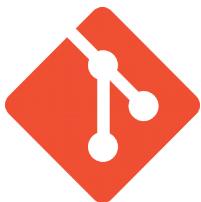


Git + GitHub

**Prof. David Fernandes de Oliveira
Instituto de Computação
UFAM**

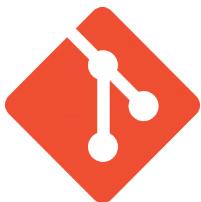
Controle de Versão

- **Sistemas de Controle de Versão** registram as mudanças feitas em um conjunto de arquivos ao longo do tempo
- Usar um sistema de controle de versão é uma decisão sábia, já que ele permite:
 - Reverter arquivos ou um projeto inteiro para um estado anterior
 - Comparar mudanças feitas nos arquivos no decorrer do tempo
 - Dividir o trabalho entre outros membros de uma equipe
 - Ver quem foi o último a modificar algo que pode estar causando problemas, quem introduziu um bug e quando



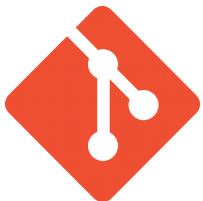
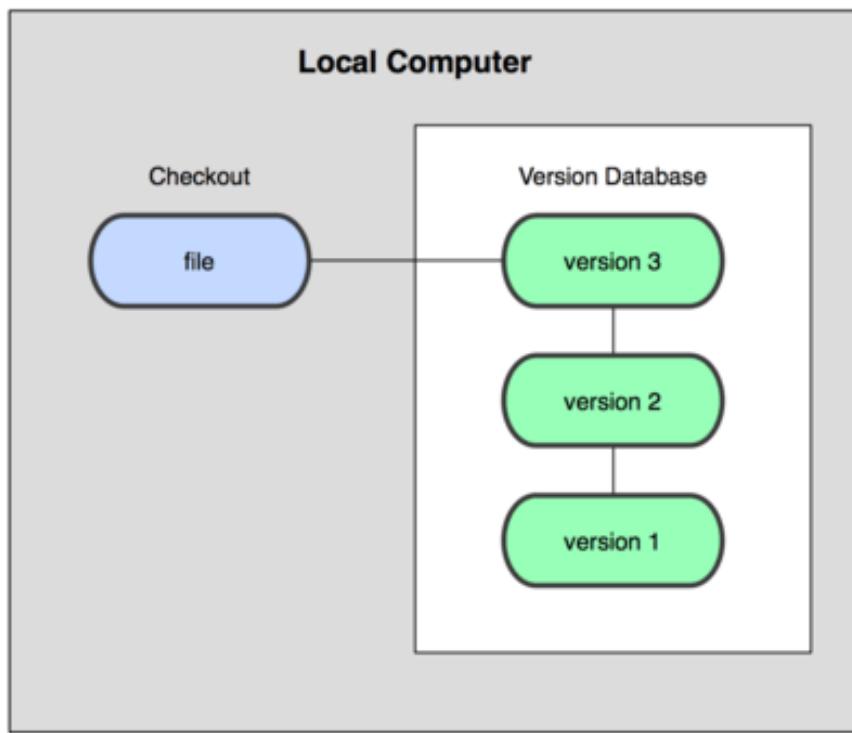
Controle de Versão Local

- O método de controle de versão mais usado é a simples **cópia dos arquivos** em outro diretório
- Esta abordagem é muito comum, mas é também a mais suscetível a erros
 - É fácil sobrescrever ou apagar arquivos sem querer



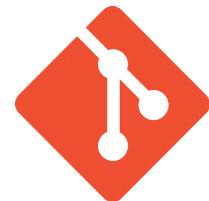
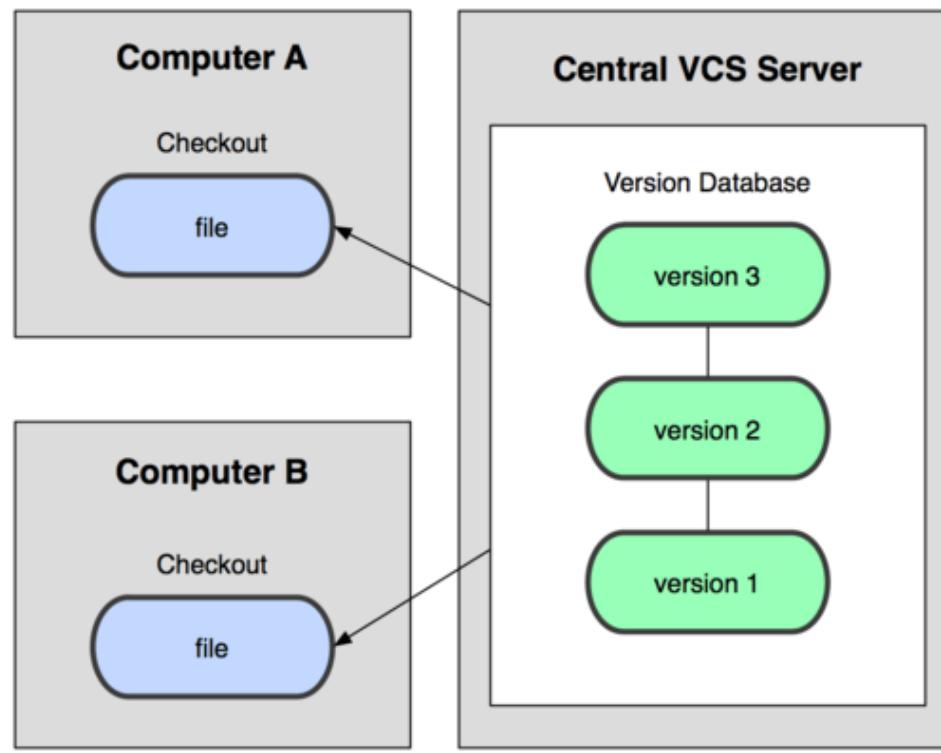
Controle de Versão Local

- Para lidar com esse problema, alguns programadores desenvolveram **sistemas de controle de versão locais**
 - Tais sistemas armazenavam todas as alterações dos arquivos em várias versões dos mesmos



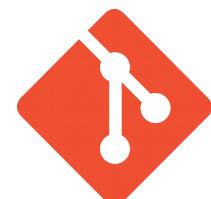
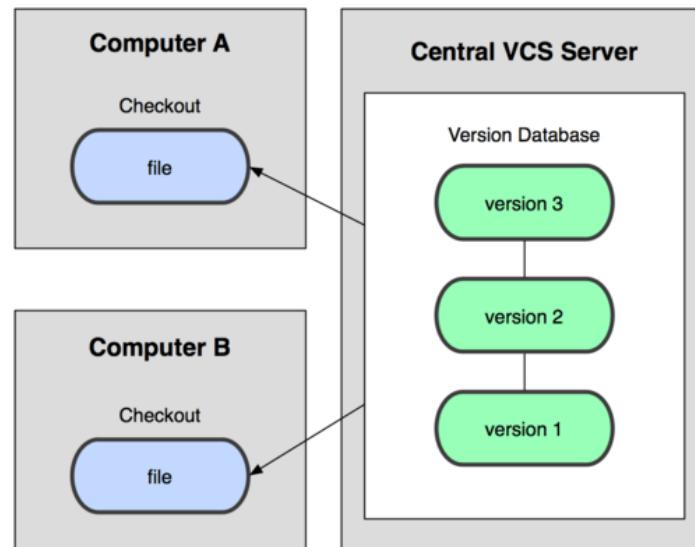
Controle de Versão Centralizado

- Em muitas ocasiões, é necessário trabalhar em conjunto com outros desenvolvedores
- Para lidar com isso, foram desenvolvidos os **sistemas de controle de versão centralizados**



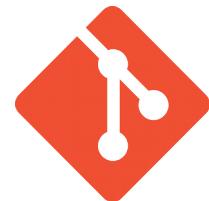
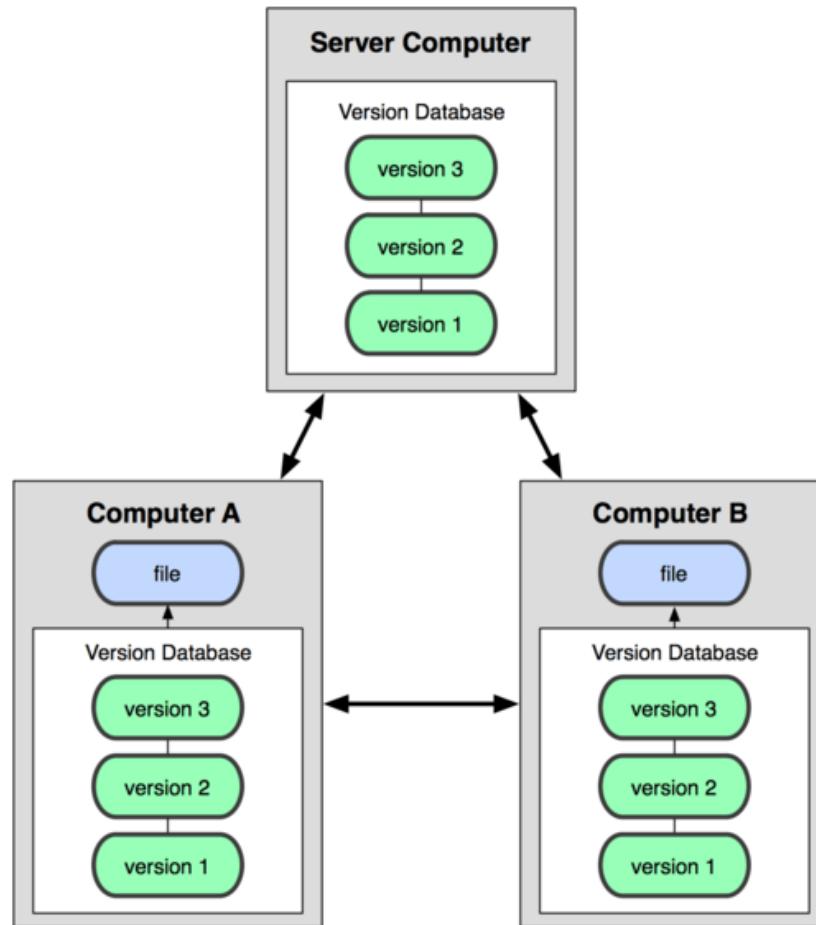
Controle de Versão Centralizado

- Problemas com o controle de versão centralizado:
 - Se o servidor ficar fora do ar, ninguém poderá salvar seus arquivos
 - Se o servidor for corrompido e não houver backup, perde-se todo o histórico de mudanças do projeto
 - Exceto pelas cópias que os desenvolvedores possuem em suas máquinas locais



Controle de Versão Distribuídos

- Na abordagem distribuída, as estações de trabalho (clientes) também possuem cópias completas do repositório



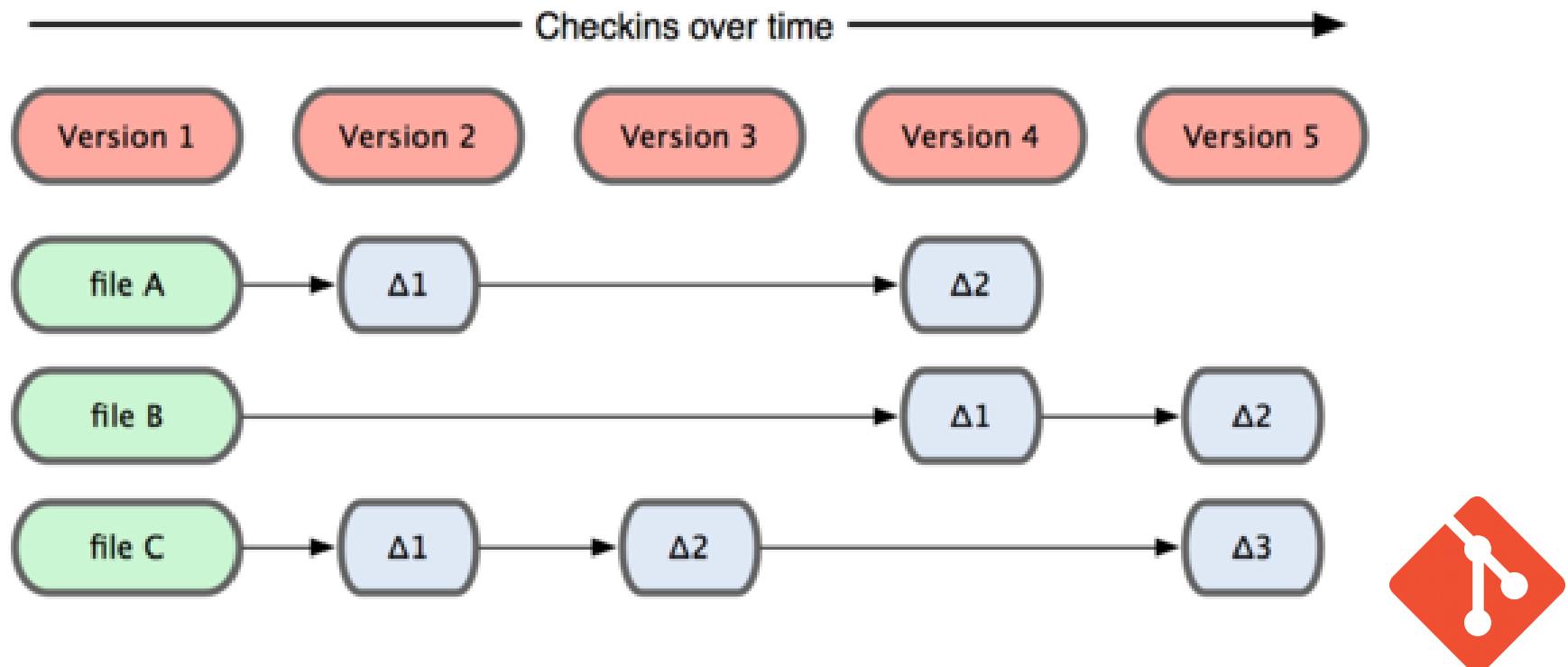
O que é o Git?

- É um **sistema de controle de versão distribuído**, desenvolvido a partir de 2005
- Foi desenvolvido pela comunidade que mantém o kernel do Linux, tendo **Linus Torvalds** como principal desenvolvedor



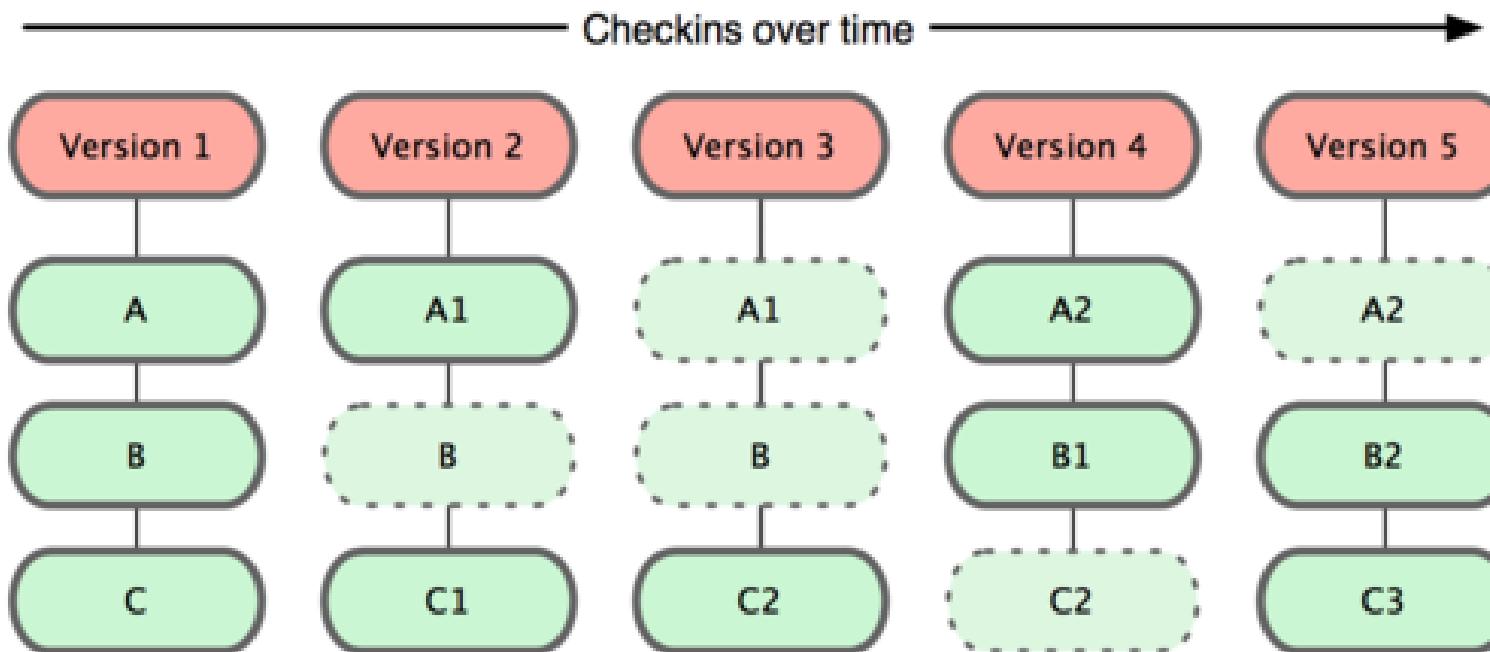
Noções Básicas de Git

- Outra diferença entre Git e outros SCV (Subversion, CVS, etc) está na forma que o Git armazena os dados
- A maior parte dos sistemas armazena informação como uma lista de mudanças por arquivo



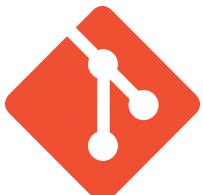
Noções Básicas de Git

- O Git, por outro lado, usa links para arquivos não foram modificados, e uma cópia completa de arquivos que foram modificados



A Maioria das Operações são Locais

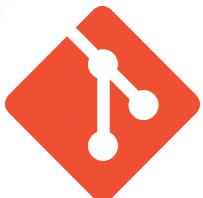
- A maior parte das operações do Git são executadas na máquina local do usuário
- Por exemplo, para navegar no histórico de versões do projeto, o Git não precisa acessar o servidor
 - Isso significa que há poucas coisas que você não possa fazer caso esteja offline



Instalando o Git no Linux

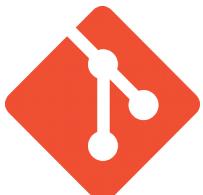
- Para instalar o Git no Linux, basta usar a ferramenta de gerenciamento de pacotes de sua distribuição
- No caso do Ubuntu, você pode usar o **apt-get**

```
david@coyote: ~
david@coyote:~$ sudo apt-get install git
```



Configuração Inicial do Git

- O Git vem com uma ferramenta chamada `git config` que permite a definição de variáveis de configuração
- Essas variáveis podem ser armazenadas em três lugares diferentes:
 - `/etc/gitconfig`: contém valores para todos usuários do sistema e todos os seus repositórios (opção `--system`)
 - `~/.gitconfig`: contém valores para todos os repositórios de seu usuário (opção `--global`)
 - `.git/config`: contém valores de um dado repositório específico (opção `--local`)
- Cada nível sobreponem o valor do nível anterior



Sua Identidade

- A primeira coisa que você deve fazer quando instalar o Git é definir o seu nome de usuário e endereço de e-mail
 - Todos os commits no Git utilizam essas informações, para informar ao repositório quem fez as mudanças nos arquivos

```
david@coyote:~$ git config --global user.name "David Fernandes"
david@coyote:~$ git config --global user.email david@icomp.ufam.edu.br
david@coyote:~$ cat .gitconfig
[user]
    email = david@icomp.ufam.edu.br
    name = David Fernandes
david@coyote:~$ █
```



Inicializando um Repositório

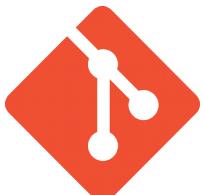
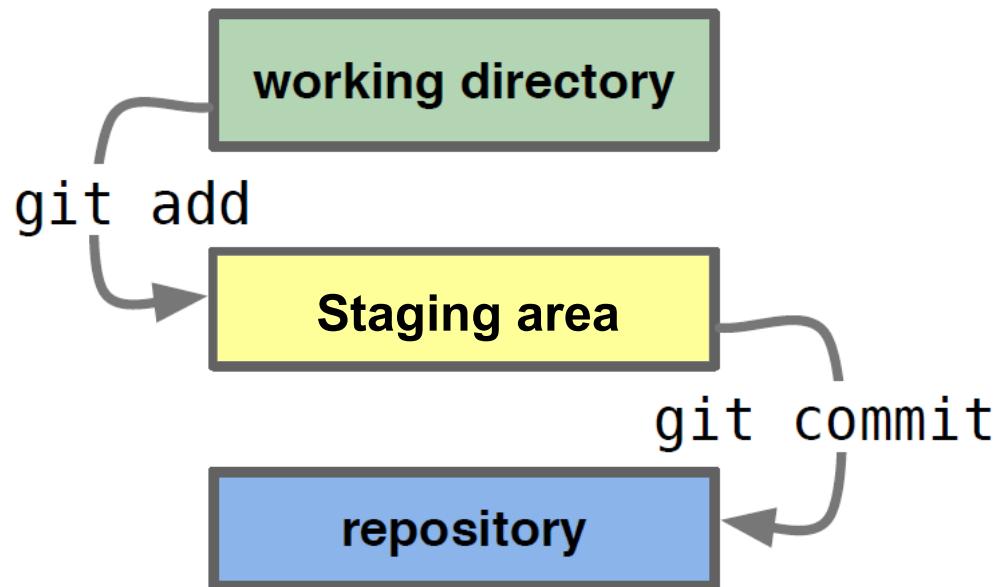
- Caso você esteja iniciando um projeto existente com Git, você precisa ir para o diretório do projeto e digitar:
 - `git init`
 - Isso cria um novo subdiretório chamado `.git` que contém todos os arquivos necessários para o controle de versão do Git

```
david@coyote:~/pilha
david@coyote:~/pilha$ ls -a
. .. main.c Makefile pilha_lib.c pilha_lib.h
david@coyote:~/pilha$ git init
Initialized empty Git repository in /home/david/pilha/.git/
david@coyote:~/pilha$ ls -a
. .. .git main.c Makefile pilha_lib.c pilha_lib.h
david@coyote:~/pilha$ █
```



Inicializando um Repositório

- Uma vez que o repositório foi criado, você pode seguir o seguinte fluxo para edição de arquivos:
 - Selecionar os arquivos que deseja versionar (com `git add`)
 - Consolidar tais arquivos no repositório (com `git commit`)



O Primeiro Commit

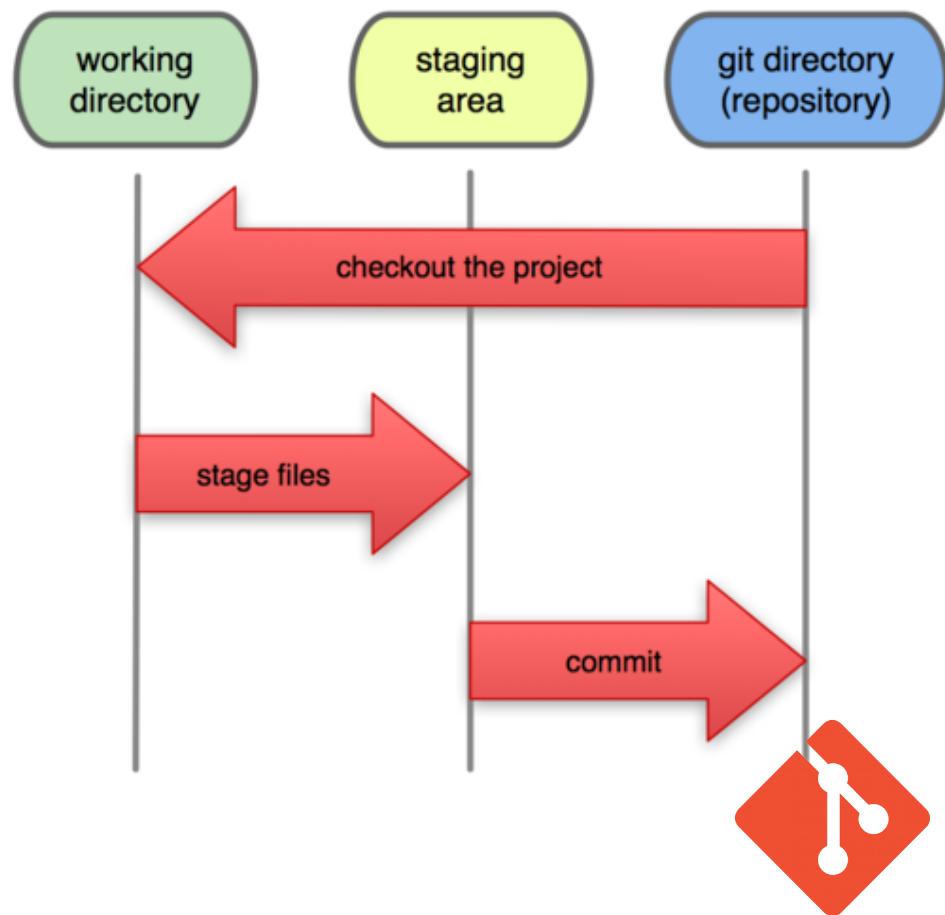
- Caso queira começar a controlar o versionamento dos arquivos existentes, você deve adicionar os arquivos desejados no **staging area**, seguido de um **commit**
 - O comando `git add .` adiciona todos os arquivos do diretório local no staging area

```
david@coyote:~/pilha$ git add .
david@coyote:~/pilha$ git commit -m "Versão inicial do projeto"
[master (root-commit) b0222f6] Versão inicial do projeto
 4 files changed, 88 insertions(+)
  create mode 100644 Makefile
  create mode 100644 main.c
  create mode 100644 pilha_lib.c
  create mode 100644 pilha_lib.h
david@coyote:~/pilha$ █
```



Os Três Estados

- Desta forma, o Git faz com que seus arquivos sempre estejam em um dos três estados fundamentais:
 - **Modificado** (modified), quando o arquivo sofreu mudanças mas ainda não foi selecionado
 - **Selecionado** (staged), quando você marca um arquivo modificado para que ele faça parte do próximo commit (consolidação)
 - **Consolidado** (committed), quando os arquivos estão seguramente salvos em seu repositório local



Adicionando um arquivo

- É possível incluir novos arquivos no repositório local normalmente, usando editores ou recursos do SO
- A ferramenta `git status` pode ser usada para identificar os estados dos arquivos do repositório

```
david@coyote:~/pilha
david@coyote:~/pilha$ echo "Implementação de pilha em C" > README.md
david@coyote:~/pilha$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README.md

nothing added to commit but untracked files present (use "git add" to track)
david@coyote:~/pilha$ git add README.md
david@coyote:~/pilha$
```



```
david@coyote:~/pilha
david@coyote:~/pilha$ echo "Implementação de pilha em C" > README.md
david@coyote:~/pilha$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README.md

nothing added to commit but untracked files present (use "git add" to track)
david@coyote:~/pilha$ git add README.md
david@coyote:~/pilha$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README.md

david@coyote:~/pilha$ git commit -m "Inclusão do arquivo README.md"
[master a411974] Inclusão do arquivo README.md
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
david@coyote:~/pilha$ git status
On branch master
nothing to commit, working directory clean
david@coyote:~/pilha$
```



Alterando um arquivo

- Os arquivos também podem ser editados usando editores de texto ou outros recursos de seu SO
- Novamente, usamos a ferramenta `git status` para identificar os estados dos arquivos do repositório

```
david@coyote: ~/pilha
david@coyote:~/pilha$ echo "Implementação de pilha em C (v2.0)" > README.md
david@coyote:~/pilha$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
david@coyote:~/pilha$ █
```



```
david@coyote:~/pilha
david@coyote:~/pilha$ echo "Implementação de pilha em C (v2.0)" > README.md
david@coyote:~/pilha$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
david@coyote:~/pilha$ git add README.md
david@coyote:~/pilha$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

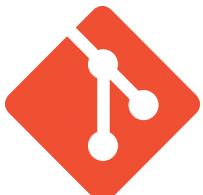
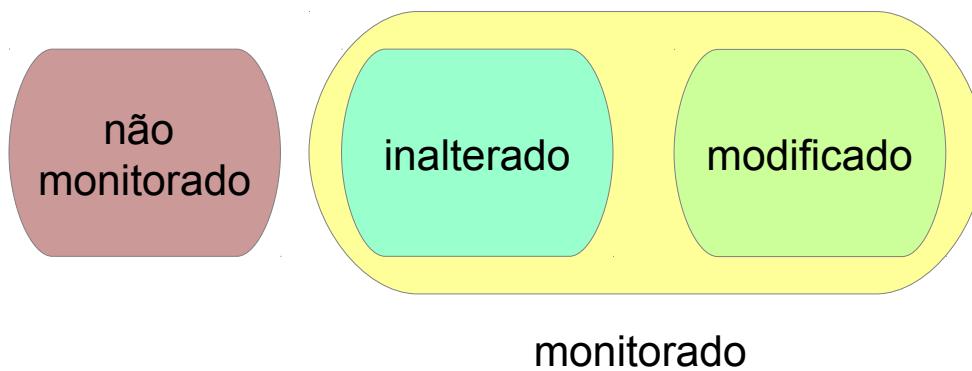
    modified:   README.md

david@coyote:~/pilha$ git commit -m "Nova versão do arquivo README.md"
[master cb26dad] Nova versão do arquivo README.md
 1 file changed, 1 insertion(+), 1 deletion(-)
david@coyote:~/pilha$ git status
On branch master
nothing to commit, working directory clean
david@coyote:~/pilha$
```



Gravando Alterações no Repositório

- É preciso efetuar um commit em seu repositório cada vez que o projeto atinge um estado no qual você queira gravar
- Na hora do commit, cada arquivo do diretório de trabalho pode estar **monitorado** ou **não monitorado**
 - Arquivos monitorados são: (i) arquivos que estavam no último commit, podendo estar **inalterados** ou **modificados**; (ii) arquivos adicionados via `git add`



Fazendo Commit de Suas Mudanças

- Quando sua área de seleção estiver do jeito que você quer, você pode fazer o commit de suas mudanças
 - Note que todos os arquivos criados ou modificados que não foram selecionados através do `git add` não fará parte deste commit
- Para fazer o commit, usamos o comando `git commit`:
 - `$ git commit`
- Ao fazer isso, um editor de texto é acionado para que você informe a mensagem do commit
 - Você pode configurar o git para usar um editor de sua preferência, através do comando `git config --global core.editor`
- Você também pode informar a mensagem de commit através do comando `git commit`, através da flag `-m`:
 - `$ git commit -m "Correção do bug 165"`



Ignorando Arquivos

- É comum ter arquivos que você não quer que o Git adicione ou mostre como arquivos não monitorados
 - Normalmente estes arquivos são gerados automaticamente como arquivos de log ou produzidos pelo seu sistema de build

```
david@coyote:~/pilha
david@coyote:~/pilha$ make
gcc -c main.c
gcc -c pilha_lib.c
gcc -o pilha main.o pilha_lib.o
david@coyote:~/pilha$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    main.o
    pilha
    pilha_lib.o

nothing added to commit but untracked files present (use "git add" to track)
david@coyote:~/pilha$
```



Ignorando Arquivos

- É comum ter arquivos que você não quer que o Git 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 `.gitignore` com uma lista de padrões a serem ignorados pelo Git

```
# ignorar arquivos terminados em .o
*.o

# ignorar o arquivo pilha
pilha

# ignorar todos os arquivos de log/
log/
```



Ignorando Arquivos

- Após criar o `.gitignore` e adicioná-lo no staging area, o git passa a ignorar todos os arquivos listados em seu interior

```
david@coyote:~/pilha
david@coyote:~/pilha$ vi .gitignore
david@coyote:~/pilha$ git add .gitignore
david@coyote:~/pilha$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   .gitignore

david@coyote:~/pilha$ git commit -m "Adicionei o .gitignore"
[master 11c4ac3] Adicionei o .gitignore
 1 file changed, 6 insertions(+)
  create mode 100644 .gitignore
david@coyote:~/pilha$ █
```



Visualizando o Histórico de Commits

- Depois que você tiver criado vários commits, você provavelmente vai querer ver o que aconteceu
- A ferramenta mais básica e poderosa para fazer isso é o comando `git log`



```
david@coyote: ~/pilha
```

```
david@coyote:~/pilha$ git log
```

```
commit 11c4ac3b452c876dd262ce4bac79cf4a02ea15fc
```

```
Author: David Fernandes <david@icomp.ufam.edu.br>
```

```
Date: Mon Nov 7 09:40:34 2016 -0400
```

Adicionei o .gitignore

```
commit cb26dadcb6706b17a6fe3478fa6bbac731f5fc9c
```

```
Author: David Fernandes <david@icomp.ufam.edu.br>
```

```
Date: Mon Nov 7 09:23:17 2016 -0400
```

Nova versão do arquivo README.md

```
commit a41197476d1b64127cba9f5b7d060a5b640a6eba
```

```
Author: David Fernandes <david@icomp.ufam.edu.br>
```

```
Date: Mon Nov 7 09:15:32 2016 -0400
```

Inclusão do arquivo README.md

```
commit b0222f60e97e2e66864670c494fd2941b8cadca3
```

```
Author: David Fernandes <david@icomp.ufam.edu.br>
```

```
Date: Mon Nov 7 08:46:49 2016 -0400
```

Versão inicial do projeto

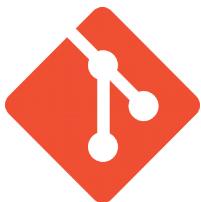
```
david@coyote:~/pilha$
```



Visualizando o Histórico de Commits

- Se você deseja que cada registro de log apareça em uma única linha, use a opção **--oneline**

```
david@coyote: ~/pilha
david@coyote:~/pilha$ git log --oneline
11c4ac3 Adicionei o .gitignore
cb26dad Nova versão do arquivo README.md
a411974 Inclusão do arquivo README.md
b0222f6 Versão inicial do projeto
david@coyote:~/pilha$ █
```



Visualizando suas Mudanças

- O comando `git status` informa quais arquivos foram modificados em um dado momento do fluxo de trabalho
- Para obter informações sobre o que foi modificado dentro dos arquivos você pode usar o `git diff`
- Por exemplo, para verificar as diferenças entre os arquivos do diretório de trabalho e a área de seleção:

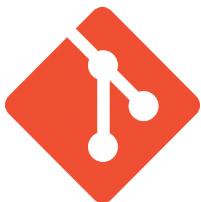
```
david@coyote:~/pilha
david@coyote:~/pilha$ echo "Implementação de pilha em C (v3.0)" > README.md
david@coyote:~/pilha$ git diff
diff --git a/README.md b/README.md
index 7d033da..141804a 100644
--- a/README.md
+++ b/README.md
@@ -1 +1 @@
-Implementação de pilha em C (v2.0)
+Implementação de pilha em C (v3.0)
david@coyote:~/pilha$
```



Visualizando suas Mudanças

- O `git diff` também é usado para listar as diferenças entre os arquivos do diretório de trabalho e o último commit

```
david@coyote: ~/pilha
david@coyote:~/pilha$ git diff HEAD
diff --git a/README.md b/README.md
index 7d033da..141804a 100644
--- a/README.md
+++ b/README.md
@@ -1 +1 @@
-Implementação de pilha em C (v2.0)
+Implementação de pilha em C (v3.0)
david@coyote:~/pilha$
```



Removendo Arquivos

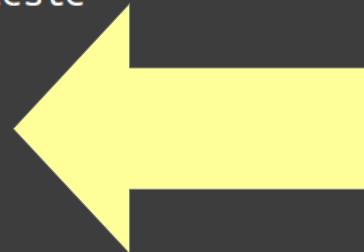
- Para remover um arquivo do repositório, você pode removê-lo usando as ferramentas do seu SO



```
david@coyote:~/pilha
```

```
david@coyote:~/pilha$ echo "Apenas um teste" > teste.txt
david@coyote:~/pilha$ git add .
david@coyote:~/pilha$ git commit -m "Inclusão do arquivo de teste"
[master 59f09c2] Inclusão do arquivo de teste
 1 file changed, 1 insertion(+)
  create mode 100644 teste.txt
david@coyote:~/pilha$ rm teste.txt
david@coyote:~/pilha$ git status
On branch master
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:    teste.txt
```



```
no changes added to commit (use "git add" and/or "git commit -a")
```

```
david@coyote:~/pilha$ git add .
david@coyote:~/pilha$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
  (use "git reset HEAD <file>..." to unstage)
```

```
    deleted:    teste.txt
```

```
david@coyote:~/pilha$ git commit -m "Remoção do arquivo de teste"
```

```
[master 7919c19] Remoção do arquivo de teste
 1 file changed, 1 deletion(-)
 delete mode 100644 teste.txt
```

```
david@coyote:~/pilha$ █
```

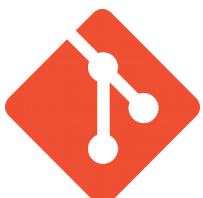


Desfazendo Coisas

- Em qualquer momento do desenvolvimento de um software, você pode querer desfazer alguma coisa
- Desta forma, veremos algumas ferramentas básicas para desfazer modificações que você fez

Cuidado, porque você não pode desfazer algumas das mudanças vistas nessa parte do curso

Essa é uma das poucas áreas no Git onde você pode perder algum trabalho se fizer errado



Desfazendo um Arquivo Modificado

- Caso faça modificações em um arquivo, é possível revertê-lo para o que era durante o último commit
- O comando `git status` diz a você como fazer isso:

```
david@coyote:~/pilha
david@coyote:~/pilha$ echo "Implementação de pilha em C (v4.0)" > README.md
david@coyote:~/pilha$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

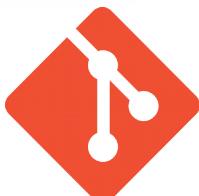
no changes added to commit (use "git add" and/or "git commit -a")
david@coyote:~/pilha$ git checkout -- README.md
david@coyote:~/pilha$ cat README.md
Implementação de pilha em C (v3.0)
david@coyote:~/pilha$
```



Vendo Versões Antigas de Arquivos

- Caso precise ver o conteúdo de versões anteriores de um arquivo, você pode usar o comando `git show`:

```
david@coyote: ~/pilha
david@coyote:~/pilha$ git log --oneline
7919c19 Remoção do arquivo de teste
59f09c2 Inclusão do arquivo de teste
f598bea Alteração no arquivo README.md
11c4ac3 Adicionei o .gitignore
cb26dad Nova versão do arquivo README.md
a411974 Inclusão do arquivo README.md
b0222f6 Versão inicial do projeto
david@coyote:~/pilha$ git show a411974:README.md
Implementação de pilha em C
david@coyote:~/pilha$ █
```



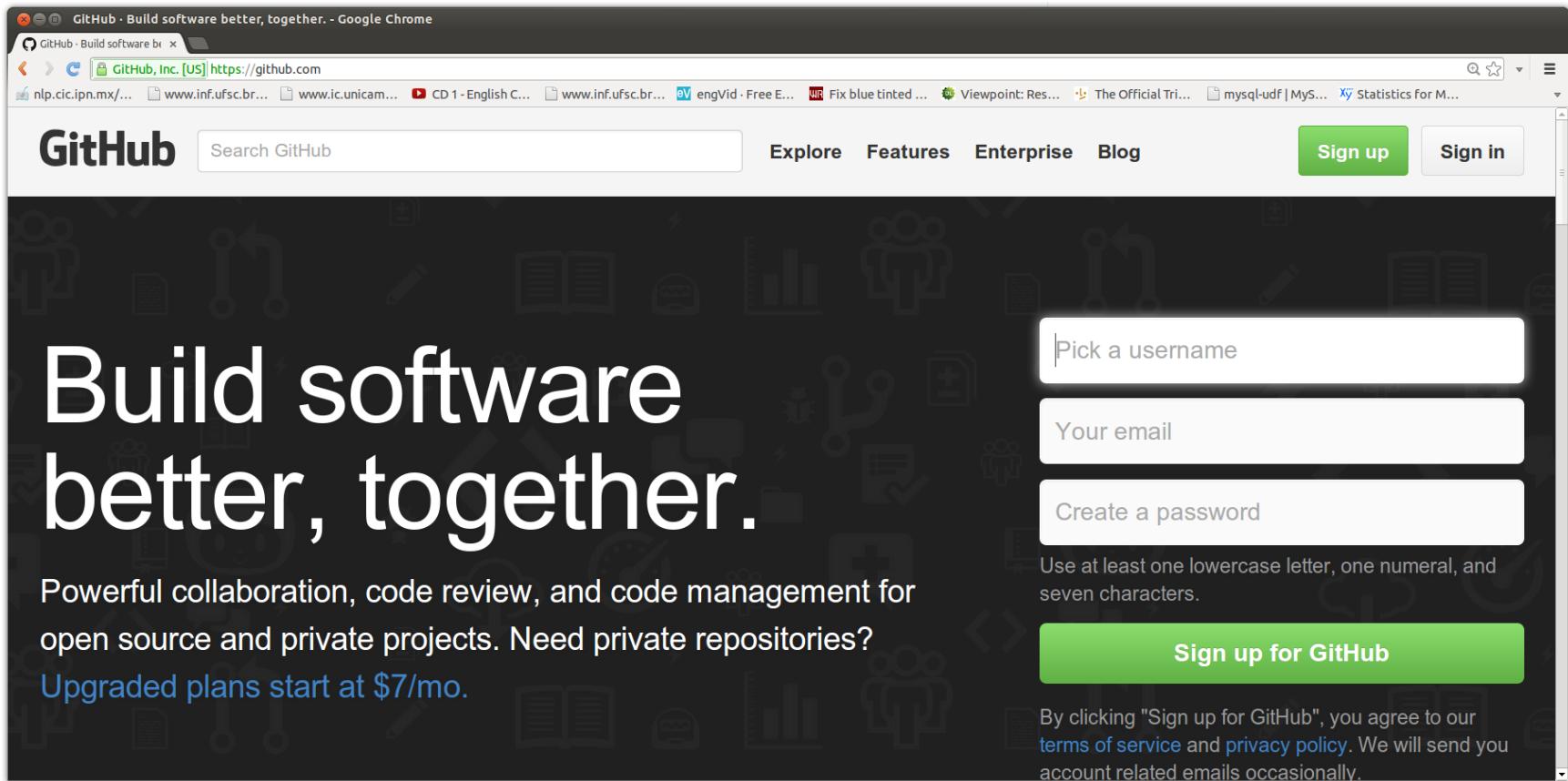
Vendo Versões Antigas de Arquivos

- O comando `git checkout` também pode ser usado para recuperar versões anteriores de arquivos

```
david@coyote:~/pilha$ git log --oneline
7919c19 Remoção do arquivo de teste
59f09c2 Inclusão do arquivo de teste
f598bea Alteração no arquivo README.md
11c4ac3 Adicionei o .gitignore
cb26dad Nova versão do arquivo README.md
a411974 Inclusão do arquivo README.md
b0222f6 Versão inicial do projeto
david@coyote:~/pilha$ git show a411974:README.md
Implementação de pilha em C
david@coyote:~/pilha$ git checkout a411974 README.md
david@coyote:~/pilha$ cat README.md
Implementação de pilha em C
david@coyote:~/pilha$ git checkout 7919c19 README.md
david@coyote:~/pilha$ cat README.md
Implementação de pilha em C (v3.0)
david@coyote:~/pilha$
```



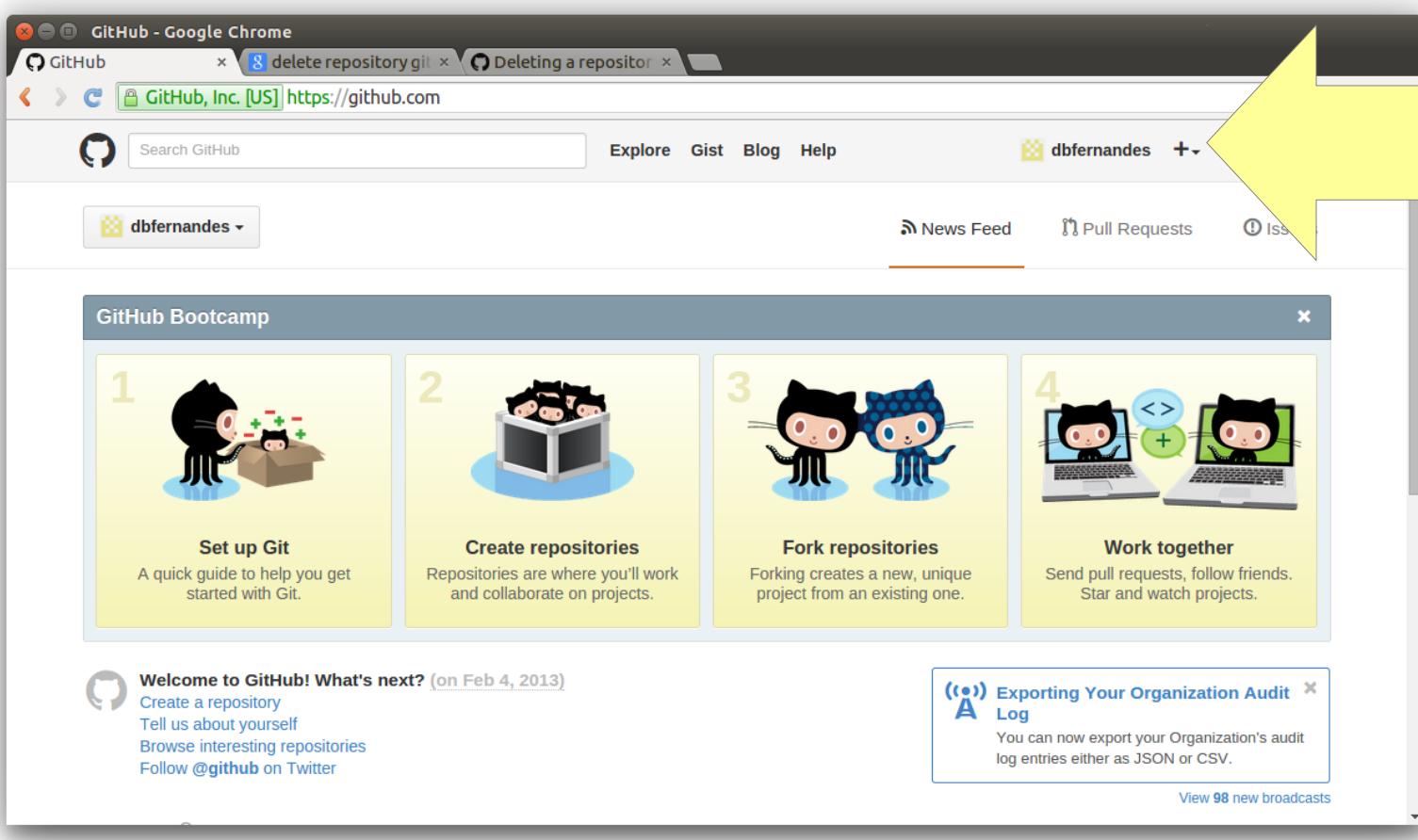
O GitHub



A screenshot of a Google Chrome browser window showing the GitHub sign-up page. The address bar shows the URL <https://github.com>. The GitHub logo is at the top left, followed by a search bar containing "Search GitHub". To the right are navigation links for "Explore", "Features", "Enterprise", and "Blog", along with "Sign up" and "Sign in" buttons. The main content features a large white text "Build software better, together." on a dark background with various icons. Below it, descriptive text reads: "Powerful collaboration, code review, and code management for open source and private projects. Need private repositories? Upgraded plans start at \$7/mo." To the right, there are three input fields: "Pick a username", "Your email", and "Create a password", each with a placeholder text. A note below the password field says: "Use at least one lowercase letter, one numeral, and seven characters." A large green "Sign up for GitHub" button is at the bottom right. At the very bottom, a small note states: "By clicking 'Sign up for GitHub', you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally."



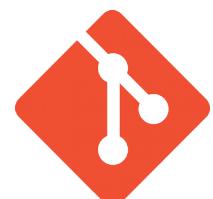
Criando um novo Repositório



A screenshot of a Google Chrome browser window displaying the GitHub homepage. The address bar shows three tabs: 'GitHub', 'g delete repository gil', and 'Deleting a repository'. The main navigation bar includes 'Explore', 'Gist', 'Blog', 'Help', and a user profile for 'dbfernandes'. A yellow arrow points from the top right towards the central content area. Below the navigation is a sidebar with a dropdown for 'dbfernandes'. The main content area features a 'GitHub Bootcamp' guide with four numbered steps:

- 1 Set up Git**: A quick guide to help you get started with Git.
- 2 Create repositories**: Repositories are where you'll work and collaborate on projects.
- 3 Fork repositories**: Forking creates a new, unique project from an existing one.
- 4 Work together**: Send pull requests, follow friends. Star and watch projects.

Below the bootcamp guide, there's a 'Welcome to GitHub! What's next?' section with links to 'Create a repository', 'Tell us about yourself', 'Browse interesting repositories', and 'Follow @github on Twitter'. To the right, a modal window titled 'Exporting Your Organization Audit Log' is open, stating 'You can now export your Organization's audit log entries either as JSON or CSV.' A message at the bottom right says 'View 98 new broadcasts'.



Salvando seu Repositório

Create a New Repository - Google Chrome

Create a New Repo

GitHub, Inc. [US] | https://github.com/new

Search GitHub

Pull requests Issues Gist

David

Create a new repository

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

Owner: dbfernandes / Repository name: pilha

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

Description (optional):

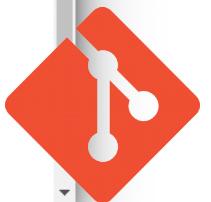
Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

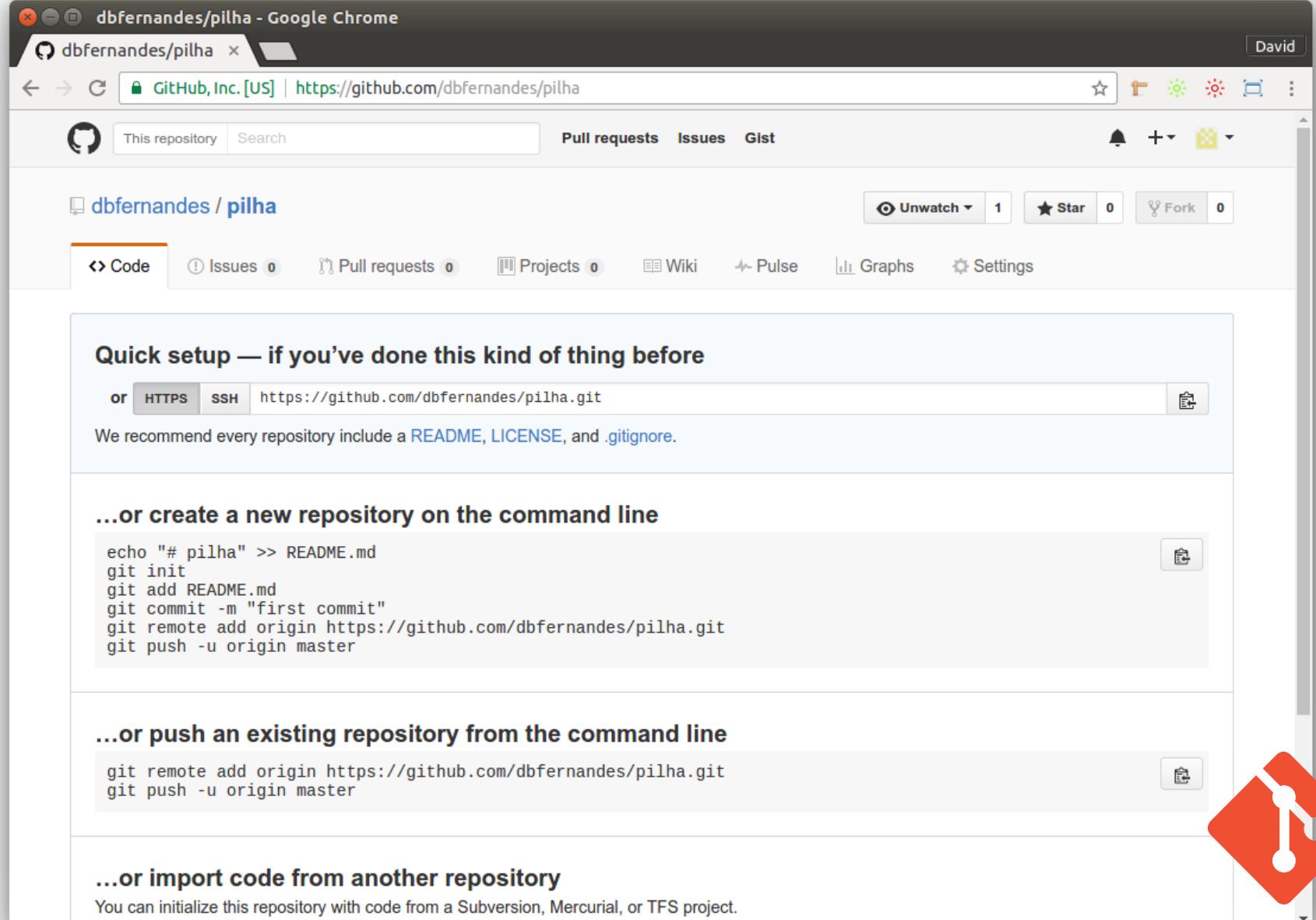
Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** Add a license: **None**

Create repository



Salvando seu Repositório



dbfernandes/pilha - Google Chrome
dbfernandes/pilha | GitHub, Inc. [US] | https://github.com/dbfernandes/pilha

This repository Search Pull requests Issues Gist

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

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

or HTTPS SSH https://github.com/dbfernandes/pilha.git

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

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

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

```
git remote add origin https://github.com/dbfernandes/pilha.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.

Salvando seu Repositório

The screenshot shows a GitHub repository page for 'dbfernandes/pilha' in a Google Chrome browser. The page includes navigation links for 'Pull requests', 'Issues', and 'Gist'. Below the header, there's a search bar and sections for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. A terminal window is overlaid on the page, displaying the following command and its execution:

```
david@coyote:~/pilha$ git remote add origin https://github.com/dbfernandes/pilha.git
david@coyote:~/pilha$ git push -u origin master
Username for 'https://github.com': dbfernandes
Password for 'https://dbfernandes@github.com':
Counting objects: 22, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (18/18), done.
Writing objects: 100% (22/22), 2.54 KiB | 0 bytes/s, done.
Total 22 (delta 6), reused 0 (delta 0)
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/dbfernandes/pilha.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
david@coyote:~/pilha$
```

At the bottom of the terminal window, there's a red diamond-shaped icon with a white gear and wrench symbol.

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

dbfernandes/pilha - Google Chrome

dbfernandes/pilha

GitHub, Inc. [US] | https://github.com/dbfernandes/pilha

Pull requests Issues Gist

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

No description or website provided. — Edit

7 commits 1 branch 0 releases 0 contributors

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

David Fernandes Remoção do arquivo de teste Latest commit 7919c19 an hour ago

📄 .gitignore	Adicionei o .gitignore	2 hours ago
📄 Makefile	Versão inicial do projeto	2 hours ago
📄 README.md	Alteração no arquivo README.md	an hour ago
📄 main.c	Versão inicial do projeto	2 hours ago
📄 pilha_lib.c	Versão inicial do projeto	2 hours ago
📄 pilha_lib.h	Versão inicial do projeto	2 hours ago

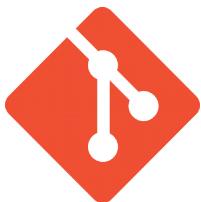
README.md

Implementação de pilha em C (v3.0)



Clonando um Repositório Existente

- Caso você queira copiar um repositório Git já existente, o comando necessário é `git clone`
 - Cada versão de cada arquivo no histórico do projeto é obtida quando você roda `git clone`
- Por exemplo, caso você queria clonar o projeto pilha do Github em outro computador, você pode fazê-lo da seguinte forma:
 - `git clone https://github.com/dbfernandes/pilha`
- Isso cria um diretório chamado `pilha`, com um diretório `.git` contendo todos os dados do repositório



Clonando um Repositório Existente

```
david@coyote: ~/implementacoes/pilha
david@coyote:~/implementacoes$ git clone https://github.com/dbfernandes/pilha
Cloning into 'pilha'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 22 (delta 6), reused 22 (delta 6), pack-reused 0
Unpacking objects: 100% (22/22), done.
Checking connectivity... done.
david@coyote:~/implementacoes$ cd pilha/
david@coyote:~/implementacoes/pilha$ git log --oneline
7919c19 Remoção do arquivo de teste
59f09c2 Inclusão do arquivo de teste
f598bea Alteração no arquivo README.md
11c4ac3 Adicionei o .gitignore
cb26dad Nova versão do arquivo README.md
a411974 Inclusão do arquivo README.md
b0222f6 Versão inicial do projeto
david@coyote:~/implementacoes/pilha$ 
```



Atualizando um Repositório Existente

- Você pode usar o comando `git push` para salvar as alterações do repositório local no repositório remoto

```
david@coyote:~/implementacoes/pilha
david@coyote:~/implementacoes/pilha$ echo "Implementação de pilha em C (v4.0)" > README.md
david@coyote:~/implementacoes/pilha$ git add .
david@coyote:~/implementacoes/pilha$ git commit -m "Nova versão do README.md"
[master 8f6068f] Nova versão do README.md
 1 file changed, 1 insertion(+), 1 deletion(-)
david@coyote:~/implementacoes/pilha$ git push origin master
Username for 'https://github.com': dbfernandes
Password for 'https://dbfernandes@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/dbfernandes/pilha
 7919c19..8f6068f  master -> master
david@coyote:~/implementacoes/pilha$
```



dbfernandes/pilha - Google Chrome

dbfernandes/pilha GitHub, Inc. [US] | https://github.com/dbfernandes/pilha

This repository Search Pull requests Issues Gist

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

No description or website provided. — Edit

8 commits 1 branch 0 releases 0 contributors

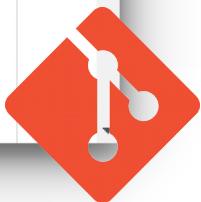
Branch: master New pull request Create new file Upload files Find file Clone or download

dbfernandes Nova versão do README.md Latest commit 8f6068f a minute ago

.gitignore	Adicionei o .gitignore	3 hours ago
Makefile	Versão inicial do projeto	4 hours ago
README.md	Nova versão do README.md	a minute ago
main.c	Versão inicial do projeto	4 hours ago
pilha_lib.c	Versão inicial do projeto	4 hours ago
pilha_lib.h	Versão inicial do projeto	4 hours ago

README.md

Implementação de pilha em C (v4.0)



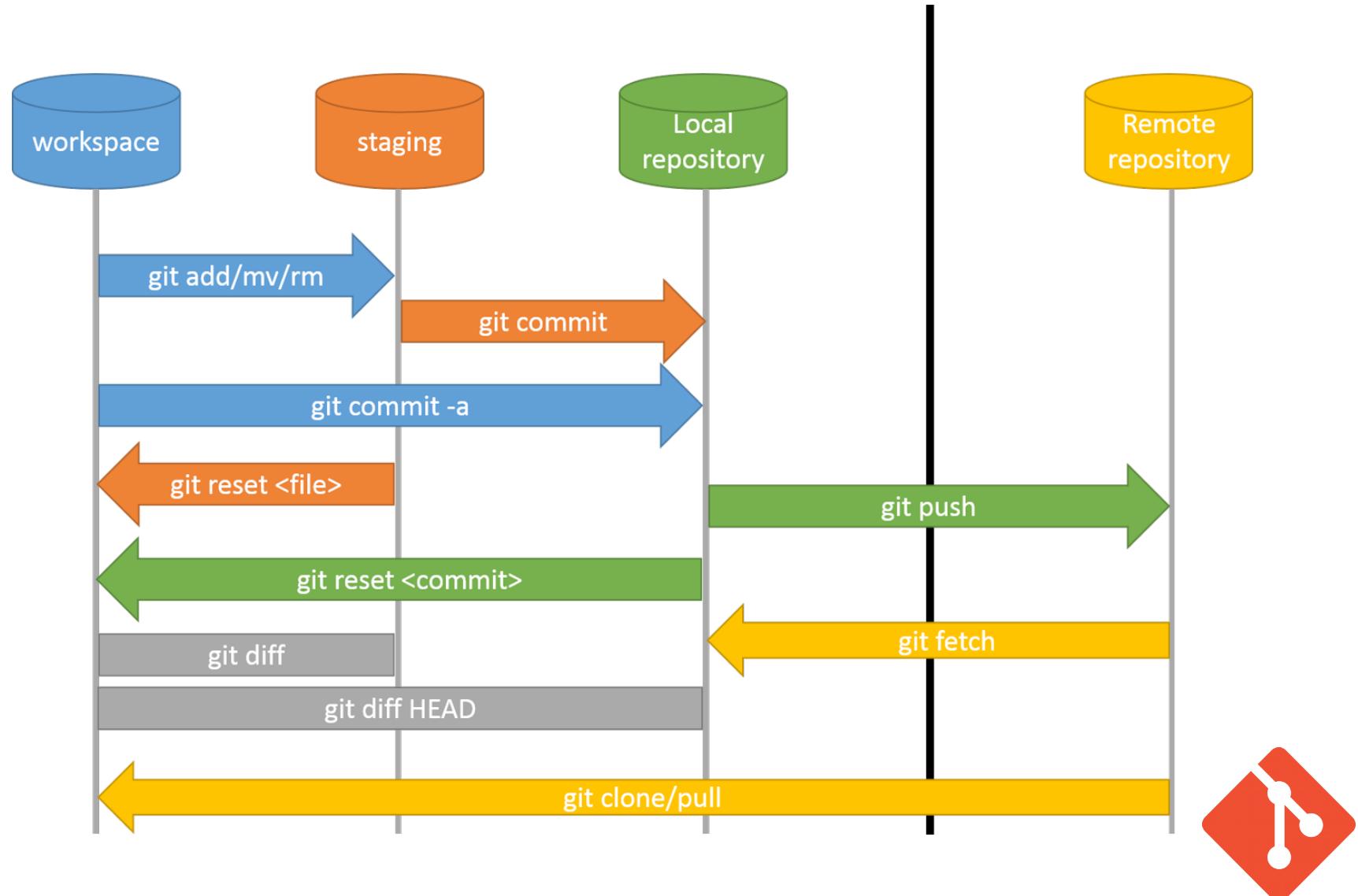
Atualizando um Repositório Existente

- Também pode usar o comando `git pull` para atualizar um repositório local a partir de um repositório remoto

```
david@coyote:~/implementacoes/pilha
david@coyote:~/pilha$ cd ~/pilha
david@coyote:~/pilha$ cat README.md
Implementação de pilha em C (v3.0)
david@coyote:~/pilha$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/dbfernandes/pilha
 * branch            master      -> FETCH_HEAD
   7919c19..8f6068f  master      -> origin/master
Updating 7919c19..8f6068f
Fast-forward
 README.md | 2 ++
 1 file changed, 1 insertion(+), 1 deletion(-)
david@coyote:~/pilha$ cat README.md
Implementação de pilha em C (v4.0)
david@coyote:~/pilha$
```



Resumo de Operações com o Git





Git Cheat Sheet

For more awesome cheat sheets
visit [rebellabs.org!](http://rebellabs.org/)



Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]:[file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
```

```
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

```
$ git push
```

Finally!

When in doubt, use git help

```
$ git command --help
```

Or visit <https://training.github.com/> for official GitHub training.

