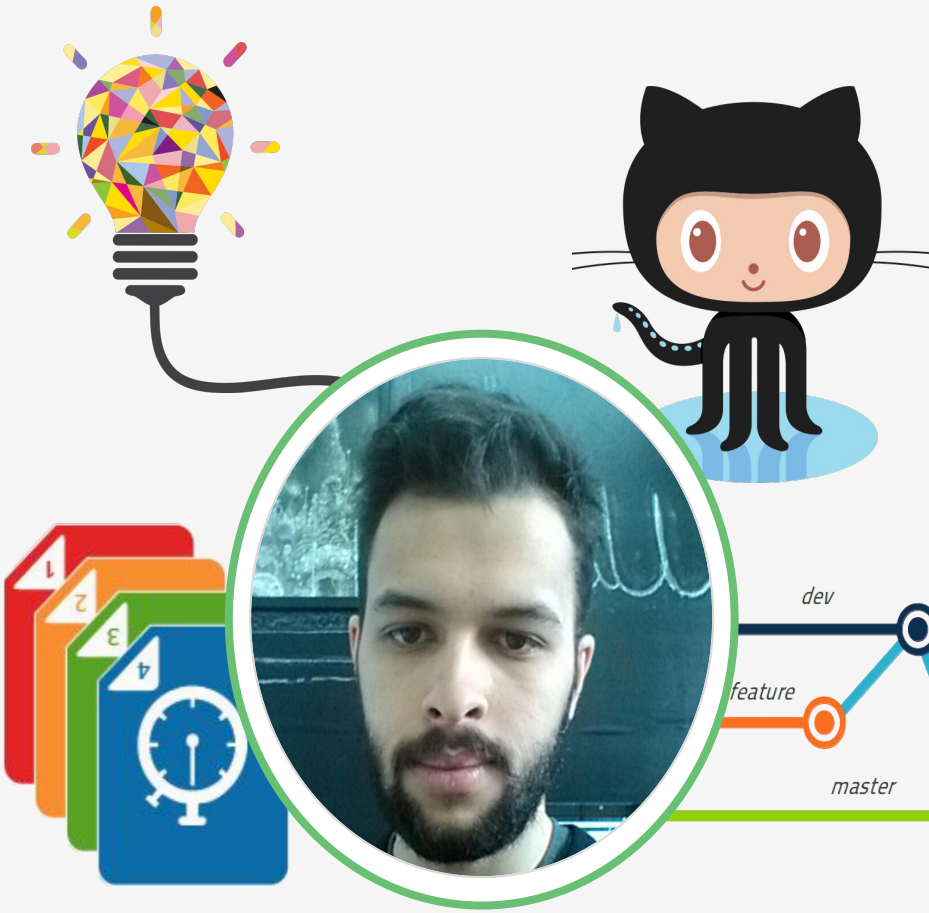
git

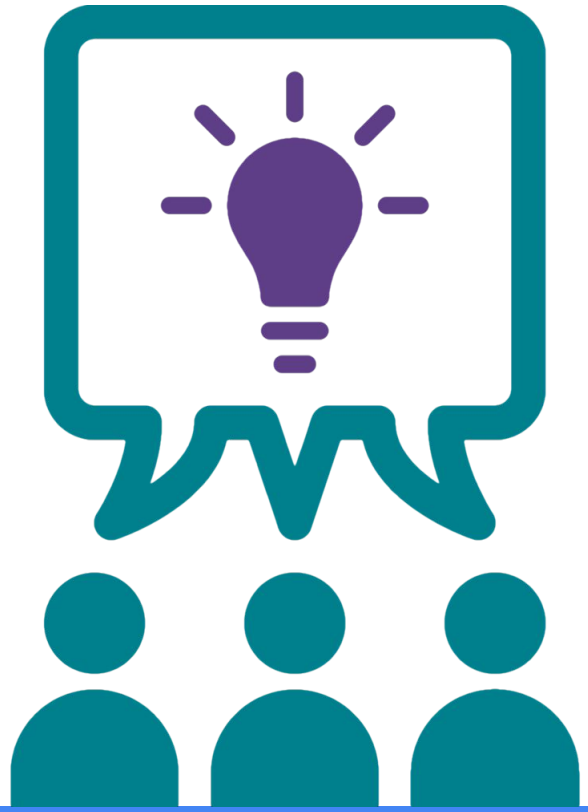
Iskailer Inaian Rodrigues

Iskailer Inaian Rodrigues

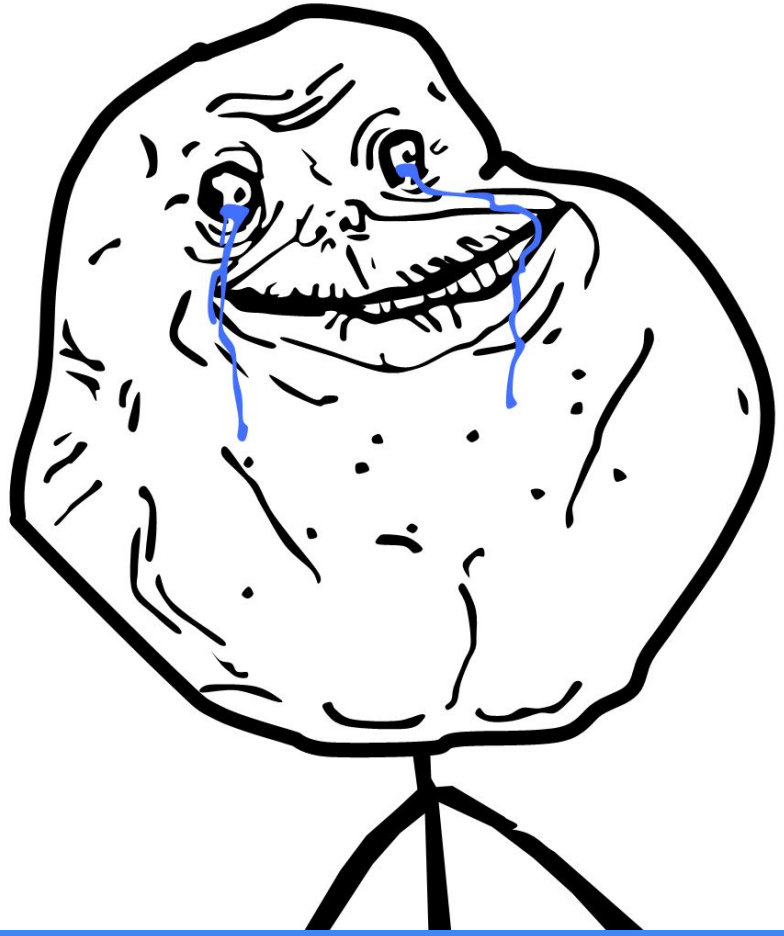
Bacharel em Sistemas de Informação pela Unoesc, com experiência no controle de qualidade e gerência de software house, é cofundador da startup Gravitwave que desenvolve soluções para o agronegócio. Responsável pelo projeto ganhador do Sinapse da Inovação, também pelo projeto entre as 100 empresas finalistas do InovAtiva Brasil. Entre 875 projetos, foi Vice campeão na categoria de Negócios Digitais no 9º concurso de plano de negócios do Sebrae-SC.



Dinâmica da Oficina!



um Pouco de História!



Um Pouco de História!

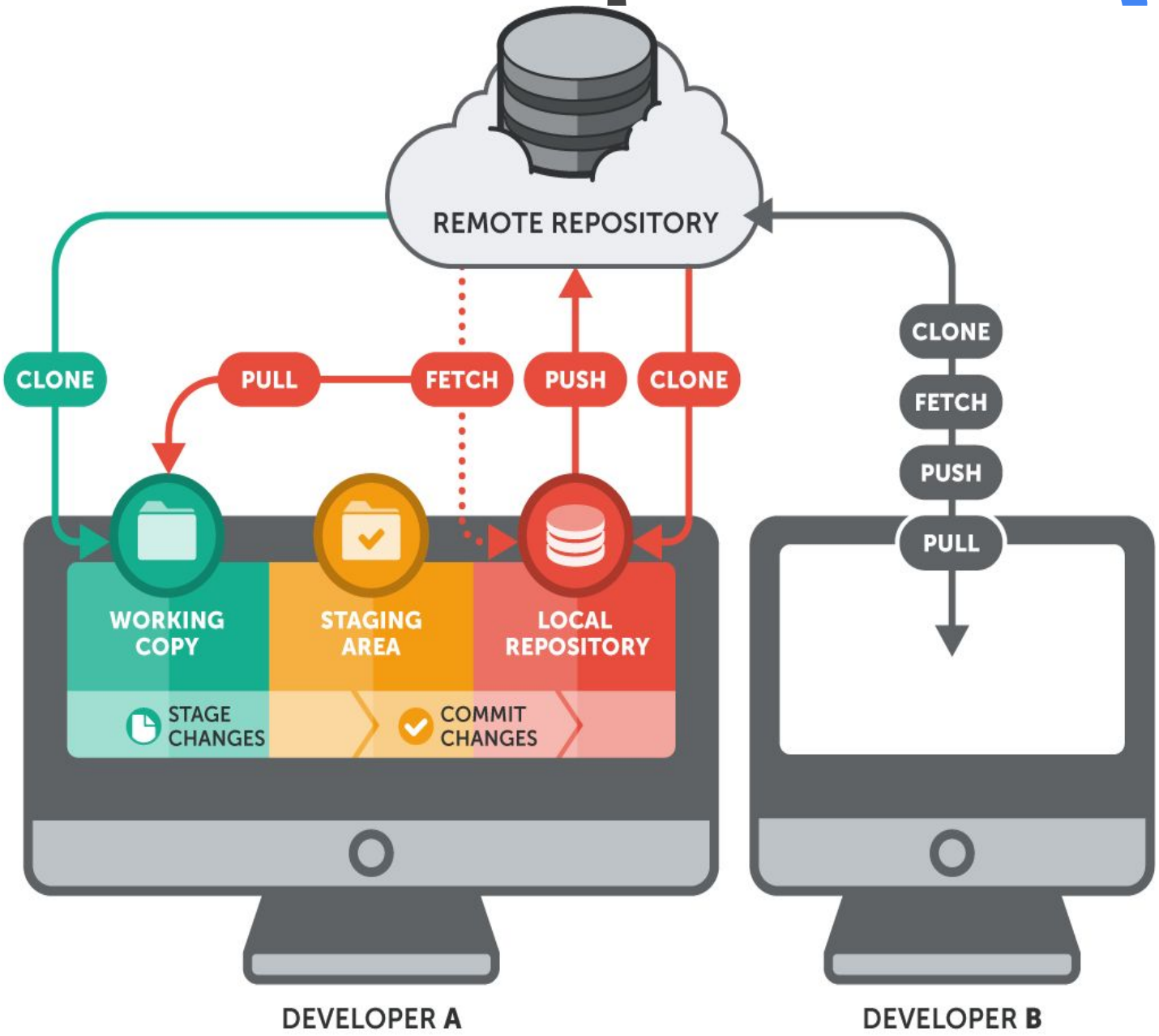
O git foi desenvolvido inicialmente por Linus Torvalds pela necessidade de ter um software capaz de controlar a versão do kernel.

O que é Git?

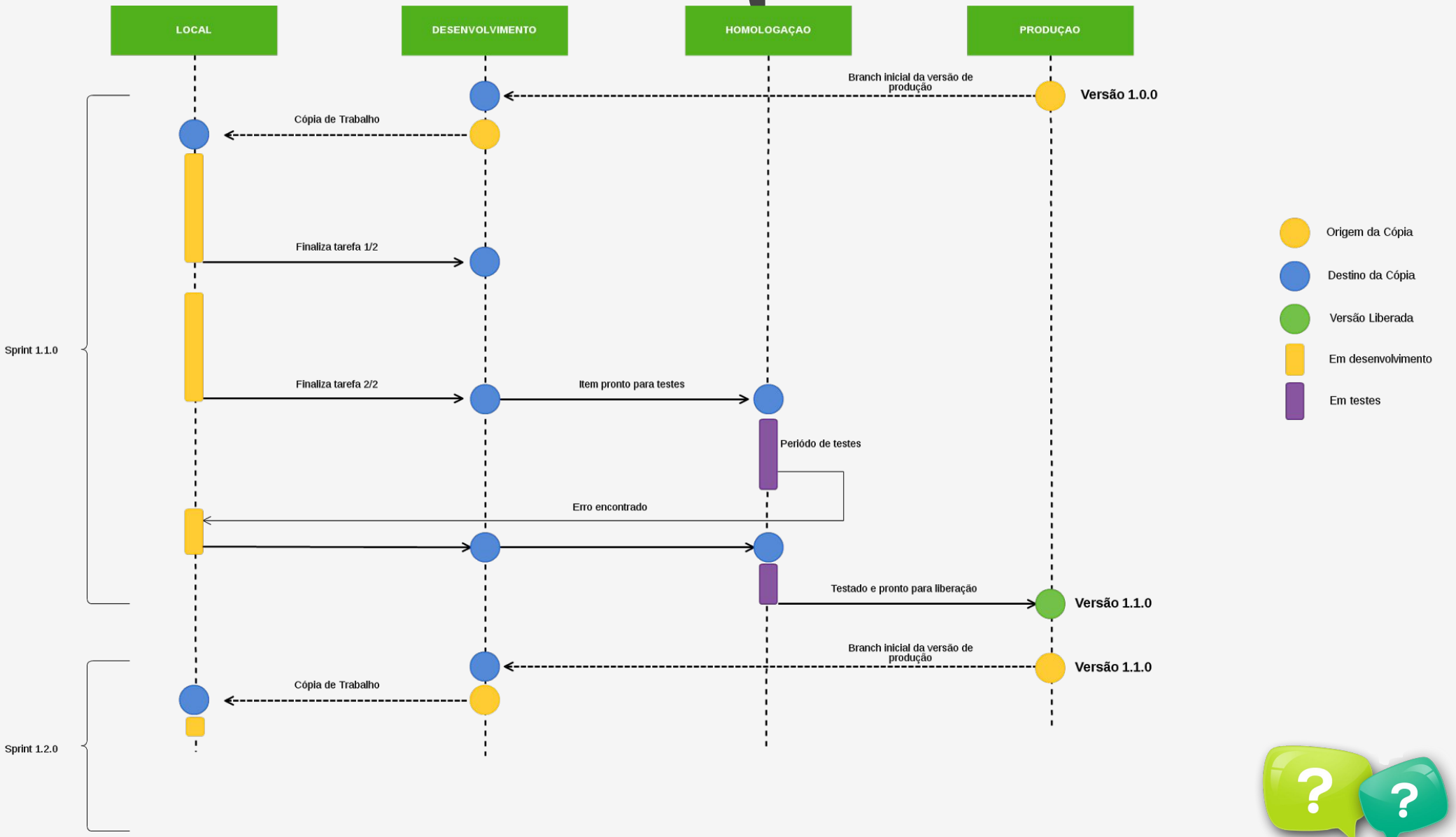
"Git é um sistema de controle de versão distribuída, rápido e escalável"

Tradução do Manual

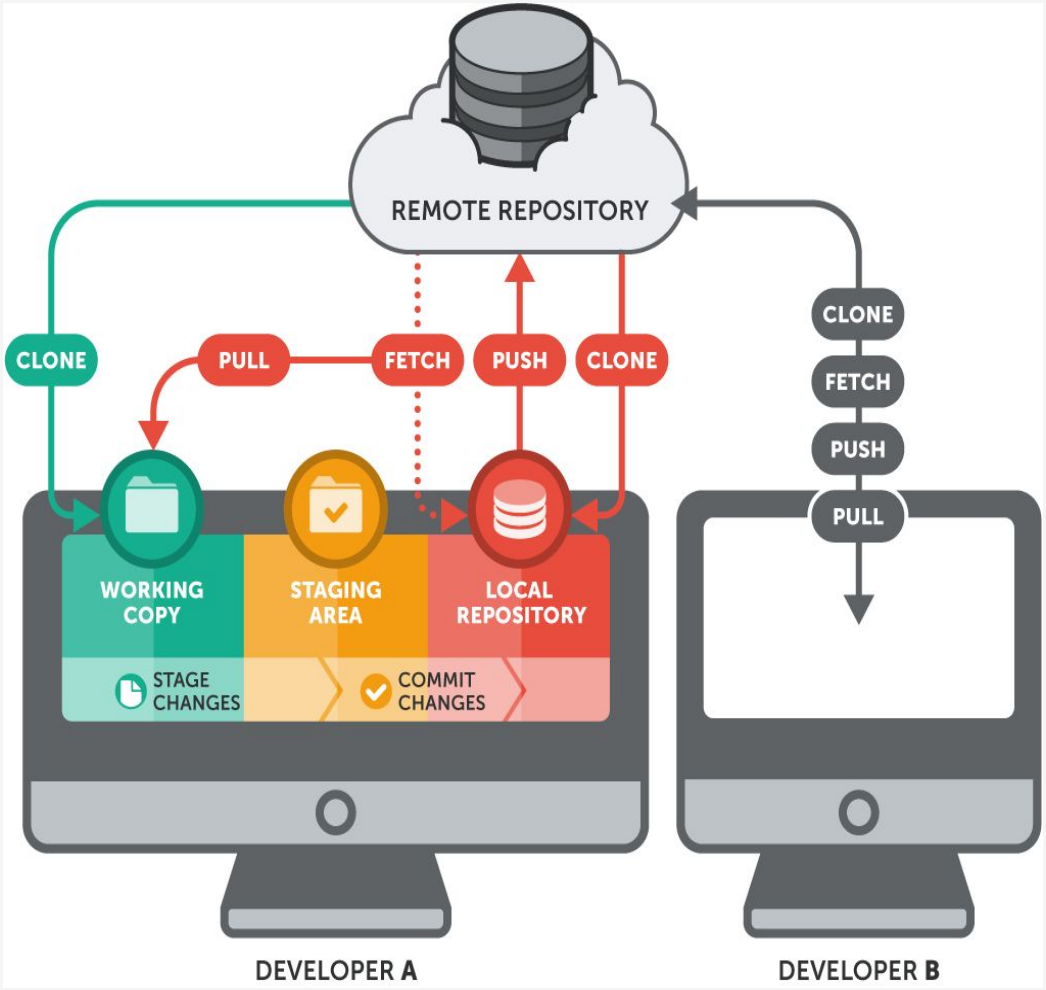
O que é Git?


REMOTE REPOSITORY
CLONE
PULL
FETCH
PUSH
CLONE
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
STAGE CHANGES
COMMIT CHANGES
DEVELOPER A
CLONE
FETCH
PUSH
PULL
DEVELOPER B

O que é Git?


LOCAL
DESENVOLVIMENTO
HOMOLOGAÇAO
PRODUÇAO
Branch inicial da versão de produção
Versão 1.0.0
Cópia de Trabalho
Finaliza tarefa 1/2
Sprint 1.1.0
Finaliza tarefa 2/2
Item pronto para testes
Período de testes
Erro encontrado
Testado e pronto para liberação
Versão 1.1.0
Branch inicial da versão de produção
Versão 1.1.0
Cópia de Trabalho
Sprint 1.2.0
Origem da Cópia
Destino da Cópia
Versão Liberada
Em desenvolvimento
Em testes

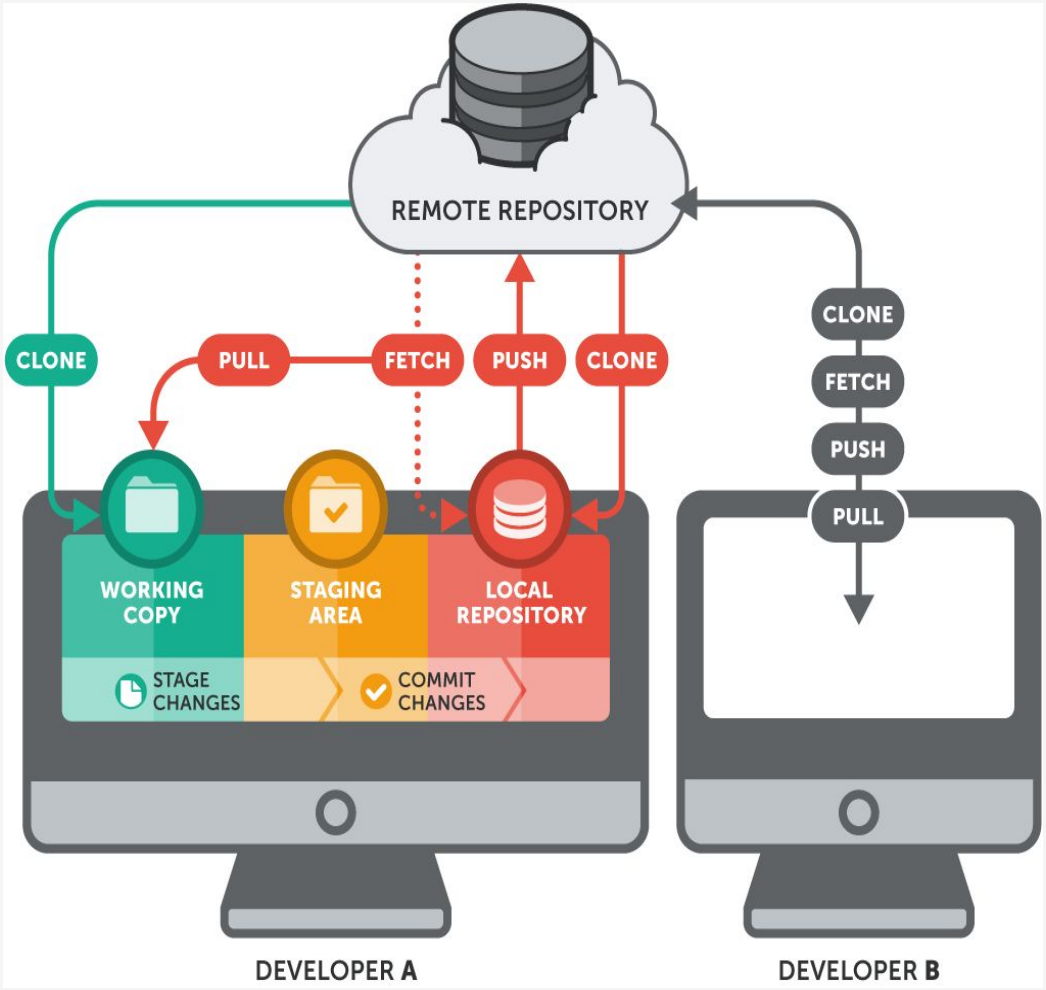
Termos



Repository: Local onde ficam todos os arquivos do projeto, inclusive os históricos e versões.

Commit: Coleção de alterações realizadas, é como se fosse um "checkpoint" do seu projeto, sempre que necessário você pode retroceder até algum commit.

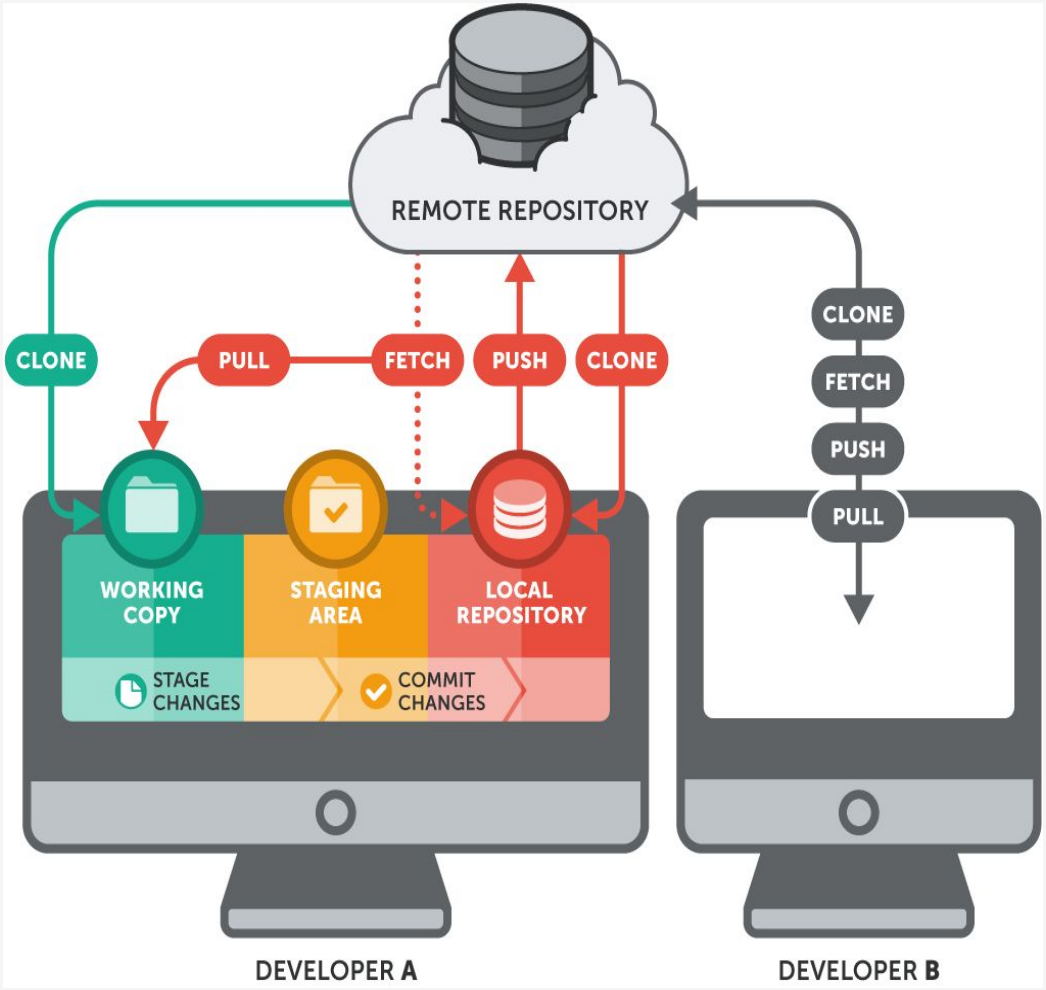
Termos



Branch: É uma ramificação do seu projeto, cada branch representa uma versão do seu projeto, e podemos seguir uma linha de desenvolvimento a partir de cada branch.

Fork: Basicamente é uma bifurcação, uma cópia de um projeto existente para seguir em uma nova direção.

Termos



Merge: É a capacidade de incorporar alterações do git, onde acontece uma junção dos branchs.

Pull: É quando você solicita ao servidor que lhe envie as últimas alterações do fonte.

Push: Você manda as últimas alterações para o servidor

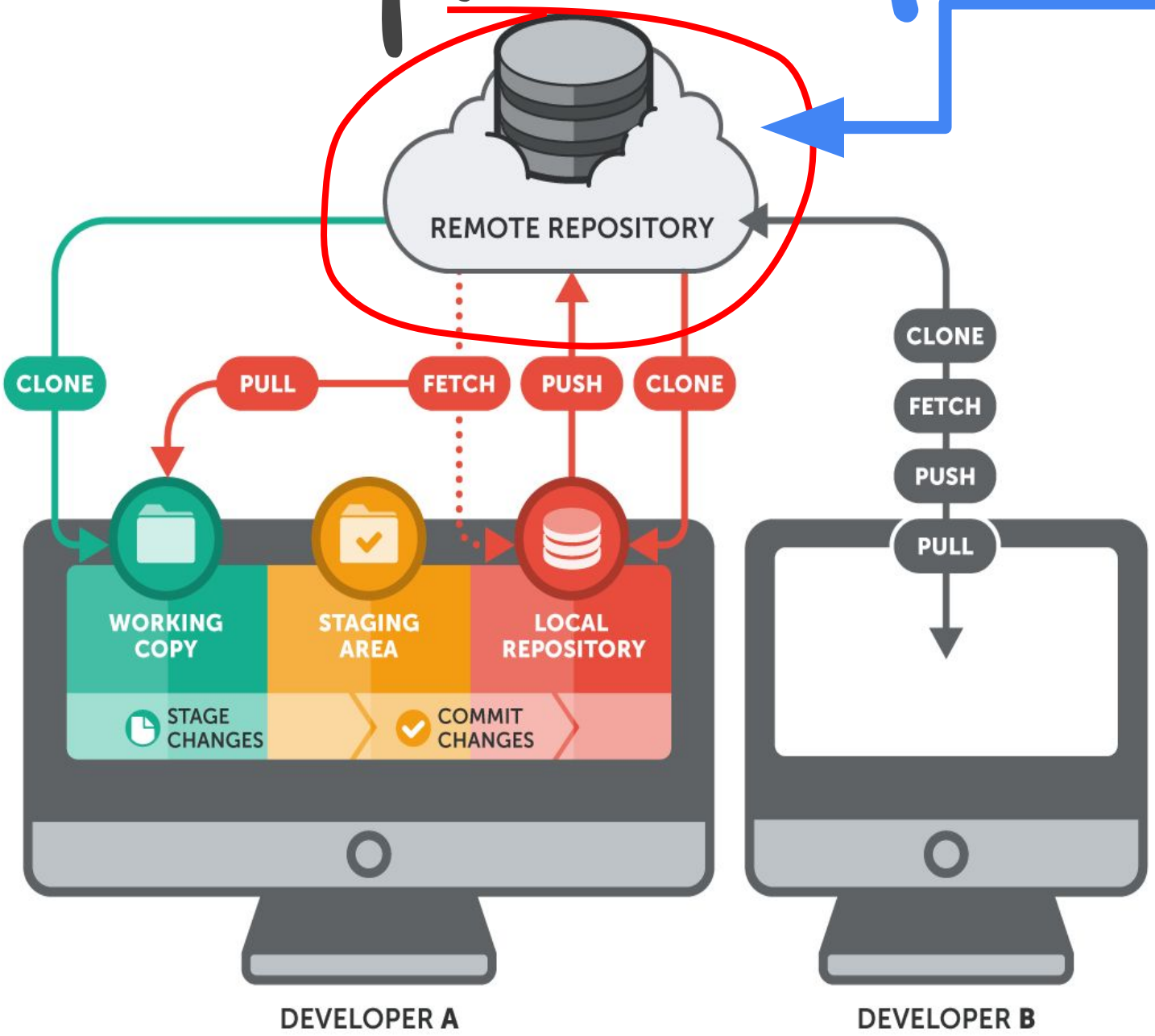

github
SOCIAL CODING

O que é GitHub?

"GitHub é um Serviço de Web Hosting compartilhado para projetos que usam o controle de versionamento Git."

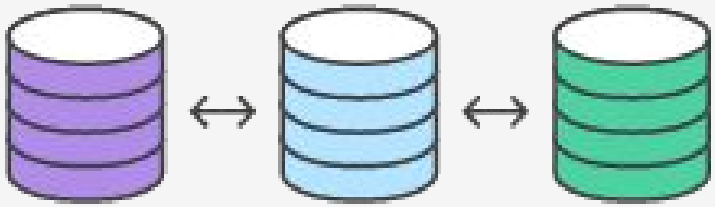
Wikipédia

O que é GitHub?



Fluxos de trabalho

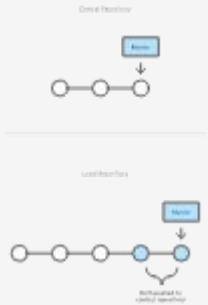
- Fluxo de Trabalho Centralizado
- Fluxo de Trabalho Feature Branch
- Fluxo de Trabalho GitFlow
- Fluxo de Trabalho Forking

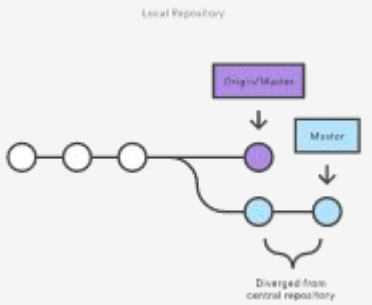


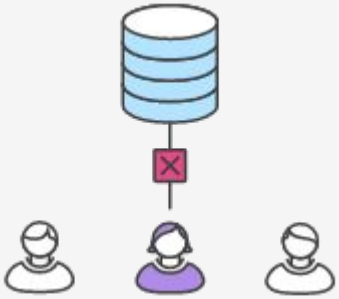
Mais detalhes em:
https://www.atlassian.com/git/tutorials/comparing-workflows

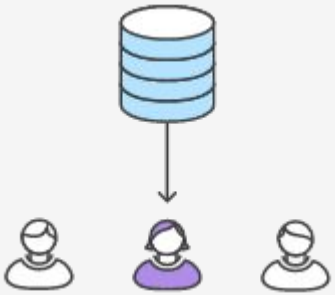
Fluxos de trabalho

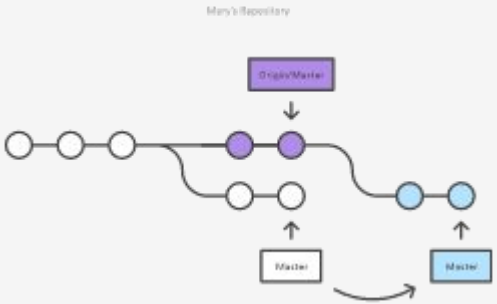
Fluxo de Trabalho Centralizado





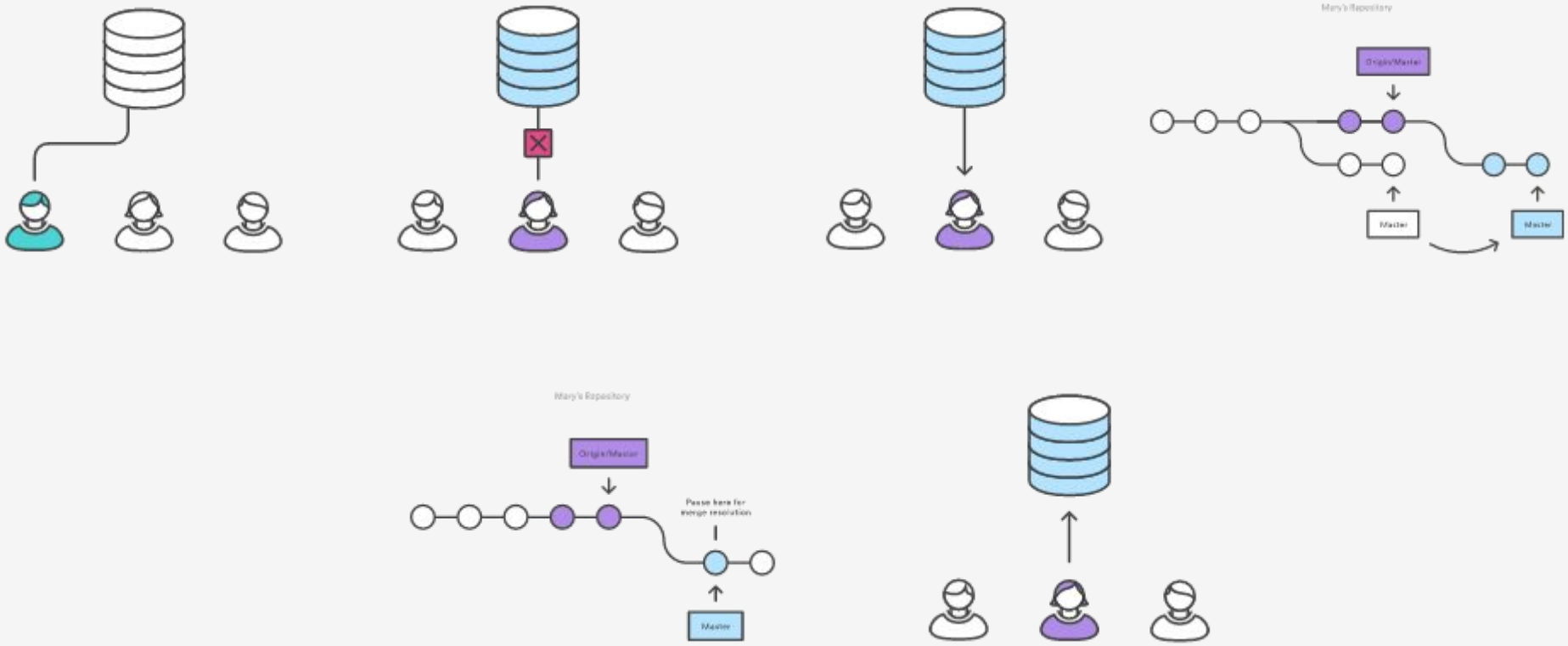






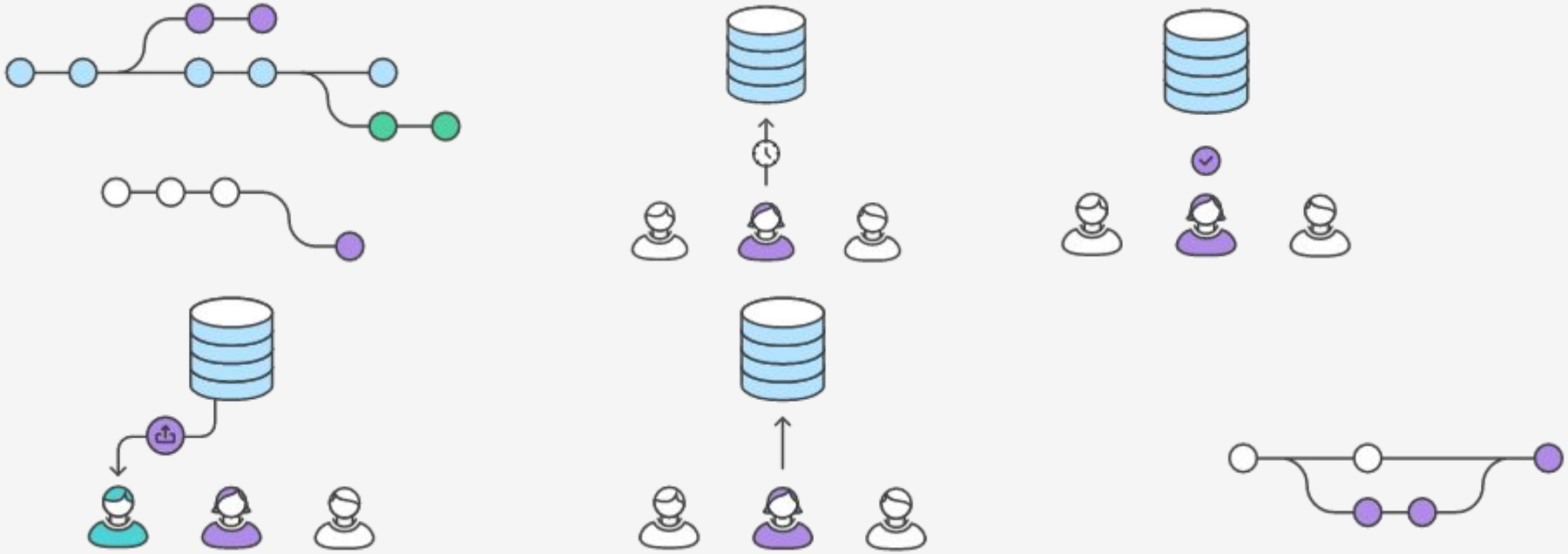
Fluxos de trabalho

Fluxo de Trabalho Centralizado



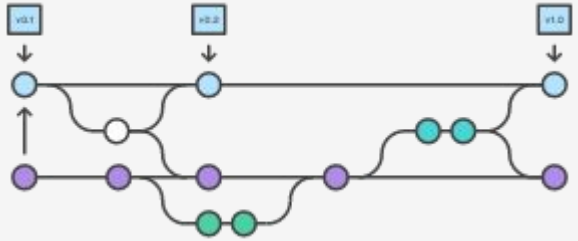
Fluxos de trabalho

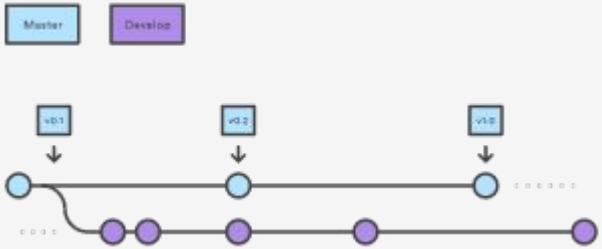
Fluxo de Trabalho Feature Branch

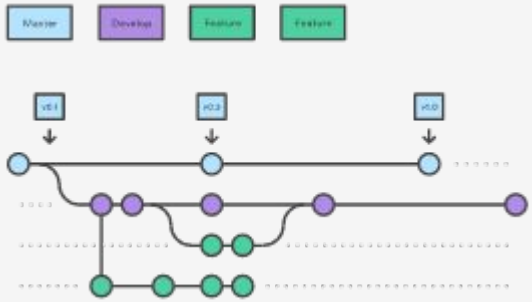


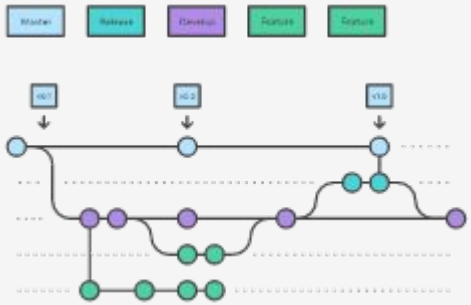
Fluxos de trabalho

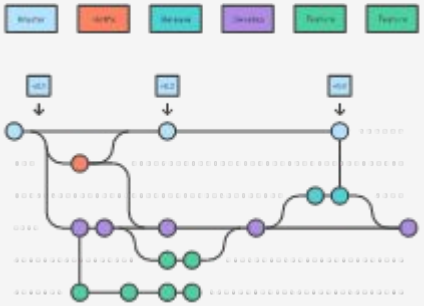
Fluxo de Trabalho GitFlow





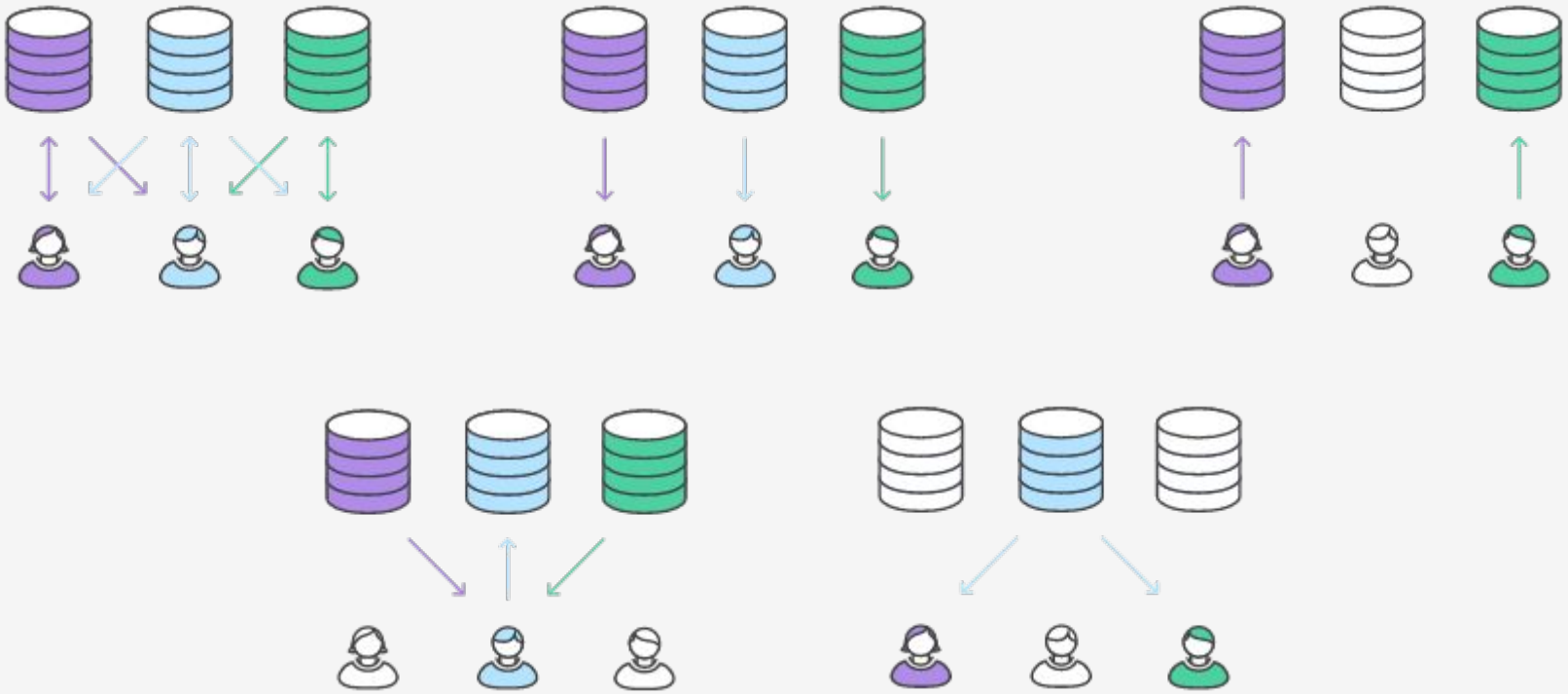






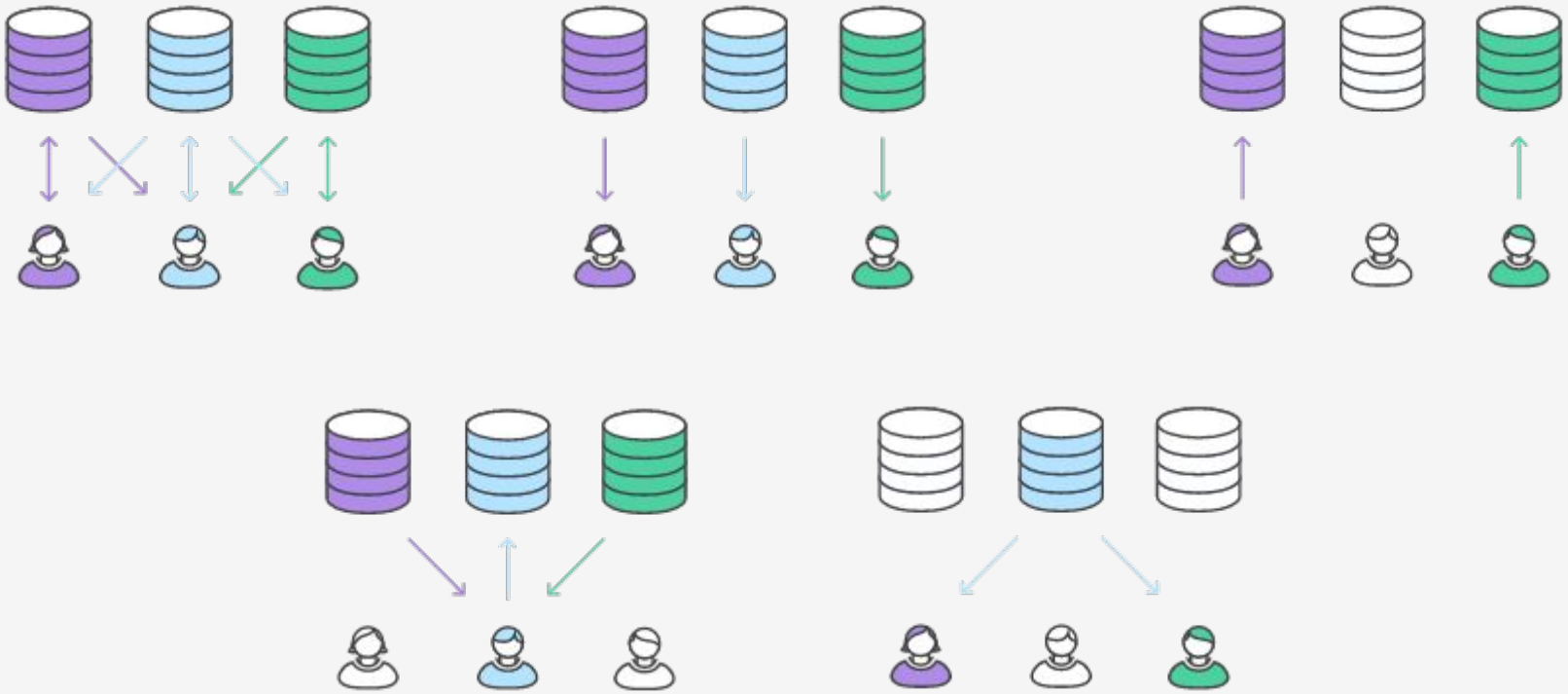
Fluxos de trabalho

Fluxo de Trabalho Forking



Fluxos de trabalho

Fluxo de Trabalho Forking



Mão na massa

Acessar e fazer a instalação:
http://bit.ly/2cgT7b9

- **Instalando no Linux**
- **Instalando no Mac**
- **Instalando no Windows**

GitHub

Acessar e criar uma conta: https://github.com/

- **Tempo máximo 3 Minutos**

Começando

Criação de um novo diretório e em seguida iniciar o repository git:

```
mkdir NovoProjeto
cd NovoProjeto

git init
```

Todos os arquivos dentro do diretório NovoProjeto fazem parte do repository

Configurando

Informações do desenvolvedor, isto é feito para identificar os desenvolvedores em cada commit específico:

git config user.name “Nome”
git config user.email “email@email.com”

OBS.: foi criada a pasta “.git” dentro do diretório, neste local ficam os registros de todo o projeto.

Exemplificando

Criação de um novo arquivo:
touch exemplo.txt

git add exemplo.txt

Ou, caso o programador queira enviar todos os arquivos:

git add .

Exemplificando

O git é inteligente o suficiente para apenas realizar commits se detectar alguma alteração nos arquivos que foram indicados pelo (git add), dessa forma o único commit que ele realiza sem alguma alteração é o primeiro.

Nosso Primeiro Commit...

git commit -a -m “Nosso primeiro commit”

Passamos o parâmetro -a para informar que todos os arquivos alterados devem ser incluidos do commit.

Passamos o paramêtro -m para adicionar uma mensagem explicativa para o commit.

Comandos úteis

Depois que fizemos nosso primeiro commit existem alguns comandos que podemos utilizar para verificar o mesmo, são eles:

git log // listar todos os commits já realizados.

git status // mostrar os arquivos alterados desde o último commit.

git show // mostrar as alterações realizadas no último commit.

Branchs

Todo repositório, quando criado, possui um branch default chamado “master”.

Para listar todos os branchs neste repositório:

git branch

*master

Branchs

Criação de branch:

git branch working

Criação de branch baseado em outros. Criamos o branch "outro" como base o branch "working" e vamos nos manter no branch **corrente**:

git branch outro working

Branchs

Para criar um novo branch e já entrarmos nele é utilizado o “checkout”:

git checkout -b “outro”

Por hora devemos ter os seguintes branchs:

master
*outro

o branch sinalizado com “” é o branch corrente.*

Branchs

Mudando de branch corrente:

git checkout master

Agora a saída fica assim:

*master
outro

Para deletarmos um branch fazemos dessa maneira:

git branch -d outro

Não é possível deletar o branch corrente.

Exemplificando Merge

Digamos que o arquivo exemplo.txt foi alterado no branch “outro”.

git checkout outro // alterando para o branch outro.

git add . // indicamos todos os arquivos do commit.

git commit -a -m “alteração exemplo.txt” // commit.

Exemplificando Merge

Agora queremos que nossa alteração no exemplo.txt seja refletida a nossa branch master.

Primeiro vamos para o branch que queremos as alterações refletidas:

git checkout master

Agora realizamos o merge das alterações do branch outro para o current branch:

git merge outro

Exemplificando Merge

Saída:

Updating 642caa9..851gb82

Fast forward
exemplo.txt | 1 +
1 files changed, 1 insertions(+), 0 deletions(-)

Exemplificando Merge

Confirmando a alteração no arquivo:

cat exemplo.txt

Nesse momento toda a alteração realizada no branch “outro” será refletida no branch master inclusive nossos commits que fizemos no branch outro e agora está no master.

Exemplificando Merge

Todo branch possui um branch pai, que é a base a partir de então, sabendo disso e se precisássemos fazer um merge em um branch master que já tivesse alguma alteração depois que o branch outro foi criado? Dessa forma o branch outro não estava refletindo a última versão de master, como proceder então?

Exemplificando Merge

Nesse caso a melhor forma seria o “rebase” ou seja pegar o último commit do branch master, e trazer para o branch outro e aplicar todos os seus commits nele, feito isso agora nosso branch outro está refletindo a última versão do master

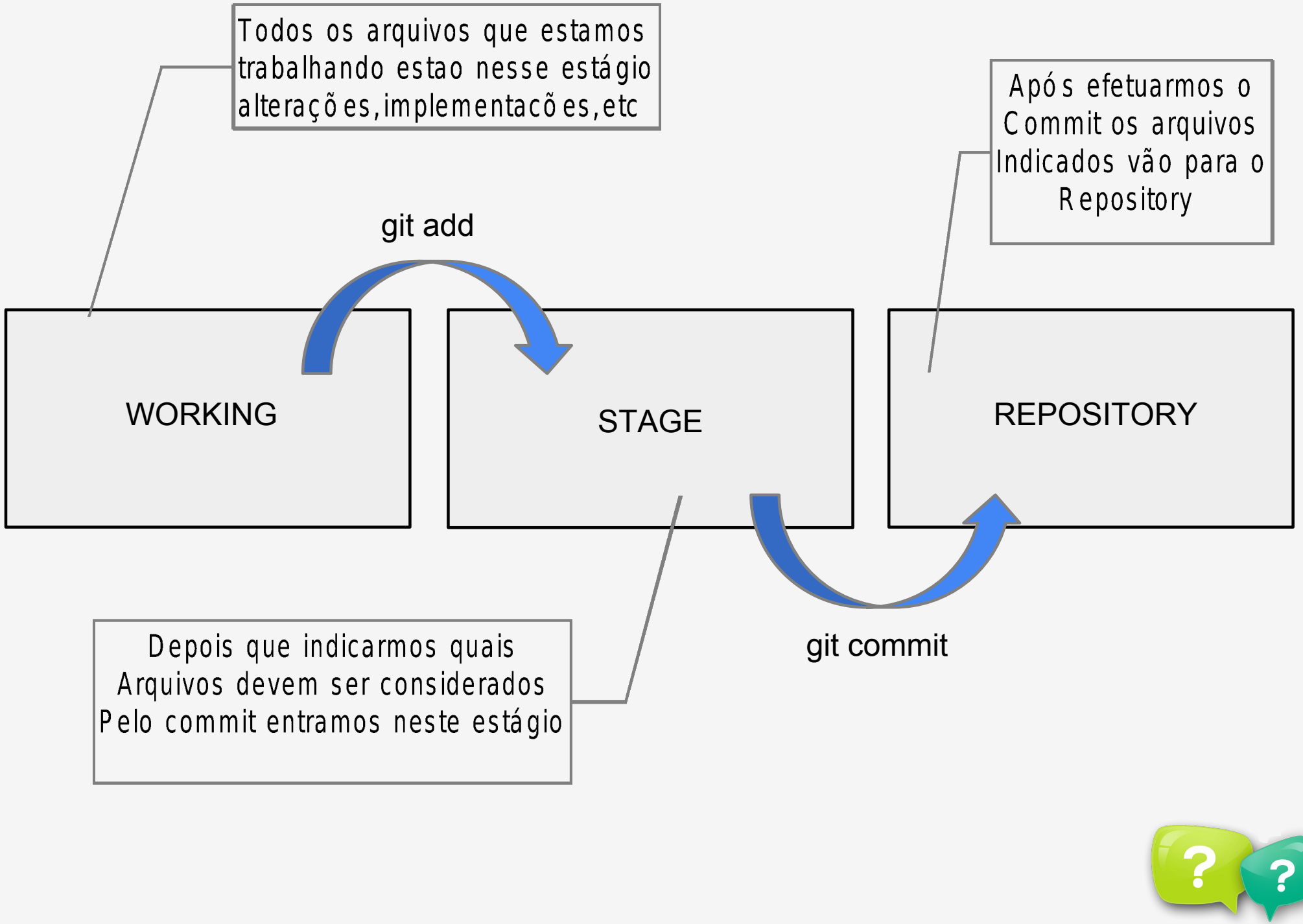
Poderíamos fazer o merge também, mas talvez poderia causar conflitos e a última coisa que queremos são conflitos.

Exemplificando Merge

git checkout outro
git rebase master

Agora podemos realizar o merge:

git checkout master
git merge outro


Todos os arquivos que estamos trabalhando estao nesse estágio alteraçõ es,implementacõ es,etc
Após efetuarmos o Commit os arquivos Indicados vão para o Repository
git add
WORKING
STAGE
REPOSITORY
Depois que indicarmos quais Arquivos devem ser considerados Pelo commit entramos neste estágio
git commit

Repositórios Remotos

São repositórios hospedados em servidores na internet ou outra máquina.

github.com

-> Usar a conta do GitHub.

Clonando um Projeto

```
git clone
https://github.com/MeuUsuario/MeuProjeto
```

Clonando um Projeto

Veja os repositórios remotos disponíveis:

```
cd MeuProjeto
git branch –r
```

```
origin/HEAD
origin/master
origin/working
```

Clonando um Projeto

Devemos criar um branch local baseado em algum branch remoto antes de começar a efetuarmos nossas alterações.

git checkout –b outro origin/working

Clonando um Projeto

Agora vamos supor que nós efetuamos várias alterações neste repositório e durante esse tempo o projeto principal também foi alterado não correspondendo mais a base que nós possuímos agora, e então desejamos sincronizar com a última versão disponível do projeto.

Clonando um Projeto

Primeiramente recuperamos a versão recente do projeto:

git fetch

Agora efetuamos o merge com o branch atual:

git merge origin/master


Seus projetos!

Dúvidas?

Sugestões?


about.me/iskailer