

CIVECCES
1º CICLO VIRTUAL DE PALESTRAS DE
ENGENHARIA DE COMPUTAÇÃO E
ENGENHARIA DE SOFTWARE DA UEPG

Workshop Introdução ao GIT

Profº Luiz Pedro Petroski

petroskilp@gmail.com

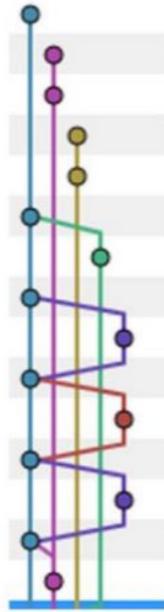
Introdução



QUANDO VOCÊ
PERCEBE QUE DÁ
PRA FAZER UM
SOLO DE GUITARRA
COM O SEU
REPOSITÓRIO...

VIDA DE
PROGRAMADOR
.com.br

(IMAGEM REAL DE REPOSITÓRIO
ENVIADA POR DAGNELSON)



fonte: <https://vidadeprogramador.com.br/2016/09/22/gitar-hero/>

Controle de Versão

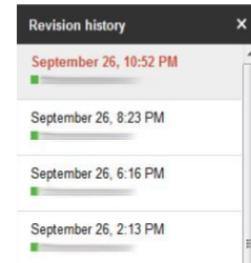
- Como que você faz o controle de versão de um arquivo? E o compartilhamento?

Controle de Versão

- Como que você faz o controle de versão de um arquivo? E o compartilhamento?

 Crash_Course_on_GIT_and_GitHub	29-Apr-15 23:55
 Crash_Course_on_GIT_and_GitHub	30-Apr-15 01:11
 Crash_Course_on_GIT_and_GitHub_rev01	30-Apr-15 09:11
 Crash_Course_on_GIT_and_GitHub_rev02	30-Apr-15 13:27
 Crash_Course_on_GIT_and_GitHub_rev03	30-Apr-15 20:29
 Crash_Course_on_GIT_and_GitHub_rev04	05-May-15 13:36
 Crash_Course_on_GIT_and_GitHub_rev05	07-May-15 19:21

Google docs



Dropbox

...e desenvolvimento de software?

Mudanças

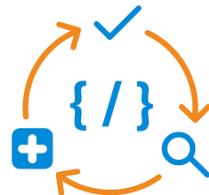
- não são **analisadas** antes de ser feitas
- não são **registradas** antes de ser implementadas
- não são **relatadas** àqueles que precisam saber
- ou não são **controladas** de forma que melhore a qualidade e reduza erros



Gerenciamento de Configuração de Software (GCS)

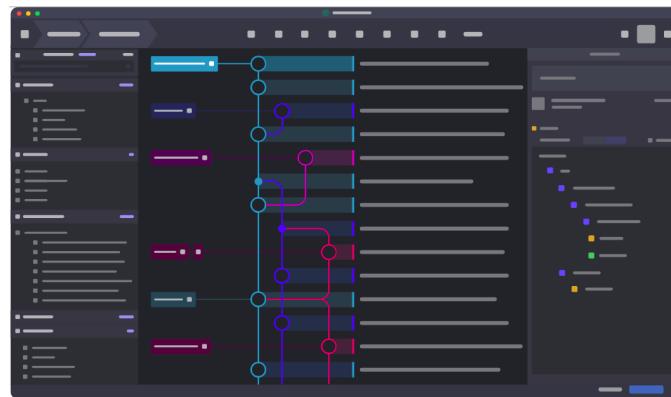
Durante o desenvolvimento do software queremos saber:

- O que mudou e quando? (controle de versão)
- Por que mudou? (controle de mudanças)
- Quem fez a mudança? (auditoria de configuração)
- Podemos reproduzir esta mudança? (auditoria de configuração)



Gerenciamento de Configuração de Software

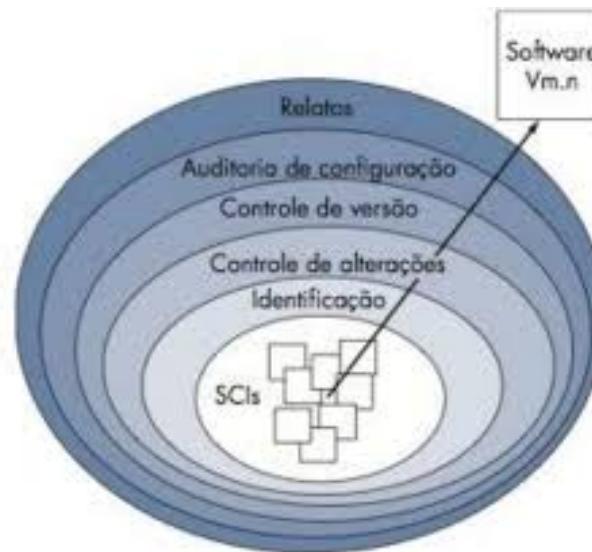
- Coordenar o desenvolvimento para minimizar a confusão!
- “Arte de identificar, organizar e controlar modificações no software que está sendo criado, maximizando a produtividade e reduzindo os erros”



Gerenciamento de Configuração de Software

As atividades são desenvolvidas para:

- Identificar a alteração
- Controlar a alteração
- Assegurar que a alteração seja implementada corretamente
- Relatar as alterações aos outros interessados



O que é uma configuração?

- Configuração de um sistema é uma coleção de versões específicas de itens de configuração (hardware, firmware ou software) que são combinados de acordo com procedimentos específicos de construção para servir a uma finalidade particular.



Conceito

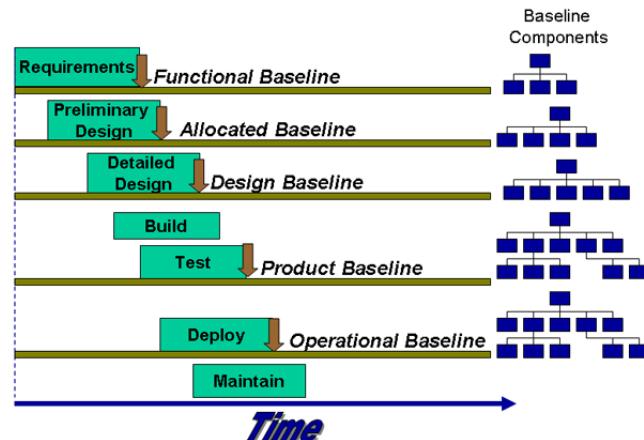
Item de configuração: Elemento unitário ou um grupo de elementos para efeito de controle de versão.

- Código
- Documentação
- Diagramas, planos, ferramentas, casos de teste e etc

Todos os documentos que podem ser úteis para a evolução futura do sistema

Conceito

- **Baseline:** configuração formalmente aprovada que servirá de referência para desenvolvimento posterior A configuração do software em um ponto discreto no tempo.
- Quando um conjunto de artefatos de software se torna um item de configuração?



Qual a importância do controle de versões?

Gerência de Configuração

Controle de Versão

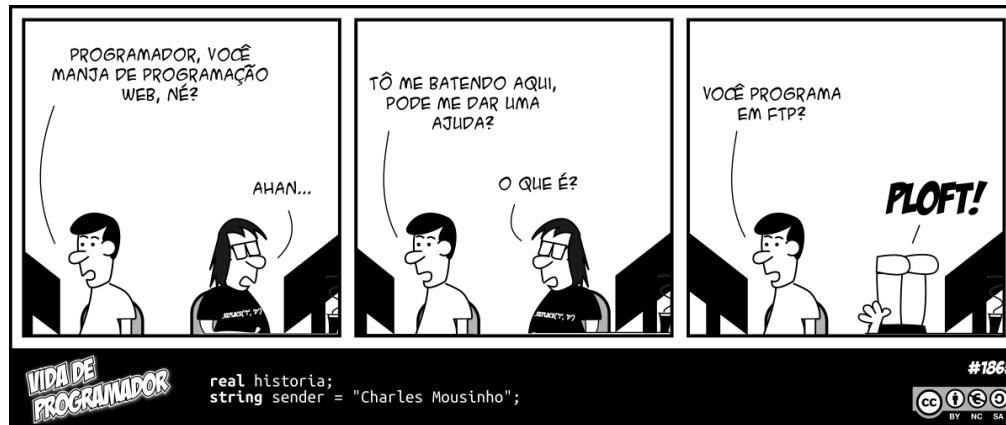
Controle de Mudanças

Integração Contínua

Uma situação problema

Você precisa editar um site hospedado em um servidor:

- Você faz o download via FTP
- Faz as alterações necessárias
- Envia os arquivos alterados para o servidor via FTP



Uma situação problema

Agora outro desenvolvedor também deve fazer alterações no site...

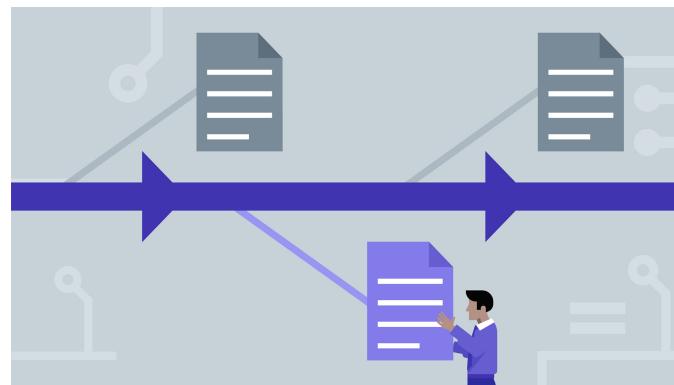
1. • Baixa o mesmo arquivo junto com você
2. • Edita e manda para o servidor depois de você
3. • Sobrescreve suas alterações!!!



Controle de versões

Principais objetivos do Controle de Versões do Sistema

- Trabalhar simultaneamente
- Evitar conflitos em alterações
- Manter versões



Controle de Mudanças

Objetivo: Garantir que todas as mudanças ocorram de maneira controlada e otimizada

- Manter histórico de mudança
- Justificar mudança

Exemplo:

Mudanças da v2.2 para v2.3:

- correção do defeito D345
- correção do defeito D346
- adicionada a funcionalidade do RF44

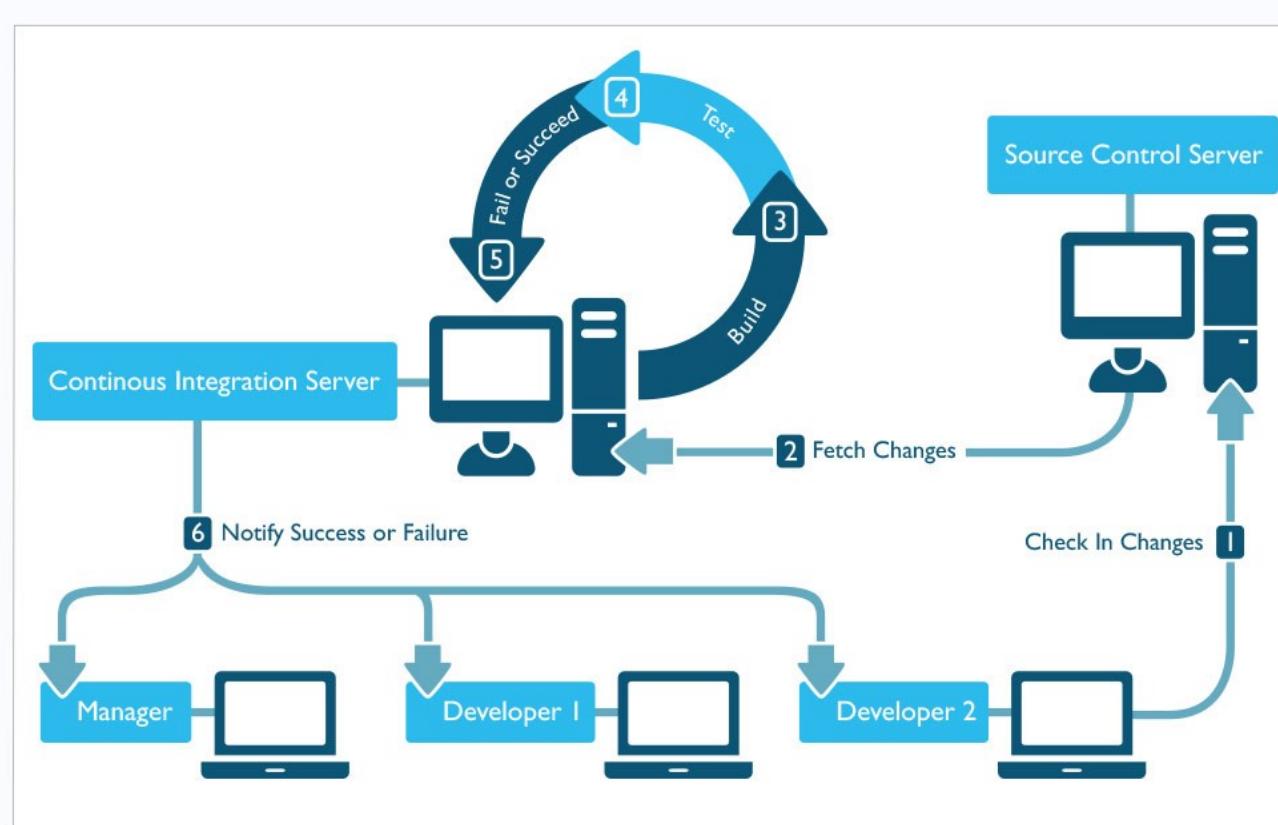
Pendências:

- melhorar usabilidade da interface I43

Integração contínua

- Uma das práticas mais importantes do desenvolvimento ágil!
- Agilizar tarefas demoradas como a compilação de um projeto e a execução dos seus testes automatizados
- Tarefas são executadas a cada mudança no repositório de código e, em caso de erros de compilação ou falhas nos testes automatizados, todos os desenvolvedores são alertados rapidamente

Integração Contínua



Ferramentas de apoio

Tipo de Ferramenta	Open Source	Comercial
Controle de versão	Mercurial, Git, Subversion, CVS	Team Foundation Server, Team Concert, ClearCase, StarTeam, Perforce, BitKeeper
Controle de Mudança	Trac, Redmine, Mantis, Bugzilla	Jira, FogBUGZ, CaliberRM, Perforce
Integração Contínua	Jenkins, Bitten, Scons, Ant, Maven, CruiseControl, Gump	AntHill Pro, FinalBuilder, BuildForge

GIT

Visão geral do Git e Github O que é Git?

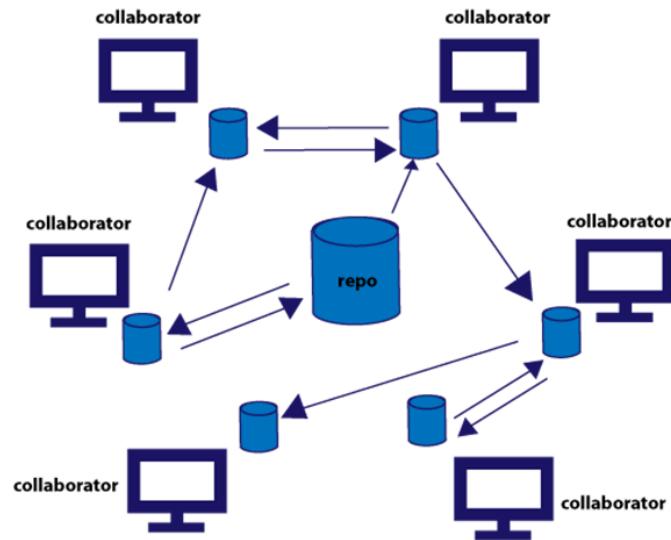
- Sistema de Controle de Versão Distribuído
- Criado por Linus Torvalds (2005)
- Auxiliar no Desenvolvimento do Linux O que é Github?
- Serviço de Web Hosting compartilhado para projetos que usam o controle de versionamento Git

Algumas empresas que utilizam

git



Controle de Versões distribuído

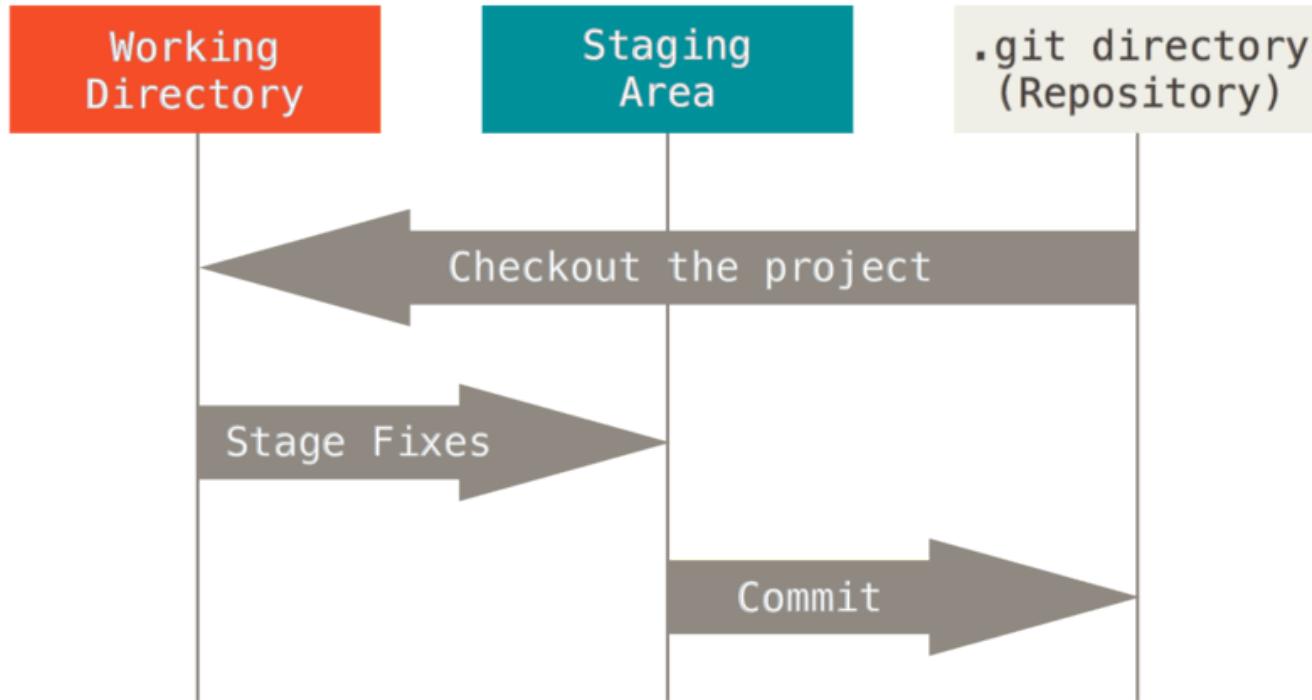


Instalando o git

- Windows
- Linux
- macOS:

<https://git-scm.com/download>

3 Estados do git



Configurações do git

- É importante se identificar no git, pois é uma das etapas do controle de versão (auditoria)

```
git config l  
git config --global <variavel> <valor>
```

Exemplo:

```
git config --global user.name 'Luiz Pedro Petroski'  
git config --global user.email 'petroskilp@gmail.com'
```

```
git help  
git help <parametro>
```

Iniciando o versionamento

Para transformar um diretório local em um repositório Git, basta abrir o diretório via terminal e executar o comando:

```
git init
```

- git init irá iniciar a estrutura de versionamento no diretório atual

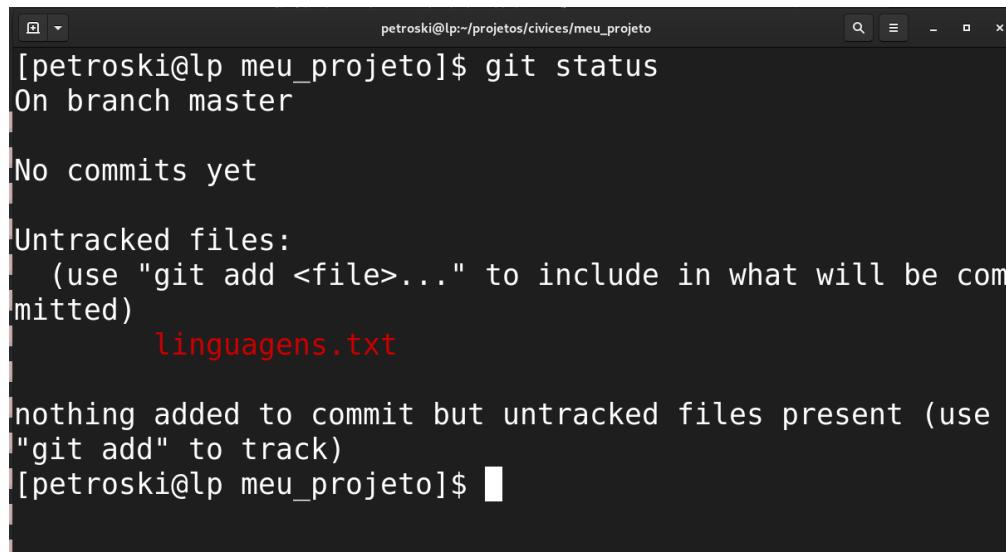
```
git init meu_projeto
```

- git init meu_projeto irá criar um diretório de nome "meu_projeto" e criar a estrutura dentro deste diretório.

Monitorando mudanças

- Supondo que adicionamos o arquivo linguagens.txt no diretório "meu_projeto"

```
git status
```



```
petroski@lp:~/projetos/civices/meu_projeto$ git status
On branch master

No commits yet

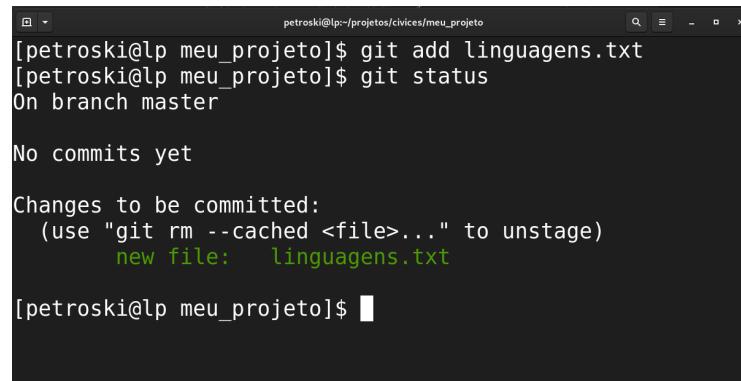
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    linguagens.txt

nothing added to commit but untracked files present (use
"git add" to track)
[petroski@lp meu_projeto]$ █
```

Enviando para a Staging Area

- Para que um arquivo seja rastreado, devemos executar o seguinte comando:

```
git add carros.txt  
#ou  
git add .  
# o ponto (.) inclui todos os arquivos não rastreados ou modificados
```



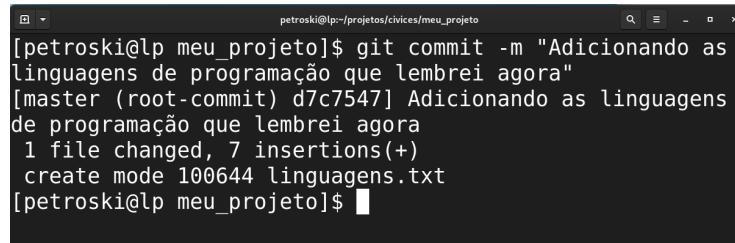
A screenshot of a terminal window titled "petroski@lp:~/projetos/civices/meu_projeto". The window shows the following command-line session:

```
[petroski@lp meu_projeto]$ git add linguagens.txt  
[petroski@lp meu_projeto]$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
(use "git rm --cached <file>..." to unstage)  
  new file:   linguagens.txt  
  
[petroski@lp meu_projeto]$ █
```

Gravando um arquivo no repositório (commit)

- Para gravar as mudanças no repositório, execute o comando:

```
git commit -m "Adicionando as linguagens de programação que lembrei agora"
```



A screenshot of a terminal window titled 'petroski@lp:~/projetos/civices/meu_projeto'. The window displays the command 'git commit -m "Adicionando as linguagens de programação que lembrei agora"' followed by its output: '[master (root-commit) d7c7547] Adicionando as linguagens de programação que lembrei agora' and '1 file changed, 7 insertions(+)'.

```
petroski@lp:~/projetos/civices/meu_projeto
[petroski@lp meu_projeto]$ git commit -m "Adicionando as linguagens de programação que lembrei agora"
[master (root-commit) d7c7547] Adicionando as linguagens de programação que lembrei agora
 1 file changed, 7 insertions(+)
  create mode 100644 linguagens.txt
[petroski@lp meu_projeto]$
```

Histórico de commits

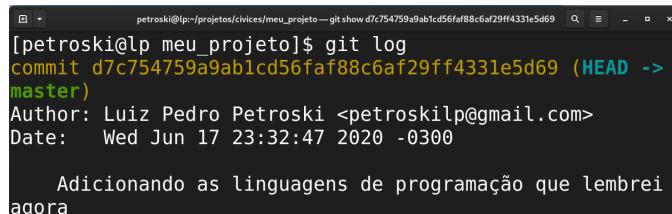
- Para visualizar o histórico de commits

```
git log
```

- Para mostrar as alterações de um commit

```
git show <commit>
```

- Commit deve ser especificado pela chave



```
petroski@lp:~/projetos/civices/meu_projeto$ git log  
[petroski@lp meu_projeto]$ commit d7c754759a9ab1cd56faf88c6af29ff4331e5d69 (HEAD ->  
master)  
Author: Luiz Pedro Petroski <petroskilp@gmail.com>  
Date:   Wed Jun 17 23:32:47 2020 -0300  
  
        Adicionando as linguagens de programação que lembrei  
        agora
```

Ramificações (branching)

- Branch é uma ramificação do repositório
- Alterações (commits) ocorrem na branch
- Muito útil para trabalhos colaborativos
- Branchs de desenvolvimento facilitam o controle

Para listar as branches do nosso repositório, devemos executar o comando:

```
git branch
```



A screenshot of a terminal window titled 'petroski@lp:~/projetos/civices/meu_projeto'. The window shows the command 'git branch' being run, with the output displaying a single branch named 'master' preceded by an asterisk (*). The terminal has a dark background and light-colored text.

```
petroski@lp:~/projetos/civices/meu_projeto]$ git branch
* master
[petroski@lp meu_projeto]$
```

Listar e criar branch

- Como listar todas as branches?

```
git branch -v
```

- Como criar uma branch?

```
git branch nome_da_branch
```

- Modo mais rápido de criar e deixá-la como corrente:

```
git checkout -b nome_da_branch
```

Alterar branch corrente e apagar

- Como mudar de branch?

```
git checkout nome_da_branch
```

- Apagar uma branch em seu repositório local

```
git branch -d nome_da_branch
```

- Apagar uma branch remota

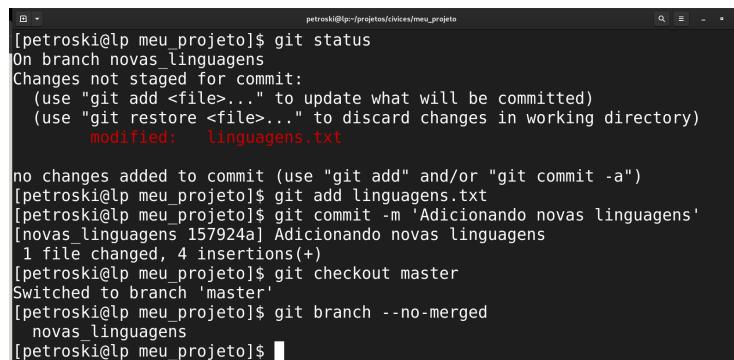
```
git push origin --delete nome_da_branch
```

Mesclar branch

- Supondo que estamos na branch master, podemos verificar as branches ainda não mescladas da seguinte forma:

```
git branch --no-merged
```

- Temos uma branch com modificações em relação a branch master!

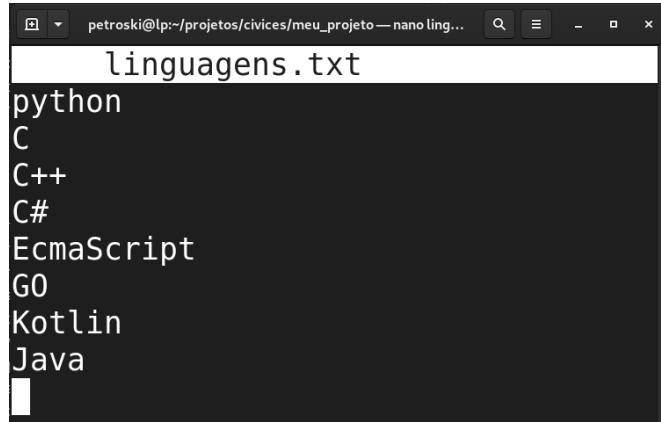


```
petroski@lp meu_projeto]$ git status
On branch novas_linguagens
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   linguagens.txt

no changes added to commit (use "git add" and/or "git commit -a")
[petroski@lp meu_projeto]$ git add linguagens.txt
[petroski@lp meu_projeto]$ git commit -m 'Adicionando novas linguagens'
[novas_linguagens 157924a] Adicionando novas linguagens
 1 file changed, 4 insertions(+)
[petroski@lp meu_projeto]$ git checkout master
Switched to branch 'master'
[petroski@lp meu_projeto]$ git branch --no-merged
  novas_linguagens
[petroski@lp meu_projeto]$
```

Mesclar branch

branch master



```
linguagens.txt
python
C
C++
C#
EcmaScript
GO
Kotlin
Java
```

branch branch_ordenado



```
linguagens.txt
C
C++
C#
EcmaScript
GO
Kotlin
Python
```

- Queremos mesclar a branch master com a branch_ordenado
- O que vai acontecer?

Mesclar branch

- Para mesclar podemos usar o comando:

```
git merge branch_ordenado -m "Mesclando com a branch_ordenado"
```

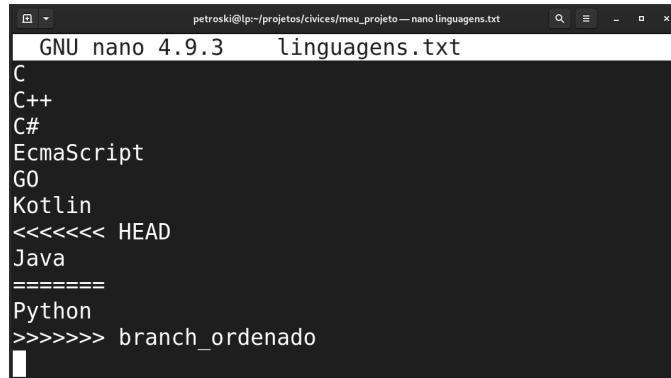


A screenshot of a terminal window titled 'petroski@lp:~/projetos/civices/meu_projeto'. The window shows the command 'git merge branch_ordenado' being run. The output indicates an automatic merge failed due to a conflict in the file 'linguagens.txt', specifically in the content of the file. It prompts the user to fix the conflicts and then commit the result.

```
[petroski@lp meu_projeto]$ git merge branch_ordenado
Auto-merging linguagens.txt
CONFLICT (content): Merge conflict in linguagens.txt
Automatic merge failed; fix conflicts and then commit the result.
[petroski@lp meu_projeto]$ █
```

Resultado do conflito

- As partes conflitantes são marcadas
- E você pode escolher se mantém a versão do branch master, ou da branch_ordenada
- Ou até mesmo as duas ao mesmo tempo
- Ou uma terceira versão que você rescrever

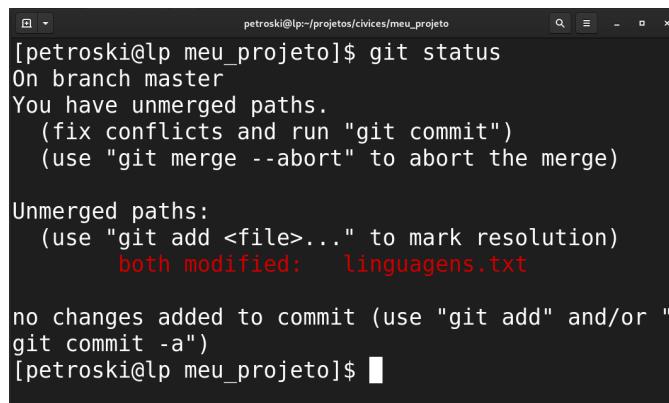


A screenshot of a terminal window titled "GNU nano 4.9.3 linguagens.txt". The window shows a list of programming languages: C, C++, C#, EcmaScript, GO, Kotlin, Java, Python, and branch_ordenado. Between "Java" and "Python", there are two sets of conflict markers: "<<<<< HEAD" and ">>>>> branch_ordenado". The "Python" line is partially visible below the markers.

Comitando a resolução dos conflitos

- Faça as alterações no arquivo mesclado e depois:

```
git add .
git commit -m "mensagem"
```



A screenshot of a terminal window titled 'petroski@lp:~/projetos/civices/meu_projeto'. The window displays the following command and its output:

```
[petroski@lp meu_projeto]$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified: linguagens.txt

no changes added to commit (use "git add" and/or "git commit -a")
[petroski@lp meu_projeto]$ █
```

Ignorando arquivos (.gitignore)

- Basta criar um arquivo chamado .gitignore no diretório principal do nosso projeto com os nomes dos arquivos ou extensões que queremos ignorar.
- Ex.:
.log
imagens/*.bmp
- Em projetos Java, arquivos .class, .jar e .war são exemplos de arquivos que devem ser ignorados.
- Detalhe importante: o .gitignore também envoluirá com o projeto.
- Assim, o que precisamos fazer?

Visualizando Logs

git log

```
[petroski@lp meu_projeto]$ git log
commit 26ed1aa4e8ae6aab84b0422c5c531c7c7b3d066 (HEAD -> master)
Merge: 4359c60 7fd9ccb
Author: Luiz Pedro Petroski <petroskilp@gmail.com>
Date:   Thu Jun 18 00:49:57 2020 -0300

    Resolvendo conflitos

commit 4359c60630b0a895319dd9e4570be8cce17d6c5b4
Author: Luiz Pedro Petroski <petroskilp@gmail.com>
Date:   Thu Jun 18 00:44:06 2020 -0300

    Adicionando java

commit 7fd9cbbb77895bfc5c0d69a836965638b1e12516 (branch_ordenado)
Author: Luiz Pedro Petroski <petroskilp@gmail.com>
Date:   Thu Jun 18 00:27:46 2020 -0300

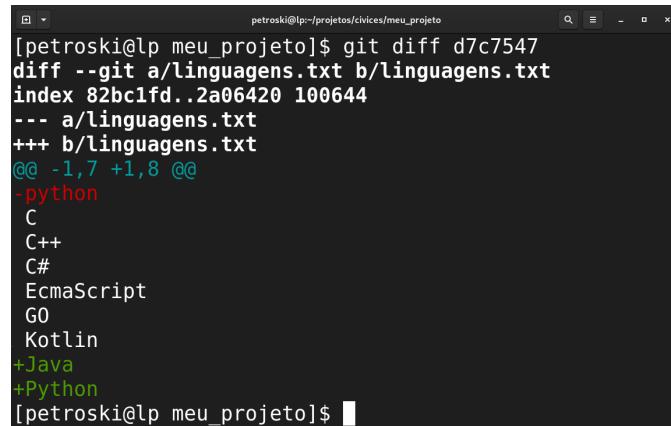
    Odenando em ordem alfabetica
```

git log - --graph - -oneline

```
[petroski@lp meu_projeto]$ git log --graph --oneline
* 26ed1aa (HEAD -> master) Resolvendo conflitos
|\ \
| * 7fd9cbb (branch_ordenado) Odenando em ordem alfabetica
| * 4359c60 Adicionando java
|/
* d7c7547 (outro_ramo, novo_ramo) Adicionando as linguagens de programação
que lembrei agora
[petroski@lp meu_projeto]$
```

Verificar diferenças entre commits

```
git diff d7c7547
```



```
[petroski@lp meu_projeto]$ git diff d7c7547
diff --git a/linguagens.txt b/linguagens.txt
index 82bc1fd..2a06420 100644
--- a/linguagens.txt
+++ b/linguagens.txt
@@ -1,7 +1,8 @@
-python
 C
 C++
 C#
 EcmaScript
 GO
 Kotlin
+Java
+Python
[petroski@lp meu_projeto]$
```

Trabalho em equipe



fonte: <https://vidadeprogramador.com.br/2017/07/19/mensagem-do-commit/>

Revisão de conceitos

- Merge?
- Commit?
- Branch?
- Config?
- Log?
- Checkout?

GitHub

Remote

- Repositório remoto, hospedado em um servidor
- São referenciados por uma URL
- Podem receber vários commits
- Sincronizar o trabalho colaborativo
- Autenticação com usuário e senha (https) ou par de chaves (SSH)

```
git remote add origin <$URL>
```



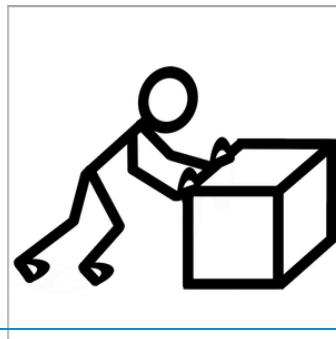
Push

- Enviar alterações (commits) da branch para o repositório remoto

```
git push <$remote> <$branch>
```

- O envio é rejeitado se o repositório local não estiver sincronizado

```
git push origin master
```



Workshop GIT

Pull

- Baixa as alterações do repositório remoto e realiza o Merge automático com o repositório local
- Mantém o repositório sincronizado com os últimos commits de uma branch
- Permite baixar novas branches que não foram criadas localmente

```
git pull  
git pull <$remote> <$branch>
```



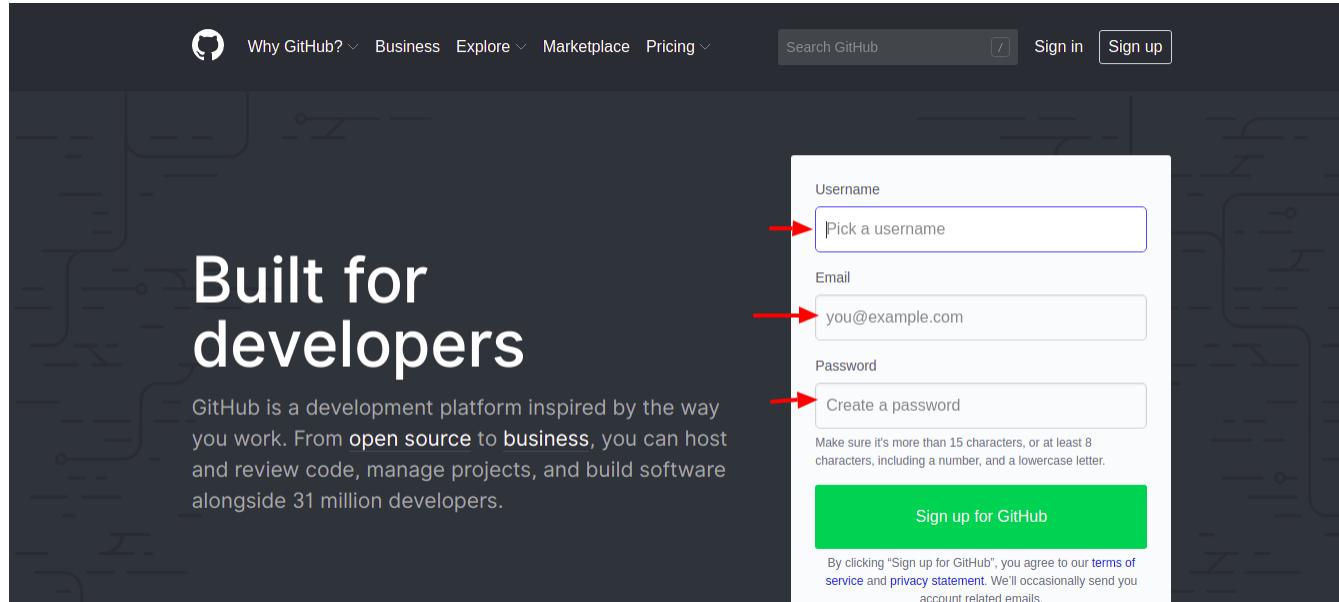
Clone

- Baixa o repositório remoto
- Outra forma de criar um repositório local
- Já vem com o remote configurado

```
git clone <$URL>
```



Criar uma conta no github



Criar um repositório

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



petroskilp / ✓

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

Description (optional)

Testando criar um repositório

Public

Anyone can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: C ▾

Add a license: Apache License 2.0 ▾



Create repository

Utilizando o repositório

The screenshot shows a GitHub repository page for 'petroskilp / meu_projeto'. The page includes a navigation bar with links for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security (0), Insights, and Settings. A prominent section titled 'Quick setup — if you've done this kind of thing before' provides instructions for setting up the repository. It includes a code editor with examples for creating a new repository on the command line and pushing an existing one. There is also a 'Import code' button.

Quick setup — if you've done this kind of thing before

or [HTTPS](https://github.com/petroskilp/meu_projeto.git) [SSH](ssh://github.com/petroskilp/meu_projeto.git) https://github.com/petroskilp/meu_projeto.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# meu_projeto" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin https://github.com/petroskilp/meu_projeto.git  
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/petroskilp/meu_projeto.git  
git push -u origin master
```

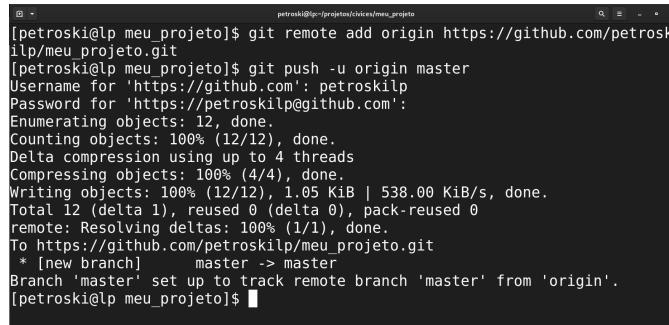
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

Usando seu repositório do GitHub

```
- git remote add $<$repository name> <$url>
- git push $<$repository name$>$ $<$branch name$>$
```



```
[petroski@lp meu_projeto]$ git remote add origin https://github.com/petrosk  
ilp/meu_projeto.git
[petroski@lp meu_projeto]$ git push -u origin master
Username for 'https://github.com': petroskilp
Password for 'https://petroskilp@github.com':
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (12/12), 1.05 KiB | 538.00 KiB/s, done.
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/petroskilp/meu_projeto.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
[petroski@lp meu_projeto]$
```

Clonando seu repositório do GitHub

- Faz o download do projeto em um novo diretório
- Clonar o repositório:

```
git clone <$url>
```

- Exibir o remote configurado:

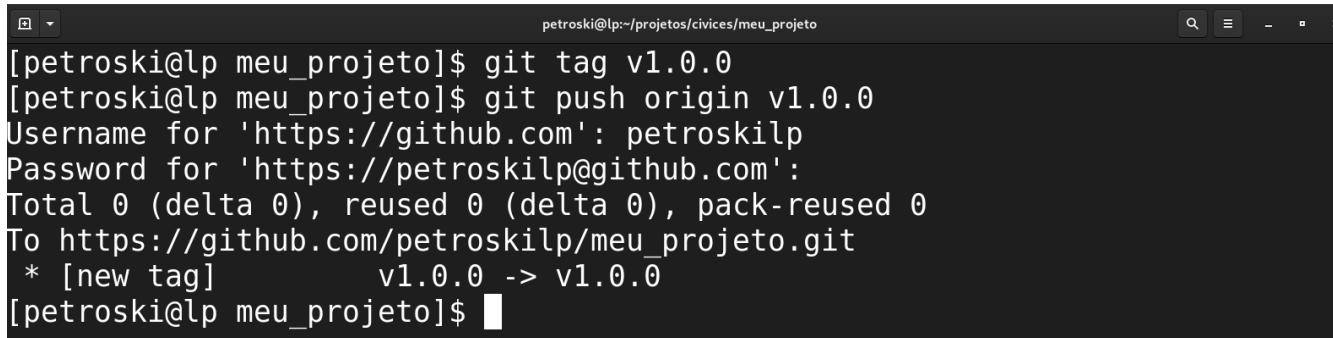
```
git remote -v
```

```
[petroski@lp teste_clonar]$ git clone https://github.com/petroskilp/meu_projeto.git
Cloning into 'meu_projeto'...
Username for 'https://github.com': petroskilp
Password for 'https://petroskilp@github.com':
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 12 (delta 1), reused 12 (delta 1), pack-reused 0
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (1/1), done.
[petroski@lp teste clonar]$
```

tagging

- Marcar um commit com uma tag

```
git tag <$tagname>
```



A screenshot of a terminal window titled "petroski@lp:~/projetos/civices/meu_projeto". The window shows the command "git tag v1.0.0" being run, followed by "git push origin v1.0.0". It prompts for GitHub credentials ("Username for 'https://github.com': petroskilp" and "Password for 'https://petroskilp@github.com':"). It then displays the push results: "Total 0 (delta 0), reused 0 (delta 0), pack-reused 0", "To https://github.com/petroskilp/meu_projeto.git", and "* [new tag] v1.0.0 -> v1.0.0". The terminal ends with "[petroski@lp meu_projeto]\$ █".

```
[petroski@lp meu_projeto]$ git tag v1.0.0
[petroski@lp meu_projeto]$ git push origin v1.0.0
Username for 'https://github.com': petroskilp
Password for 'https://petroskilp@github.com':
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/petroskilp/meu_projeto.git
 * [new tag] v1.0.0 -> v1.0.0
[petroski@lp meu_projeto]$ █
```

tagging

petroskilp / meu_projeto Private

Unwatch 1 Star 0 Fork 0

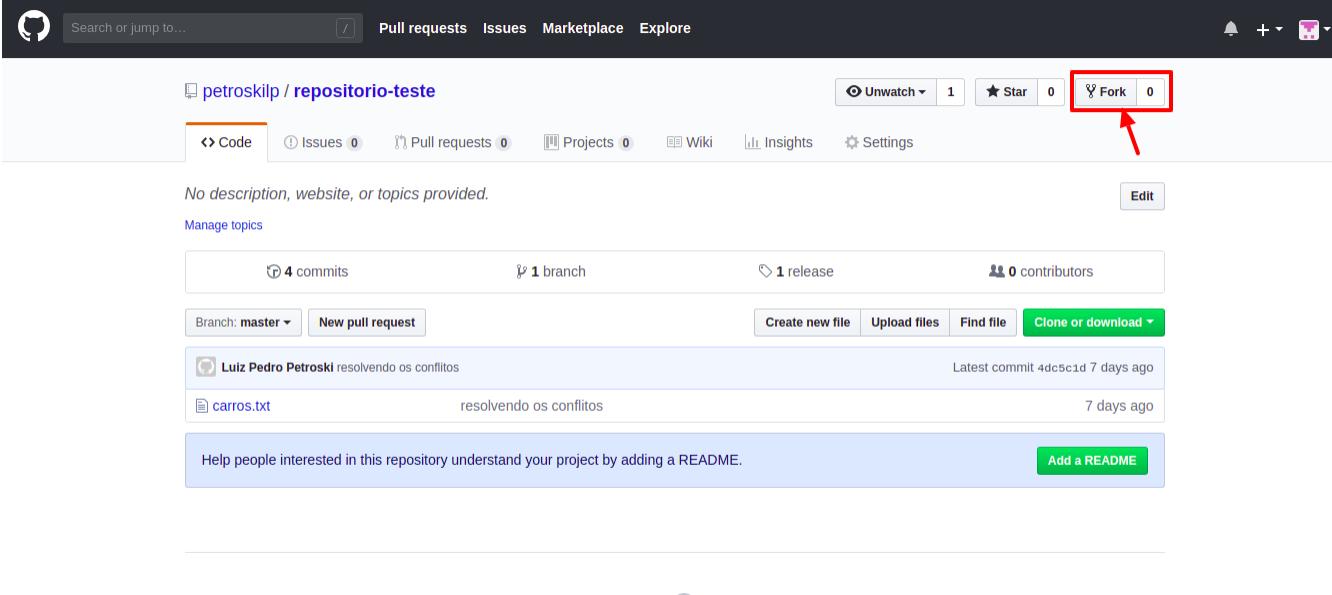
Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

Releases Tags

Tags

v1.0.0 ...
 1 hour ago 26ed1aa zip tar.gz ...

forking



A screenshot of a GitHub repository page for 'petroskilp / repositorio-teste'. The page shows basic repository statistics: 4 commits, 1 branch, 1 release, and 0 contributors. A recent commit by 'Luliz Pedro Petroski' is listed, along with a file named 'carros.txt'. A prominent red box and arrow highlight the 'Fork' button in the top right corner of the header bar.

No description, website, or topics provided.

Manage topics

4 commits 1 branch 1 release 0 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

Luliz Pedro Petroski resolvendo os conflitos Latest commit 4dc5c1d 7 days ago

carros.txt resolvendo os conflitos 7 days ago

Add a README

© 2010 GitHub Inc. Terms Privacy Security Status Help

Contact GitHub Pricing API Training Blog About

forking

A screenshot of a GitHub repository page for 'petroskilp / repositorio-teste'. The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below the navigation bar, there's a search bar and a header with the repository name 'petroskilp / repositorio-teste'. The main content area shows various metrics like Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights (selected), and Settings. A red arrow points to the 'Forks' section in the Insights tab, which shows one fork. Another red arrow points to the 'Forks' link in the sidebar menu, which also lists 'codelp / repositorio-teste'.

Outras funcionalidades do github

- Readme.md
- Wiki
- Web diff (código/imagens)
- Pull Request
- Comentários e Discussões
- Issue Tracking

readme.md

README.md

apache2 Cookbook

cookbook v5.2.1 build passing license apache2.2

This cookbook provides a complete Debian/Ubuntu style Apache HTTPD configuration. Non-Debian based distributions such as Red Hat/CentOS, ArchLinux and others supported by this cookbook will have a configuration that mimics Debian/Ubuntu style as it is easier to manage with Chef.

Debian-style Apache configuration uses scripts to manage modules and sites (vhosts). The scripts are:

- a2ensite
- a2dissite
- a2enmod
- a2dismod
- a2enconf
- a2disconf

This cookbook ships with templates of these scripts for non-Debian/Ubuntu platforms.

Cookbooks:

Wiki

 petroskilp / **repositorio-teste**

 Unwatch 1  Star 0  Fork 1

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)



Welcome to the **repositorio-teste** wiki!

Wikis provide a place in your repository to lay out the roadmap of your project, show the current status, and document software better, together.

[Create the first page](#)

© 2018 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)



[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

WebDiff

Resolvendo conflitos

master v1.0.0 [Browse files](#)

 petroskilp committed 1 hour ago
2 parents 4359c60 + 7fd9cbb commit 26ed1aa4e8ae6aab84b0422c5c531c7c7b3d066

Showing 1 changed file with 1 addition and 1 deletion.

Unified Split

2 linguagens.txt

...	00 -1,8 +1,8 00	...
1	- python	1
2	C	2
3	C++	3
4	C#	4
5	EcmaScript	5
6	GO	6
7	Kotlin	7
8	Java	8 + Python

Pull Request

nodejs / node

Watch 2,984 Star 55,867 Fork 12,299

Code Issues 564 Pull requests 224 Projects 9 Insights

First time contributing to nodejs/node? Dismiss

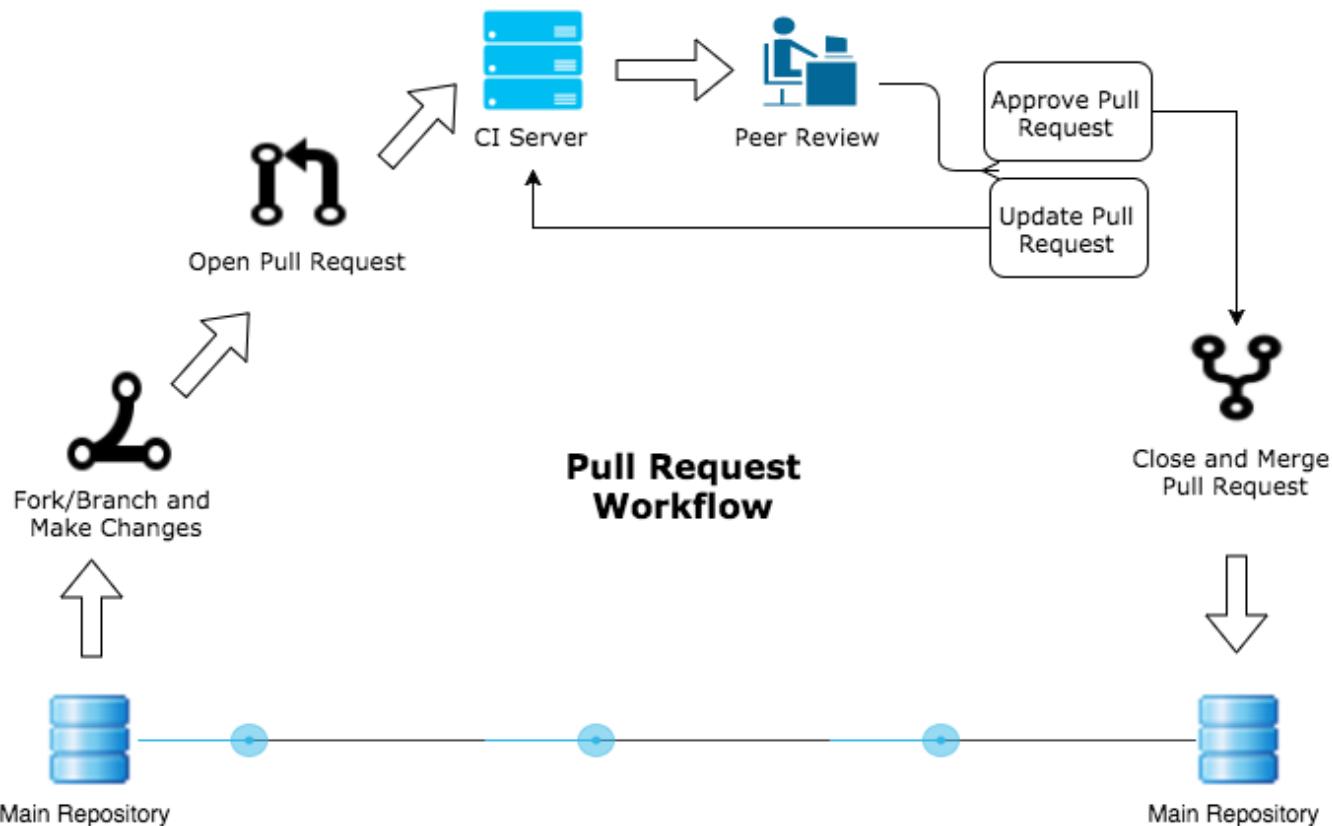
If you know how to fix an issue, consider opening a pull request for it. You can read this repository's contributing guidelines to learn how to open a good pull request.

Filters ▾ is:pr is:open Labels Milestones New pull request

224 Open ✓ 15,232 Closed Author ▾ Labels ▾ Projects ▾ Milestones ▾ Reviews ▾ Assignee ▾ Sort ▾

PR	Title	Author	Labels	Comments
#24701	src: move C++ binding/addon related code into node_binding{.h, .cc} ✘	joyecheung	2 of 2	3
#24700	doc: describe current HTTP header size limit ● author ready doc errors http http_parser	sam-github	Approved 0 of 4	7
	[v11.x] backport "tools: update remark-preset-lint-node to v1.3.1" ✘ doc tools v11.x			1

Pull Request



Issue Tracking

nodejs / node

Watch 2,984 Star 55,867 Fork 12,299

Code Issues 564 Pull requests 225 Projects 9 Insights

Want to submit an issue to nodejs/node? Dismiss

If you have a bug or an idea, read the [contributing guidelines](#) before opening an issue.
Issues labeled [help wanted](#) or [good first issue](#) can be good first contributions.

Filters ▾ is:issue is:open Labels Milestones New issue

Author	Labels	Projects	Milestones	Assignee	Sort
564 Open ✓ 8,474 Closed					
Missing API docs for recent changes in HTTP module http http_parser					
#24693 opened 6 hours ago by lirantal					
Make HTTP_MAX_HEADER_SIZE configurable http http_parser					
#24692 opened 8 hours ago by zauberpony					
npm i -g @ionic/cli npm ERR! code E401 npm ERR! 401 Unauthorized: @ionic/cli@latest npm					
#24690 opened 12 hours ago by rajeshwarpatlolla					

Prática 01

1. • Instale o git na sua máquina
2. • Crie um diretório chamado aprendendo_git e entre nele via terminal
3. • Use git config e informe seu nome e email
4. • git init
5. • git status
6. • Crie um arquivo anotacoes.txt nesse diretório, inclua seu nome nesse arquivo
7. • git add anotacoes.txt
8. • git commit -m "meu primeiro commit"
9. • git status

Prática 02

1. • Crie uma branch feedback (git checkout -b feedback)
2. • Abra o arquivo anotacoes.txt e veja se contém seu nome
3. • Altere o anotacoes.txt (nessa branch), apague seu nome e faça um texto explicando o que é o git
4. • git status
5. • git add anotacoes.txt
6. • git commit -m "meu commit na branch feedback"
7. • Mude para a branch master e adicione a sua idade no arquivo anotacoes.txt
8. • Faça o merge com a branch feedback

Prática 02

1. • Resolva o conflito (edite o arquivo anotacoes.txt)
2. • git add anotacoes.txt
3. • git commit -m "resolvendo conflitos"
4. • delete a branch feedback

Tutoriais e Guias

Tutorial auto-instrucional:

- <https://learngitbranching.js.org/>
- <https://gitexercises.fracz.com/>
- <http://git-school.github.io/visualizing-git/>
- <https://github.com/jlord/git-it-electron>

Guias e livros

- https://rogerdudler.github.io/git-guide/index.pt_BR.html
- <https://guides.github.com/introduction/git-handbook/>
- <https://git-scm.com/book/pt-br/v2>
- <http://marklodato.github.io/visual-git-guide/index-pt.html>

Referências

Livros

- Eric Sink. 2011. Version Control by Example (1st ed.). PYOW Sports Marketing. (FREE EBOOK)
<https://ericsink.com/vcbe/index.html>
- Scott Chacon e Ben Straub. 2014. Pro Git (2nd ed.). Apress, Berkely, CA, USA. (FREE EBOOK) <https://git-scm.com/book/pt-br/v2>
- Pressman, R. S. Software engineering: A practitioner's approach. 5th ed. McGraw-Hill Higher Education, 2001.
- Kemper C., Oxley, I. 2012. Foundation Version Control for Web Developers. Appress.

Perguntas

