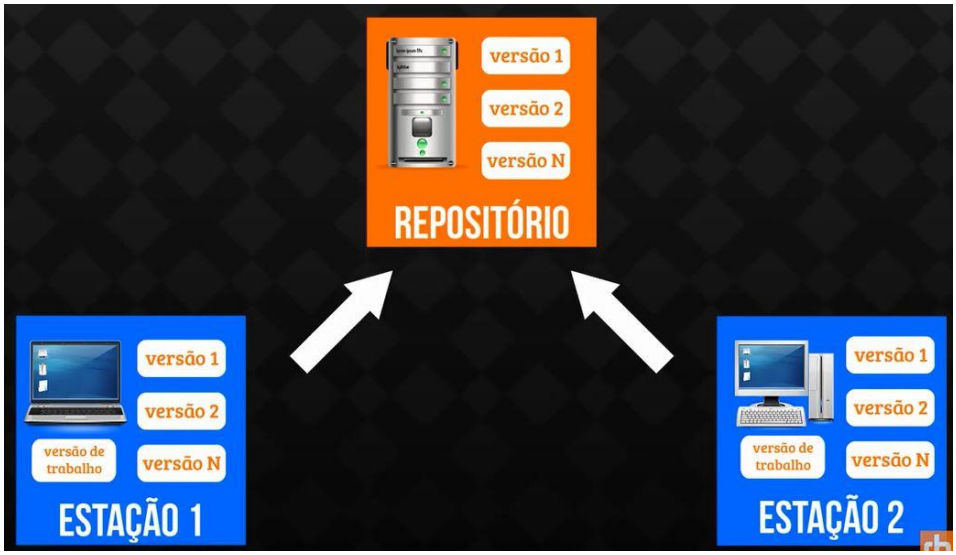
Git e GitHub
RBtech (Ricardo)

https://www.youtube.com/watch?v=WVLhm1AMeYE&list=PLInBAd9OZCzzHBJjLFZzRl6DgUmOeG3H0

Resumo do curso feito por Roberto Pinheiro

AULA 1 - INTRODUÇÃO

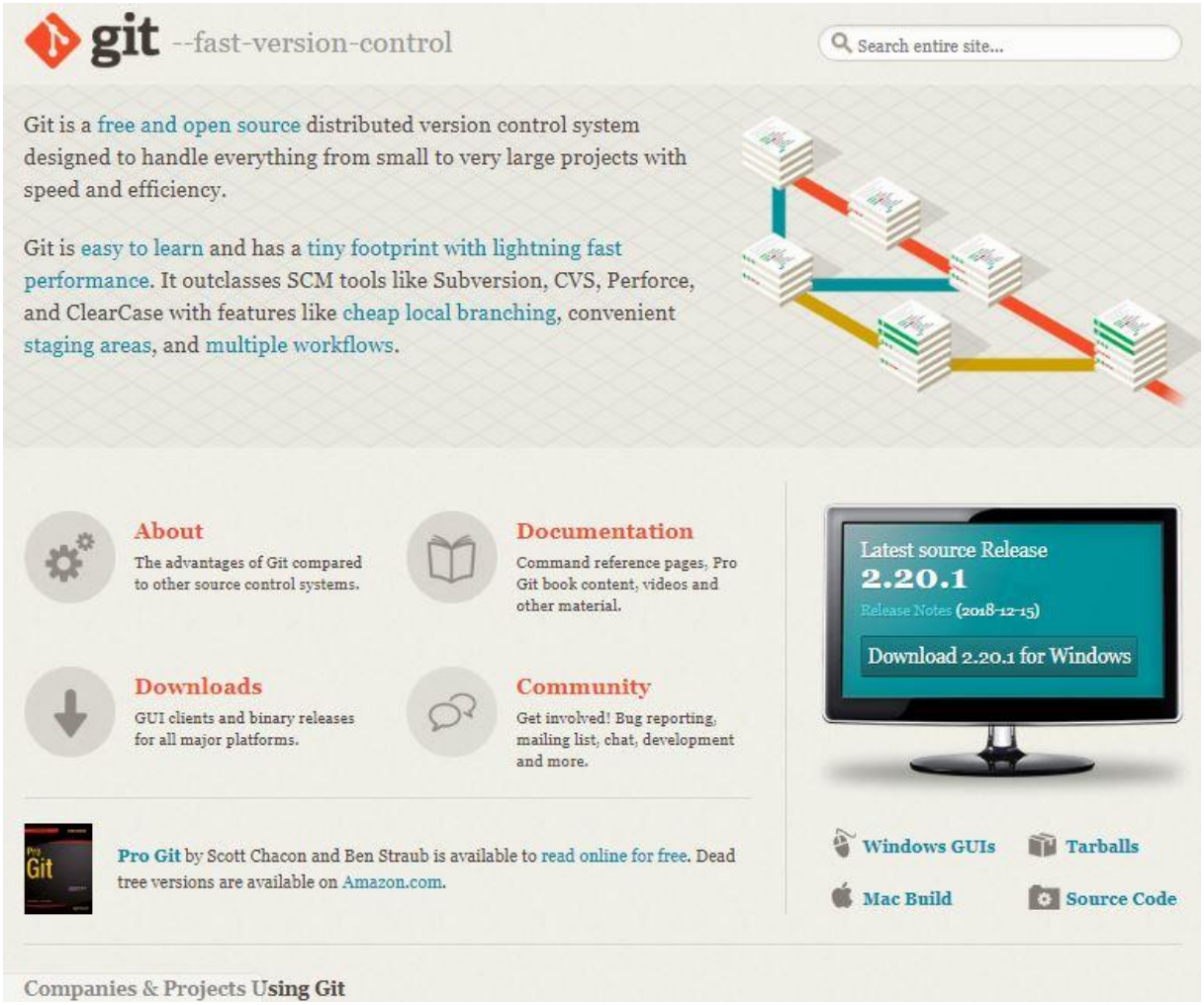
Git é um sistema de controle de versão distribuído.



Cada projeto é dividido em três estágios:

Site oficial do Git:

https://git-scm.com/



O gerenciamento do Git é feito por linha de comando, mas existem interfaces gráficas para facilitar o uso do sistema.

O terminal do Git é o Git Bash.

Configurações iniciais básicas

git config --global user.name <nome_usuario>

git config --global user.email <email_usuario>

Exemplo:

git config --global user.name "Roberto Pinheiro"

git config --global user.email "betopinheiro1005@yahoo.com.br"

Confirmando se a instalação foi bem sucedida

$ git --version

git version 2.20.1.windows.1

Listando as configurações do sistema

git config --list

core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
user.name=Roberto Pinheiro
user.email=betopinheiro1005@yahoo.com.br
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true

Exibindo o diretório local

$ pwd

/c/Users/betop_000

Alterando para a pasta de trabalho

Para ir para a pasta de trabalho:

cd /c/git/

AULA 2 – COMANDOS BÁSICOS INICIAIS DO GIT

Para limpar comandos:

clear

Criação de um repositório

Para criar um repositório, entrar na pasta onde ele será criado e digitar o comando:

git init

Será criada dentro da pasta, uma subpasta oculta chamada .git

Status do repositório

Digite o comando:

git status

Adicionando arquivos do projeto

Para adicionar um arquivo específico digite o comando:

git add <nome do arquivo>

Para adicionar todos os arquivos com a extensão php digite o comando:

git add *.php

Para adicionar todos os arquivos da pasta, digite o comando:

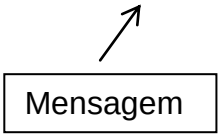
git add .

Ou digite o comando:

git add *

Confirmando (gravando) as mudanças no projeto

git commit –m "Commit inicial"



Ignorando arquivos no commit

Criar arquivo com o nome .gitignore com o nome dos arquivos a serem ignorados:

.gitignore
resumo.txt

Fazendo commit sem passar pela etapa do add

git commit –a –m "linhas adicionadas no arquivo readme"

AULA 3 – VISUALIZAÇÃO DE ALTERAÇÕES

Visualizando as alterações feitas com o passar do tempo

Para consultar todas as alterações feitas em um arquivo que está na pasta de trabalho, mas que ainda não foi adicionado a stage area utilizar o comando:

git diff

Será exibido em vermelho o conteúdo antes da alteração e em verde as alterações feitas.

Exibindo o que está na stage area

git diff --staged

Exibindo todos os commits feitos no projeto

git log

Exibe um histórico de todos os commits feitos no projeto.

Exemplo:

A chave (hash) ao lado de cada commit serve para referenciá-lo sempre que for necessário (para fazer transição de commits).

Para visualizar mais detalhes (por exemplo: ver as alterações feitas em cada commit) use o comando:

git log –p

Exibe como se fosse uma junção do git log com o git diff.

Para finalizar a tela de exibição, clique na tecla q (quit).

Para limitar a quantidade de commits a serem exibidos use o comando:

git log –p -<n>

onde <n> é o número de entradas a serem exibidas. Exemplo:

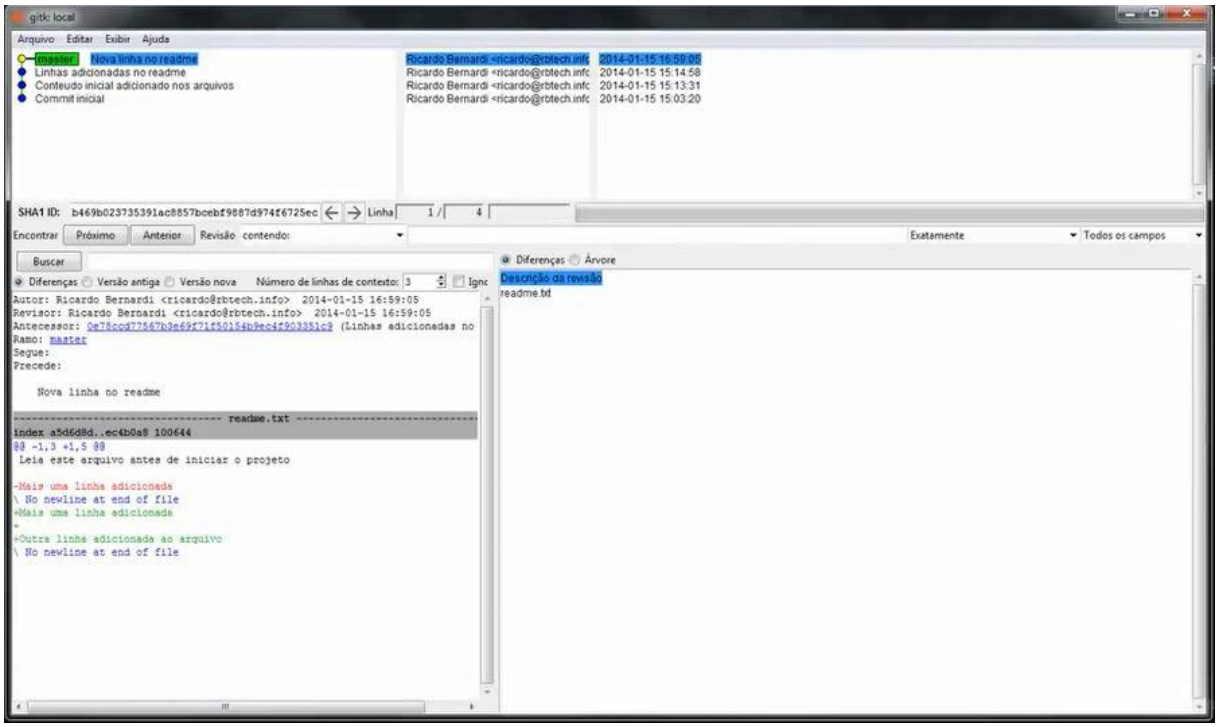
git log –p -3

Esse comando irá exibir as três últimas entradas.

Para facilitar a visualização das alterações feitas é mais adequado utilizar a interface gráfica do git com o seguinte comando:

gitk

Será aberto o visualizador de relatório detalhado de alterações do git:



AULA 4 – DESFAZENDO OPERAÇÕES REALIZADAS

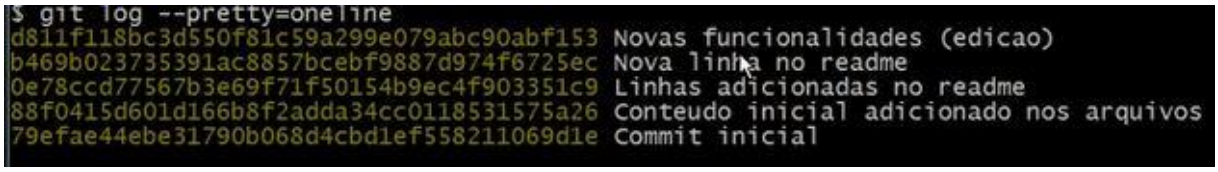
Adicionando um commit a outro já feito

git commit –a –m "Novas funcionalidades"

git commit –amend –m "Novas funcionalidades (edicao)"

Não cria um novo commit

git log –pretty=oneline
// exibe somente o hash e a mensagem de cada commit;



Removendo um arquivo do stage area

git reset HEAD <nome_arquivo>

Exemplo:

git reset HEAD novo.php

Revertendo um arquivo ao seu status original

Para descartar as mudanças feitas em um arquivo no diretório de trabalho:

git checkout -- <file>

Exemplo:

git checkout – novo.php

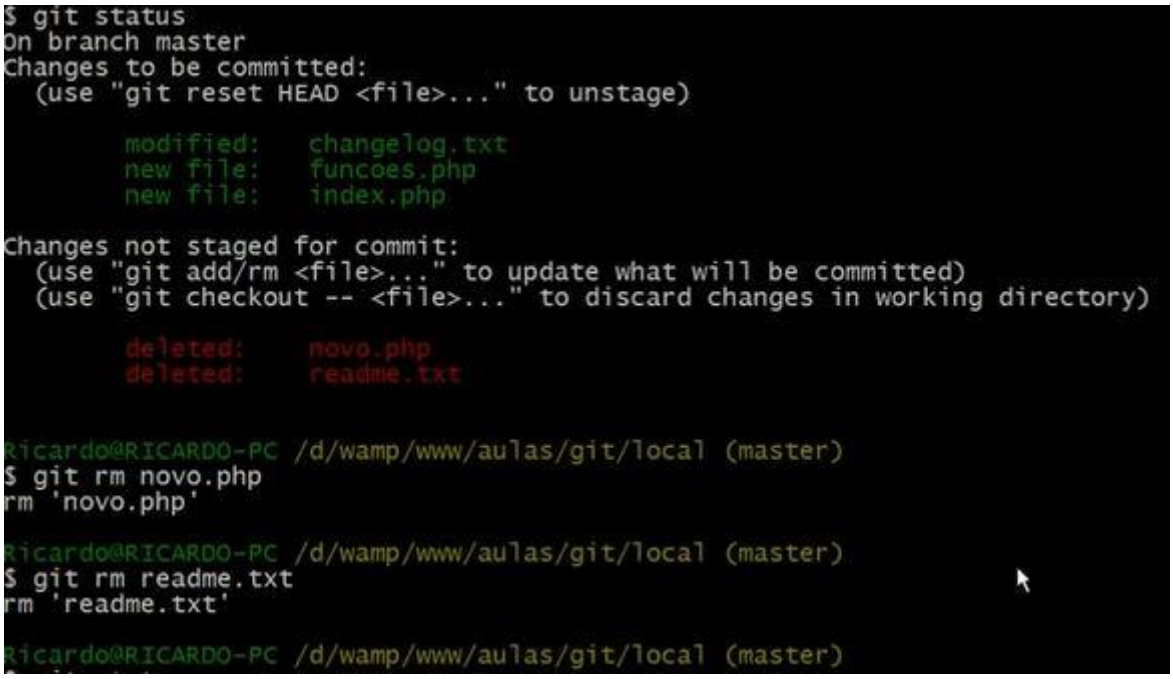
Reverte as alterações feitas dentro do arquivo.

Removendo arquivo do repositório

Para remover um arquivo do repositório:

git rm <arquivo>

Exemplo:


$ git status
On branch master
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
modified: changelog.txt
new file: funcoes.php
new file: index.php
Changes not staged for commit:
(use "git add/rm <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)
deleted: novo.php
deleted: readme.txt
Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)
$ git rm novo.php
rm 'novo.php'
Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)
$ git rm readme.txt
rm 'readme.txt'
Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)


$ git rm novo.php
rm 'novo.php'
Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)
$ git rm readme.txt
rm 'readme.txt'
Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)
$ git status
On branch master
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
modified: changelog.txt
renamed: novo.php -> funcoes.php
new file: index.php
deleted: readme.txt

AULA 5 – TAGS E BRANCHES

Tags

Uma tag é uma etiqueta, um ponto de atalho para um determinado status do sistema.

Geralmente as tags são usadas para criar marcações nas versões diferentes de um sistema.

Utilizando uma tag você tem um atalho, uma forma facilitada de trocar o status, de reverter o status de seu sistema para uma determinada tag ou poder fazer consultas rápidas no código fonte nas diferentes versões.

Exibindo as tags existentes

Para listar as tags que estão no sistema digite o seguinte comando.

git tag

Criando uma tag

git tag –a <nome_tag>

Exemplo:

git tag –a v1.0 –m "Versão 1.0"

-a cria uma tag anotada

Exemplo:

```
Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)
$ git log --pretty=oneline
ade64afaaa03d9aa718fc40231f7f8113635fafb Base do sistema criada
bb3861cf5b5e0a2ad7dcfbd4553bccd4ef133c8e Novo arquivo de funcoes
d811f118bc3d550f81c59a299e079abc90abf153 Novas funcionalidades (edicao)
b469b023735391ac8857bcebf9887d974f6725ec Nova linha no readme
0e78ccd77567b3e69f71f50154b9ec4f903351c9 Linhas adicionadas no readme
88f0415d601d166b8f2adda34cc0118531575a26 Conteudo inicial adicionado nos arquivos
79efae44ebe31790b068d4cbd1ef558211069d1e Commit inicial

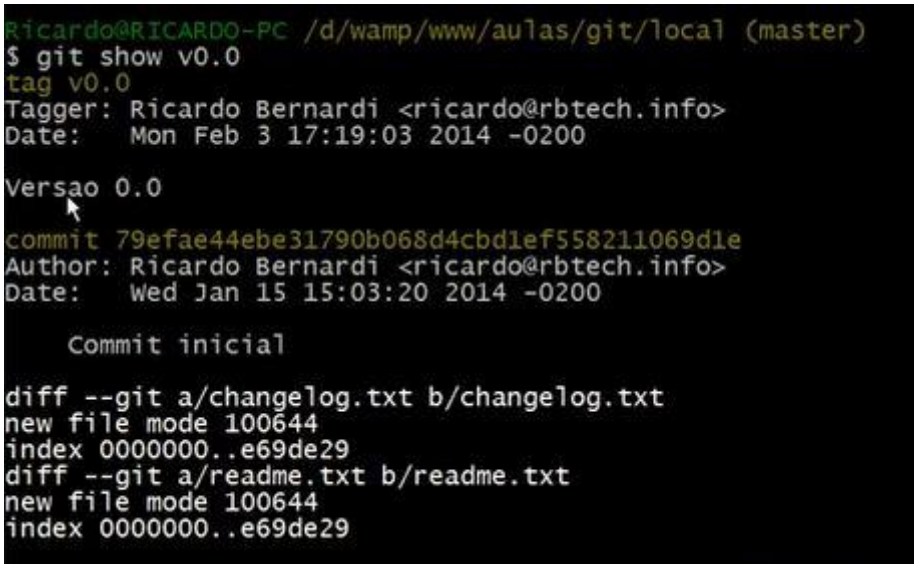
Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)
$ git tag -a v0.0 79efae44ebe31790b068d4cbd1ef558211069d1e -m "Versao 0.0"

Ricardo@RICARDO-PC /d/wamp/www/aulas/git/local (master)
$ git tag
v0.0
v1.0
```

Exibindo uma tag específica

git show <tag>

Exemplo:



Como usar as tags

git checkout v-0.0

Voltando ao padrão branch master

git checkout master

Deletando uma tag

git tag –d <nome_tag>

Exemplo:

git tag –d v0.0

Tag é um ponteiro para um ponto específico do sistema, porém para fazer testes, criar uma nova versão não é interessante utilizar as tags para controlar isso ou para criar ambientes diferentes de desenvolvimento, por exemplo. Neste caso é melhor utilizar os branches.que são as ramificações dentro de um controle de versão.

Branches

O branch permite que se trabalhe com várias ramificações, com várias segmentações de sistemas diferentes e você pode fazer commits em uma ramificação e esse commit não irá alterar outra ramificação.

Por padrão quando se cria um repositório no Git ele cria um branch chamado master. Esse é o branch padrão dele. É o branch principal e muitas vezes o único em um trabalho. O branch cria uma cópia de segurança.

Criando um ambiente de testes para o sistema

Digite o comando:

git branch teste

Para fazer a transição dos arquivos para o branch de teste, ou seja, para colocar os arquivos no working directory, digite o comando:

git checkout teste

Ou para fazer direto os dois comandos acima, digite:

git checkout –b teste

Ele cria o branch teste e já faz a troca para esse branch

Agora já é possível trabalhar no ambiente de testes normalmente.

Trocando o sistema para o branch master

git checkout master

O sistema branch master tem o sistema que está rodando no cliente.

Fazendo um merge do branch de teste

origem: teste
destino: master

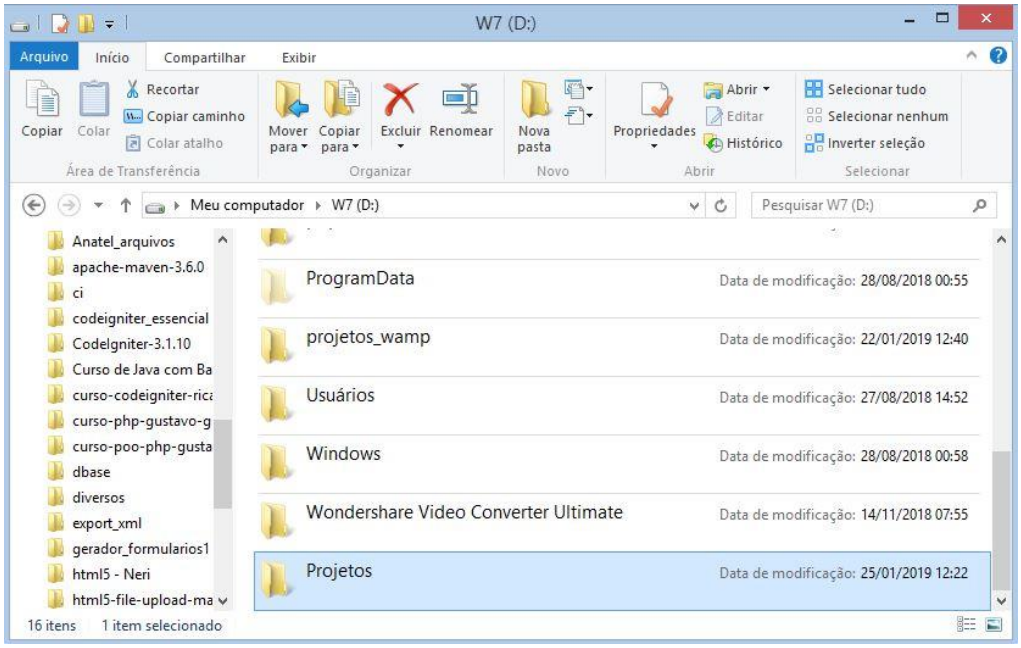
git merge teste

Deletando o branch de teste

git branch –d teste

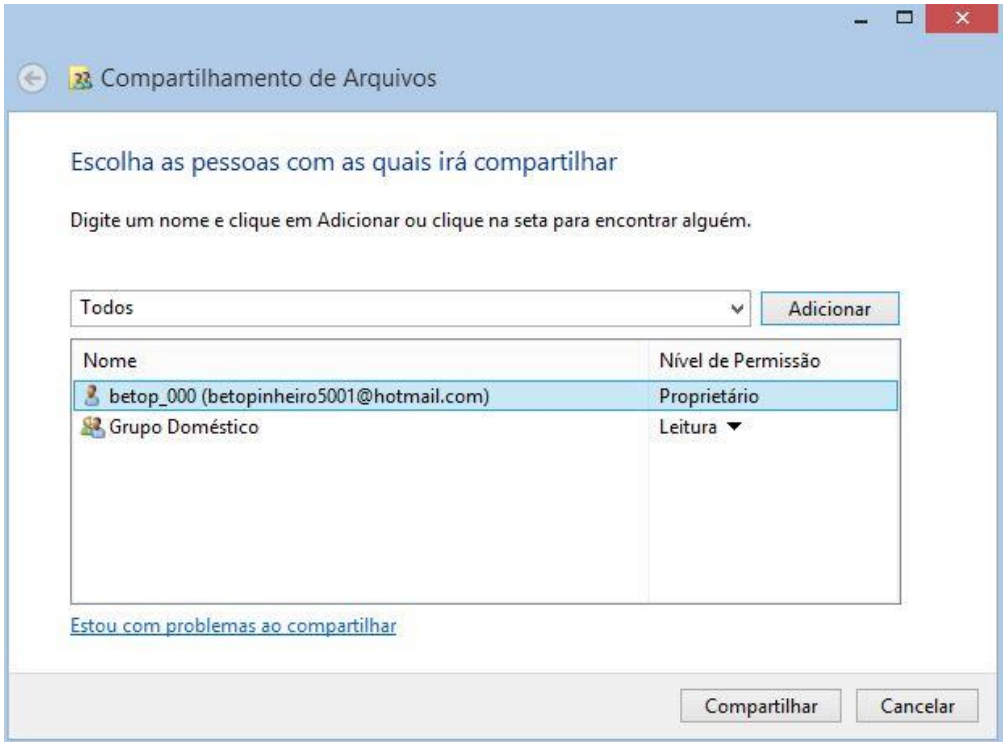
AULA 6 – GIT EM REDE LOCAL

Criar uma pasta chamada projetos no disco local D:



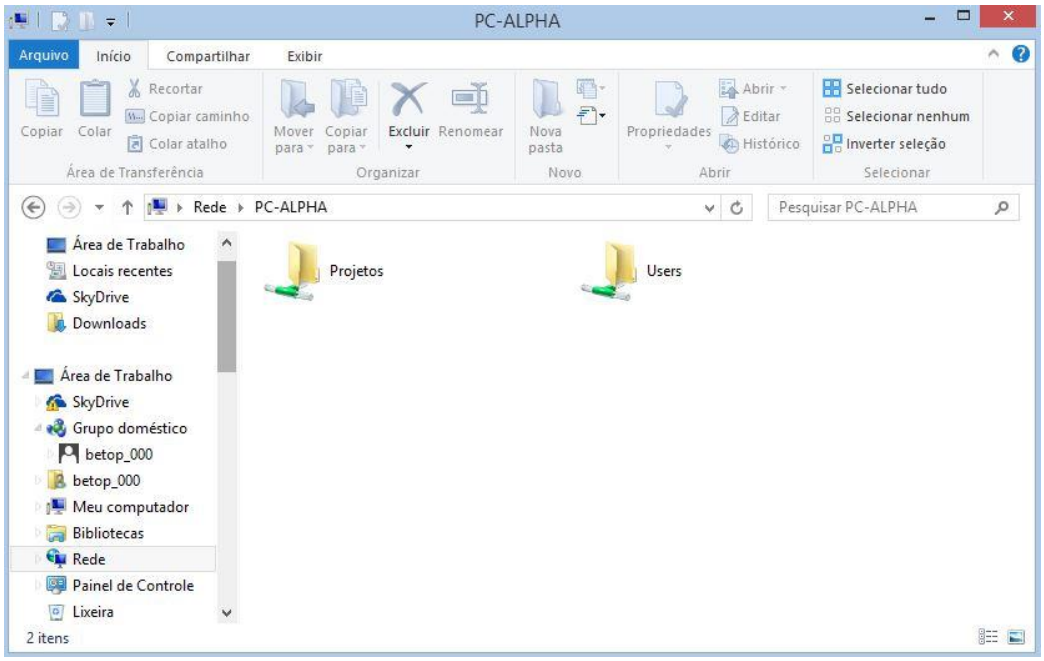
Nessa pasta, dar acesso via rede a ela, para conseguir acessar essa máquina pela rede.

Clique com o botão direito sobre a pasta "Projetos" e selecione "Compartilhar com → Pessoas específicas" e Adicione o nome "Todos".

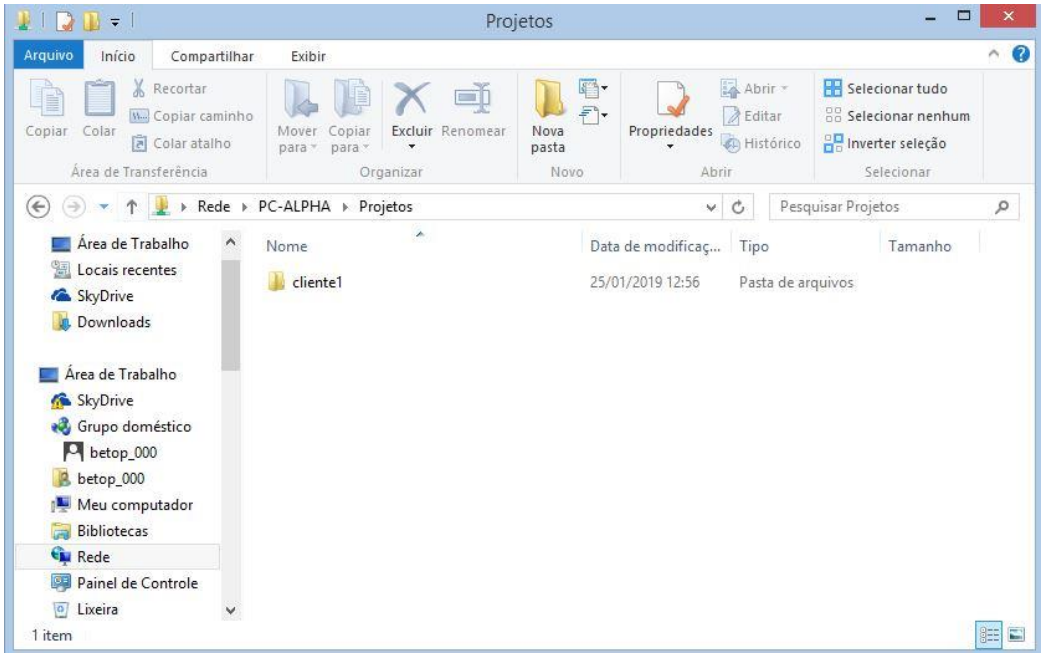


Em nível de permissão, selecionar para esse grupo, "Leitura/Gravação" e clicar no botão "Compartilhar".

Em "Meu computador" digitar: \\PC-ALPHA (onde PC-ALPHA é o nome do computador) e pressionar <enter>



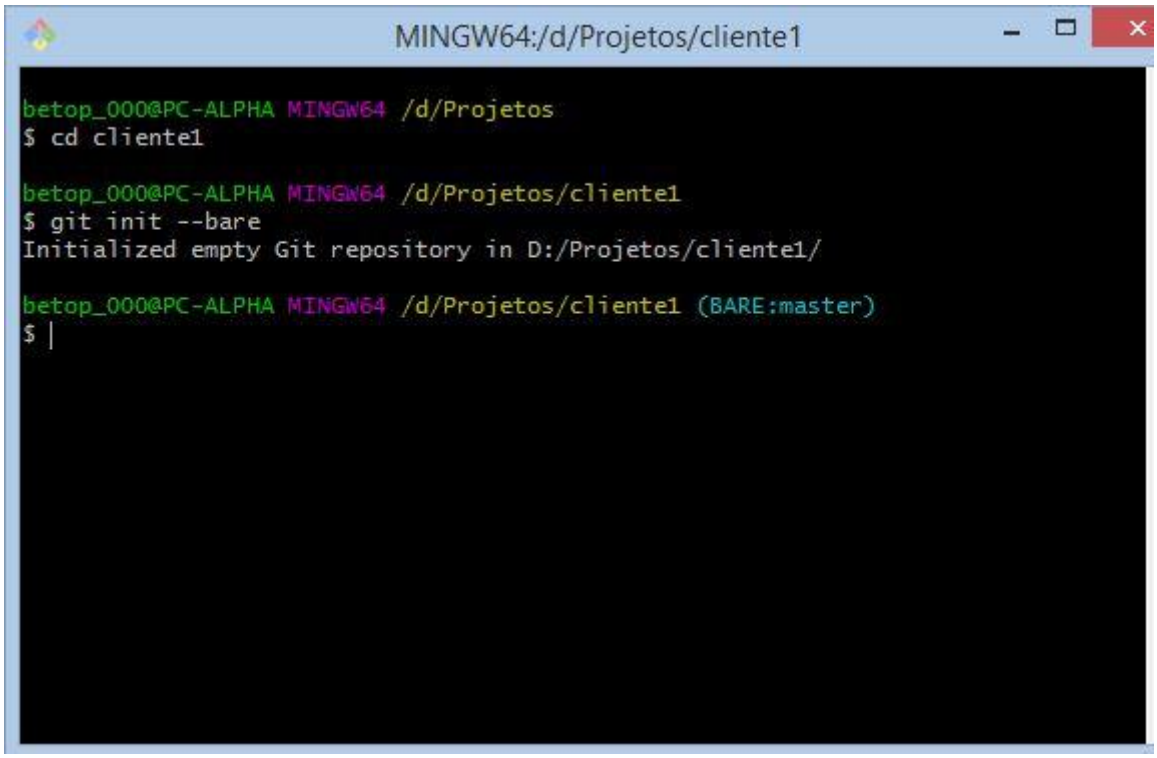
Clique na pasta "Projetos" e crie uma pasta (por exemplo: cliente1).

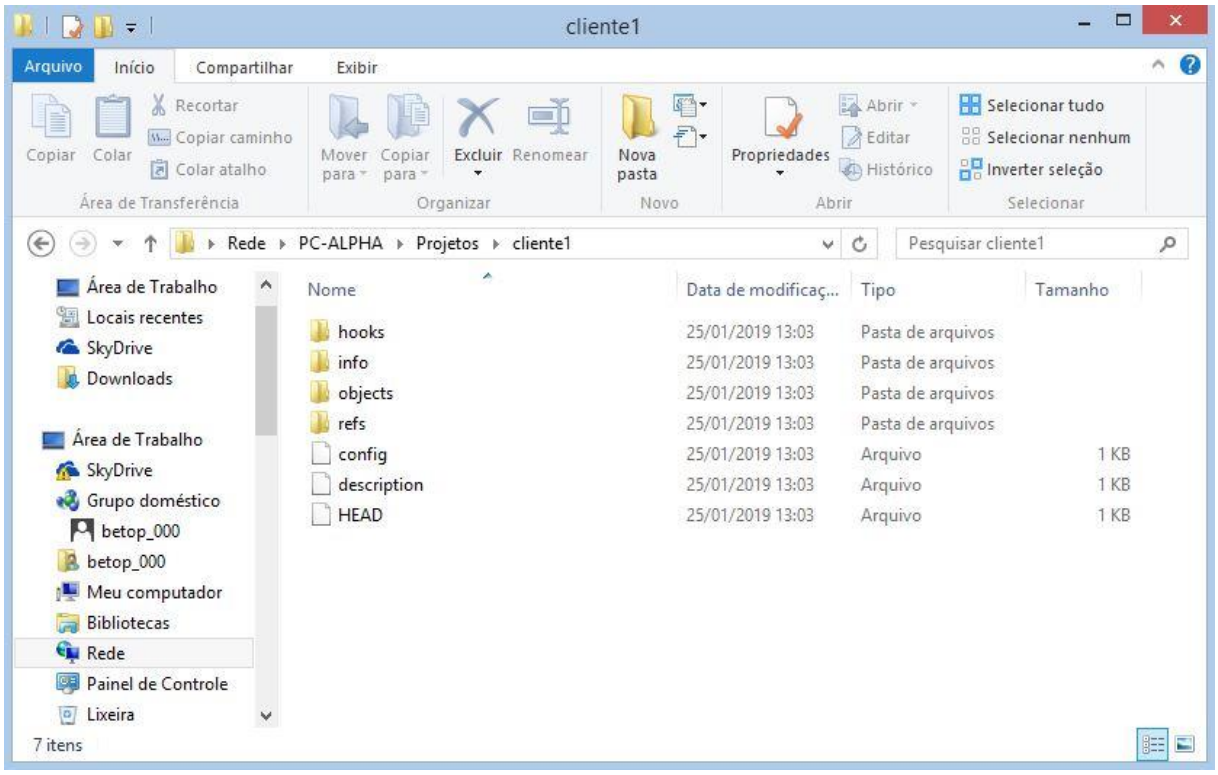


Inicie o git nessa página.

Para iniciar um repositório que será disponibilizado para outras máquinas de sua rede local, entre com o comando:

git init --bare

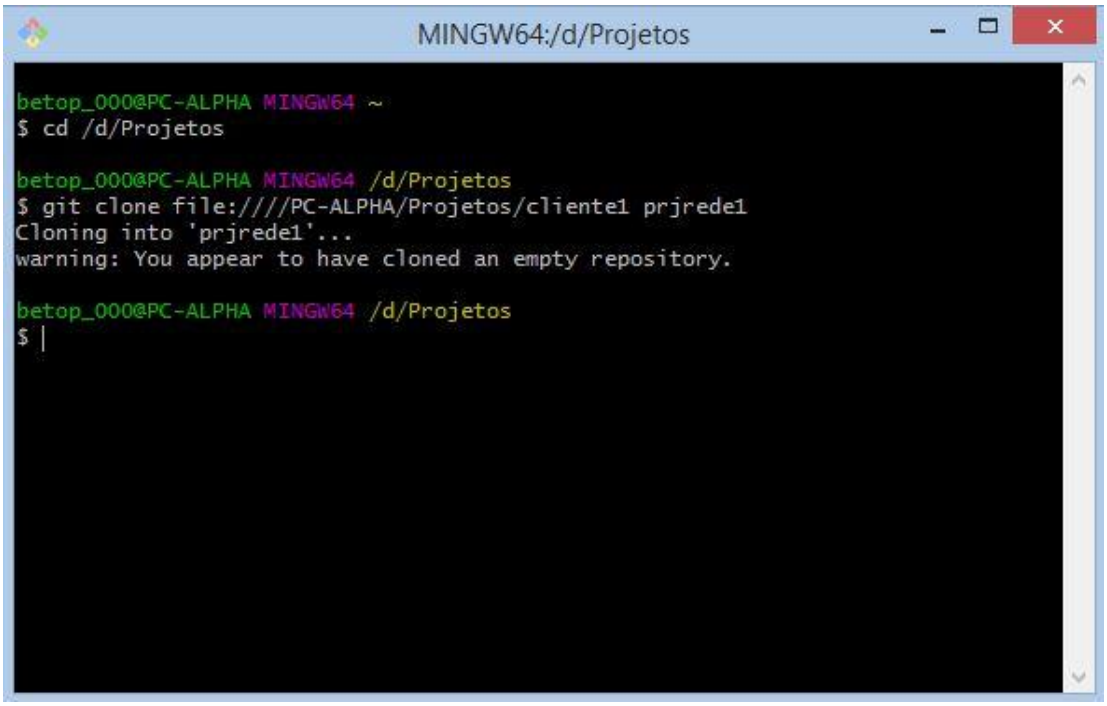


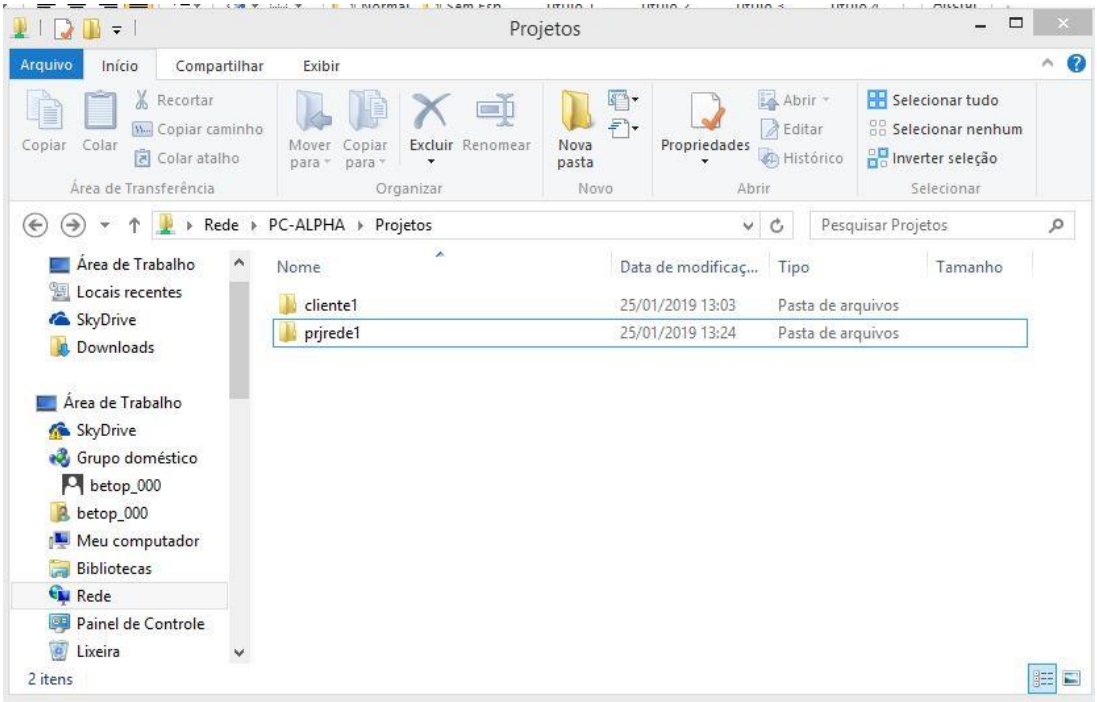


Cada estação de trabalho que vai trabalhar no projeto do cliente1 precisa, a primeira vez, fazer um clone.

git clone file:////PC-ALPHA/Projetos/cliente1 prjrede1

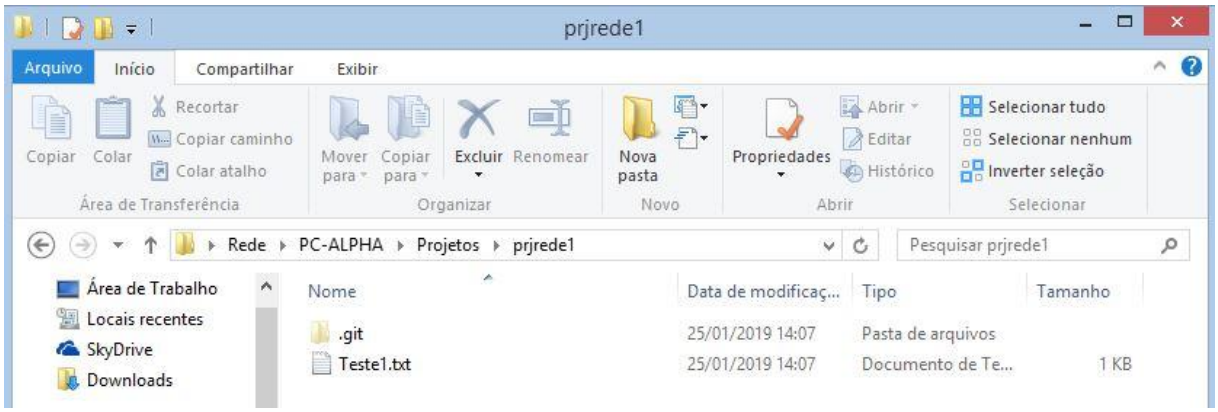
PC-ALPHA é o servidor
prjrede1 renomeia cliente1





- Insira dentro da pasta prjrede1 um arquivo chamado Teste1.txt com o seguinte texto:

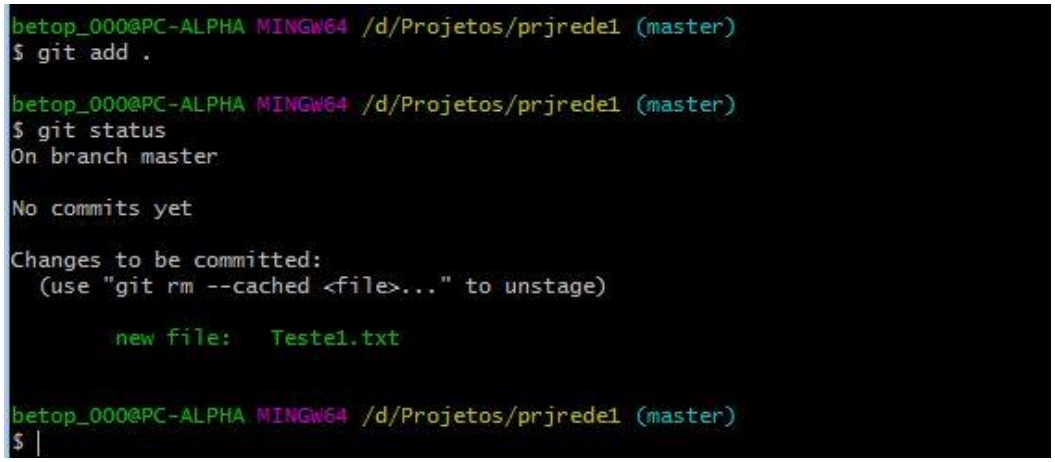
Isso é apenas um teste com o GIT/GITHUB



- Com o arquivo Teste1.txt dentro da pasta prjrede1 utilize os comandos:

git status

git add .



git status

```
betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   Teste1.txt


betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$
```

git commit –m "Projeto pronto para iniciar a fase de desenvolvimento"

```
betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$ git commit -m "Projeto pronto para iniciar a fase de desenvolvimento"
[master (root-commit) 8cf21c7] Projeto pronto para iniciar a fase de desenvolvim
ento
 1 file changed, 1 insertion(+)
 create mode 100644 Teste1.txt

betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$
```

git push origin master

```
betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 288 bytes | 288.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To file:////PC-ALPHA/Projetos/cliente1
 * [new branch]      master -> master

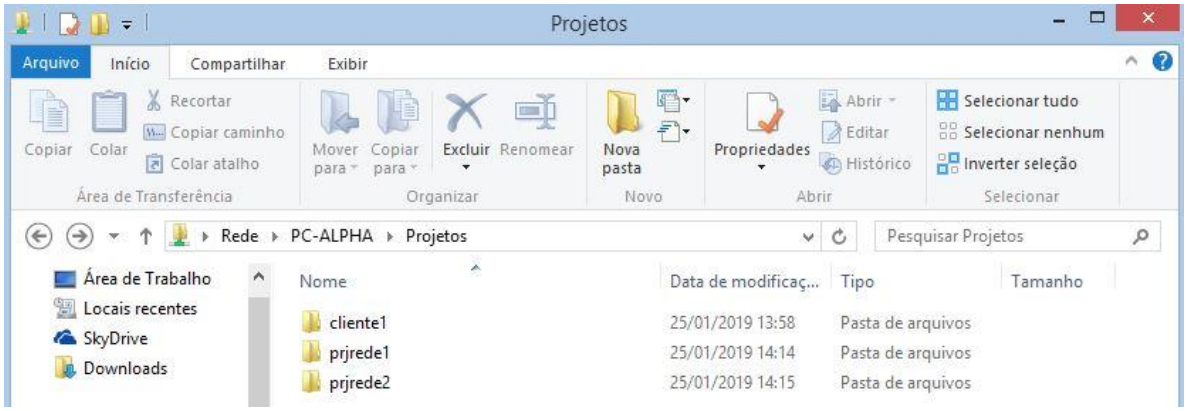
betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$
```

- Clonar novamente o projeto chamando-o de prjrede2:

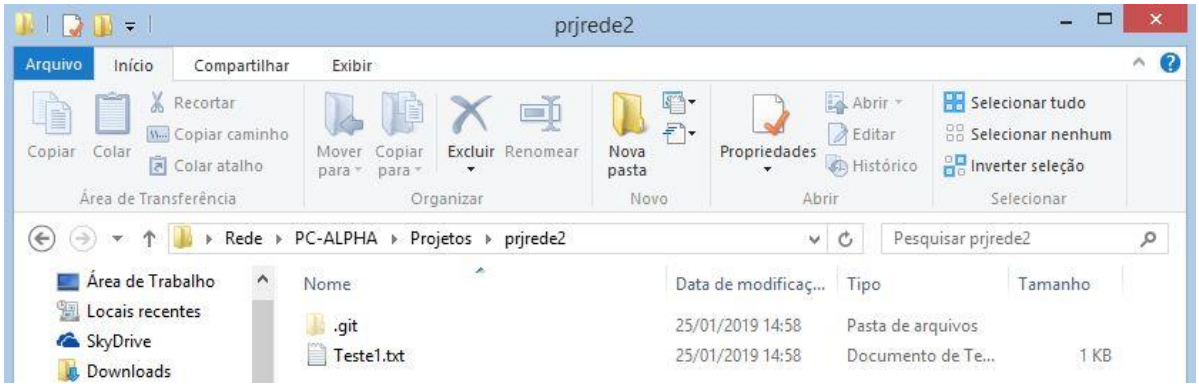
git clone file:////PC-ALPHA/Projetos/cliente1 prjrede2

```
betop_000@PC-ALPHA MINGW64 /d/Projetos
$ git clone file:////PC-ALPHA/Projetos/cliente1 prjrede2
Cloning into 'prjrede2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.

betop_000@PC-ALPHA MINGW64 /d/Projetos
$
```



Clicando na pasta prjrede2 já será possível ver o arquivo Teste1.txt



Em prjrede1 insira um arquivo chamado Teste2.txt com o seguinte conteúdo:

Outro teste no sistema.

Esse arquivo ainda não aparecerá em prjrede2.

No gitbash acesse prjrede1 e entre com os seguintes comandos:

git status

```
betop_000@PC-ALPHA MINGW64 /d/Projetos
$ cd prjrede1

betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

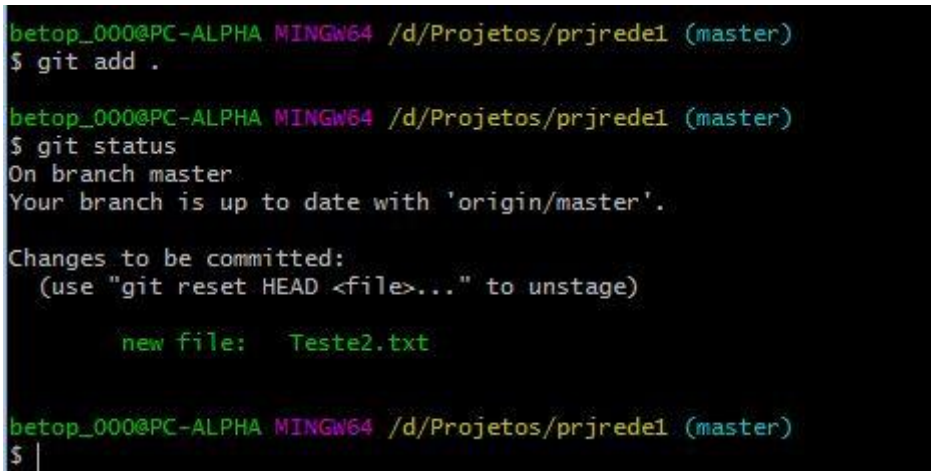
        Teste2.txt

nothing added to commit but untracked files present (use "git add" to track)

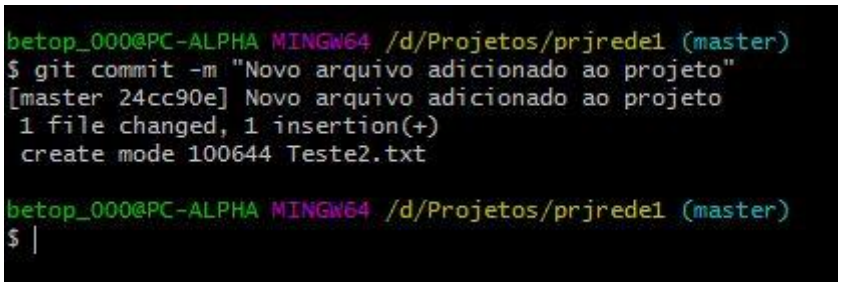
betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede1 (master)
$
```

git add .

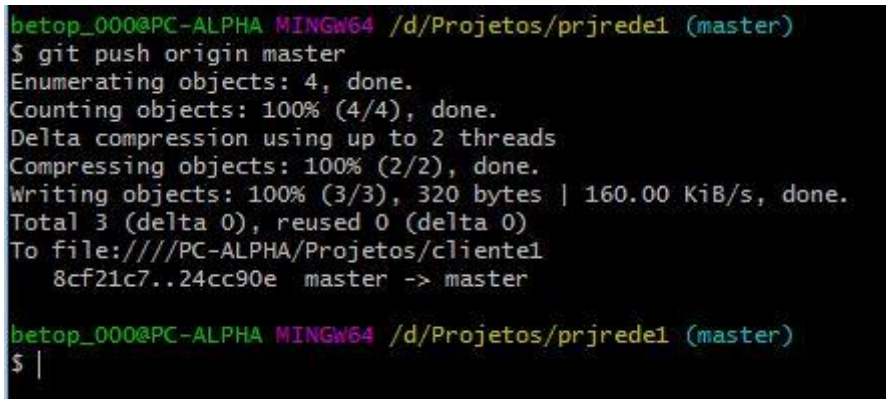
git status



git commit –m "Novo arquivo adicionado ao projeto"



git push origin master



Informe o usuário da rede prjrede2 que foi enviado esse arquivo para o servidor.

Para receber o arquivo, esse usuário, em prjrede2, deve entrar com o seguinte comando:

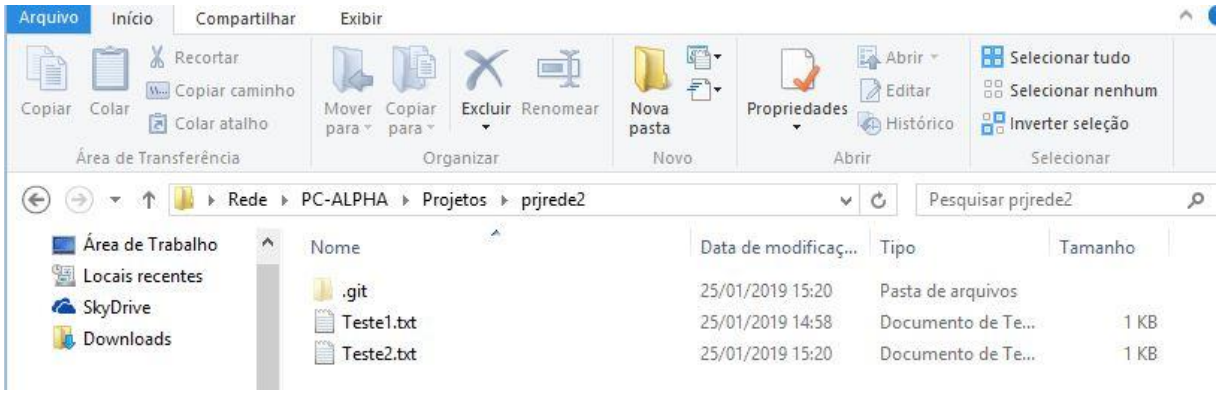
git pull origin master

```
betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede2 (master)
$ dir
Teste1.txt

betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede2 (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From file:////PC-ALPHA/Projetos/cliente1
 * branch            master     -> FETCH_HEAD
   8cf21c7..24cc90e  master     -> origin/master
Updating 8cf21c7..24cc90e
Fast-forward
 Teste2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Teste2.txt

betop_000@PC-ALPHA MINGW64 /d/Projetos/prjrede2 (master)
$
```

Agora sim, o arquivo irá aparecer na pasta prjrede2:



O Git faz automaticamente um merge dos dados que estão no servidor com o seu repositório atual, com seu diretório atual.

Quando você não quiser fazer isso é preciso ter um branch separado e fazer um fech.

Aí ao invés de entrar com o comando:

git pull origin master

Entre com o comando:

git fetch origin <branch>

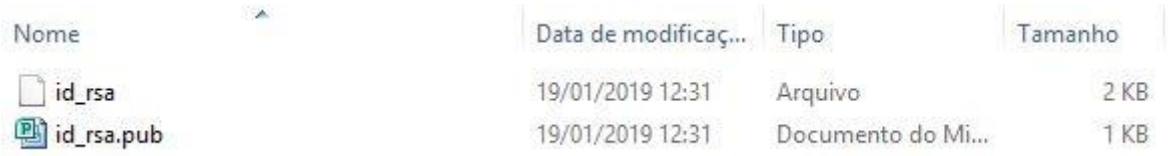
Não será feito o merge automático;

AULA 7 – TRABALHANDO COM O GITHUB E O GIT

Gerando um par de chaves SSH

Para gerar um par de arquivos SSH, entre com o comando:

ssh-keygen

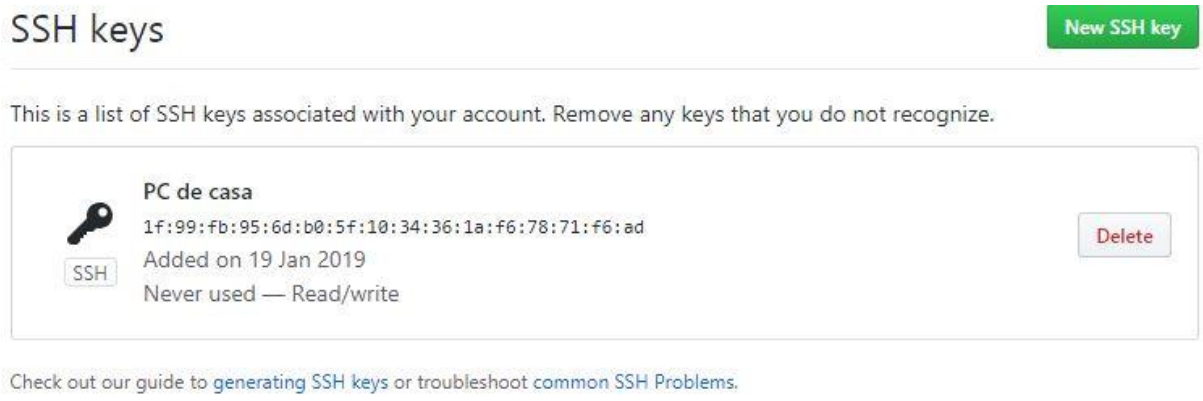


Abrir o arquivo id_rsa.pub em um editor de textos (por exemplo: notepad ++) e copiar o conteúdo para a área de trabalho.

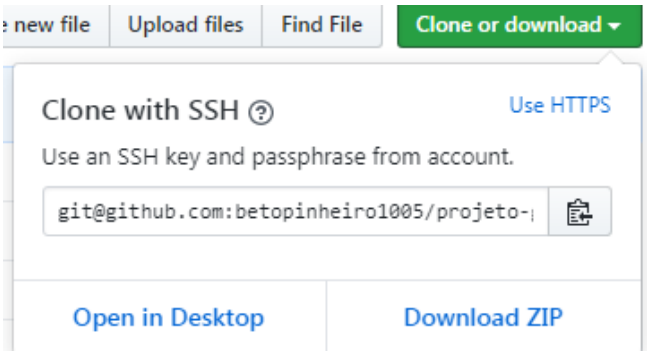
No GitHub clicar em "Settings" e escolher a opção SSH and GPG Keys.

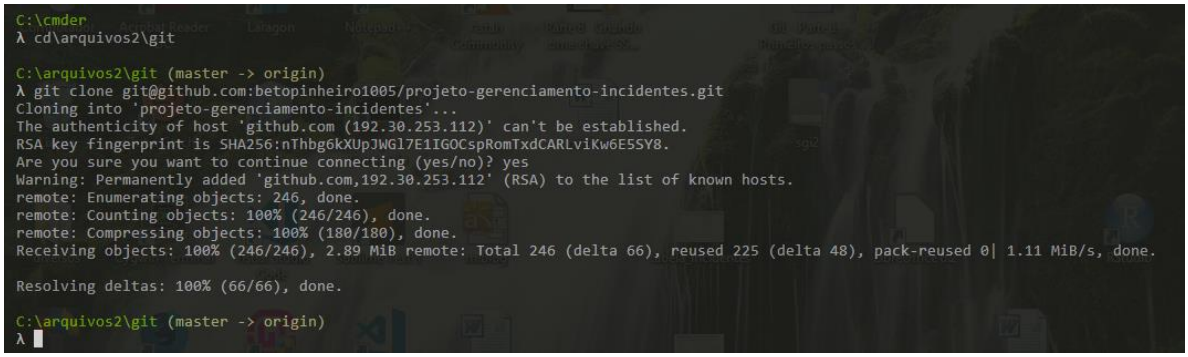
Clicar no botão New SSH Key. No campo title, digitar o título da chave (por exemplo: "PC de casa") e no campo key colar a chave copiado para a área de trabalho.

Finalmente clicar no botão Add Key.



Isso irá dar autorização para acessar o meu GitHub através do git bash local.





Para testar:

ssh -T git@github.com

Em GitHub criar um novo repositório (repositório público ou privado).

As contas gratuitas do GitHub agora podem criar repositórios privados!

Criando um repositório no GitHub

No GitHub crie um novo repositório com o nome, por exemplo, de:
curso-ajax-joao-ribeiro

Copie o endereço do repositório e no Git bash, na pasta de trabalho, entre com o seguinte comando:

$ git clone https://github.com/betopinheiro1005/curso-ajax-joao-ribeiro

```
betop_000@PC-ALPHA MINGW64 ~
$ cd /c/git

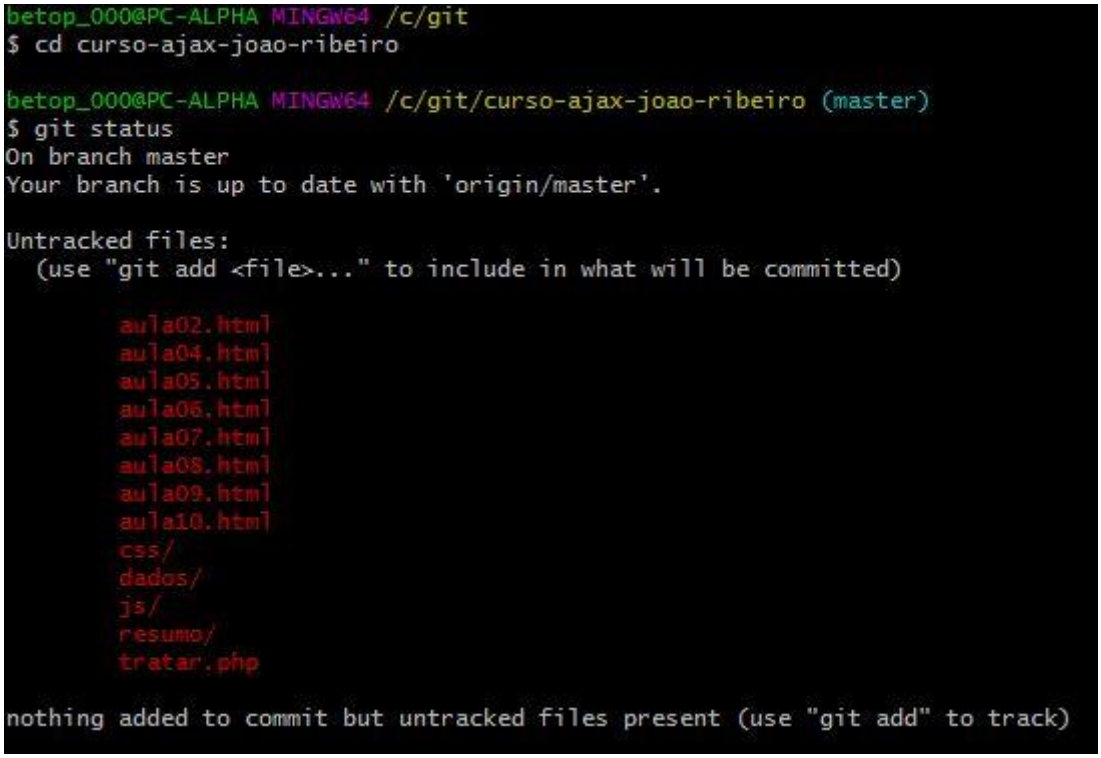
betop_000@PC-ALPHA MINGW64 /c/git
$ git clone https://github.com/betopinheiro1005/curso-ajax-joao-ribeiro
Cloning into 'curso-ajax-joao-ribeiro'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
```

Dentro da pasta C:\git será criada a subpasta curso-ajax-joao-ribeiro e dentro desta pasta será criada uma pasta oculta com nome .git

Copie os arquivos do curso para esta pasta.

Na pasta de trabalho (curso-ajax-joao-ribeiro), digite o comando:

git status

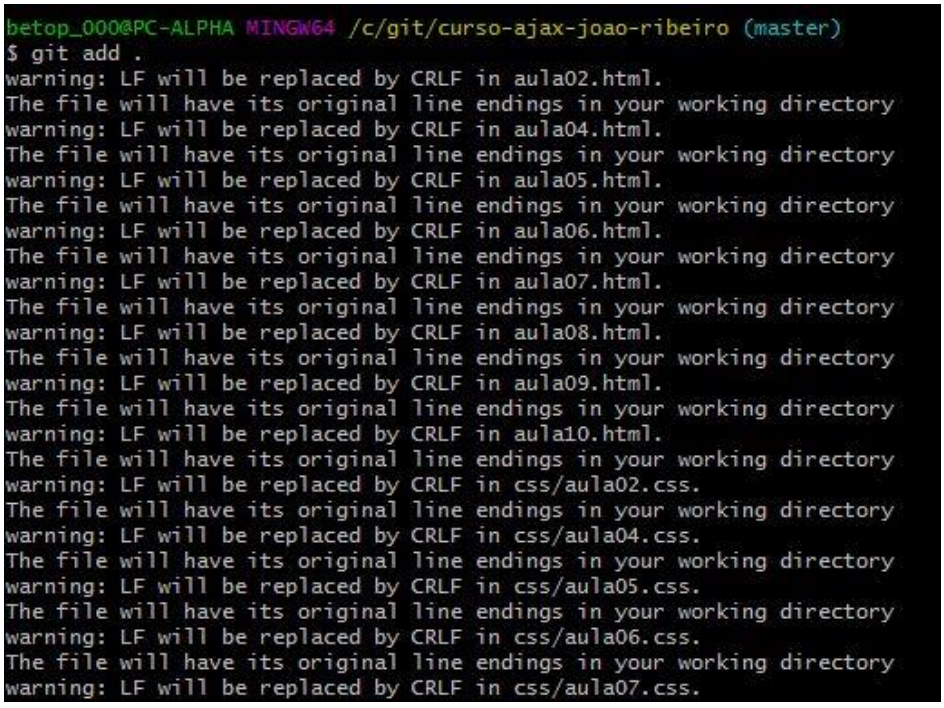


É informado que existem arquivos para serem trackeados.

Entre com o comando:

git add .

Os arquivos irão para a stage area.



Entre novamente com o comando:

git status

```
betop_OOO@PC-ALPHA MINGW64 /c/git/curso-ajax-joao-ribeiro (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   aula02.html
        new file:   aula04.html
        new file:   aula05.html
        new file:   aula06.html
        new file:   aula07.html
        new file:   aula08.html
        new file:   aula09.html
        new file:   aula10.html
        new file:   css/aula02.css
        new file:   css/aula04.css
        new file:   css/aula05.css
        new file:   css/aula06.css
        new file:   css/aula07.css
        new file:   css/aula08.css
        new file:   css/aula09.css
        new file:   css/aula10.css
        new file:   css/estilos.css
        new file:   dados/dados.html
        new file:   dados/dados.txt
        new file:   dados/dados.xml
        new file:   dados/dados1.txt
```

Faça o commit com o comando:

git commit –m "Primeiro commit das aulas"

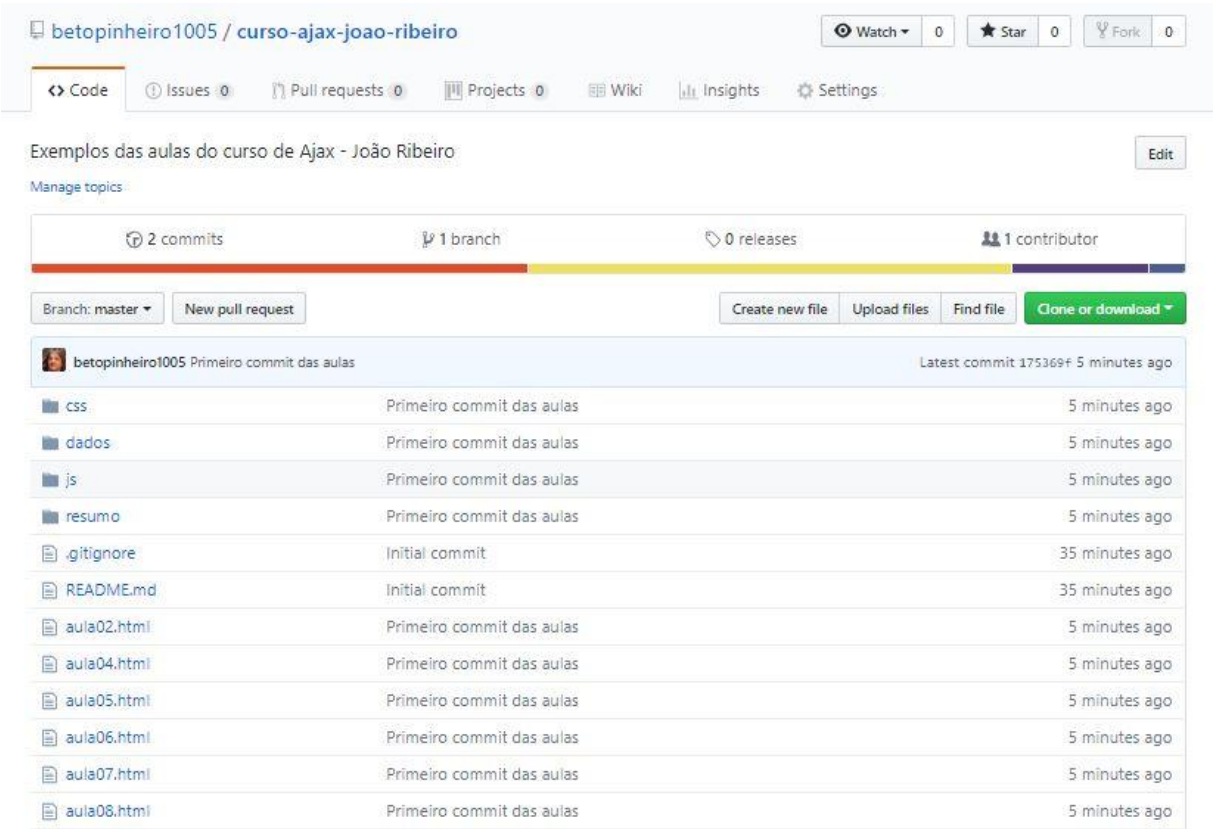
```
betop_OOO@PC-ALPHA MINGW64 /c/git/curso-ajax-joao-ribeiro (master)
$ git commit -m "Primeiro commit das aulas"
[master 175369f] Primeiro commit das aulas
 35 files changed, 10823 insertions(+)
 create mode 100644 aula02.html
 create mode 100644 aula04.html
 create mode 100644 aula05.html
 create mode 100644 aula06.html
 create mode 100644 aula07.html
 create mode 100644 aula08.html
 create mode 100644 aula09.html
 create mode 100644 aula10.html
 create mode 100644 css/aula02.css
 create mode 100644 css/aula04.css
 create mode 100644 css/aula05.css
 create mode 100644 css/aula06.css
 create mode 100644 css/aula07.css
 create mode 100644 css/aula08.css
 create mode 100644 css/aula09.css
 create mode 100644 css/aula10.css
 create mode 100644 css/estilos.css
 create mode 100644 dados/dados.html
 create mode 100644 dados/dados.txt
 create mode 100644 dados/dados.xml
 create mode 100644 dados/dados1.txt
 create mode 100644 dados/dados2.txt
 create mode 100644 dados/dados_lorem.txt
```

Para enviar os arquivos para o GitHub entre com o seguinte comando:

git push origin master

```
betop_000@PC-ALPHA MINGW64 /c/git/curso-ajax-joao-ribeiro (master)
$ git push origin master
Enumerating objects: 41, done.
Counting objects: 100% (41/41), done.
Delta compression using up to 2 threads
Compressing objects: 100% (36/36), done.
Writing objects: 100% (40/40), 479.95 KiB | 5.27 MiB/s, done.
Total 40 (delta 11), reused 0 (delta 0)
remote: Resolving deltas: 100% (11/11), done.
To https://github.com/betopinheiro1005/curso-ajax-joao-ribeiro
   6893ccb..175369f  master -> master
```

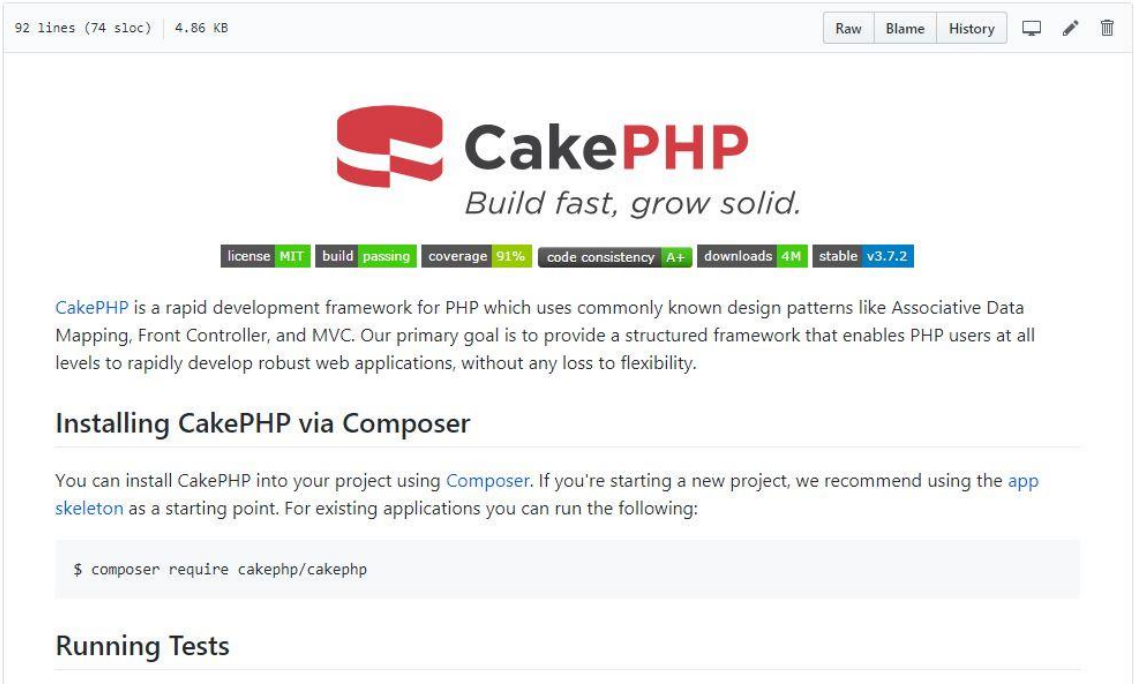
Entre no repositório do GitHub e confirme se os arquivos foram enviados:



Personalizando o arquivo readme.md

É possível inserir links e imagens nesse arquivo.

Entre no arquivo readme.md do cakePHP.



Clique no botão Raw. Irá aparecer o código.

```
<p align="center">
  <a href="https://cakephp.org/" target="_blank" >
    <img alt="CakePHP" src="https://cakephp.org/v2/img/logos/CakePHP_Logo.svg" width="400" />
  </a>
</p>
<p align="center">
    <a href="LICENSE" target="_blank">
        <img alt="Software License" src="https://img.shields.io/badge/license-MIT-brightgreen.svg?style=flat-square">
    </a>
    <a href="https://travis-ci.org/cakephp/cakephp" target="_blank">
        <img alt="Build Status" src="https://img.shields.io/travis/cakephp/cakephp/master.svg?style=flat-square">
    </a>
    <a href="https://codecov.io/github/cakephp/cakephp" target="_blank">
        <img alt="Coverage Status" src="https://img.shields.io/codecov/c/github/cakephp/cakephp.svg?style=flat-square">
    </a>
    <a href="https://squizlabs.github.io/PHP_CodeSniffer/analysis/cakephp/cakephp/" target="_blank">
        <img alt="Code Consistency" src="https://squizlabs.github.io/PHP_CodeSniffer/analysis/cakephp/cakephp/grade.svg">
    </a>
    <a href="https://packagist.org/packages/cakephp/cakephp" target="_blank">
        <img alt="Total Downloads" src="https://img.shields.io/packagist/dt/cakephp/cakephp.svg?style=flat-square">
    </a>
    <a href="https://packagist.org/packages/cakephp/cakephp" target="_blank">
        <img alt="Latest Stable Version" src="https://img.shields.io/packagist/v/cakephp/cakephp.svg?style=flat-square&label=stable">
    </a>
</p>

[CakePHP](https://cakephp.org) is a rapid development framework for PHP which
uses commonly known design patterns like Associative Data
Mapping, Front Controller, and MVC.  Our primary goal is to provide a structured
framework that enables PHP users at all levels to rapidly develop robust web
applications, without any loss to flexibility.

## Installing CakePHP via Composer

You can install CakePHP into your project using
[Composer](https://getcomposer.org).  If you're starting a new project, we
recommend using the [app skeleton](https://github.com/cakephp/app) as
a starting point. For existing applications you can run the following:

``` bash
```

Basta usá-lo como base para fazer alterar o seu arquivo readme.md
```

AULA 8 – COLABORANDO COM PROJETOS GITHUB

Conceito de fork e pull request

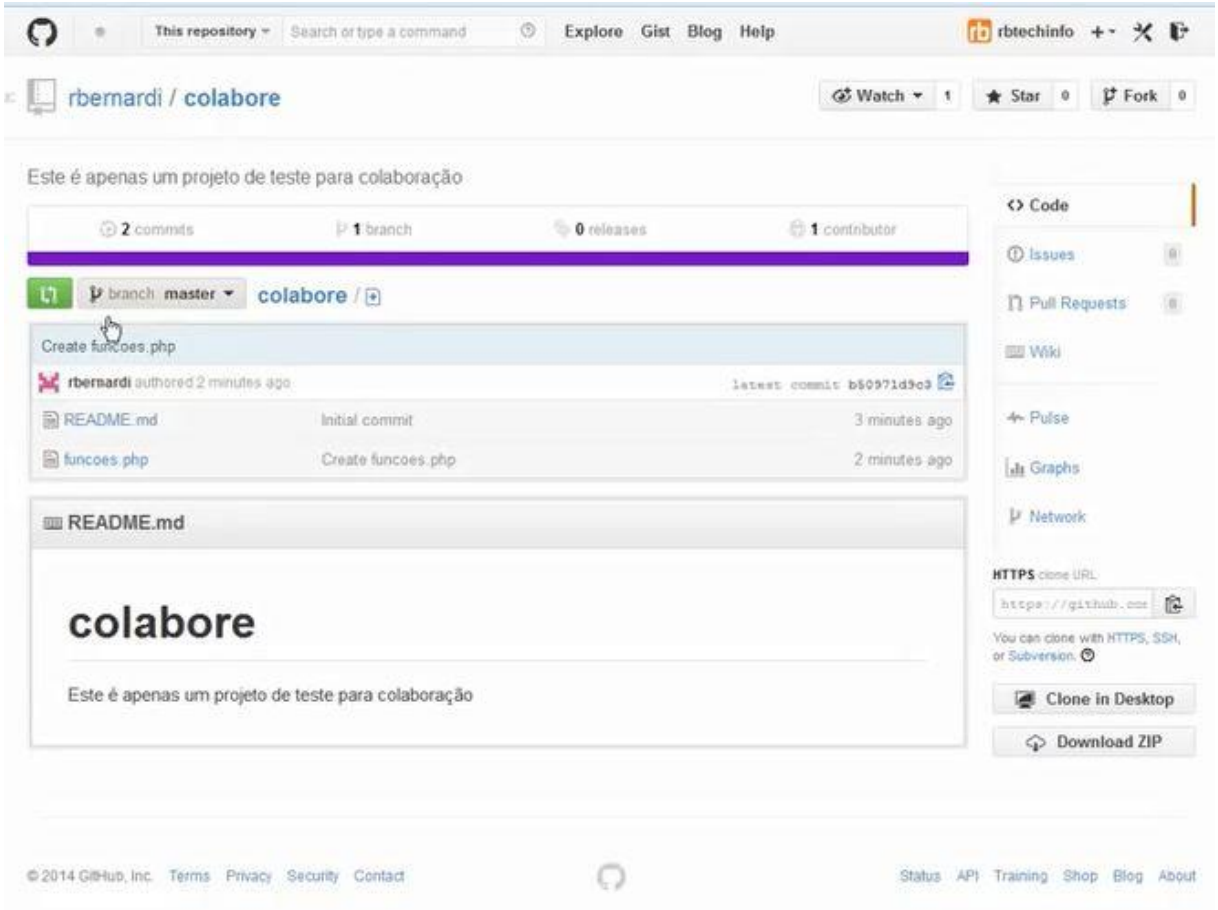
Fork é o procedimento que se faz para clonar aquele repositório para o qual você quer colaborar dentro da sua conta do GitHub.

Depois que fizer todas as alterações normais nos repositórios que estão na sua conta faça um pull request, ou seja, envie tudo que fez de alterações para que o autor do projeto avalie e decida se vai colocar isso com o projeto original, ou não.

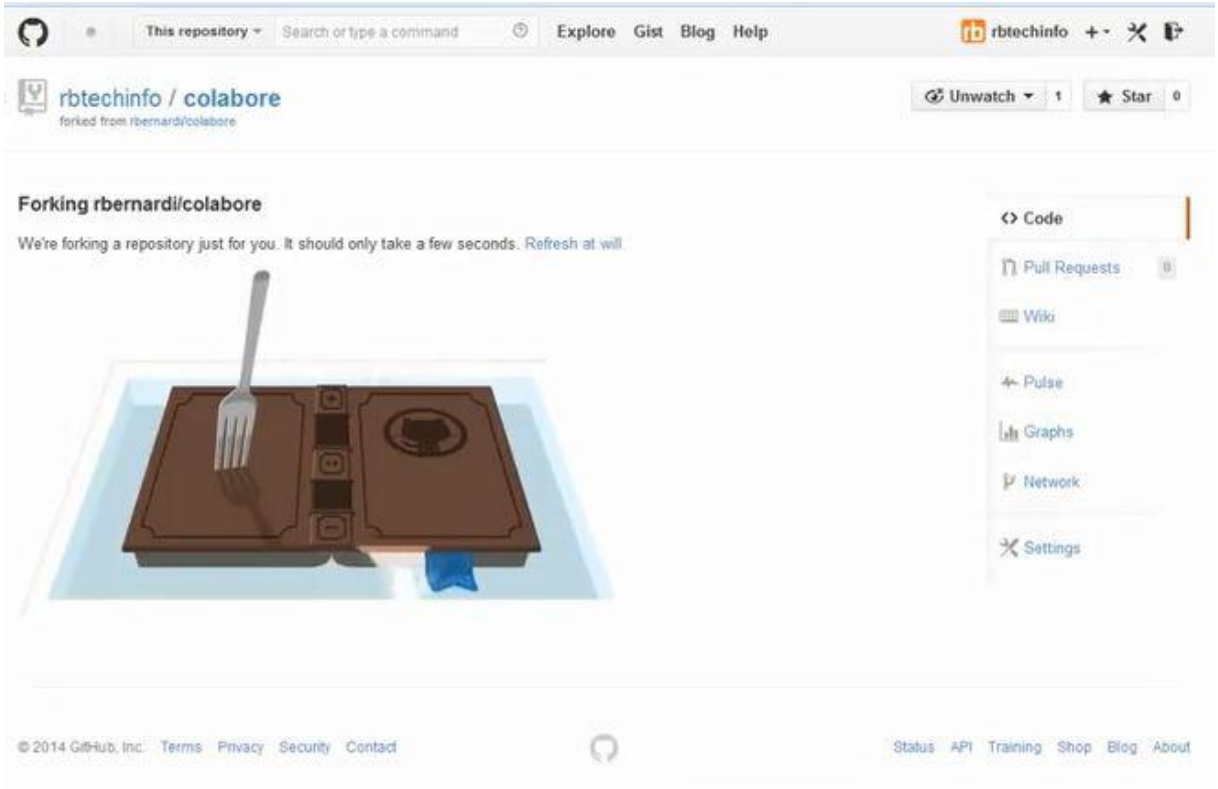
Procedimento

Na URL do projeto que se deseja colaborar copie o nome de usuário e do projeto.

Cole na sua página no GitHub e pressionar a tecla < enter >.



Clique no botão fork.



Depois que tudo estiver pronto você será redirecionado para a tela do projeto já dentro do seu repositório.

Faça um clone do projeto para os seus repositórios.

Agora basta fazer as alterações no projeto e enviar para o autor. Depois:

1. Clique no botão Pull Request.
2. Em seguida, clique no botão New Pull Request.
3. Clique no botão Create Pull Request.
4. Dê um título para a requisição que será enviada e faça um comentário das alterações realizadas no projeto.
5. Finalmente clique no botão Send Pull Request.

O trabalho de colaboração está concluído. Cabe ao autor aprovar ou não.

O autor, ao abrir a lista de Pull Request, vai visualizar o que foi enviado e para aceitar a colaboração mesclando ao projeto basta clicar no botão Merge Pull Request.