

FACULDADE DE TECNOLOGIA SENAC GOIÁS

ANALISE E DESENVOLVIMENTO DE SISTEMAS MODULO II

CONTROLE DE VERSÃO

O que é git?

Git é um sistema de controle de versão de arquivos. Através deles podemos desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente no mesmo, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.

Se não houver um sistema de versão, imagine o caos entre duas pessoas abrindo o mesmo arquivo ao mesmo tempo. Uma das aplicações do git é justamente essa, permitir que um arquivo possa ser editado ao mesmo tempo por pessoas diferentes. Por mais complexo que isso seja, ele tenta manter tudo em ordem para evitar problemas para nós desenvolvedores.

O que é github?

O Github é um serviço web que oferece diversas funcionalidades extras aplicadas ao git. Resumindo, você poderá usar gratuitamente o github para hospedar seus projetos pessoais. Além disso, quase todos os projetos/frameworks/bibliotecas sobre desenvolvimento open source estão no github, e você pode acompanhá-los através de novas versões, contribuir informando bugs ou até mesmo enviando código e correções.

TUTORIAL GITHUB II

Instalando git

O git é um programa que pode ser instalado neste link para Windows, neste para Mac, ou então através do comando `sudo apt-get install git` para plataformas Linux/Debian, como o Ubuntu.


Após criar a conta, você verá um botão verde +New Repository na qual poderá criar um repositório de acordo com a tela a seguir.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)



Owner

Repository name *

 **jhonzeras** /

Great repository names are short and memorable. Need inspiration? How about [studious-umbrella?](#)

Description (optional)

- ☒  **Public**
Anyone can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

- ☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

Add a license: **None** ▾

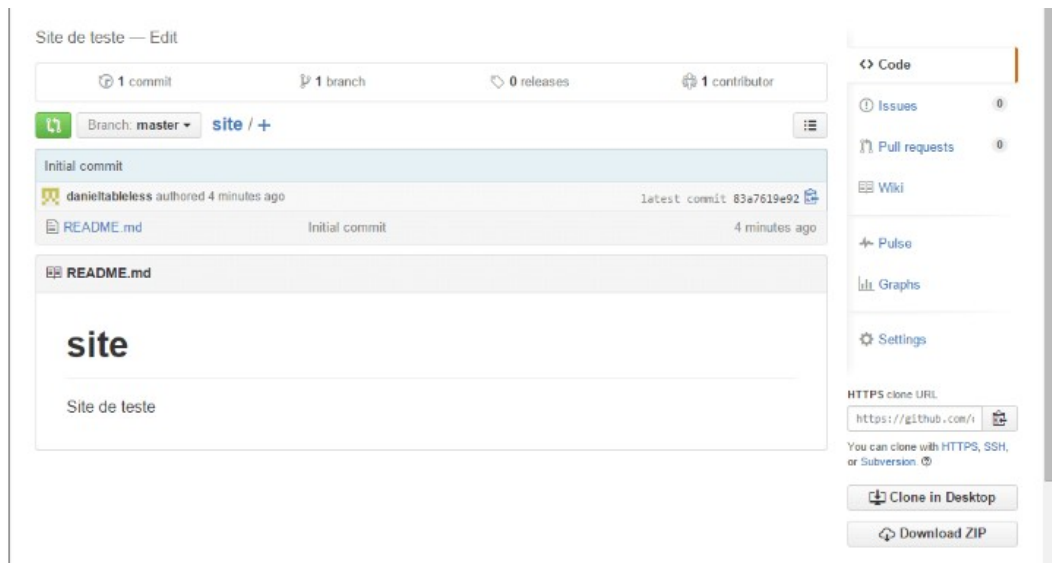


Create repository

Nesta imagem estamos criando um repositório cujo nome é site, de domínio público (podem ser criados reps privados pagando uma mensalidade), e com o arquivo README.md embutido, que contém uma descrição do seu projeto. Para que

possamos começar a entender como o git funciona, é fundamental criar um repositório como este para os nossos testes.

Após a criação do repositório, ele estará disponível no endereço <https://github.com/<username>/site>, onde username é o login que você usou para se cadastrar. Acessando esta url temos a seguinte resposta:



Temos muitas informações nesta tela, pois ela é a tela principal do seu projeto. Explicaremos algumas informações ao longo deste artigo, por enquanto repare apenas no botão HTTPs Clone Url na parte inferior à direita. Esta URL será necessária para que possamos “clonar” este projeto em nosso ambiente de estudo (sua máquina windows, mac, linux ou a vm). Clique no botão de copiar URL e perceba que a seguinte URL está na área de transferência: <https://github.com/<username>/site.git>

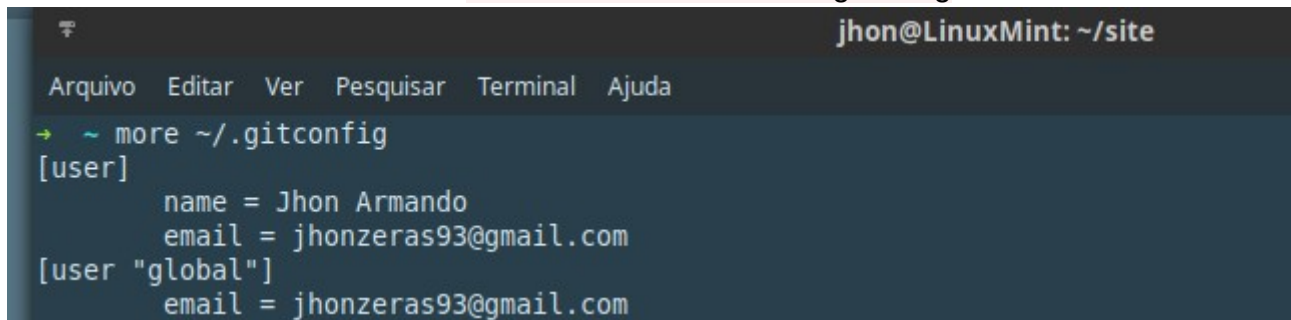
Configurando o git

Existem 2 pequenos passos para configurar o seu GIT para ter um acesso mais simplificado ao github. Aqui estaremos estabelecendo que, sempre que necessitar, você irá fornecer o seu login e senha ao GitHub. Existem meios para salvar a senha em local seguro, mas vamos pular esta etapa. Para abrir um terminal GIT no Windows, basta criar uma pasta no seu sistema e, nela, clicar com o botão direito do mouse e escolher Git Bash Here. Em sistemas mac/linux você já está acostumado a usar o terminal/console, o git estará lá disponível. Neste artigo estaremos utilizando a máquina virtual cloud9, que você pode aprender a usá-la neste [artigo](#).

Então, com o seu terminal git aberto, vamos digitar:

```
$ git config --global user.name "YOUR NAME"
$ git config --global user.email "YOUR EMAIL ADDRESS"
```

Estas configurações ficam alocadas no arquivo `~/.gitconfig`, onde o `~` é o seu diretório home. No Windows, ele fica em `c:\Usuarios\<username>\.gitconfig`.

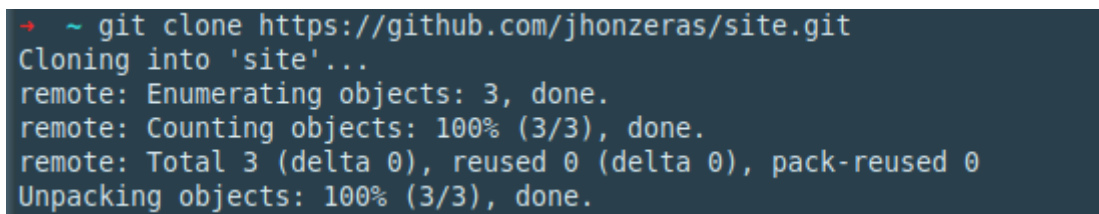


```
jhon@LinuxMint: ~/site
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
→ ~ more ~/.gitconfig
[user]
    name = Jhon Armando
    email = jhonzeras93@gmail.com
[user "global"]
    email = jhonzeras93@gmail.com
```

Clonar!

Então o que temos até agora é o git configurado para utilizar o github e o projeto no github criado. Precisamos trazer este projeto para o nosso git, e este processo se chama clonar. Então, quando você quiser começar um projeto utilizando git, você cria ele no github e clona na sua máquina. O comando para clonar o projeto é `git clone "url"`, veja:

```
git clone https://github.com/<username>/site.git
```



```
→ ~ git clone https://github.com/jhonzeras/site.git
Cloning into 'site'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

Perceba que, ao fazer o `git clone`, o projeto é baixado para a sua máquina, e uma pasta com o nome do projeto é criada.

Com o repositório na sua máquina, vamos aprender 4 comandos iniciais que farão parte da sua vida a partir de agora:

- `git add <arquivos...>` Este comando adiciona o(s) arquivo(s) em um lugar que chamamos de INDEX, que funciona como uma área do git no qual os arquivos possam ser enviados ao Github. É importante saber que ADD não está adicionando um arquivo novo ao repositório, mas sim dizendo que o arquivo (sendo novo ou não) está sendo preparado para entrar na próxima revisão do repositório.
- `git commit -m "comentário qualquer"` Este comando realiza o que chamamos de "commit", que significa pegar todos os arquivos que estão naquele lugar INDEX que o comando add adicionou e criar uma revisão com um número e um comentário, que será vista por todos.
- `git push` Push (empurrar) é usado para publicar todos os seus commits para o github. Neste momento, será pedido a sua senha.
- `git status` Exibe o status do seu repositório atual

Chegou o momento de praticar um pouco o que vimos até agora, e com bastante calma para que você possa entender cada passo. Após clonar o seu projeto, crie o arquivo `index.html` na pasta `site` que é o seu repositório git. Após criar o arquivo, execute o comando `git status`. A resposta é semelhante a figura a seguir:

```

→ ~ /home/jhon/site
→ site git:(master) touch index.html
→ site git:(master) x git status
No ramo master
Your branch is up to date with 'origin/master'.

Arquivos não monitorados:
(utilize "git add <arquivo>..." para incluir o que será submetido)

    index.html

nada adicionado ao envio mas arquivos não registrados estão presentes (use "git add" to registrar)

```

Ou seja, o comando `git status` nos trouxe várias informações, que iremos ignorar a princípio, exceto pelo `Untracked files`, dizendo que existe um arquivo que não está sendo “mapeado” pelo git. Para preparar este arquivo para o seu versionamento, usamos o comando `git add`, veja:

```

→ site git:(master) x git add index.html
→ site git:(master) x git status
No ramo master
Your branch is up to date with 'origin/master'.

Mudanças a serem submetidas:
(utilize "git reset HEAD <file>..." to unstage)

    new file:   index.html

```

Agora temos o nosso arquivo `index.html` no INDEX do repositório, ou se você quiser pensar: “preparado para um commit”. Para commitar este arquivo, usamos:

```

→ site git:(master) x git commit -m "Criação do arquivo index.html"
[master 7cblf68] Criação do arquivo index.html
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html
→ site git:(master) git status
No ramo master
Seu ramo está à frente de 'origin/master' por 1 submissão.
(utilize "git push" to publish your local commits)

nothing to commit, working tree clean

```

Após “commitar” o arquivo, ele já está presente no nosso repositório local, tanto que realizamos o comando `git status` novamente e ele retornou que não havia nada de novo no projeto. Perceba agora que, mesmo recarregando o projeto no github, nada muda. Ou seja, estas mudanças até agora foram locais, você pode realizar várias operações antes de publicá-las no github. Para publicar, usamos o comando `git push`:

```
→ site git:(master) git push
Username for 'https://github.com': jhonzeras
Password for 'https://jhonzeras@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 303.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/jhonzeras/site.git
   1b6c519..7cb1f68  master -> master
→ site git:(master) █
```

Após realizar o git push podemos ver no site github as mudanças realizadas no projeto:

The screenshot shows the GitHub interface for the repository 'jhonzeras / site'. At the top, there are buttons for 'Unwatch', 'Star' (0), and 'Fork' (0). Below this is a navigation bar with tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Actions', 'Projects' (0), 'Wiki', 'Security', 'Insights', and 'Settings'. The main content area shows the repository name 'Site de Teste' with an 'Edit' button. Below this, there are statistics: '2 commits', '1 branch', '0 packages', '0 releases', and '1 contributor'. A section for the 'master' branch shows a 'New pull request' button and buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A list of commits is shown, with the latest commit '7cb1f68' from 'jhonzeras' titled 'Criação do arquivo index.html' made 19 minutes ago. Below the commits, the 'README.md' file is displayed, showing the text 'site' and 'Site de Teste'.

Errei a mensagem do commit

Imagine que você tenha errado a mensagem que escreveu no commit ou simplesmente queira melhorar a descrição do seu trabalho. Você já comitou a mensagem mas ainda não fez o push das suas modificações para o servidor. Nesse caso você usa a flag `--amend`. Fica assim:

```
$ git commit --amend
```

O `git commit --amend` modifica a mensagem do commit mais recente, ou seja, o último commit feito por você no projeto. Além de você mudar a mensagem do commit, você consegue adicionar arquivos que você se esqueceu ou retirar arquivos comitados por engano. O git cria um commit totalmente novo e corrigido.

O git pull

Ainda existe um comando importante neste processo, que é o `git pull`. Ele é usado para trazer todas as modificações que estão no github para o seu projeto local. Isso é vital quando existem projetos mantidos por mais de uma pessoa, ou se você possui duas máquinas e precisa manter a sincronia entre elas. Supondo que você possui uma máquina no trabalho e outra em casa. Ambas tem o repositório local ligado ao github. Quando você executar um `git push` em uma das máquinas, terá que realizar um `git pull` na outra.