Parte 2

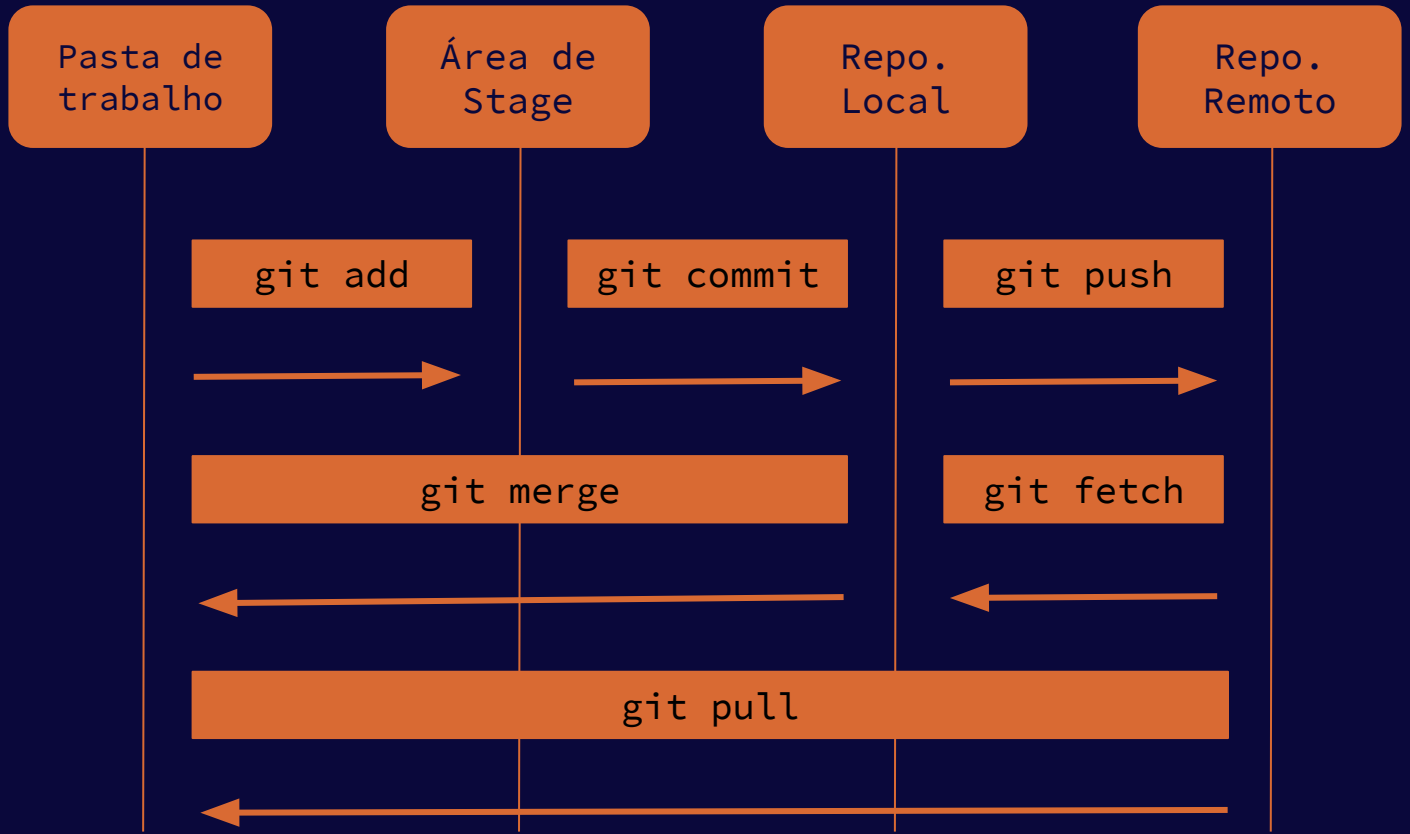
Luiz Carlos C. Filho, Wagner A.A. Cotta

Universidade Federal do Espírito Santo (UFES)

***Campus* Vitória**

Nos capítulos anteriores...

- Controle de versão;
  - conceito;
- Sistema de controle de versão (git);
  - Código fonte (repositório);
  - Estados do código (commits);
- Hospedagem de repositórios (github);
  - Repositórios remotos;
  - fluxo de trabalho usando git e github;
  - Comandos básicos (pull, add, commit, push).

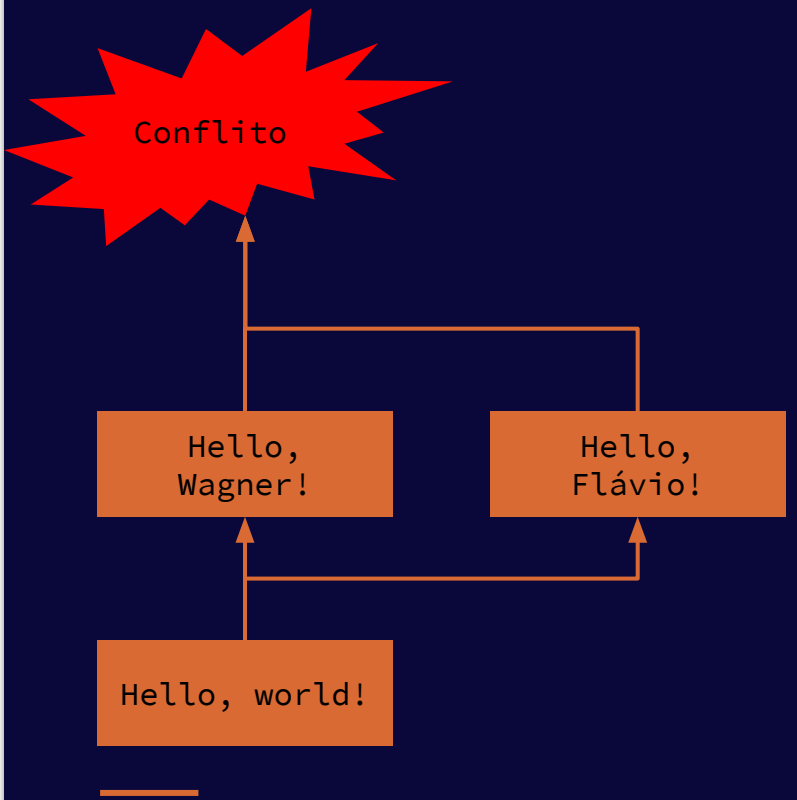

Pasta de trabalho
Área de Stage
Repo. Local
Repo. Remoto
git add
git commit
git push
git merge
git fetch
git pull

Capítulo de hoje

- Histórico e conflitos
  - visualizar histórico
  - resolvendo conflitos unindo históricos de desenvolvimento
- Trabalhando em equipe
  - ramificações (branches)
  - unindo branches (merge)
  - unindo branches no github (pull request)
  - resolvendo conflitos
- Versões
  - criando versões com tags

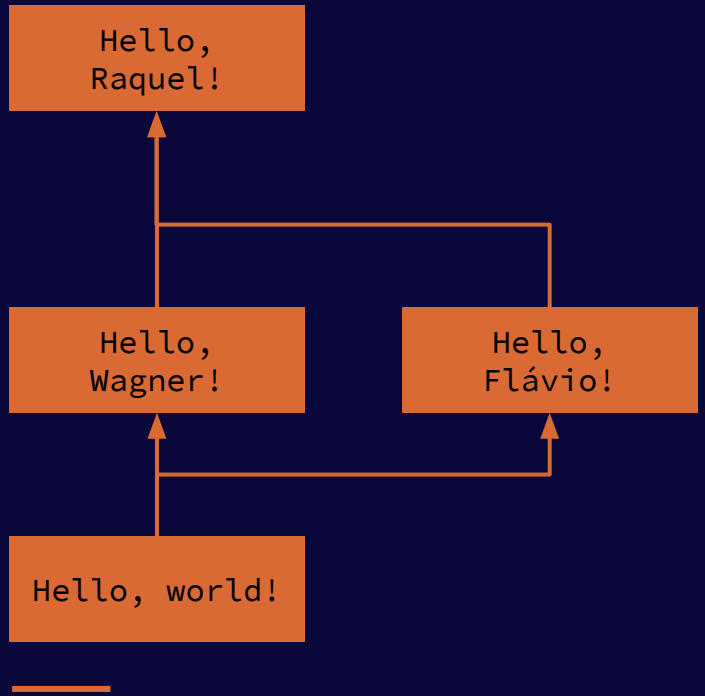
Conflitos

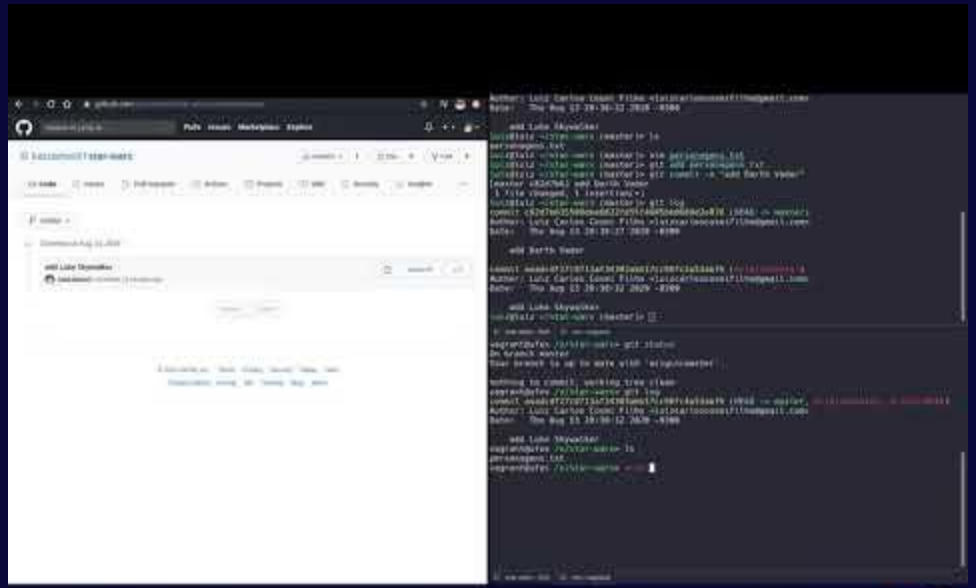
- Conflitos ocorrem por históricos de desenvolvimento diferentes onde as mudanças são feitas na mesma linha do arquivo.



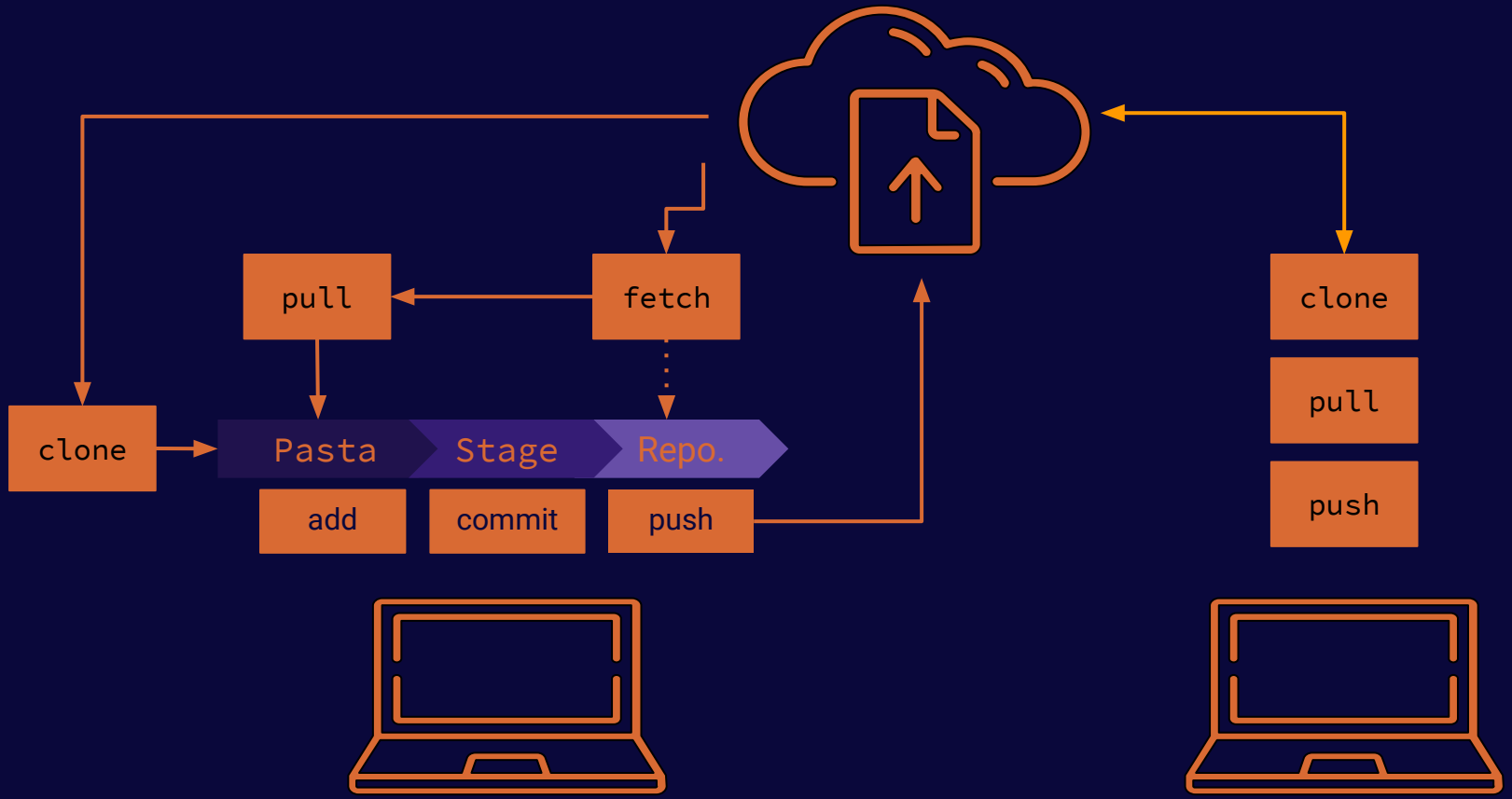
Resolvendo conflitos

- quando acontecem, o git irá indicar onde ocorreram;
- é necessário criar um commit para resolver os conflitos e decidir o que irá ficar;





Tutorial Git e Github - Visualizando Histórico e Resolvendo Conflitos

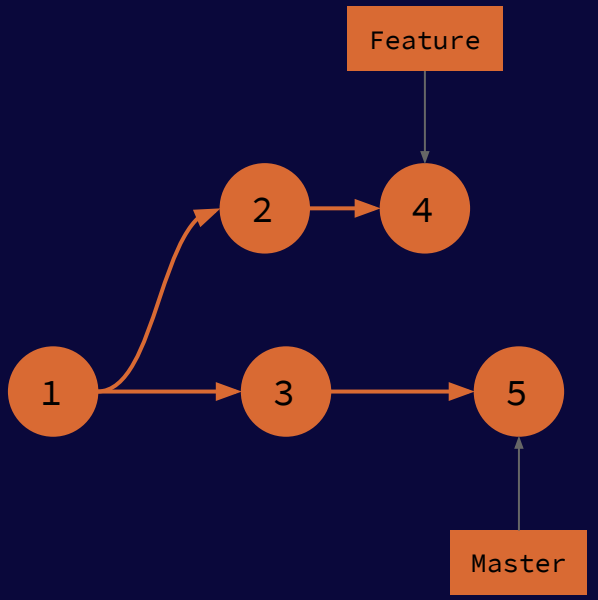

pull
fetch
clone
Pasta
Stage
Repo.
add
commit
push
clone
pull
push

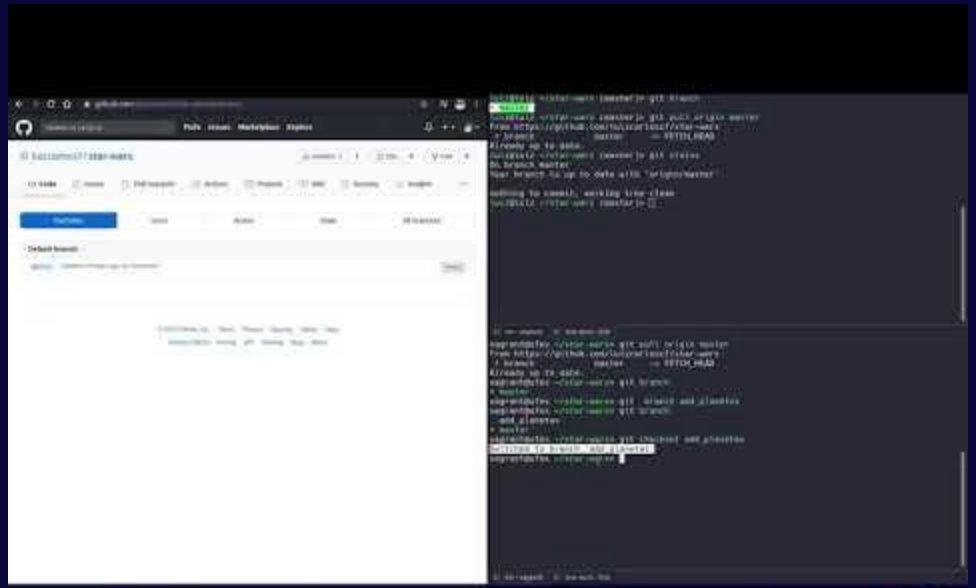
Trabalhando em equipe

- Muitos conflitos;
- Necessidade de algum recurso que possibilite uma estabilidade;
  - resolver um bug;
  - nova funcionalidade;
  - melhorar algum código.

Branches

- O histórico de desenvolvimento é uma lista de commits;
- A lista principal de desenvolvimento é chamada de “master”;
- Podem existir múltiplas linhas desenvolvimento.

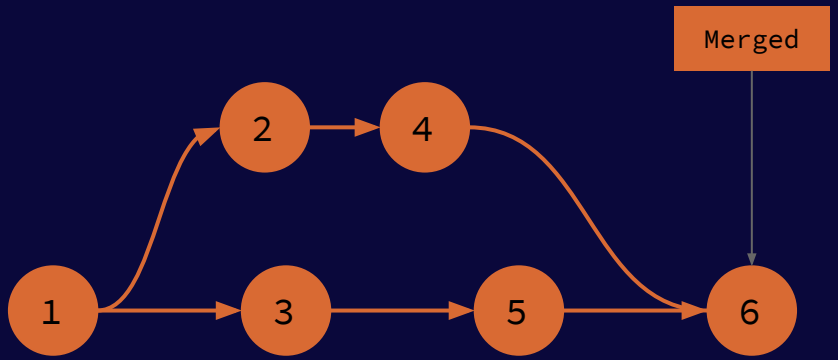


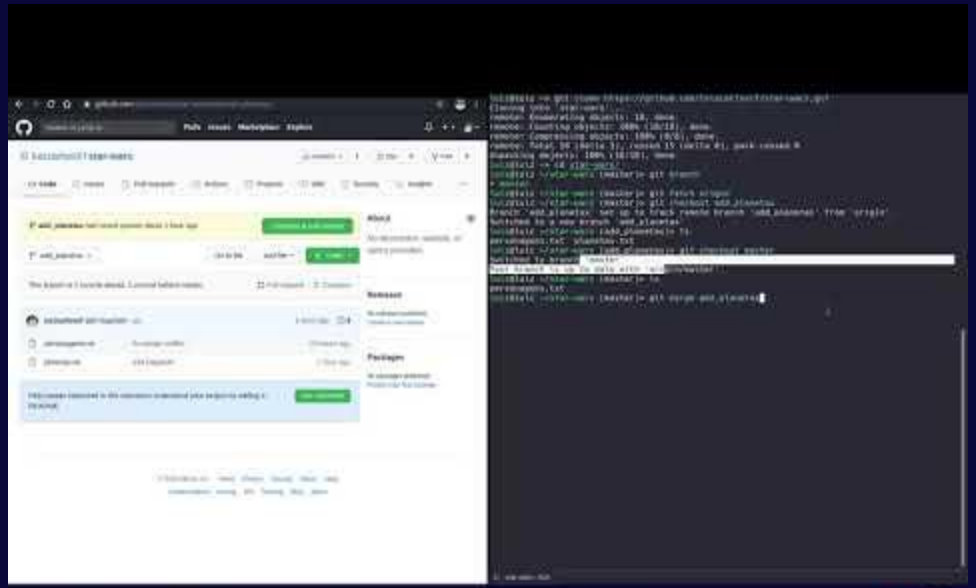


Tutorial Git e Github - Trabalhando com branches diferentes

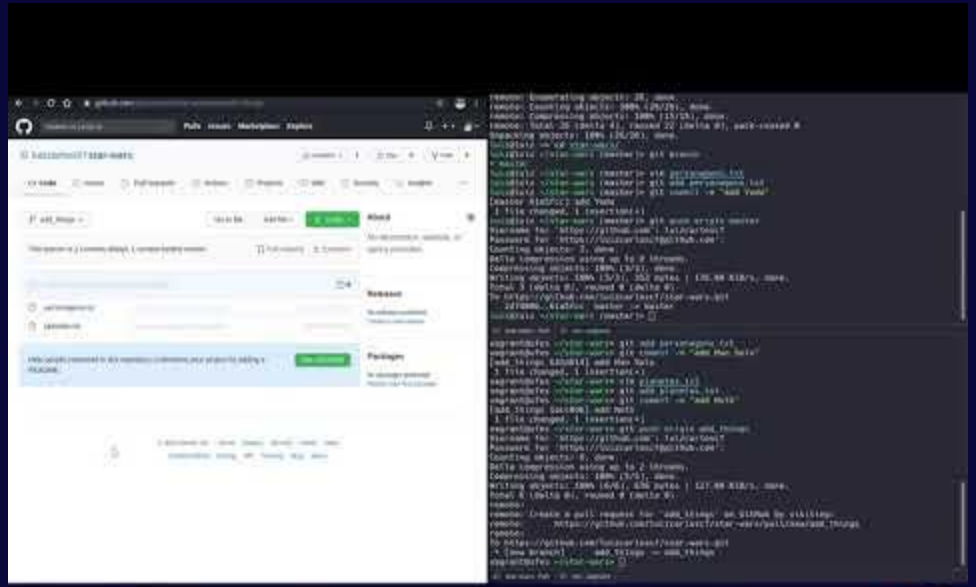
Merge

- O procedimento de unir branches é chamado de “merge”;
- Se caso em 4 e 5, as mesmas linhas forem editadas conflitos irão ocorrer! Será necessário resolver antes de fazer o merge.

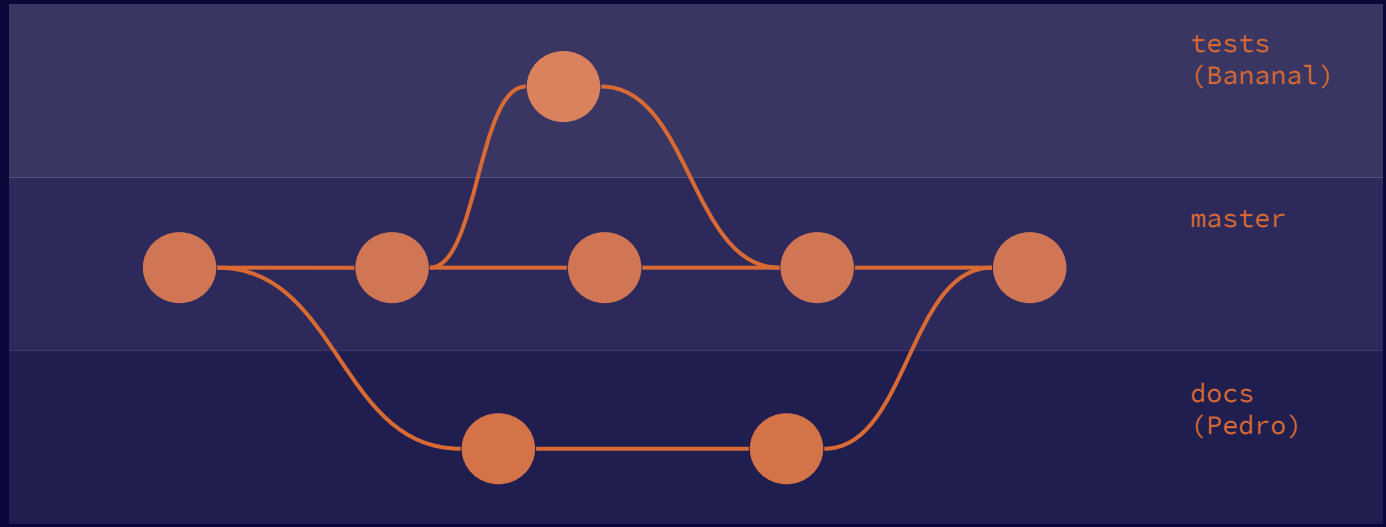




Tutorial Git e Github - Unindo branches com merge

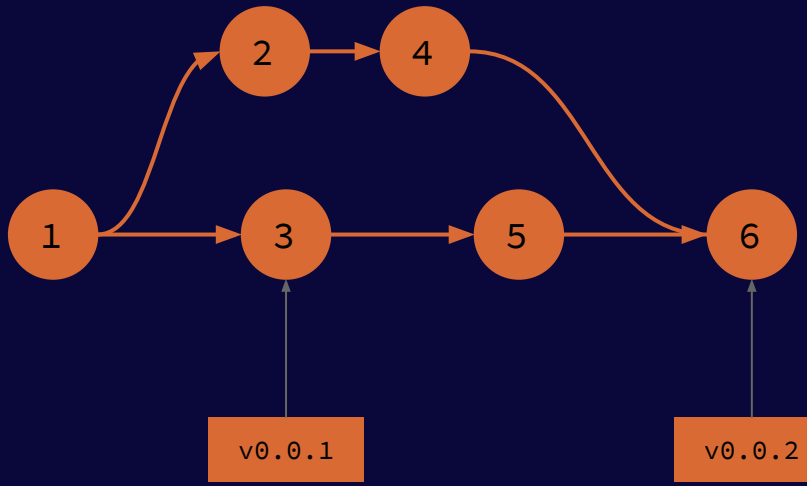


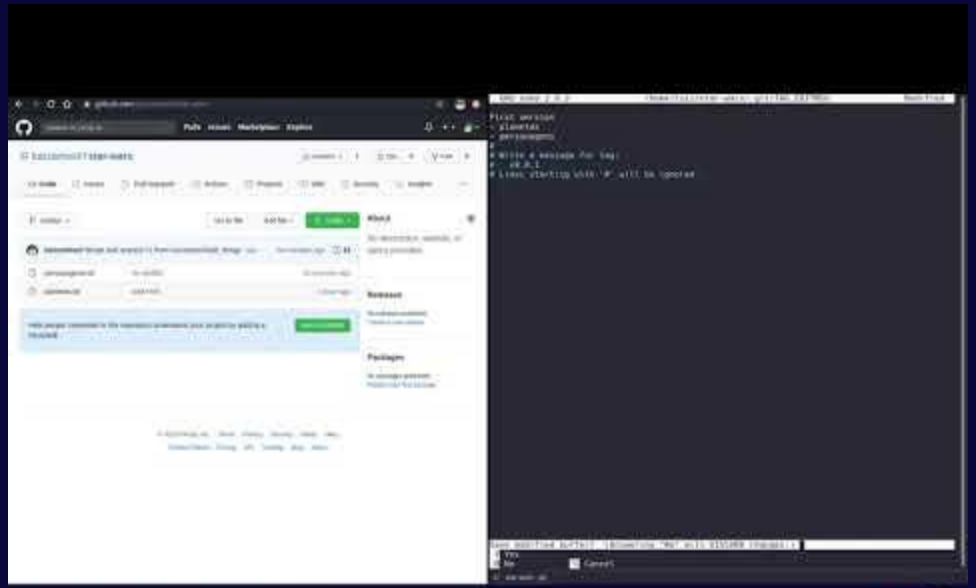
Tutorial Git e Github - Unindo branches com pull request


tests
(Bananal)
master
docs
(Pedro)

Tags

- tags são usadas para gerenciar versões em um repositório.
- criação de versões estáveis.





Tutorial Git e Github - Criando versões

Resumo de hoje

- git log
  - lista commits no repositório local
- git branch
  - lista, cria ou delete branches
- git checkout
  - troca de branches
- git merge
  - junta dois ou mais históricos de desenvolvimento
- git tag
  - gerenciar versões em um repositório

Para os curiosos

- git ignore
  - documentation
  - collection of .gitignore templates
- documentação
  - README.md guide
- projeto da apresentação
  - github.com/luizcarloscf/star-wars

GIT CHEATSHEET

```
# configuration
git config --global user.name “Your name”
git config --global user.email “you@gmail.com”
```

```
# starting a project
git init [project name]
git clone [project url]
```

```
# remove file from directory
git rm [file]
```

```
# status of working directory
git status
```

```
# add a file to the staging area
git add [file]
```

```
# discard changes in working directory
git checkout -- [file]
```

```
# commit to local
git commit
```

```
# revert your repository
git reset [file]
```

```
# list all local branches
git branch [-a]
```

```
# fetch changes from the remote and merge current branch with its upstream
git pull [remote]
```

```
# join specified [from name] branch
git merge [from name]
```

```
# create a new branch
git branch [branch_name]
```

```
# push local changes to the remote
git push [--tags] [remote]
```

```
# remove selected branch
git branch -d [name]
```

```
# fetch changes from the remote
git fetch [remote]
```

```
# switch current branch to specificied branch
git checkout [-d] [branch_name]
```

FIM!

DÚVIDAS, SUGESTÕES E CRÍTICAS?