



Conceitos
básicos

GIT

Baseado no material preparado pelo Prof. Edson



organizaç ão

Históric

Contextualizaç

Criando repositórios

Operações de
versionamento

Histórico

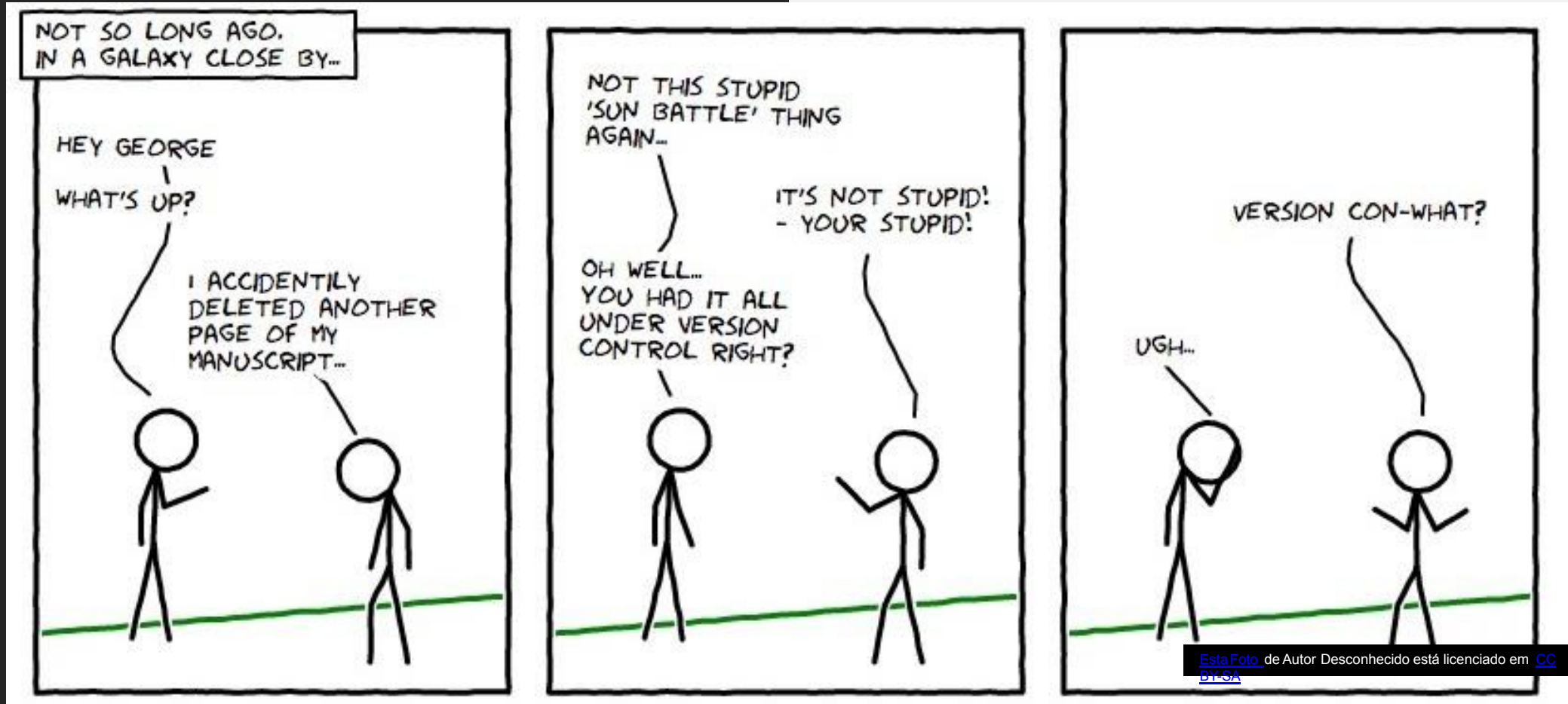
- ^O GIT – Global Information Tracker (mas não necessariamente)
 - Sistema elaborado inicialmente por Linus Torvalds
 - Projeto GIT iniciou em 2005, como solução a um “empasse”
 - Projeto Linux não usava controle de versão (1991-2002)
 - Após pressão da comunidade mantenedora, buscou-se uma solução
 - Em 2002, adotou-se a solução proposta pela bitKeeper
 - Desenvolvido como um sistema distribuído de controle de versão
 - Ia ao encontro do que Linus Torvalds buscava para garantir escalonamento do projeto
 - Era uma solução gratuita para a comunidade open-source
 - Em 2005, Bitkeeper removeu a política de “free-of-charge”
 - Criou-se o impasse e o gatilho acionador da elaboração de um sistema próprio (GIT)

Context

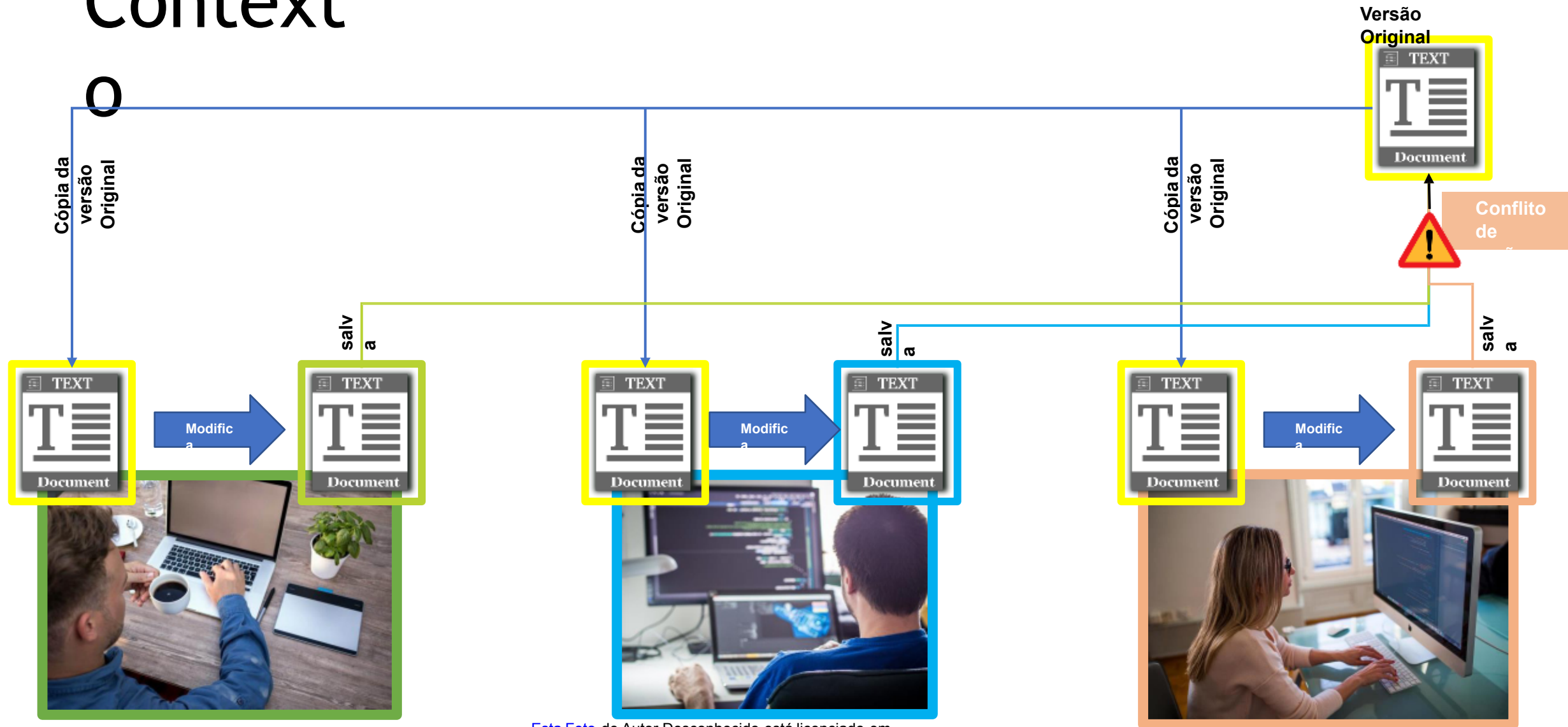
- **O** O que é o GIT
 - Sistema de controle de versões
- Porque precisamos de um sistema de controle de versões?
 - Cenário
 - Trabalhamos com arquivos texto
 - Cada modificação feita em um arquivo e salvo, é uma versão diferente
 - Várias pessoas contribuem na elaboração de um sistema
 - Potencialmente mexem no mesmo arquivo
 - Cada pessoa tem uma versão diferente em seu computador
 - Solução
 - Precisamos que algo **AJUDE** a controlar esta “bagunça”
- O que um sistema de controle de versão garante
 - **AJUDA** para capturar, criar, editar, lidar com conflitos e salvar versões de um ou mais arquivos

Context 0

—
Porque usar um
sistema de controle
de versão?



Context

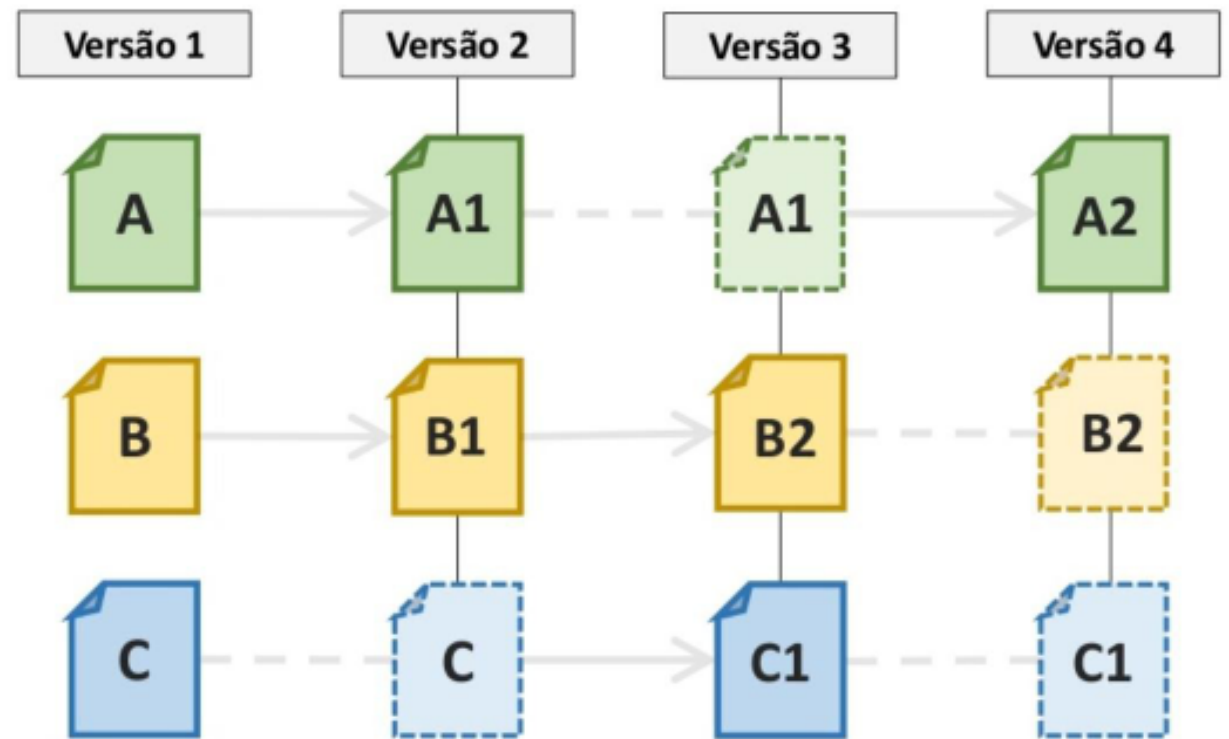


26/10/202

context

0

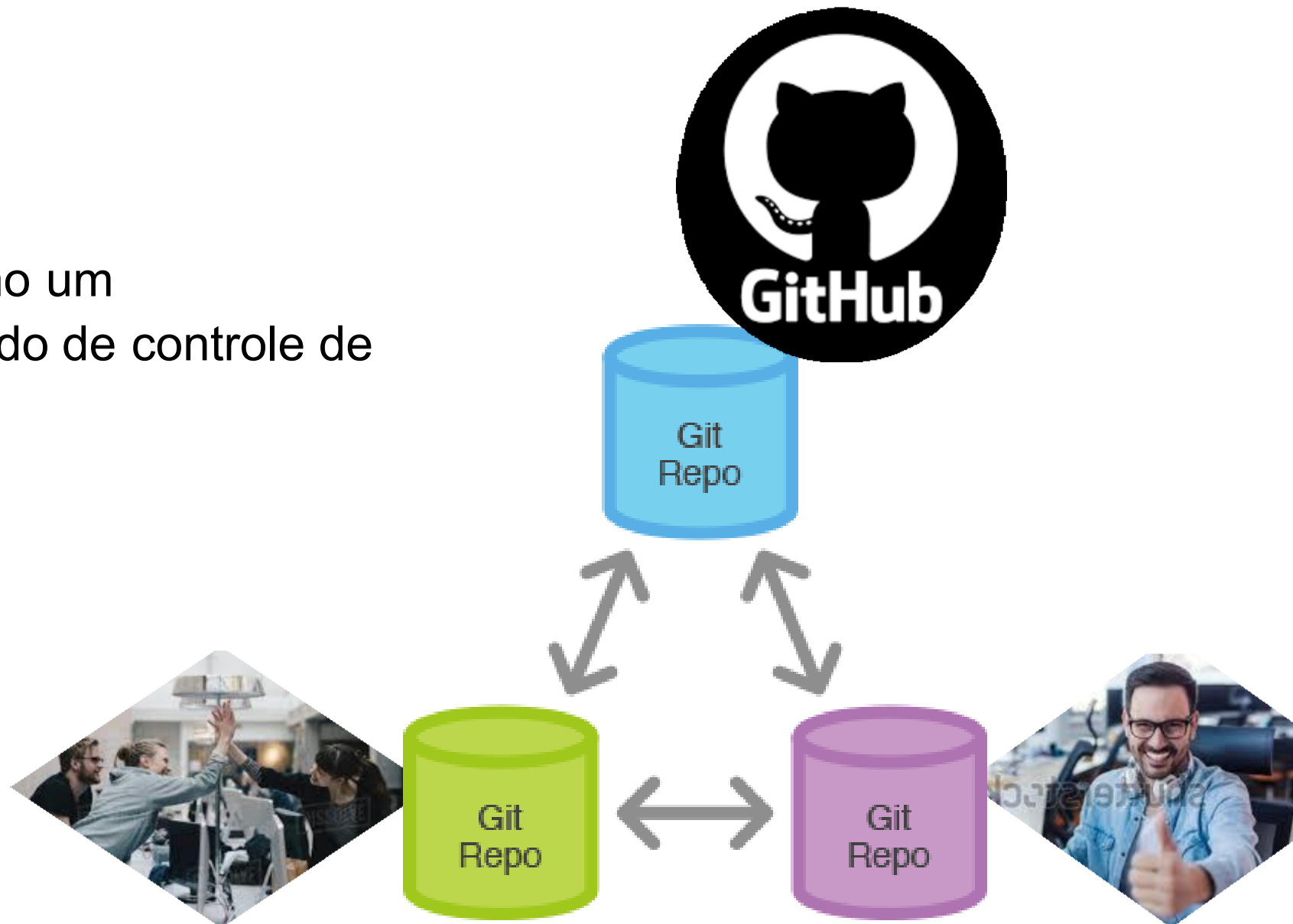
- GIT
 - Controle de versão



Context

0 • GIT

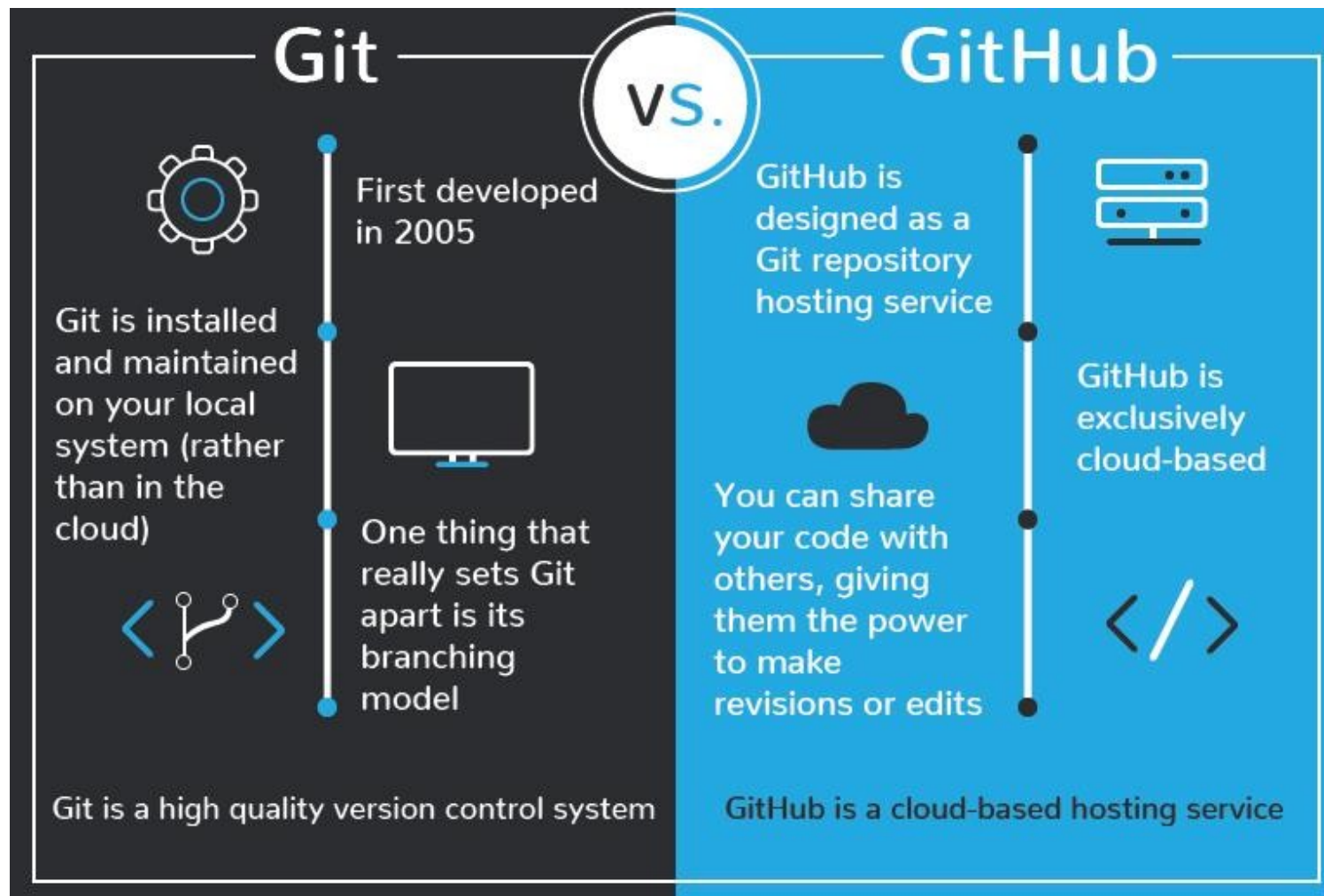
- Desenvolvido como um sistema distribuído de controle de versão



Serviços de hospedagem

GIT

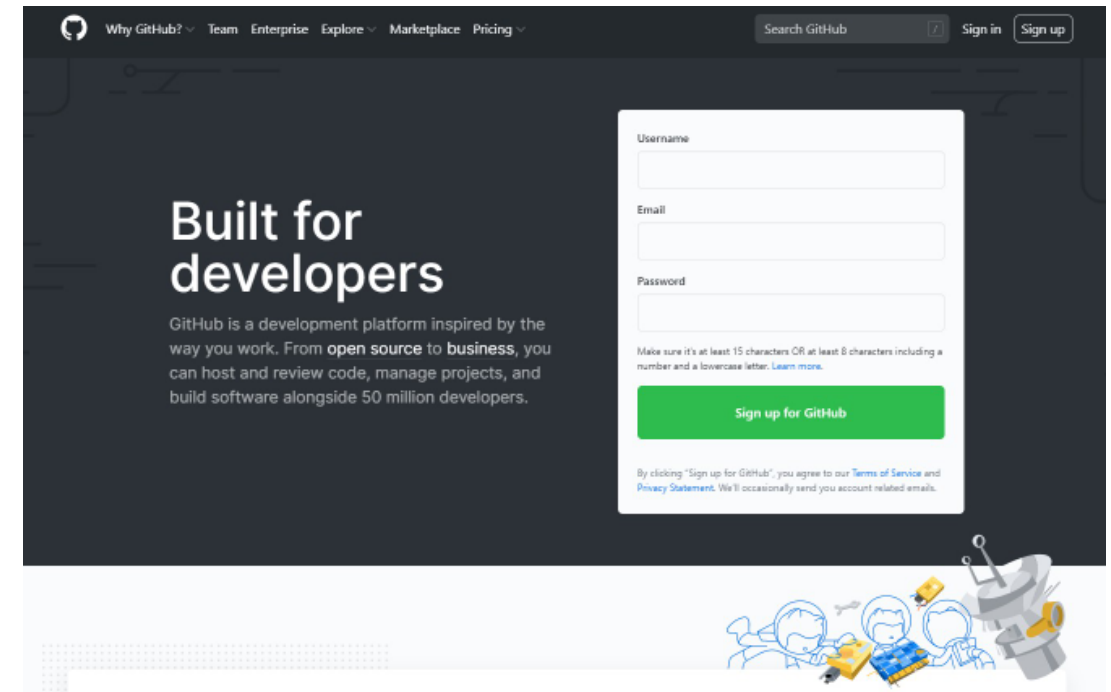
- [BerliOS](#)
- [GitHub](#) □ Serviço que usaremos nesta apresentação
- [Gitorious](#)
- [Sourceforge](#)
- [GNU Savannah](#)
- [Project Kenai](#)
- [Unfuddle](#)
- [SourceRepo](#)
- [Google Code](#)
- [Bitbucket](#)
- [GitLab](#)
- [Azure DevOps](#)



Diferencian
do os
serviços

Usando um sistema

- Criando uma conta no github
 - Acesse o site <http://github.com>
 - Informe nome, email e senha
 - Pronto... Sua conta está criada
- Instalando GIT na sua máquina
 - Execute os passos propostos no site do github
 - <https://github.com/git-guides/install-git>
 - Instalação no Windows
 - Baixar a última versão do link <https://gitforwindows.org/>
 - Siga as instruções passadas durante a execução do instalador
 - Abrir o prompt de comandos do Windows (shortcut +r e em seguida cmd)
 - Teste se a instalação foi bem sucedida
 - Digite o comando "git help"

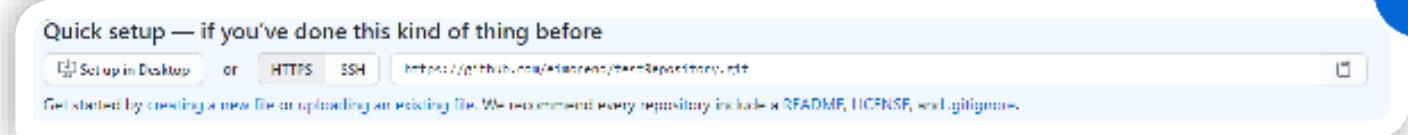
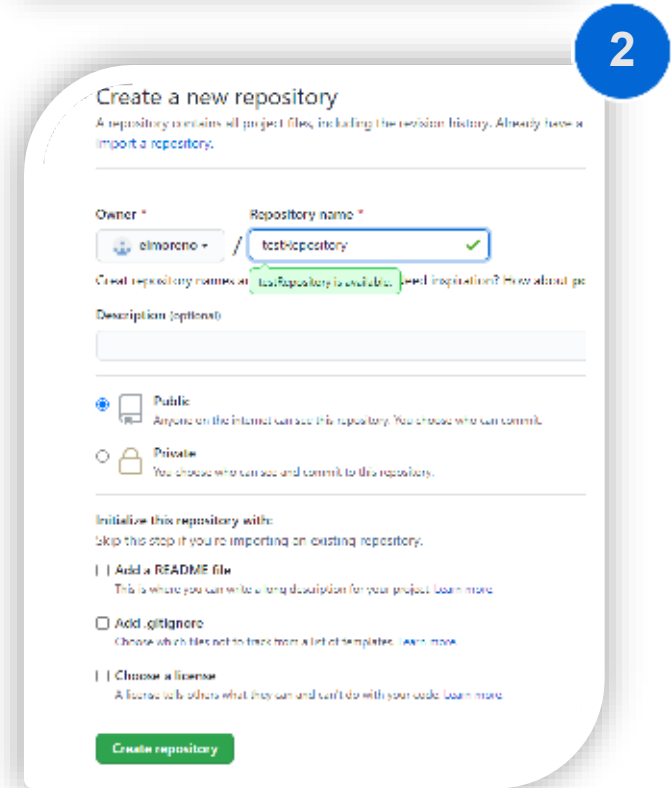
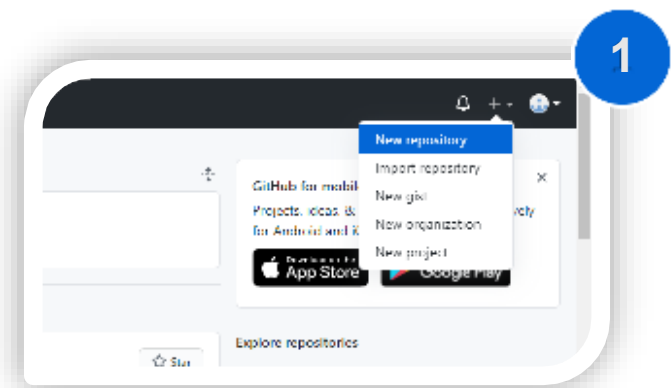


Configuração inicial

- É interessante registrar seu username, criado no github
 - Não é obrigatório
 - Se não realizado, seu user e password será pedido quando for transacionar como github
 - Será usado pelo git quando for realizado um commit
 - Definindo nome e email globalmente
 - `git config --global user.name "Bugs Bunny"`
 - `git config --global user.email bugs@gmail.com`
 - Como resultado, todos projetos terão estes user anotados, bastando apenas a senha

Criando um repositório

- Criando localmente um repositório do zero
 - `git init`
 - Comando de inicialização
 - Este cria um diretório .git na pasta corrente que será seu repositório
 - Este repositório está local a sua máquina neste instante
 - Adicione arquivos iniciais ao repositório
 - Passo a passo
 - `git add filename`
 - Comando sinaliza arquivos que farão parte do repositório
 - `git commit -m "commit message"`
 - Comando faz uma foto das mudanças, e salva o repositório localmente
 - Crie o repositório remoto no github
 - Faça o login na sua conta github
 - No canto superior direito, selecione “new repository” (1)
 - Na tela de criação de repositório, de um nome em “repository name” e clique “create repository” (2)
 - Na tela seguinte, copie o link que finaliza com .git (3)
 - Salve o repositório local no github
 - `git remote add origin <link com final .git>`
 - `git branch -M master`
 - `git push -u origin master`



Criando um repositório

```
C:\WINDOWS\system32\cmd.exe - git push -u origin master

c:\Recursos\github\primeiroRepositorio>git init
Initialized empty Git repository in C:/Recursos/github/primeiroRepositorio/.git/

c:\Recursos\github\primeiroRepositorio>git add readme.txt

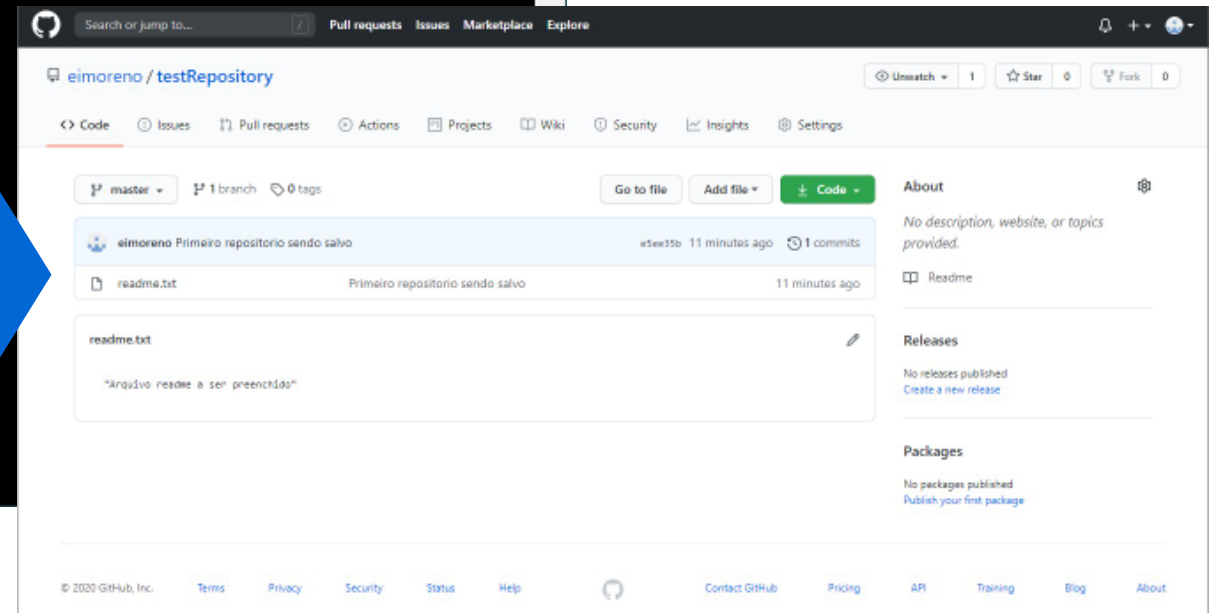
c:\Recursos\github\primeiroRepositorio>git commit -m "Primeiro repositorio sendo salvo"
[master (root-commit) e5ee35b] Primeiro repositorio sendo salvo
1 file changed, 1 insertion(+)
create mode 100644 readme.txt

c:\Recursos\github\primeiroRepositorio>git remote add origin https://github.com/eimoreno/testRepository.git

c:\Recursos\github\primeiroRepositorio>git branch -M master

c:\Recursos\github\primeiroRepositorio>git push -u origin master
Username for 'https://github.com': eimoreno
Password for 'https://eimoreno@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 260 bytes | 260.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/eimoreno/testRepository.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

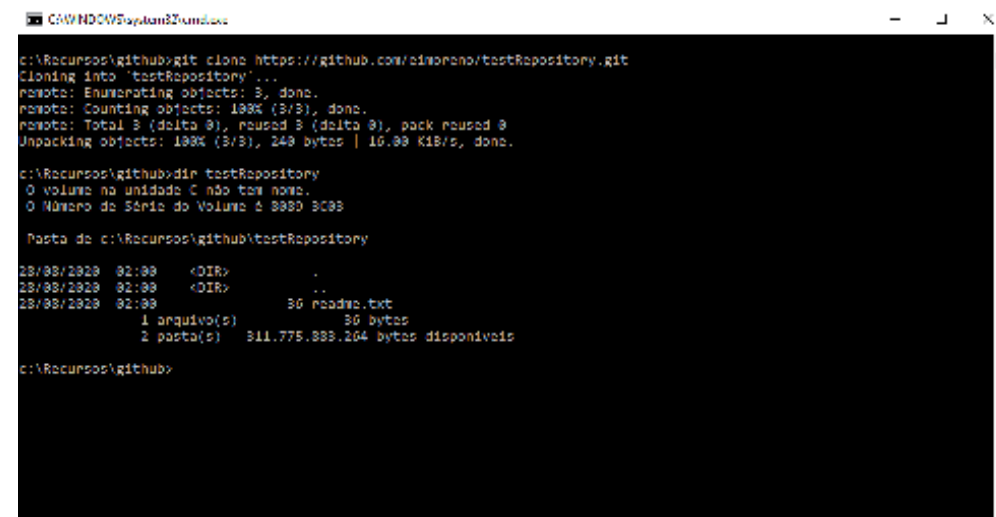
c:\Recursos\github\primeiroRepositorio>
```



Clonando um

repositório

- Considerando a pré existência de um projeto no github
 - A criação de um repositório local fica mais fácil
 - Navegar até o diretório onde deseja depositar o repositório
 - git clone <link com final .git>



```
C:\WINDOWS\system32\cmd.exe

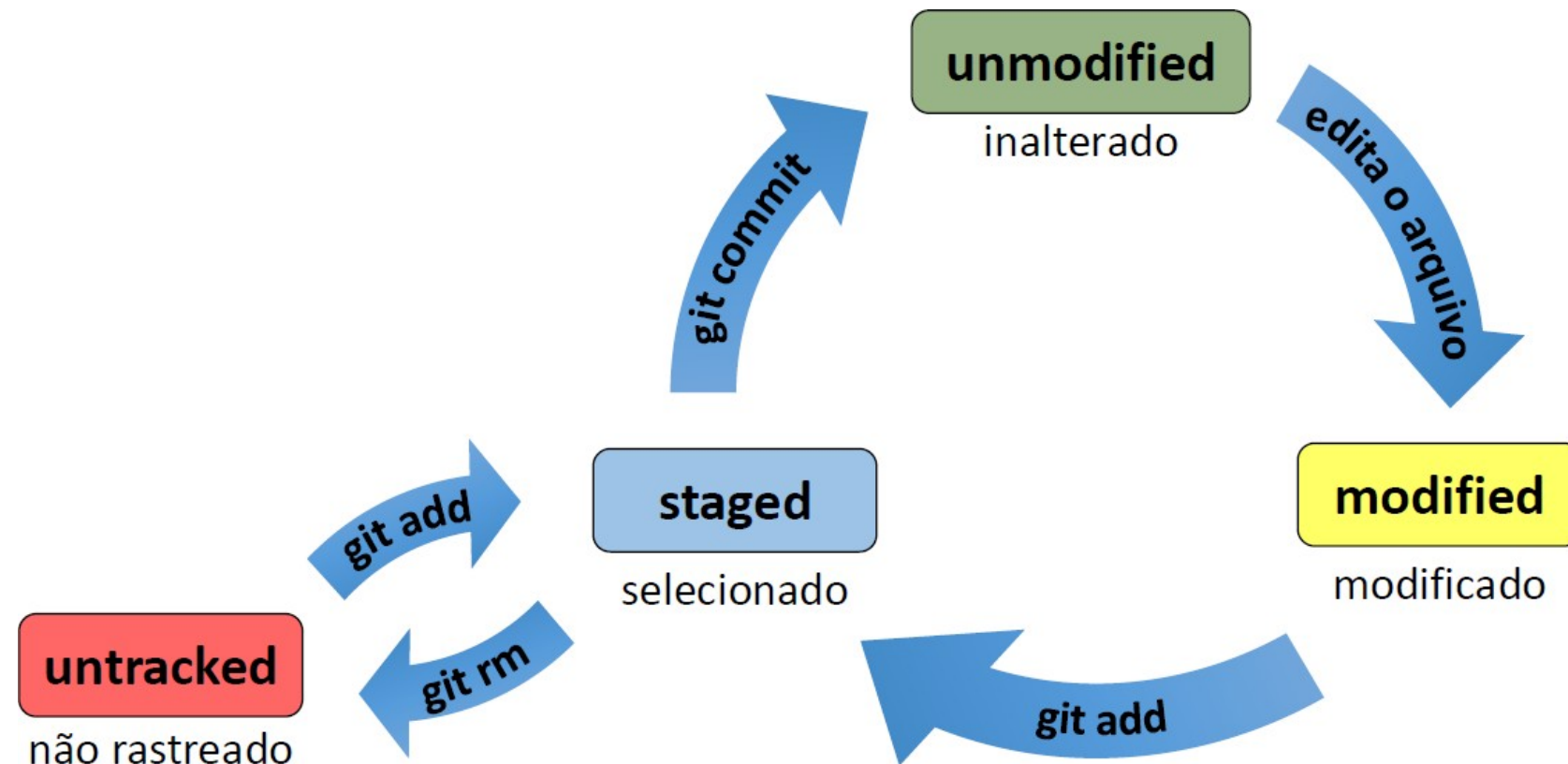
c:\Recursos\github>git clone https://github.com/simorano/testRepository.git
Cloning into 'testRepository' ...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack reused 0
Unpacking objects: 100% (3/3), 240 bytes | 10.00 KiB/s, done.

c:\Recursos\github>dir testRepository
O volume na unidade C não tem nome.
O Número de Série do Volume é 8880 3C98

Pasta de c:\Recursos\github\testRepository

28/08/2020  02:00    <DIR>          .
28/08/2020  02:00    <DIR>          ..
28/08/2020  02:00                36 readme.txt
               1 arquivo(s)      36 bytes
               2 pasta(s)  311.775.888.104 bytes disponíveis

c:\Recursos\github>
```



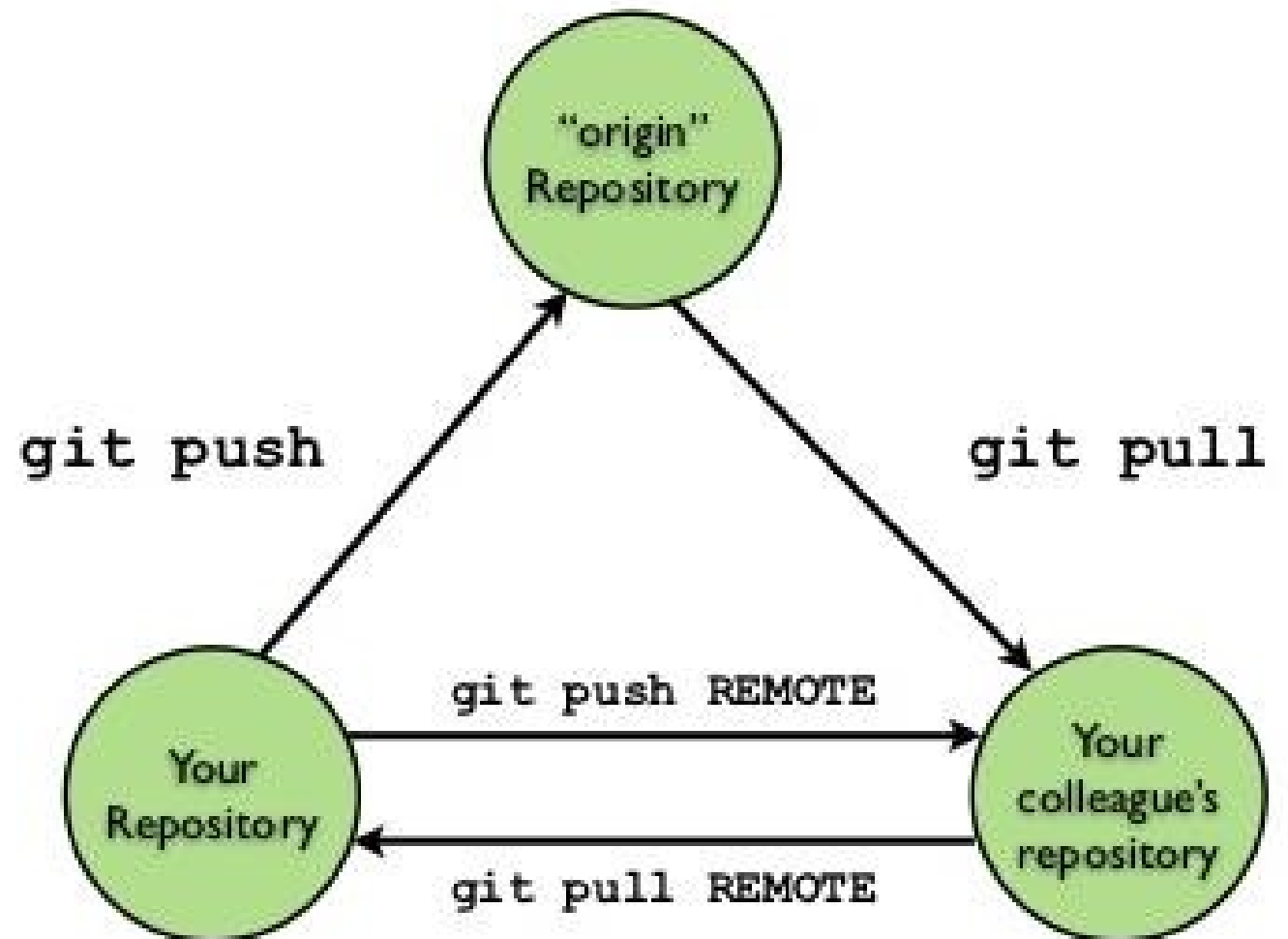
No ambiente versionado local

- Estado dos arquivos
- Arquivos podem estar em diferentes estados
- Para saber o estado
 - `git status`

Desenvolvimento colaborativo

Atualização dos projetos pode ser feito a partir de qualquer repositório participante do projeto

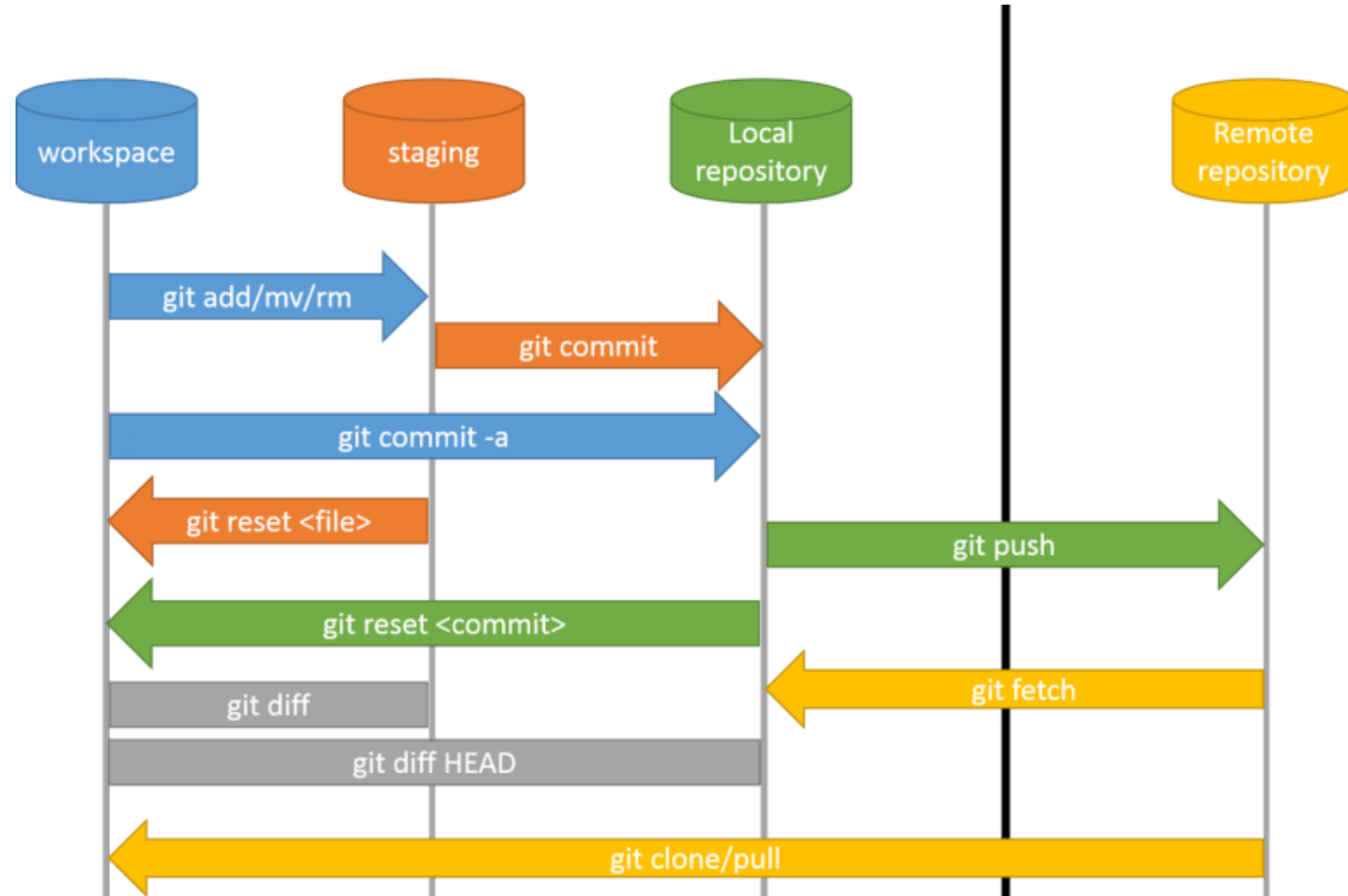
10/26/2021

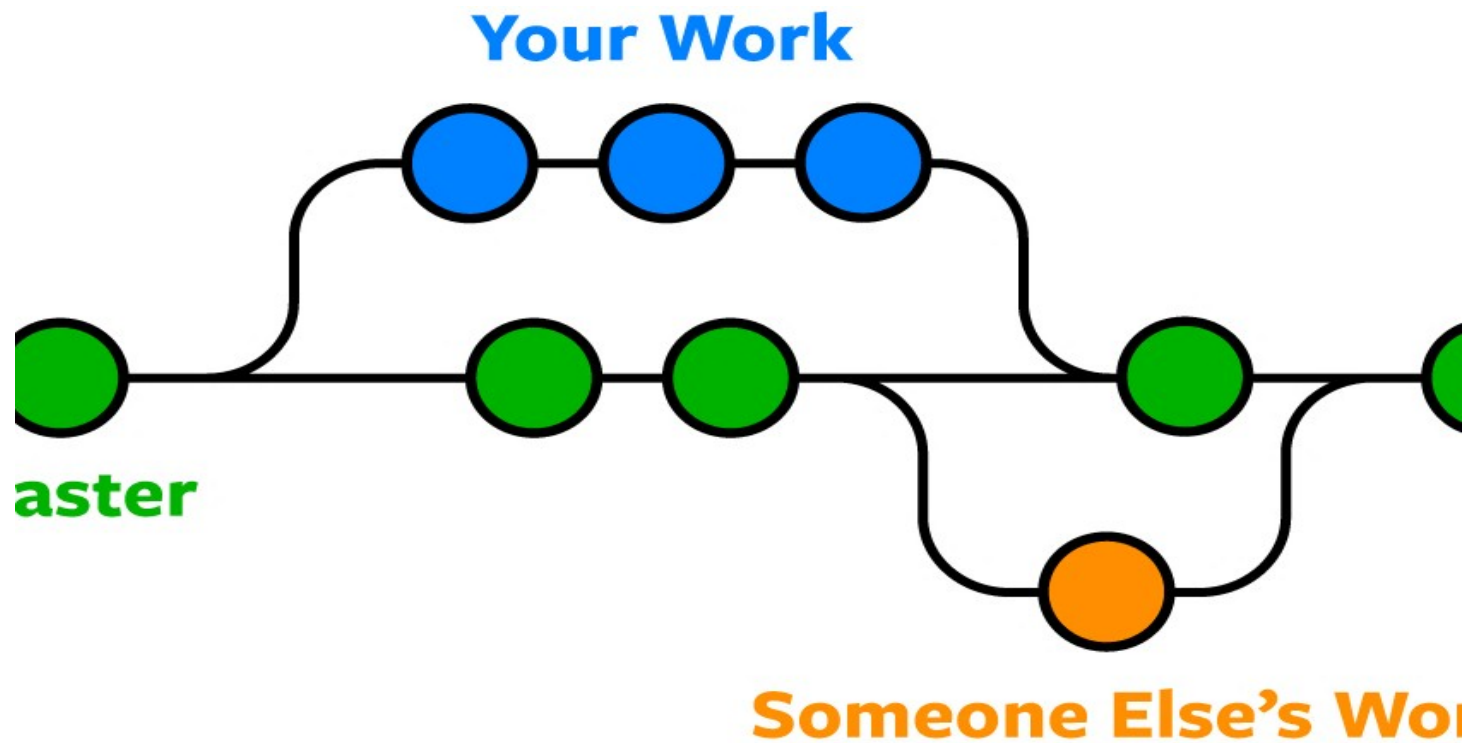


No ambiente versionado

geral

- Comandos básicos no git e suas aplicações



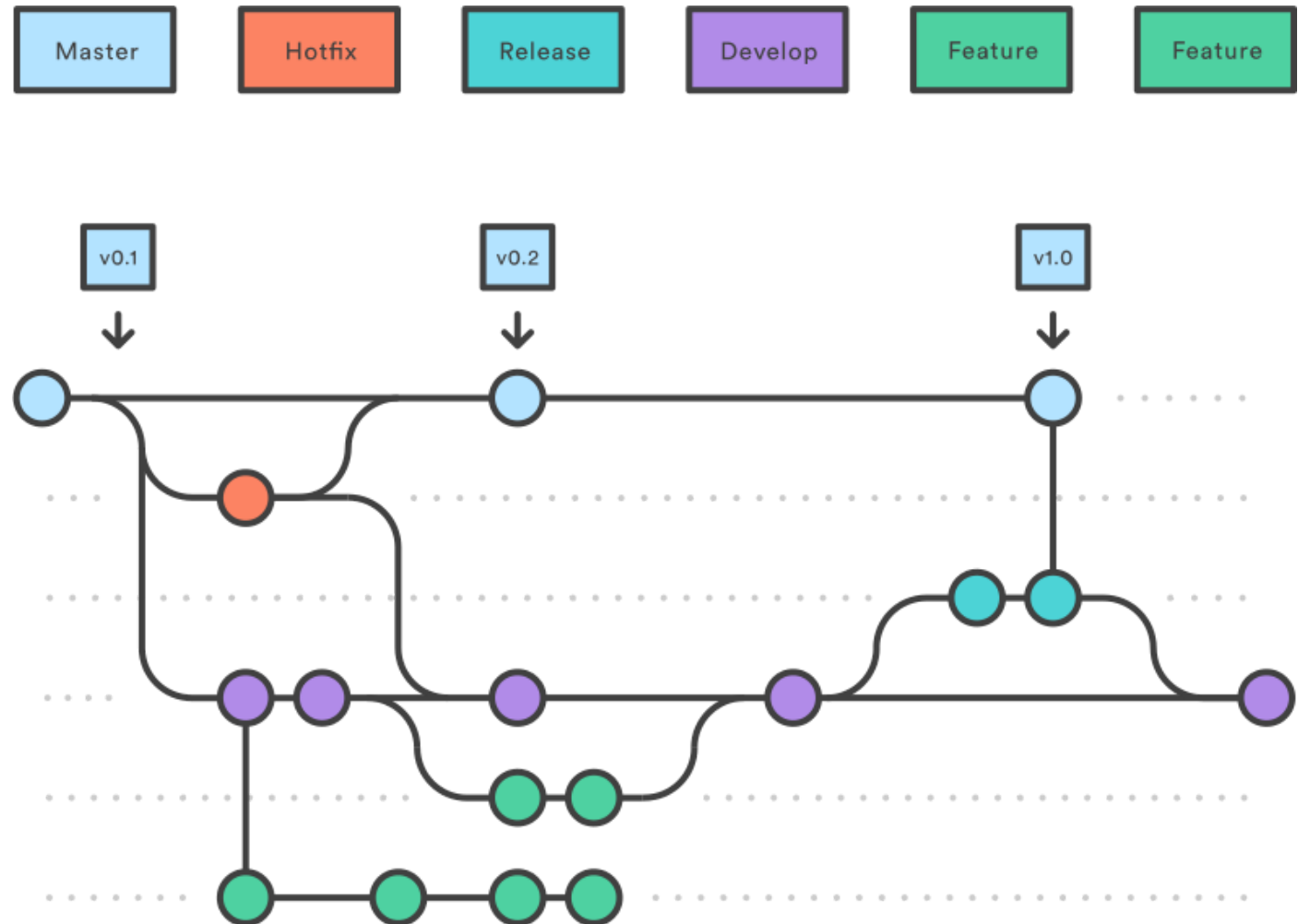


Proposta de fluxo de trabalho

- Proposta de criação básica de trabalho para garantia de boa trilha de desenvolvimento

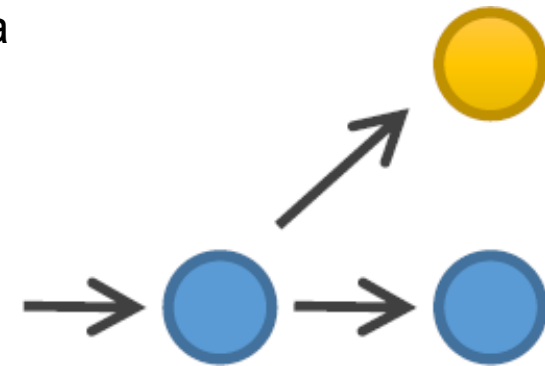
Proposta de fluxo de trabalho

- Proposta de criação mais completa de branches para garantia de boa trilha de desenvolvimento



Criando uma branch

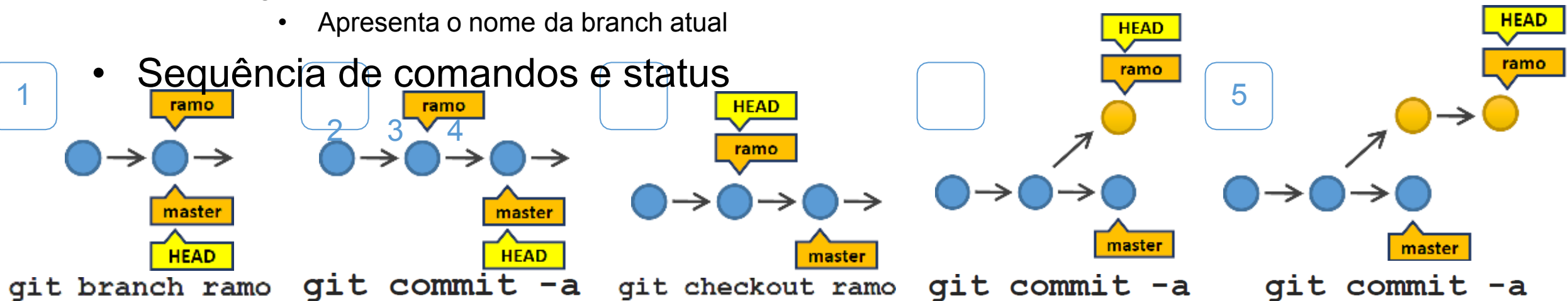
- Cria uma foto da branch em que se está no momento
 - Para saber quais branches existem
 - `git branch`
 - Branch que se está atualmente estará destacada na lista
 - Para criar a branch
 - Alternativas
 - `git branch nomeDaBranch`
 - Cria branch cópia da atual
 - Mantém desenvolvedor na branch atual
 - `git checkout -b nomeDaBranch`
 - Cria branch cópia da atual
 - Desloca o desenvolvedor para a branch criada



Navegando entre branches

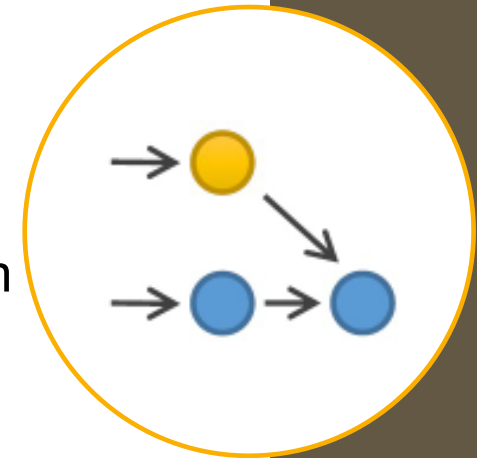
- Comando previamente utilizado
 - `git checkout nomeDaBranch`
 - Deslocada da branch atual para a branch nome da Branch
 - Para saber em qual branch se encontra
 - `git branch`
 - Observar o nome em destaque
 - `git branch --show-current`
 - Apresenta o nome da branch atual

Sequência de comandos e status



Mesclando branches

- Para a mescla de duas branches
 - É primordial que haja relação entre elas
 - Podem ocorrer conflitos que terão de ser resolvidos
- Comandos
 - Deslocar para a branch que receberá a mescla
 - e.g. deslocar para a branch master, que será mesclada com branch nomeDaBranch
 - `git checkout master`
 - Comandar a mescla
 - `git merge nomeDaBranch`
 - Caso o merge não tenha gerado conflito ou os conflitos tenham sido resolvidos,
 - esta pode ser eliminada
 - `git branch -d nomeDaBranch`
- Relação entre as branches
 - `git show-branch`



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 commits

```
$ 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

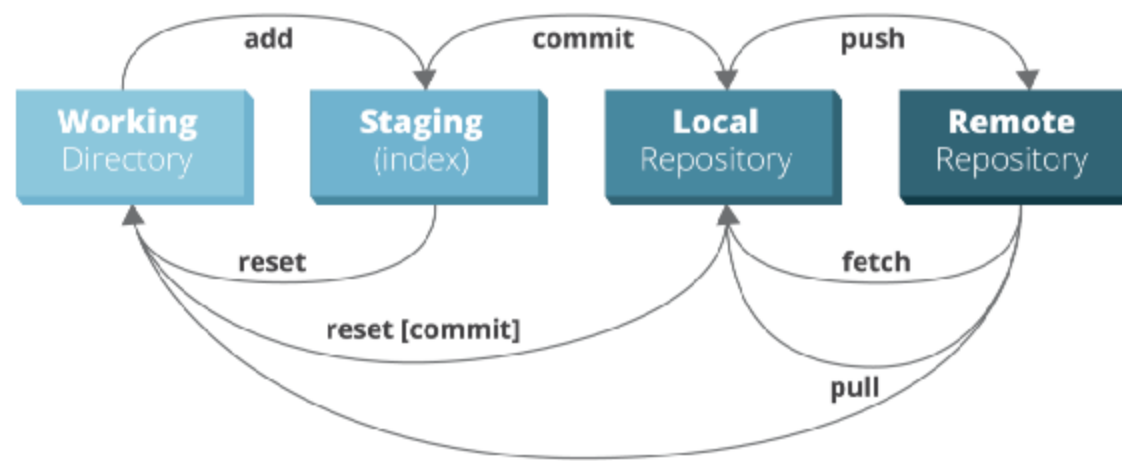
```
$ git push
```

Finally!

When in doubt, use git help

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

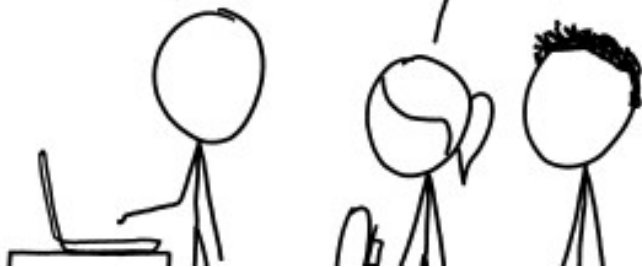
Or visit <https://training.github.com/> for official Git-Hub training.



THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL
COMMANDS AND TYPE THEM TO SYNC UP.
IF YOU GET ERRORS, SAVE YOUR WORK
ELSEWHERE, DELETE THE PROJECT,
AND DOWNLOAD A FRESH COPY.



Em caso de dúvidas