincronizado com o original

Depois que você forka um projeto no Github, você faz as alterações necessárias e envia um Pull Request para o repositório original.

Se você precisa fazer mais alguma alteração no mesmo repositório no futuro, mas já tem outras colaborações no repositório original, como você procede para manter seu repositório atualizado? Copia as alterações manualmente para o seu repositório? Apaga e forka novamente? Chora?

git remote add upstream [ENDEREÇO DO REPOSITÓRIO PRINCIPAL] git fetch upstream git merge upstream/master

git remote set-url origin [git@gitlab.com:fatecweb/docs.git](mailto:git@gitlab.com:fatecweb/docs.git)

Inicialize o Git

Até aqui a pasta que você abriu no git é um diretótio como qualquer outro, o Git ainda não o reconhece como um repositório local. Para identificar a pasta como repositório local git, você precisa inicializar o git. No Git Bash, digite:

git init

Após isso, será criado um arquivo oculto .git em sua pasta. Este arquivo é o que guradará suas configurações, não o delete.

==================

5. Configure sua conta

Agora, você configurará seus dados de conta:

git config --global user.name "seuNomeDeUsuario"

git config --global user.email "meu\_nome@email.com"

==================

6. Clonando um repositório

Neste ponto seu repositório local já está pronto para funcionar, porém ainda não sabe com o que irá sincronizar as informações que aqui estarão. Agora, iremos clonar um repositório já existente em algum site de versionamento (gitHub, GitLab..) para seu computador. Digite o link disponível no reporsitório sobre o seu projeto. Ex.:

git clone git@gitlab.com:nomeSite/nomeProjeto.git

Isto deve ser feito apenas uma vez. Uma vez clonado, o todas as alterações estarão disponíveis para enviar para o site.

# Usando o Git: Comando básicos de utilização

## 1. Entrar na pasta

No git bash, certifique-se de estar na pasta que foi definida como seu repositório local. Caso não esteja, entre:

Clique no logo do Git Bash

Entre com o caminho da pasta onde irá trabalhar: cd C:/caminho/caminho/nomepasta/

Ou, você pode abrir o Git Bash direto na pasta:

Entre na pasta que irá trabalhar

Clique com o botão direito do mouse e selecione a opção git bash

==================

## 2. Puxar os dados do repositório

Para ter seu repositório local sempre sincronizado/atualizado com o repositório online, você precisará puxar as alterações:

Para isso, digite o comando: git pull origin master

O master significa que o projeto é master. Ele pode estar como Alpha ou outro nome. Caso não tenha certeza, é possível dar pull com apenas o comando git pull origin

==================

## 3. Ver status dos arquivos

Para saber o status dos arquivos, ver quais foram modificados, se já estão add pra commit e etc. use:

git status

Sempre que for modificado qualquer arquivo de dentro da pasta do projeto, ele aparecerá em vermelho. Permanecerão assim até serem add.

Em verde, aparecerão os que já estiverem prontos para serem commitados.

==================

#### 3.1. Status com mais detalhes Existe outro comando para ver mais detalhes sobre os arquivos modificados:

git diff

Apresentará os arquivos modificados com detalhes das linhas adicionadas e excluídas (assim como aparede no github após o push).

Adcionando a referência --staged e --cached.

git diff --staged é posssivel comparar o as alterações feitas com o último commit.

git diff --cached apresenta o que está sendo mudado até o momento.

Os dois, --cached e --staged são sinônimos.

=================

## 4. Adicionando a lista de mudanças

Para adicionar arquivos à lista para commit:

Para adicionar todos os arquivos que modificou, digite o código:

git add .

Para adicionar a lista de commit os arquivos que foram escluídos:

git add -u

Para adicionar algum arquivo específico:

git add nome-do-arquivo.xx

Caso já tenha adicionado um arquivo mas depois (antes de dar commit) tenha o modificado, o adicione novamente para que as mudanças sejam atualizadas.

É possível, também, desfazer a adição de algum arquivo, para isso, use o código:

git reset HEADnome-do-arquivo.xx

Também podemos adicionar e commitar ao mesmo tempo, basta adicionar a tag -a:

git commit-a-m "Mensagem do commit"

==================

## 5. Commitando as mudanças

É preciso commitar os arquivos adicionados antes de enviá-los. Isso serve para dar um nome ao envio, e seu envio automaticamente receberá uma numeração que o identificará:

git commit -m "Descrição da mudanca feita"

faça uma descrição referente a mudança realizada.

Ao fazer o commit, é possível referenciar uma ensue criada no site de versionamento usando uma # + número da insue. Exemplo:

git commit -m "Descrição da mudanca feita #20 "

É possível, também, adicionar e commitar ao mesmo tempo, basta adicionar a tag -a:

git commit-a-m "Mensagem do commit"

Uma insue é uma tarefa criada no site. Asimm, a mudança ficará lincada à tarefa automaticamente

## 5.1. Voltando um commit

Caso tenha feito um commit e deseje desfaze-lo, digite:

git reset HEAD~1

Isso fará com que o commit último commit seja desfeito, caso deseje voltar mais commits basta mudar o número que vem depois do head~. O número corresponde ao número de commits que irá voltar.

Voltando dois: git reset HEAD~2

## 5.2. Ver histórico de commits

git log

Para que este seja mostradas as diferenças entre os commits adicione o -p

git log -p

Verificando os últimos 2 commits: git log -p -2

==================

# 6. Enviando os arquivos

Essa é a última parte da atualização, ele enviará para o repositório (gitlab, github..) as alterações feitas.

git push origin master

O master significa que o projeto é master. Ele pode estar como Alpha ou outro nome. Caso não tenha certeza, é possível dar pull com apenas o comando git pull origin