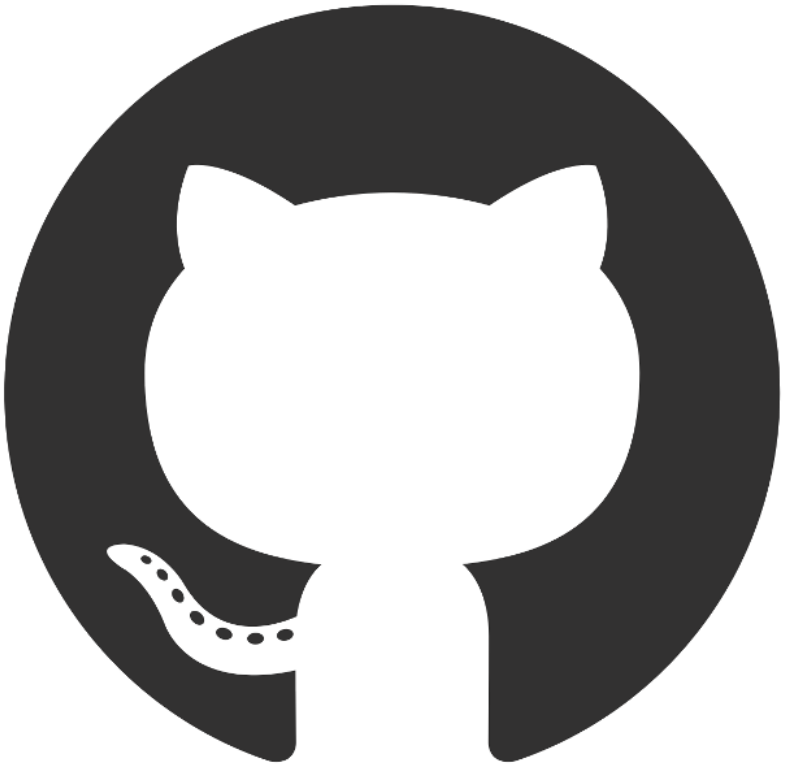
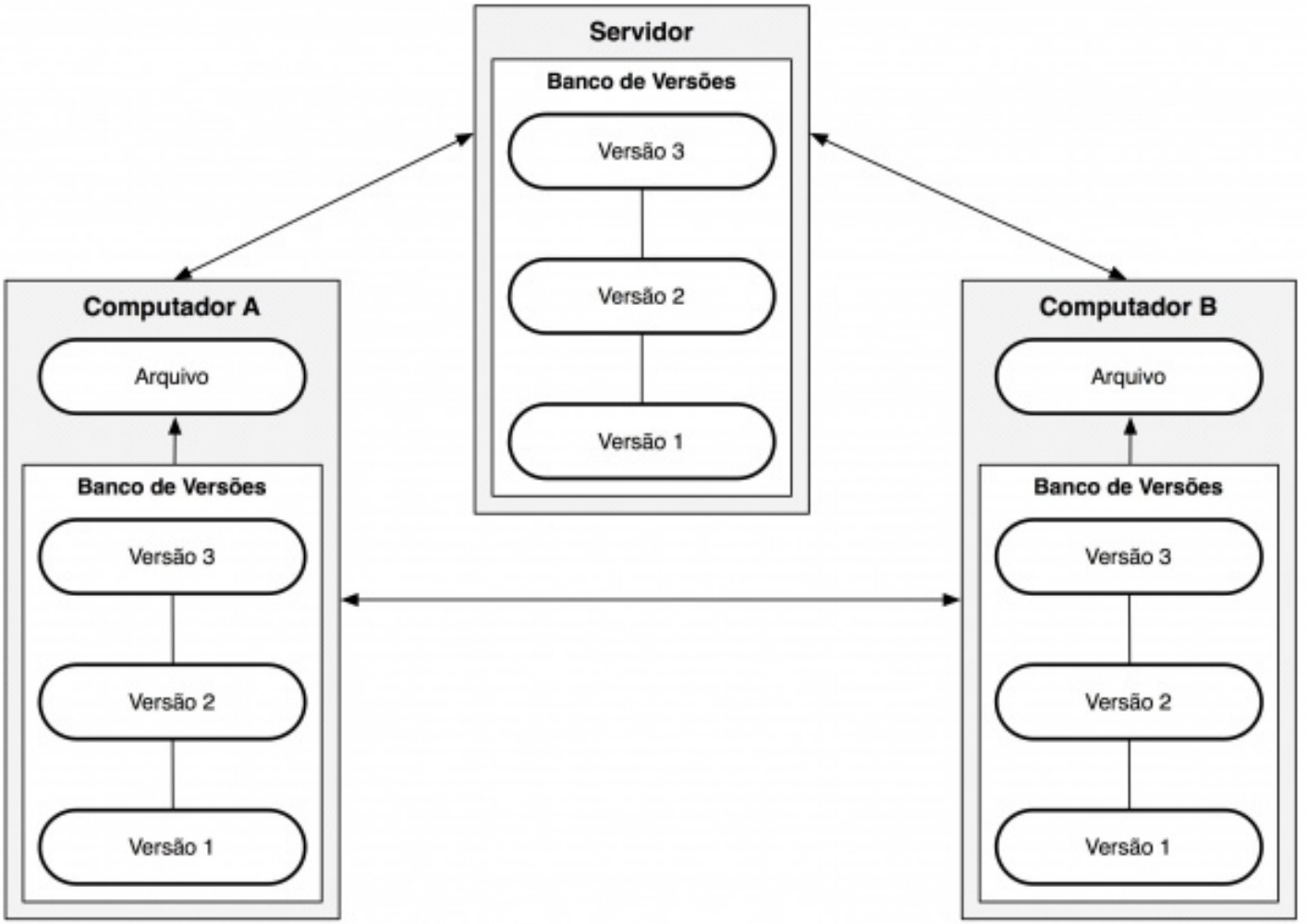
# Git e GitHub para iniciantes



- O que é Git
- O que é Github
- Controle de Versão
- Configurando o Git
- Essencial do Git
- Repositórios Remotos
- Ramificação (Branch)

# - O que é Git?

# - O que é Git?

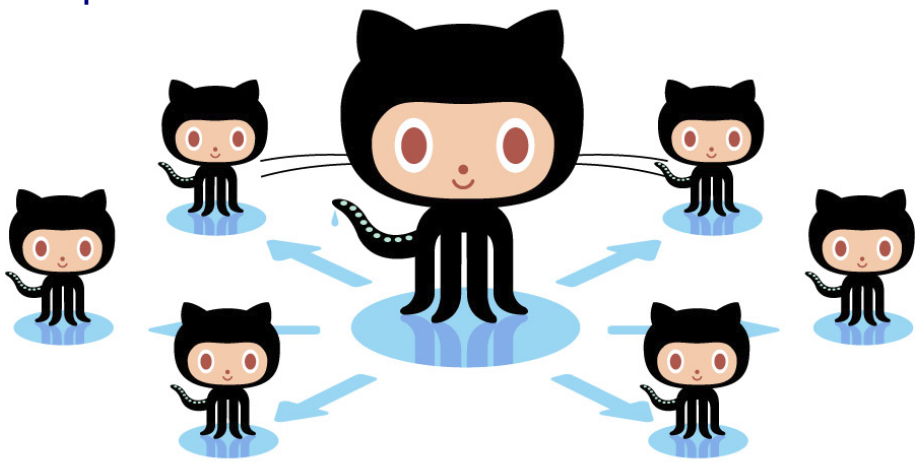

Servidor
Banco de Versões
Versão 3
Versão 2
Versão 1
Computador A
Arquivo
Banco de Versões
Versão 3
Versão 2
Versão 1
Computador B
Arquivo
Banco de Versões
Versão 3
Versão 2
Versão 1

# - O que é Github?

- O Github é um serviço web lançado em 2008 que oferece diversas funcionalidades extras aplicadas ao Git. É usado para que desenvolvedores possam hospedar seus projetos.
- O GitHub costuma ser preferido por oferecer também alguns recursos de redes sociais, já que é possível seguir projetos de outros desenvolvedores e ainda comenta-los.
- Além disso, o Github torna possivel, através do Git, compartilhar um bloco de código, trocar ideias, comentar os demais projetos e ainda pegar o código de alguém para modificar.
- Quase todos os projetos/frameworks/bibliotecas sobre desenvolvimento open source estão no github, e você pode acompanhá-los através de novas versões, contribuir informando bugs ou até mesmo enviando código e correções.
- Ele está disponível gratuitamente, com limite de armazenamento de 300MB. O serviço também oferece planos pagos, com isso, os desenvolvedores podem ter um maior controle sobre o código fonte, bem como adicionar desenvolvedores fixos e esconder os códigos dos demais membros. O GitHub funciona basicamente na nuvem, assim sendo, o projeto pode ser acessado de qualquer local.

# - Como funciona o site?

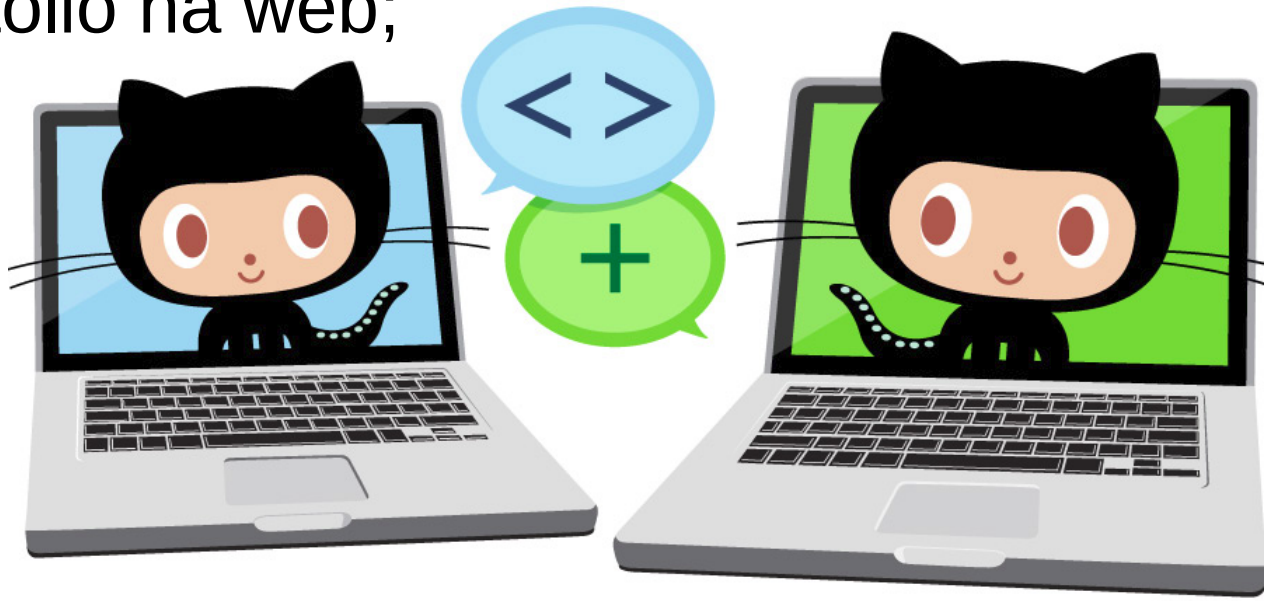
- Lugar pra aprender
  - https://guides.github.com/
  - https://github.com/tidyverse/ggplot2
  - https://github.com/justmarkham/scikit-learn-videos
  - https://education.github.com/pack
  - https://github.com/darribas/geopandas
  - https://github.com/marketplace
  - https://github.com/rstudio/rstudio
  - https://github.com/scikit-learn/scikit-learn
  - https://github.com/explore



- Importância do Github para o mercado de trabalho
  - https://thenextweb.com/dd/2018/06/04/its-official-microsoft-has-acquired-github/
  - https://thenextweb.com/apps/2012/09/02/my-github-resume-generates-resume-github-account/
  - https://github.com/douglas/IWantToWorkAtGloboCom

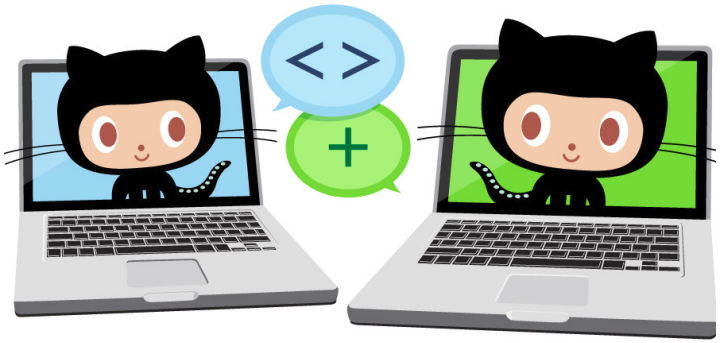
- **Um pouco de controle de versão:**

  - Controlar trabalhos da faculdade;
  - Varias cópias de um mesmo projeto com pequenas alterações
  - Ex: Apaquei um arquivo importante e não tem como recuperar
  - Portólio na web;



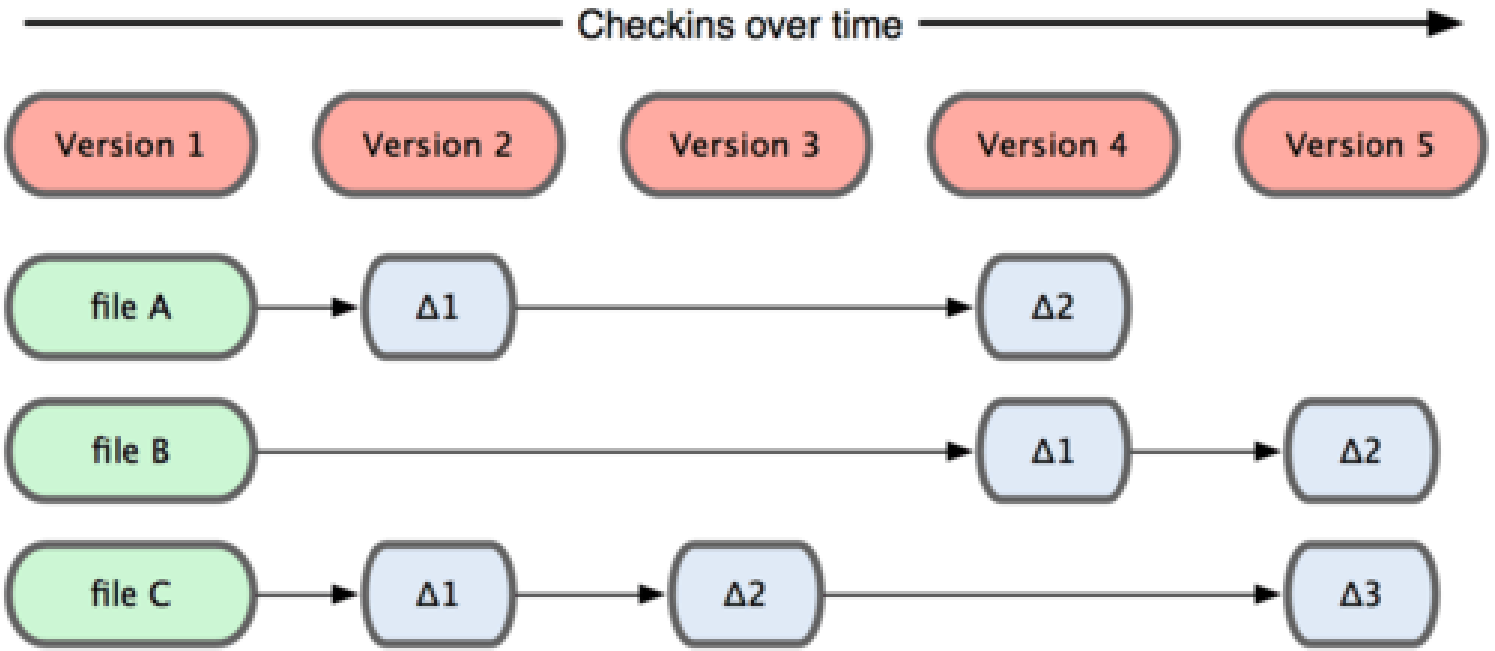
- **Um pouco de controle de versão:**

  - Sistema com a finalidade de gerenciar diferentes versões de um documento
  - Sistema responsável por criar versões com as diferentes modificações



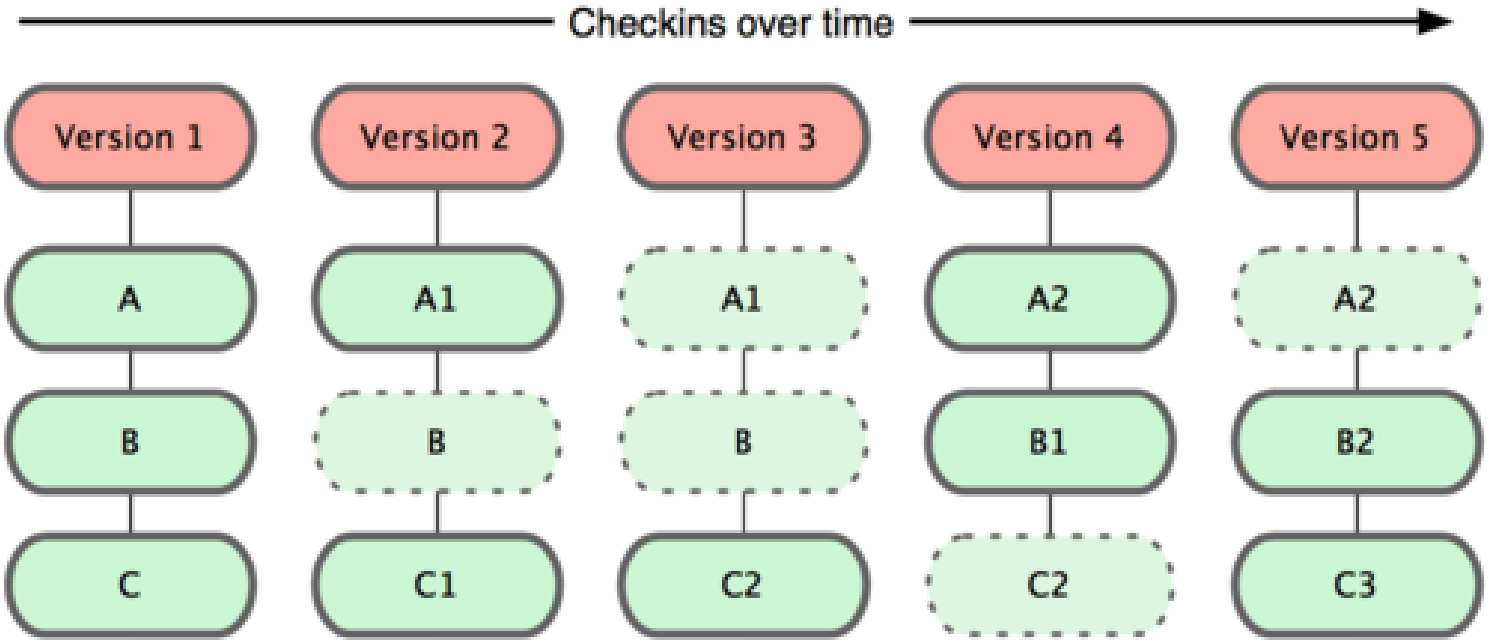
- **Como outros sistemas funcionam:**

  - Existem diferentes versões que vão sendo armazenadas. Ele compara os arquivos em diferentes momentos e salva um arquivo com as diferenças. Não há a possibilidade de criar branchs



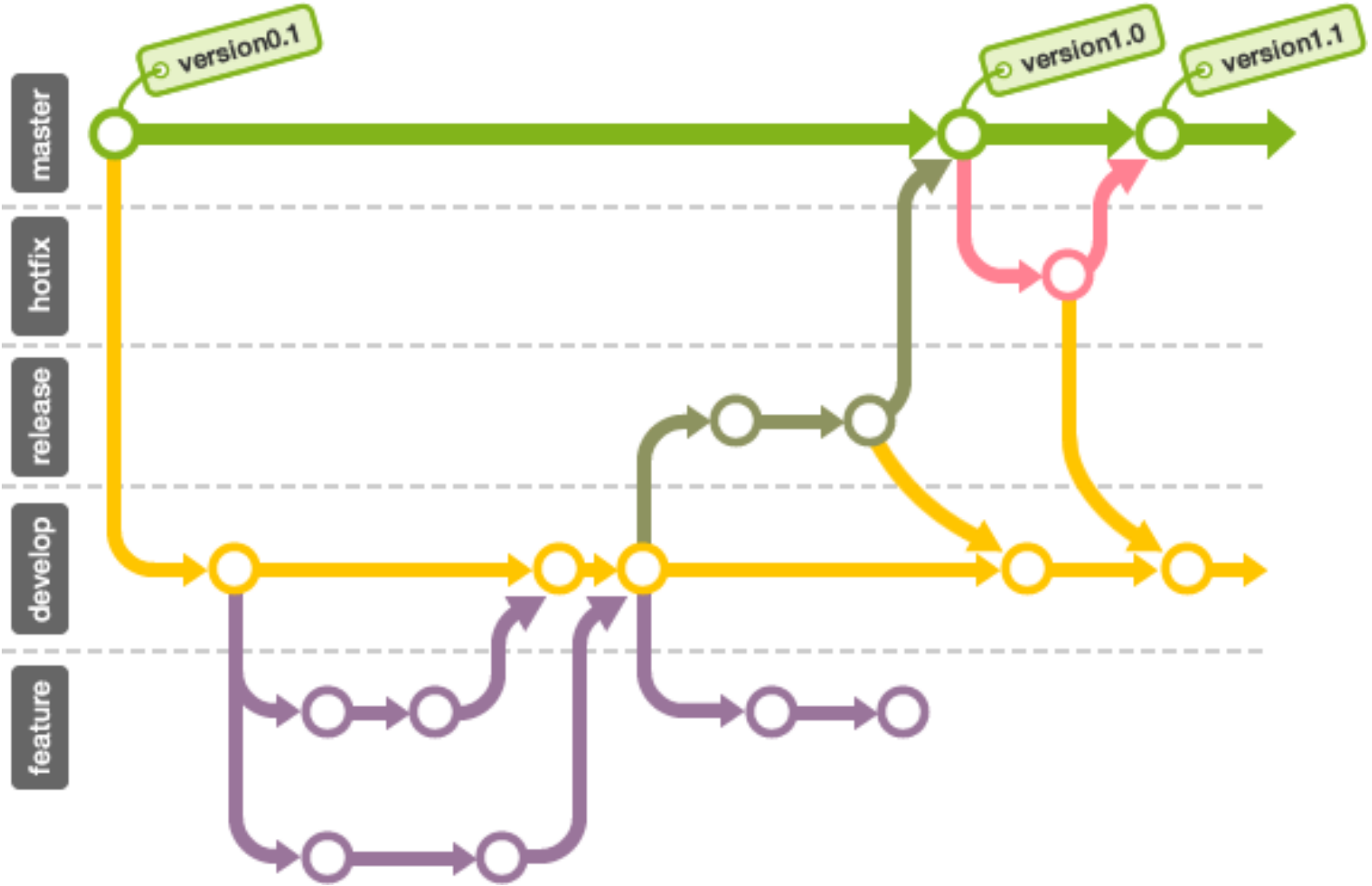
- **Sistema Git: (snapshots)**

  - Ele tira "fotos" dos arquivos e vai levando os snapshots versão a versão tendo os arquivos mudado ou não. Ele leva um link para arquivos que não foram modificados.



- **Sistema Git: (snapshots)**

  - Vantagens do sistema Git:
    - Velocidade
    - Design simples
    - Possibilidade de desenvolvimento não linear
    - Capaz de lidar com grandes progetos (Linux)


master
hotfix
release
develop
feature
version0.1
version1.0
version1.1

- **Configurando o Git:**

- Instalando o Git  http://git-scm.com
- Criando a conta no Github  https://github.com/join
- Praticando os comandos básicos  https://try.github.io/

- **Configuração inicial do Git:**

- Definir nome de usuario, email e etc. O git quarda suas informações em três lugares o git config do sistema todo, o git config do usuario e o git config do projeto
- Vamos definir as informações pra todos os repositórios de um usuario usando o git config global.

Configurar o nome de usuário:

```
$ git config --global user.name "walefmachado"
```

Configurar o e-mail:

```
$ git config --global user.email "walefm2@hotmail.com"
```

Verificar:

```
$ git config user.name
```

```
$ git config user.email
```

```
$ git config --list
```

- **Essencial do Git**

- Inicializando um repositório

Criar uma pasta:

```
$ mkdir curso-git
```

Entrar na pasta:

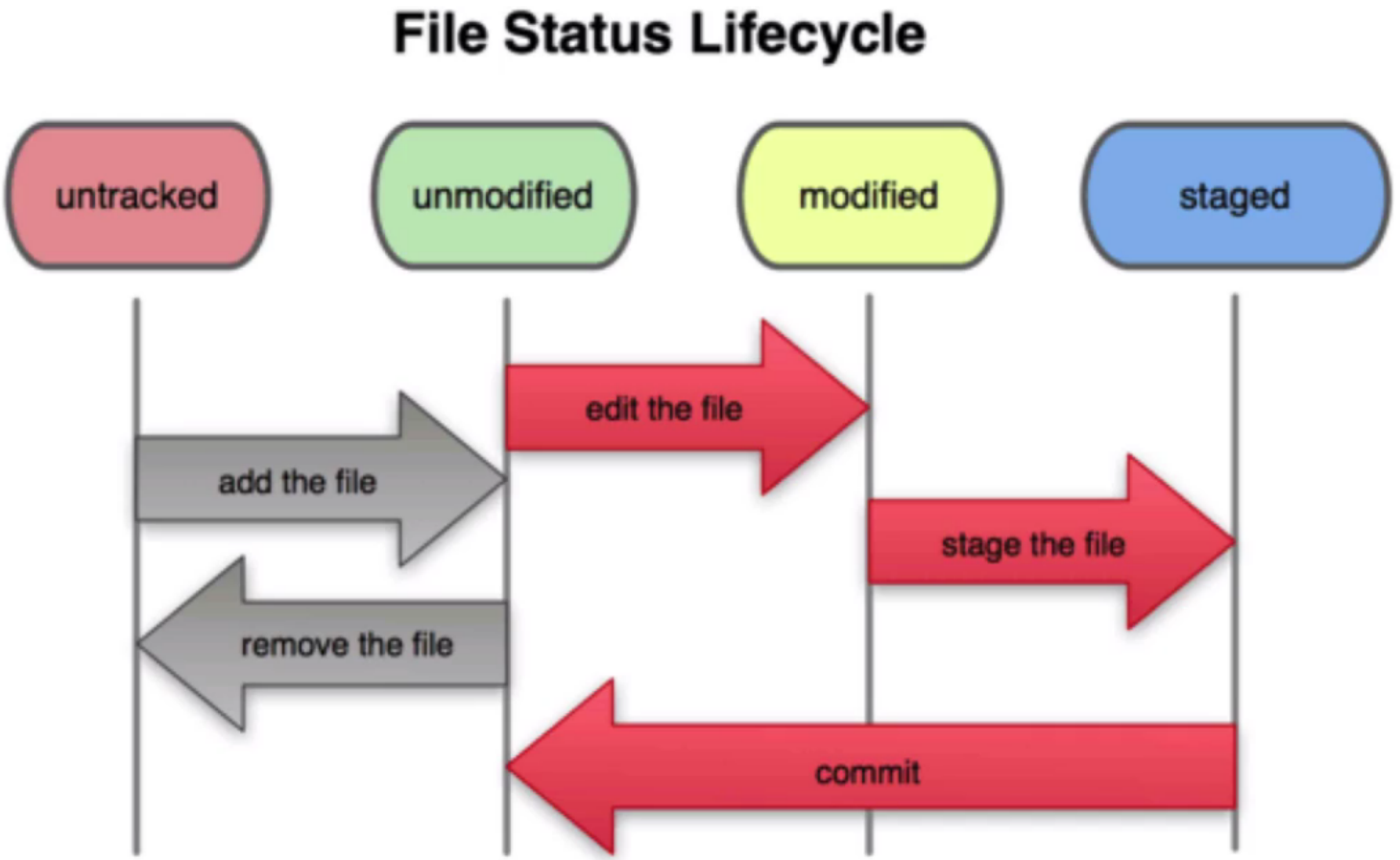
```
$ cd curso-git
```

Inicializar um repositorio git: com isso o git fica atento as mudanças

```
$ git init
```

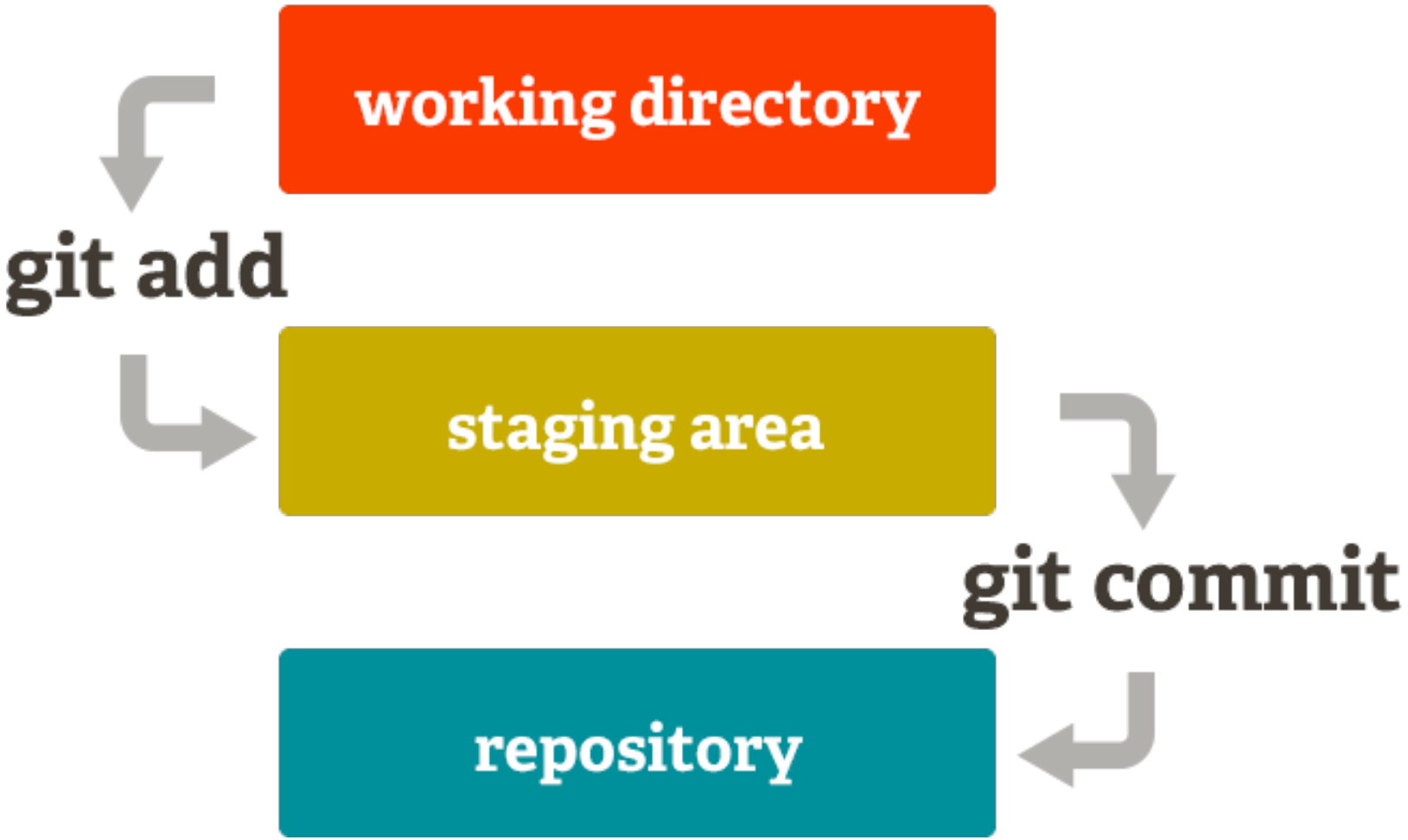
- **O ciclo de vida dos status de seus arquivos**

- Antes de começar a trabalhar com os arquivos é necessário entender o ciclo de vida dos arquivos.



- **O ciclo de vida dos status de seus arquivos**

- Untracked: Foi adicionado no repositório mas ainda não foi visto pelo git
- Unmodified: Não modificado mas já existe no git
- Modified: Existe no git e foi modificado
- Staged: Área de transferência para os arquivos modificados transfere os arquivos quando fizer commit


working directory
git add
staging area
git commit
repository

- **No git:**

Git status: Reportar o estado do repositório:

```
$ git status
```

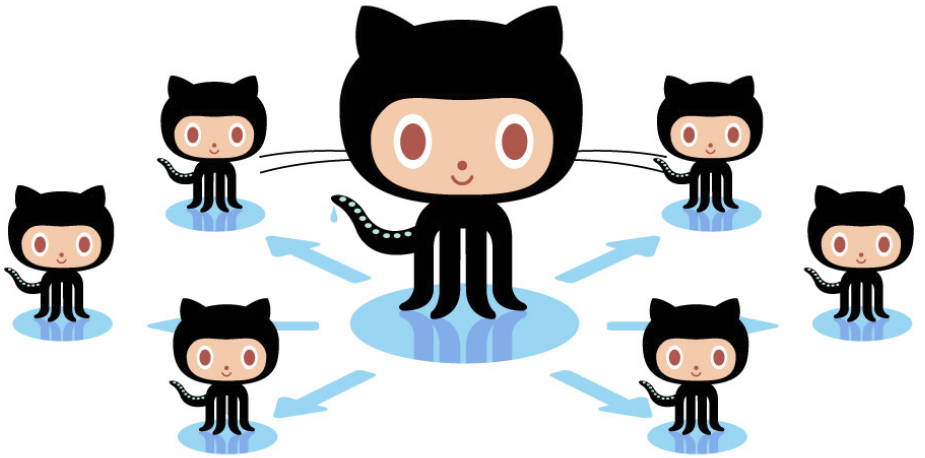
Colocar arquivo README.md na past

```
$ git status
```

Adicionar o arquivo ao untracked:

```
$ git add README.md
```

```
$ git status
```



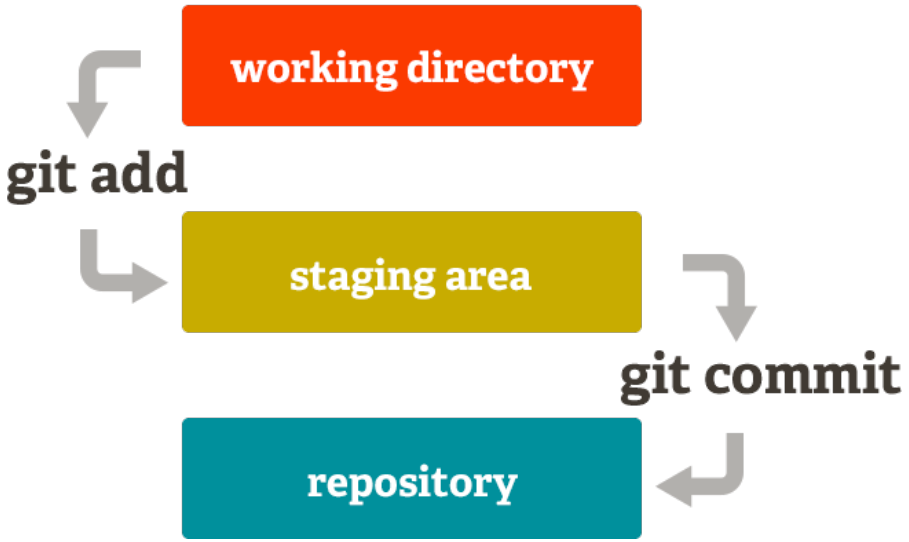
- **No git:**
  - Já existe um arquivo no working directory que pode ser adicionado a área de transferencia.
    experimentar: modificar e pedir status.

Adicionando o arquivo:

```
$ git add README.md
```

```
$ git status
```

Lembrar do ciclo:



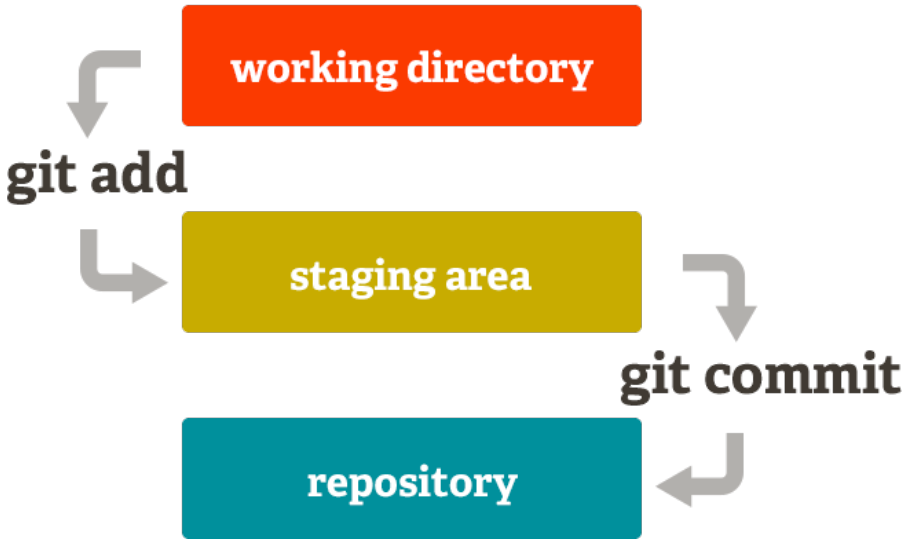

- **No git:**
  - Já existe um arquivo no sated que pode ser commitado. experimentar: modificar e pedir status.

Criar o primeiro Commit:

```
$ git commit -m "mensagem descrevendo a modificação"
```

```
$ git status
```

Lembrar do ciclo:




- # Visualizando logs
  - Depois dos primeiros commits e da criação dos primeiros arquivos é importante pode ver o historico das modificações. Para isso temos o comando git log:

```
$ git log
```

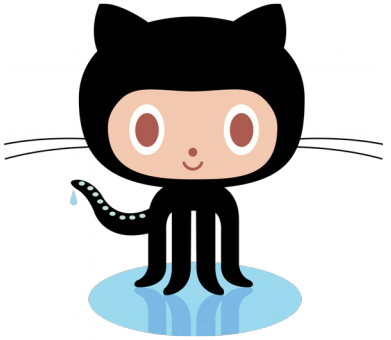
Ele nos mostra a identificação do commit, o autor e o email, e a data da modificação além da mensagem escrita no comit.

Filtrar por autor:

```
$ git log --author="walefmachado"
```

Retorna só o nome e a quantidade de commits

```
$ git shortlog -sn
```



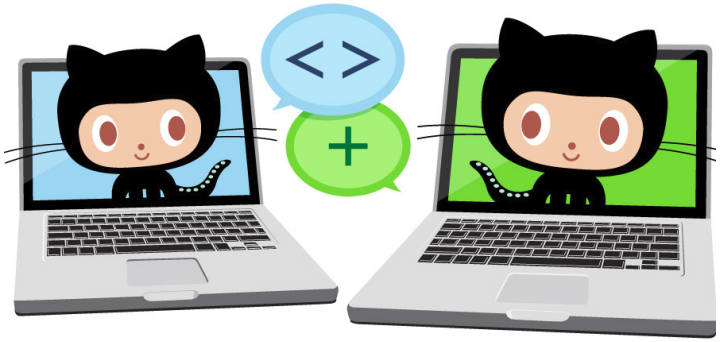
- **Visualizando o diff**
  - Ver as mudanças antes de salvar e fazer commit.
  - Editar o arquivo README.md ***

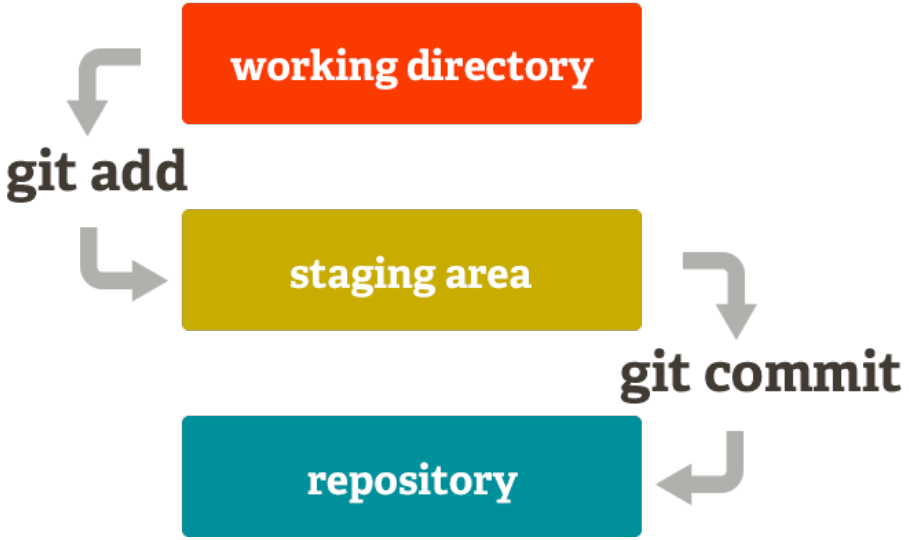
Mostra a mudança antes de commitar.

```
$ git diff
```

Reverter a modificação:

```
$ git checkout README.md
```





- **Repositórios Remotos:**
  - Criando um repositório no Github: https://github.com/

Ligando repositório local a um remoto:

```
$ git remote add origin https://github.com/walefmachado/curso-git.git
```

```
$ git push -u origin master
```

master é o branch padrão. Ele leva os arquivos para o repositório no github

- **Enviando mudanças para um repositório remoto:**
  - Se eu fizer uma modificação no computador como eu levo as alterações para o repositório remoto?

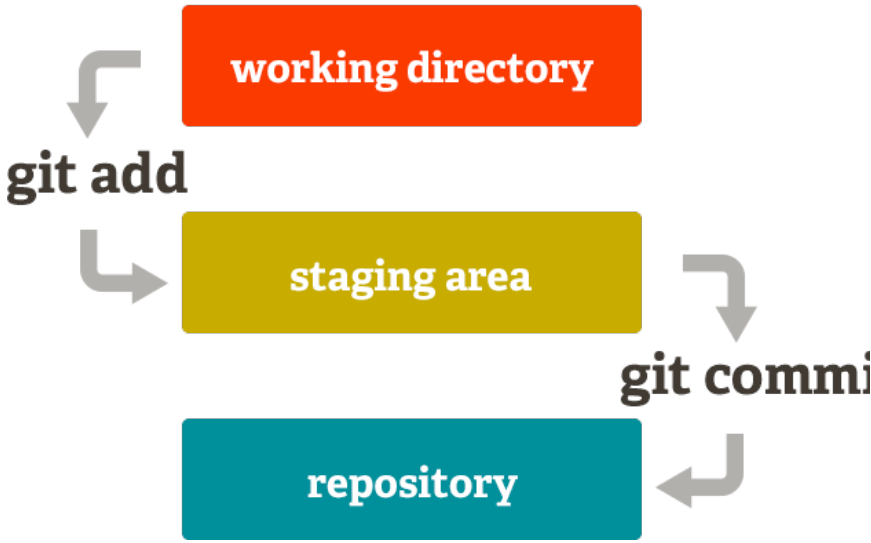
```
$ git add README.md
```

Fazer um Commit:

```
$ git commit -m "mensagem descrevendo a modificação"
```

```
$ git status
```

```
$ git push
```



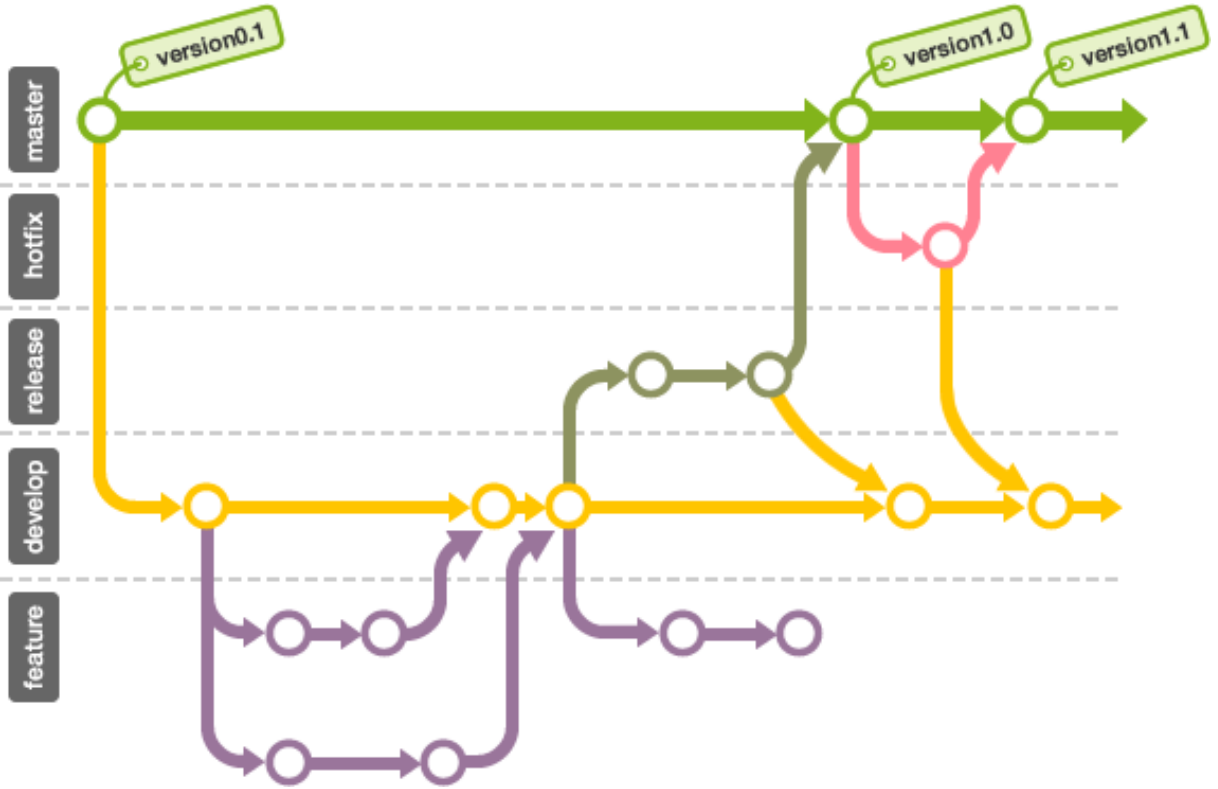
- **Clonando repositórios remotos**
  - Site do github no repositório e pegar o endereço

```
$ git clone https://github.com/walefmachado/interface-atuarial.git novo-nome
```

Pode clonar com um novo nome ou com o nome do repositório

- **Fazendo fork de um projeto:**
  - O fork pega um projeto que não é seu, esta em um repositório remoto de outro usuario, e cria uma cópia em um repositório no seu perfil do git hub.
  - Passos:
    - 1º- Fazer um fork no site do github
    - 2º- Fazer o <git clone> do seu repositório
    - 3º- Fazer as alterações que desejar
    - 4º- <git push> com as alterações para o seu fork
    - 5º- Pull request para o repositório original

- **Ramificação (Branch)**
  - O branch é um ponteiro móvel que leva a um commit. Quando iniciamos um repositório o ele é iniciado como branch master. É uma forma pra trabalhar de forma paralela em um projeto, dividindo-o em varios branchs.



- # Criando um branch
  - Vantagens:
    - Poder modificar sem alterar o branch master. Ex: Corrigir defeitos sem tirar a principal do ar.
    - Criar e apagar varios branchs facilmente.
    - Varias pessoas trabalhando em branchs diferentes evitando conflitos

- Criando um branch:

```
$ git checkout -b nome-do-branch
```

Pronto ele cria um novo branch e já entra nele pra você

```
$ git branch
```

Mostra os branchs existentes e com um asterisco mostra o que você esta no momento

Para ir para o branch nome-do-branch:

```
$ git checkout nome-do-branch
```

Volta para o branch master

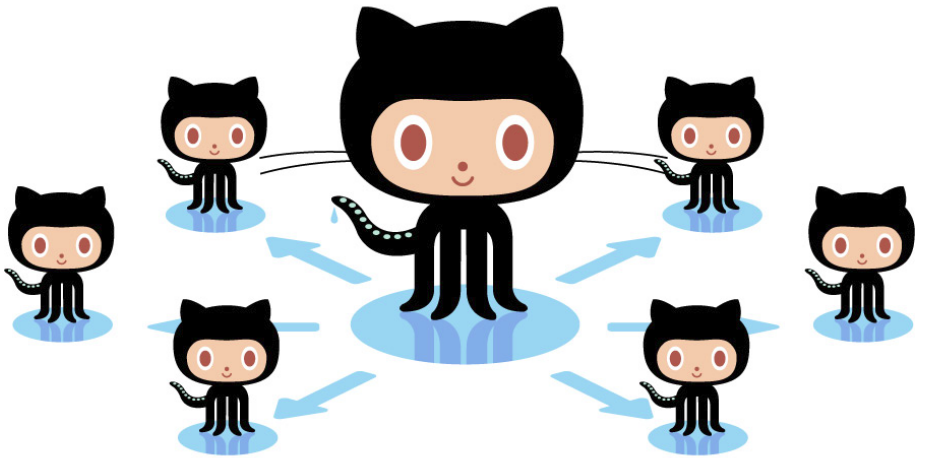
```
$ git checkout master
```

Apaga o branch nome-do-branch

```
$ git branch -D nome-do-branch
```

# - Referências


- CHACON, S.; STRAUB, B. Pro git. New York, Apress, 2014.
- GITHUB. Github, Disponível em: <http://github.com/>. Acesso em: 06 de abril, 2017.
- https://www.youtube.com/watch?v=4XpnKHJAok8&feature=youtu.be

# - Como funciona o site?

- Lugar pra aprender
  - https://guides.github.com/features/mastering-markdown/
  - https://github.com/tidyverse/ggplot2
  - https://github.com/justmarkham/scikit-learn-videos
  - https://github.com/pysal
  - https://github.com/darribas/geopandas
  - https://github.com/rstudio/shiny
  - https://github.com/rstudio/rstudio
  - https://github.com/scikit-learn/scikit-learn
  - https://github.com/marketplace