



Ganhando superpoderes com Git e GitHub



Um pouco sobre nós...

Giorgio Braz



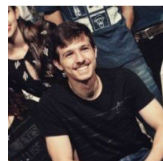
BCC

SARndbox

Cafeína Compilada

giorgiobraz.github.io

Lucas Ribeiro



BCC

Cafeína Compilada

IoT

haken

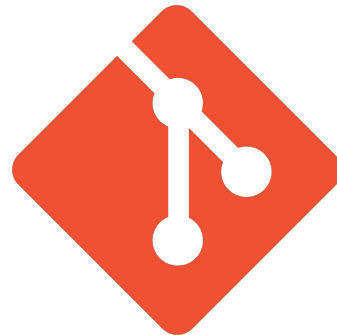
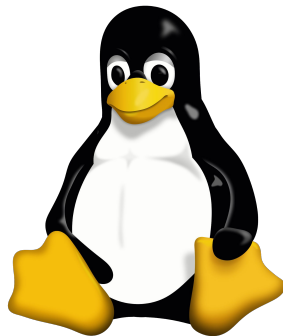
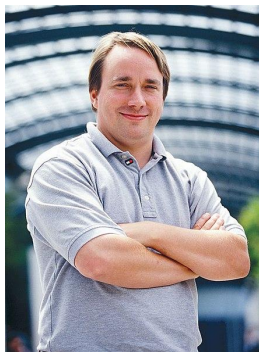


O que veremos

- O que é Git
- Por que usar
- Configuração inicial
- Controle de versão
- Visualização de alterações
- Desfazendo algo
- Tags e branches
- Introdução ao GitHub
- Colaborando com um projeto



O que é Git, afinal?



Por que usar?

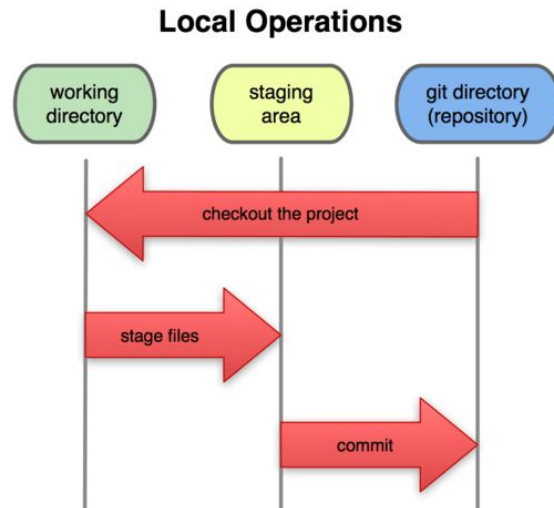
- Máquina do tempo
- Colaboração
- Velocidade
- Design simples
- Suporte robusto a desenvolvimento não linear
 - milhares de branches paralelos
- Totalmente distribuído
- Capaz de lidar eficientemente com grandes projetos como o kernel do Linux
 - velocidade e volume de dados



Git - Os três estados

Resumo do fluxo básico

1. Você modifica arquivos no computador
2. Adiciona esses arquivos na “staging area”
3. Faz o commit (armazenamento permanente)





Primeiros passos

Instalando o Git numa distribuição GNU/Linux

```
$ yum install git-core (ex: Fedora...)
```

```
$ apt-get install git (ex: Debian, Ubuntu, Mint...)
```

Instalando o Git no Windows

<http://msysgit.github.com>



Configuração Inicial

Sua identidade

```
$ git config --global user.name "John Doe"  
$ git config --global user.email johndoe@example.com
```

Verificando suas configurações

```
git config --list
```

Obtendo ajuda

```
$ git help config
```




O fluxo básico

- Criando um novo repositório: `$ git init`
- Visualizando o status: `$ git status`
- Movendo para a staged area: `$ add (.txt, .c, .py...)`
- Fazendo uma mudança permanente: `$ git commit -m "first commit"`



Ignorando arquivos

Segue um outro exemplo de arquivo `.gitignore`:

```
# um comentário - isto é ignorado
# sem arquivos terminados em .a
*.a
# mas rastreie lib.a, mesmo que você tenha ignorado arquivos terminados em .a acima
!lib.a
# apenas ignore o arquivo TODO na raiz, não o subdiretório TODO
/TODO
# ignore todos os arquivos no diretório build/
build/
# ignore doc/notes.txt mas, não ignore doc/server/arch.txt
doc/*.txt
```



Visualizando alterações

O `git status` pode ser um tanto útil, mas também é superficial

O que você alterou, mas ainda não selecionou (stage)? E o que você selecionou, que está para ser comitado?

- `$ git diff`
- `$ git diff --staged`
- `$ git commit`
- `$ git log` (seu histórico de commits) ou `$ git log --pretty=oneline` (apenas código e mensagem)
 - parâmetro `-p` (ordena cronológica decrescente + diff (o que foi feito em cada arquivo))
 - `-1` para limitar os resultados que serão exibidos

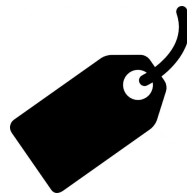


Editando e desfazendo coisas

- Fez um commit antes da hora? Edite-o!
 - `$ git add .`
 - `$ git commit --amend -m "Novas funcionalidades (edição)"`
- Remover arquivo da staged area
 - `$ git reset HEAD file_name`
- Revertendo mudanças do working directory
 - `$ git checkout -- file_name`
- Para remover: `$ git rm`
- Para renomear: `$ git mv`

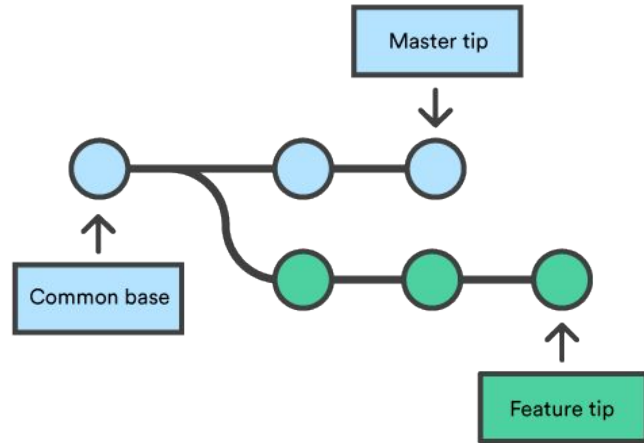


Organize seu código com TAGs

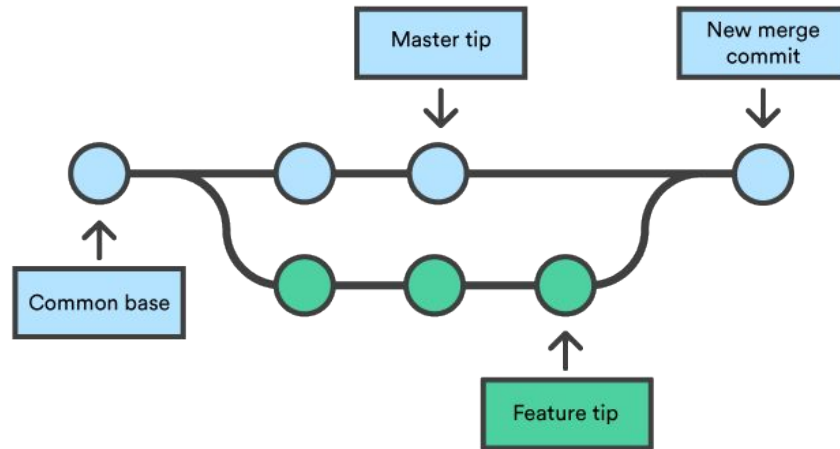


- Criar uma tag (no commit atual) `$ git tag -a v1.0 -m "Versão 1.0"`
- Criar uma tag (num commit antigo) `$ git tag -a v0.0 CHAVE -m "Versão 0.0"`
- Para listar as tags `$ git tag`
- Ver detalhes da tag `$ git show v0.0`
- Reverter a versão `$ git checkout v0.0`
- Voltando para a versão atual `$ git checkout master`
- Deletar uma tag `$ git tag -d nome_da_tag`

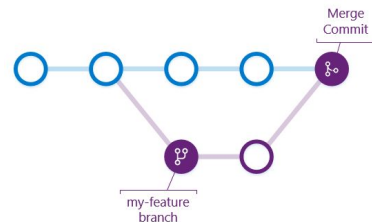
Chegou a vez dos branches



Chegou a vez dos branches



Como trabalhar com branches?



Criando testes ou novas versões utilizando um branch

- Criar um novo branch: `$ git branch teste`
- Criar e trocar de uma só vez: `$ git checkout -b teste`

Criar um novo branch `git branch teste`

- Movendo o working directory para o ambiente de teste: `$ git checkout teste`
- Para retornar: `$ git checkout master`
- Transferindo as alterações do **teste** para o **master**: `$ git merge teste`
- Deletar um branch: `$ git branch -d teste`
- Listar os branches num repositório: `$ git branch`



Microsoft



GitHub

Criando uma conta no GitHub

- <https://github.com>
- Crie seu repositório

Your Profile/Repositories/New

Público ou Privado

Com ou sem README.md



Get the best developer tools with the
GitHub Student Developer Pack



Criando um repositório por linha de comando

- Criar e adicionar texto em markdown: `$ echo "# Novo-Projeto" >> README.md`
- Criar o repositório: `$ git init`
- Adicionar o README: `$ git add README.md`
- Consolidar as alterações: `$ git commit -m "first commit"`
- Vincular ao GitHub: `$ git remote add origin https://github.com/...`
- Salvar o repositório no GitHub: `$ git push -u origin master`



Empurrar um repositório existente (linha de comando)

- `$ git remote add origin https://github.com/...`
- `$ git push -r origin master`

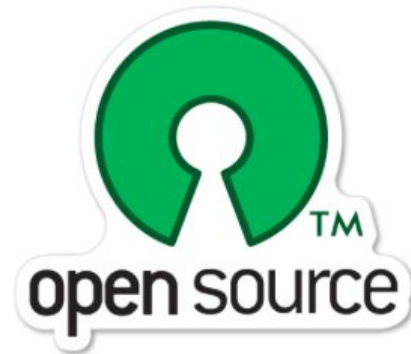


Trabalhando num time

- Conhecendo a interface do GitHub
- Trazendo um repositório para o seu computador: `$ git clone https://github.com/...`
- Atualizando alterações do time: `$ git pull`
- Um pouco sobre resoluções de conflitos
- Resolvendo problemas por conta própria:
 - Stack Overflow

Contribuir num projeto

- Fork
- Pull Request





github:pages

THE END



“Eu não falhei. Apenas descobri mil maneiras que não funcionam - Thomas Edison”

—