Programação Orientada por Objetos

Introdução à gestão de versões

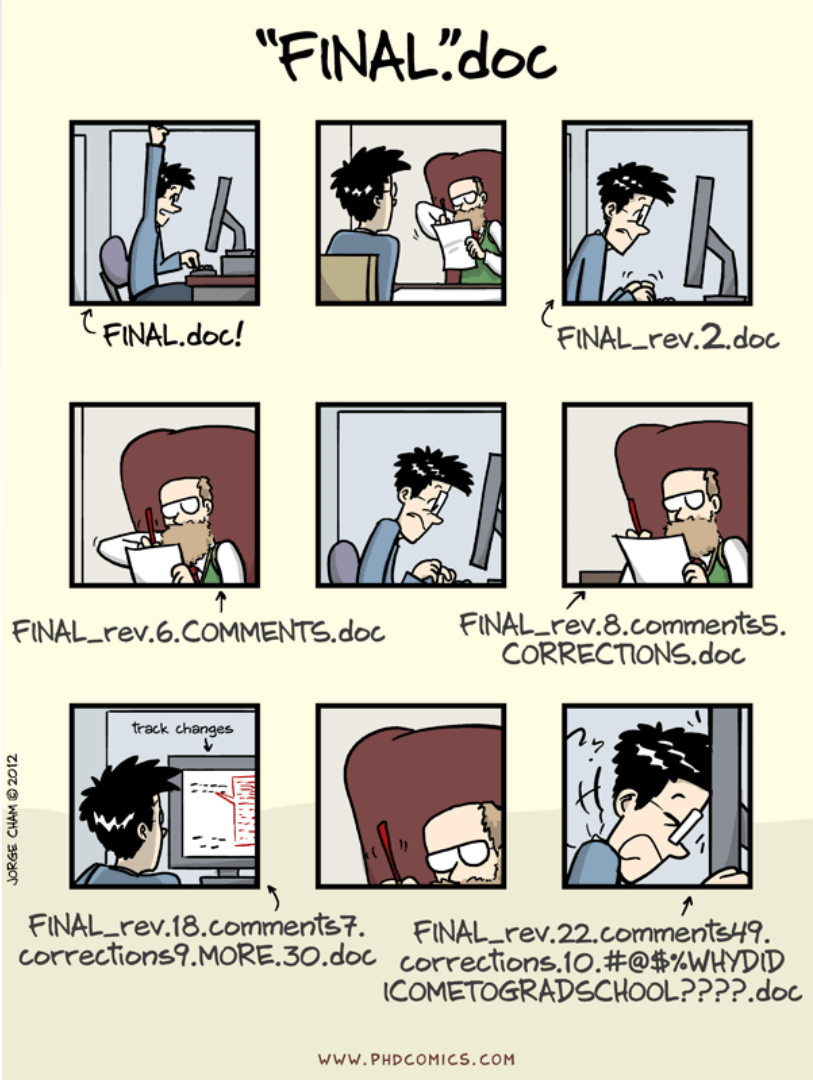
Prof. Cédric Grueau
Prof. José Sena Pereira
Departamento de Sistemas e Informática
Escola Superior de Tecnologia de Setúbal
Instituto Politécnico de Setúbal
2022/2023

Sumário

- A Gestão de Versão
- Sistemas de Gestão de Versão
- Git, GitHub, GitHub Classroom
- Utilização de Git

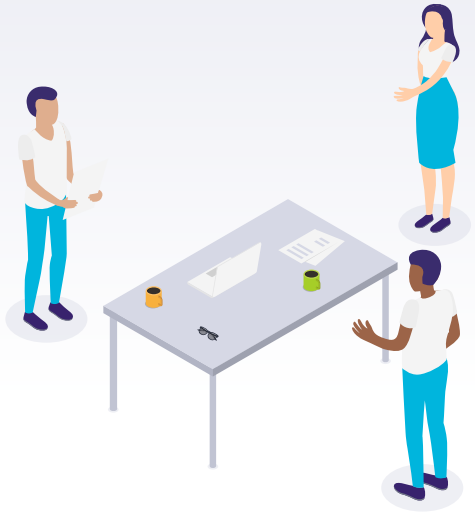
Problema

- “Piled Higher and Deeper” por Jorge Cham, http://www.phdcomics.com



Problema

- Imagine que escreveu um programa que estava a funcionar mas que depois o estragou ao implementar uma nova funcionalidade.
  - Como iria conseguir recuperar a versão funcional?
  - É mesmo possível?
  - Como fez no projeto da disciplina de IPOO?



Solução

- Recuperar a versão funcional do seu código só é possível se tiver criado uma cópia da versão antiga do código.
- O perigo de perder as versões funcionais geralmente leva ao fluxo de trabalho problemático ilustrado no cartoon PhD Comics mostrado anteriormente.

- O ***controlo de versão*** é usado para **rastrear** e **armazenar** alterações num conjunto de ficheiros sem perder o histórico de alterações anteriores.

Sistema de controlo de versão

- Um sistema de controlo de versão é uma ferramenta que rastreia essas mudanças para nós, criando efetivamente diferentes versões dos nossos ficheiros.
- Os sistemas de controlo de versões começam com **uma versão base do documento** e, a seguir, **são guardadas apenas as alterações** feitas em cada etapa do processo.

O que é controlo de versão?

- Permite-nos decidir quais mudanças serão feitas na próxima versão (cada registo dessas mudanças é chamado de commit), e mantém **metadados** úteis sobre eles.
- O histórico completo de commits para um projeto específico e os seus metadados constituem um repositório.
- Os repositórios podem ser mantidos e sincronizados em diferentes computadores, facilitando a colaboração entre diferentes pessoas.

Aspectos chave

- O controlo de versão permite:
  - Voltar facilmente à uma versão anterior
    - O controlo de versão é como um "undo" ilimitado.
  - Seguir a evolução do projeto ao longo do tempo
  - O trabalho em paralelo em partes disjuntas do projeto e gerir as modificações concorrentes
  - Facilitar a deteção de erros
  - ....

Histórico

- Anos 1980: sistemas automatizados de controle
  - RCS, CVS ou Subversion
  - usados por muitas empresas grandes.
  - Sistemas desatualizados devido a várias limitações nas suas funcionalidade.
- Sistemas mais modernos: como **Git** e **Mercurial**,
  - Distribuídos: não precisam de um servidor centralizado para alojar o repositório.
  - Ferramentas de fusão poderosas (***merging***) : vários autores trabalhem nos mesmos ficheiros simultaneamente.

Sistema local

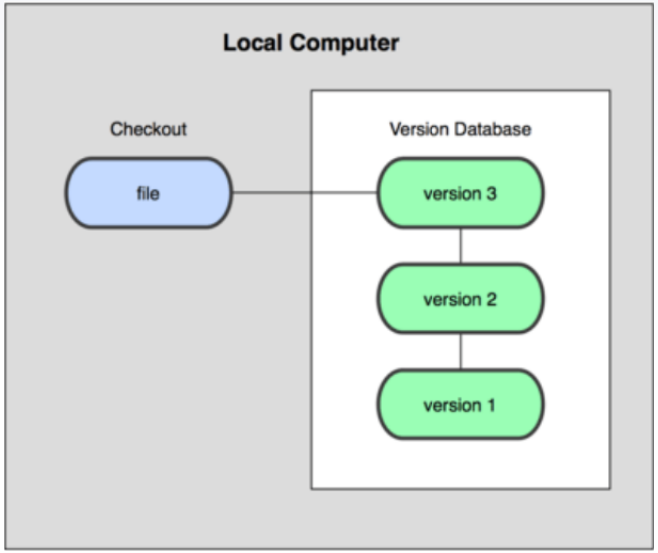
Vantagens

Gestão e utilização muito simples

Desvantagens

Muito sensível às avarias

Não permite a colaboração



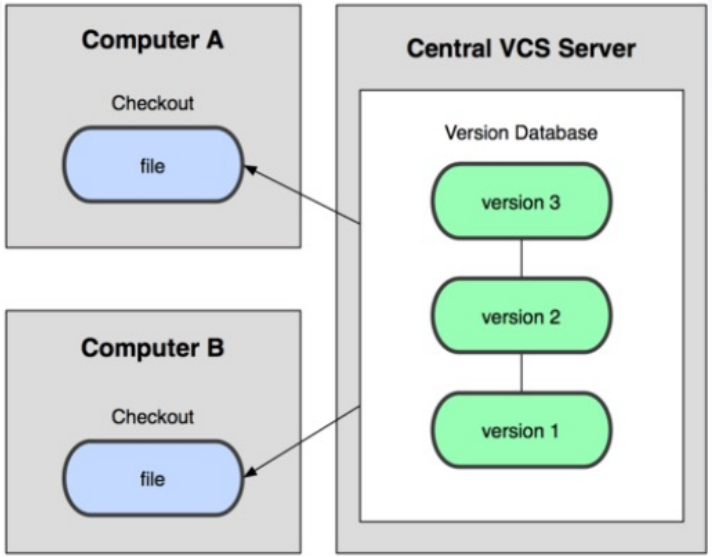
Sistema centralizado

Vantagens

- Estrutura simples
- Gestão e utilização muito simples

Desvantagens

- Muito sensível às avarias
- Não adaptado aos grandes projetos e / ou à uma estrutura muito hierarquizada a colaboração



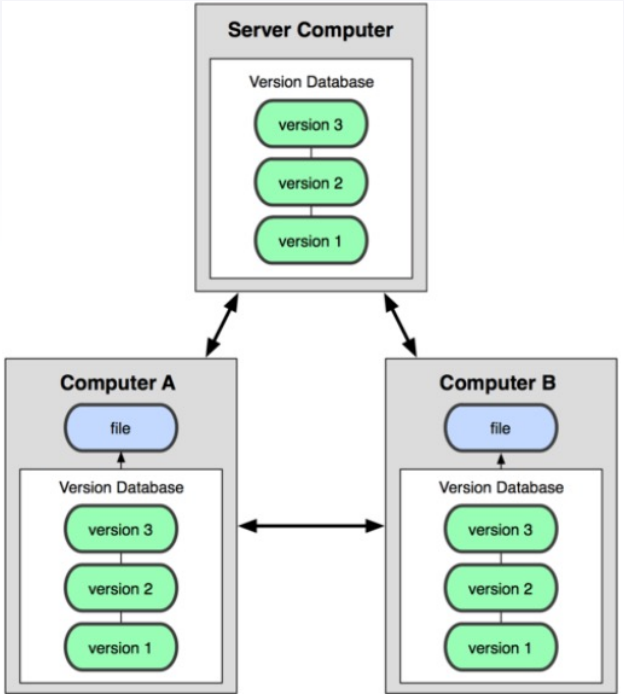
Sistema distribuído

Vantagens

- Menos sensível às avarias
- Adaptado aos grandes projetos e / ou à uma estrutura muito hierarquizada

Desvantagens

- Gestão e utilização mais complicadas
- Pode ter uma estrutura muito complexa

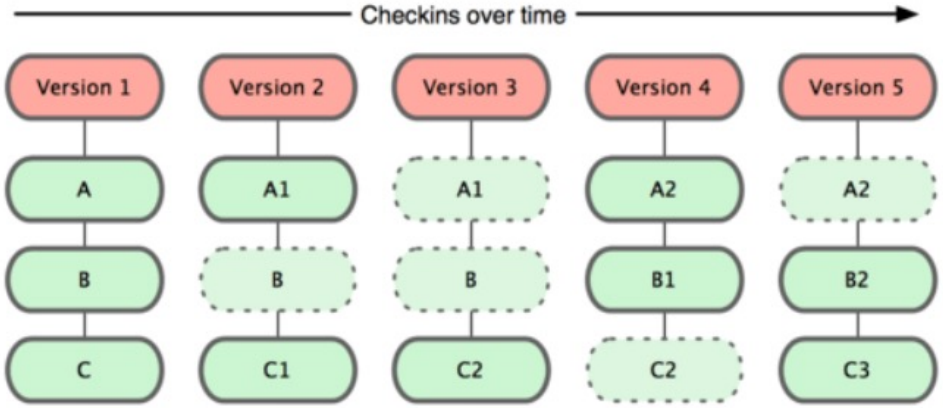


git – sistema de gestão de versão distribuído

- Histórico (Wikipedia)
  - De 1991 a 2002, o núcleo de Linux estava a ser desenvolvido sem usar sistema de gestão de versão.
  - A partir de 2002, a comunidade começou a usar BitKeeper, um DVCS proprietário.
  - Em 2005, depois de um contencioso, BitKeeper retira a possibilidade de usar gratuitamente o seu produto. Linus Torvalds lança o desenvolvimento de Git e após alguns meses de desenvolvimento, Git aloja o desenvolvimento do núcleo Linux

Git – princípios de base

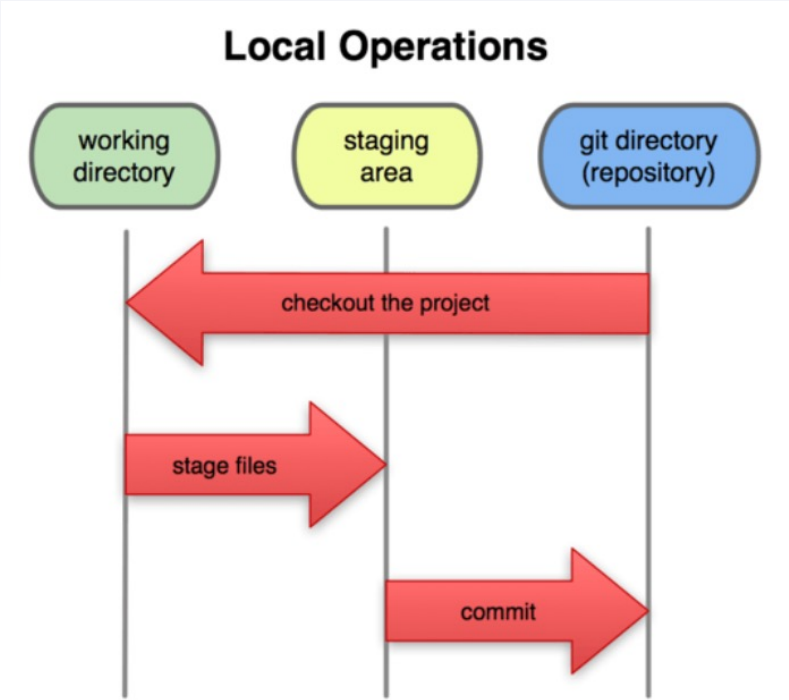
- Um **repositório** Git é uma espécie de sistema de ficheiros (base de dados), que guarda as versões dos ficheiros de um projeto em determinados momentos, ao longo do tempo sob a forma de **instantâneos** (snapshots).



Git – princípios de base

3 zonas num projeto Git

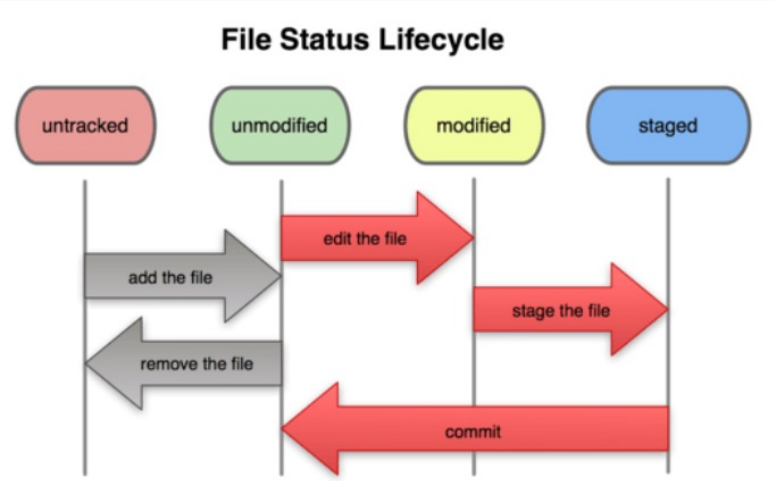
- O repositório Git: contém os metadados e a base de dados dos objetos do projeto
- O diretório de trabalho: extração única da versão do projeto desde a base de dados do repositório
- A zona de trânsito / índex: simples ficheiro que contém as informações sobre o que será considerado na próxima submissão/commit.



Git – princípios de base

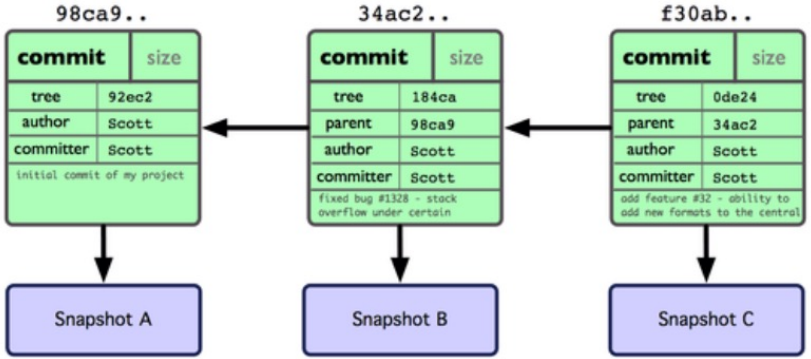
4 estados de um ficheiro no Git

- não "versionado": ficheiro não gerido ou que deixou de ser gerido pelo Git;
- não modificado: ficheiro guardado de forma segura na sua versão atual na base de dados do repositório;
- modificado: ficheiro que sofreu modificações desde a última vez que houve um commit;
- Indexado (staged): idêntico ao modificado, só que será integrado no próximo instantâneo na sua versão corrente durante o próximo commit



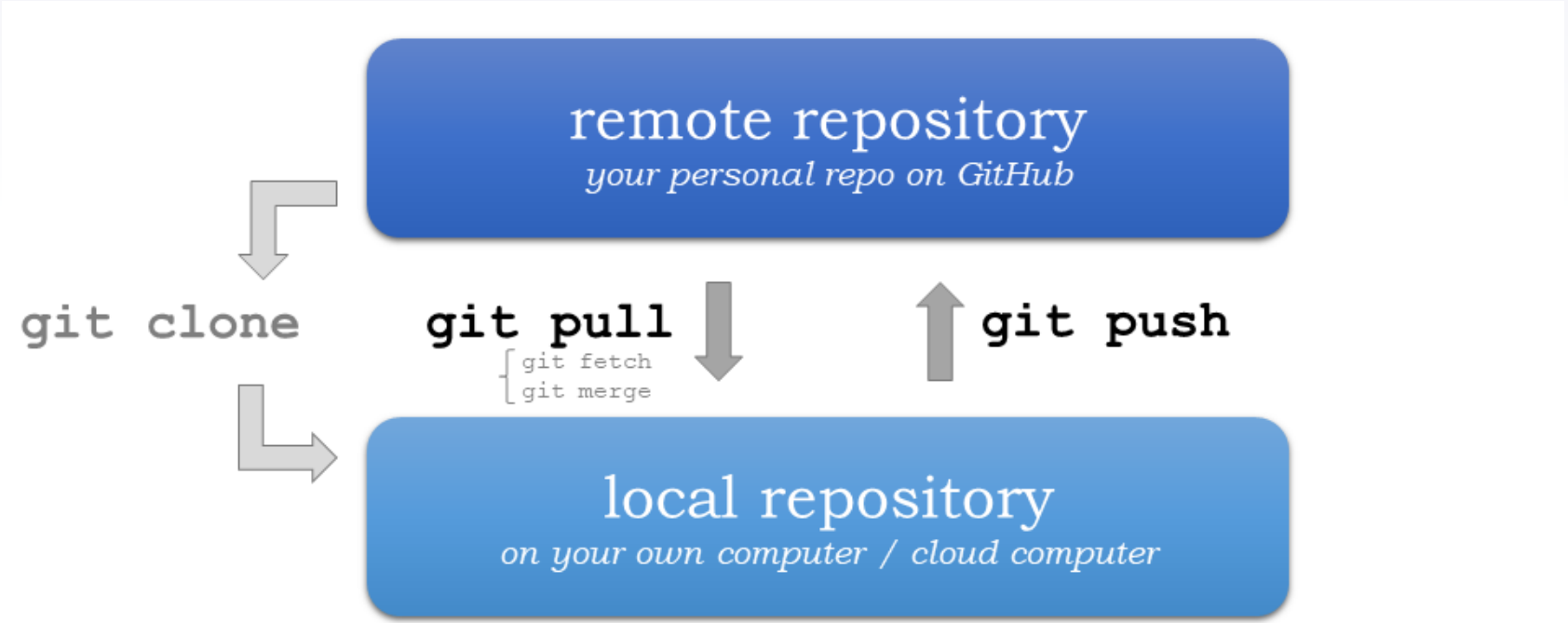
Git – princípios de base

Cada submissão (commit) dá lugar à criação de um objeto "commit" que contém um apontador para um instantâneo do conteúdo cujas modificações estavam indexadas no momento da submissão (conjunto de objetos que permitem armazenar um instantâneo dos ficheiros em causa e por outro lado eu permitem reproduzir a estrutura do repositório do projeto e dos subdiretórios), alguns metadados (autor, mensagem) e um apontador para o objeto "commit" anterior.

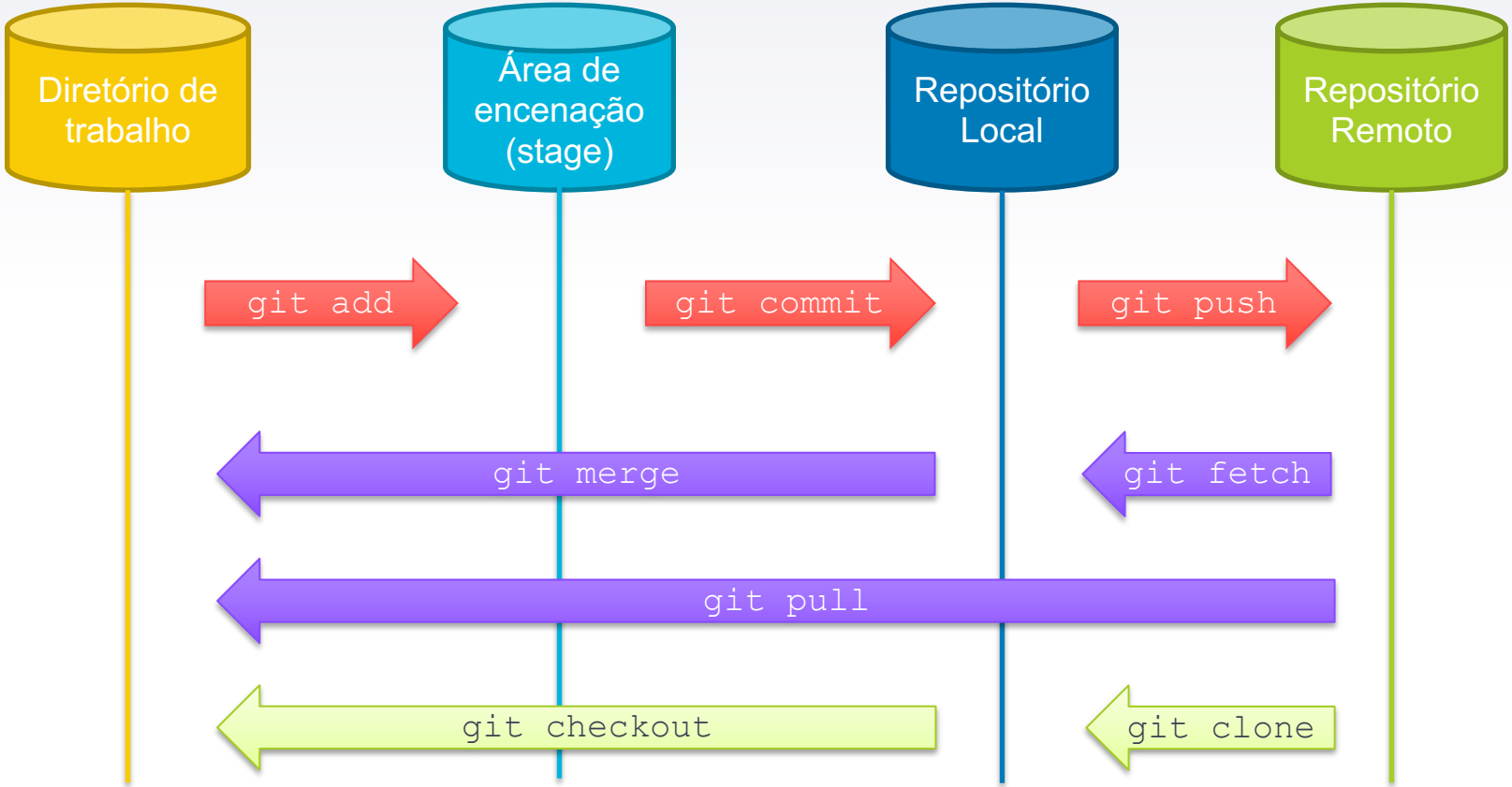


Git – princípios de base

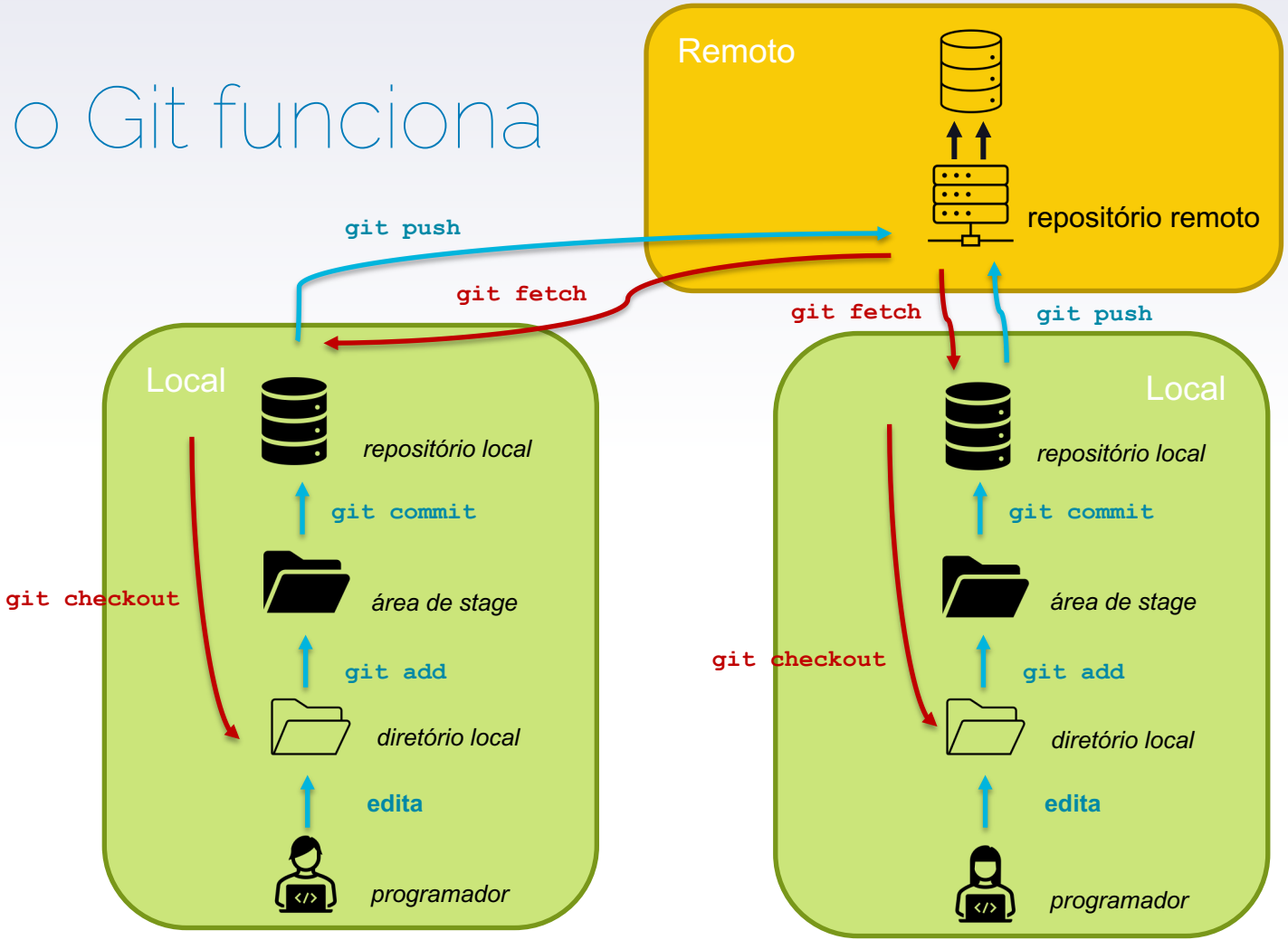
Git – princípios de base



Como os comandos Git funcionam



Como o Git funciona

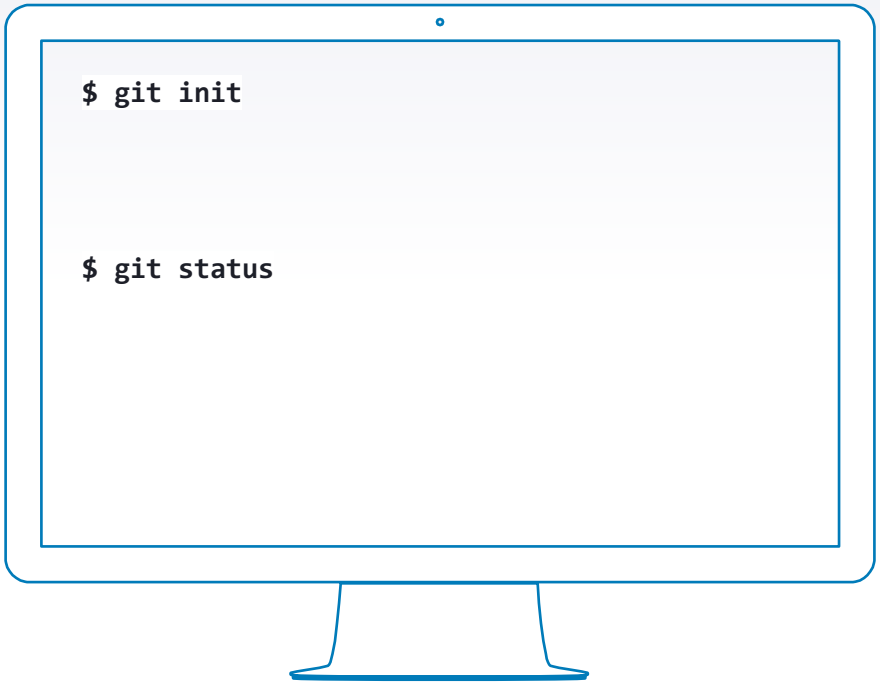


Git – comandos de base

Inicializar um repositório

Mostrar o estado dos ficheiros do repertório corrente

- *Untracked files*: ficheiros não versionados
- *Changes to be committed*: modificações (adição,remoção,mudança) carregadas na zona de trânsito (staging area), ou indexadas,
- *Changes not staged for commit*: moficiações que não foram acrregadas para a zona de trânsito (ou indexadas).



Git – comandos de base

Indexar a adição ou modificações de um ficheiro

```
$ git add [-p] <ficheiro>
```

Cancelar as modificações indexadas de um ficheiro

```
$ git reset <ficheiro>
```

Cancelar as modificações ainda não indexadas de um ficheiro

```
$ git checkout [--] <ficheiro>
```

Indexar a eliminação de um ficheiro

```
$ git rm <ficheiro>
```

"Deversionar" um ficheiro

```
$ git rm --cached <ficheiro>
```

Git – comandos de base

Mostrar o detalhe de modificações não indexadas

```
$ git diff
```

Mostrar o detalhe das modificações indexadas

```
$ git diff --staged
```

Submeter as modificações indexadas em zona de trânsito

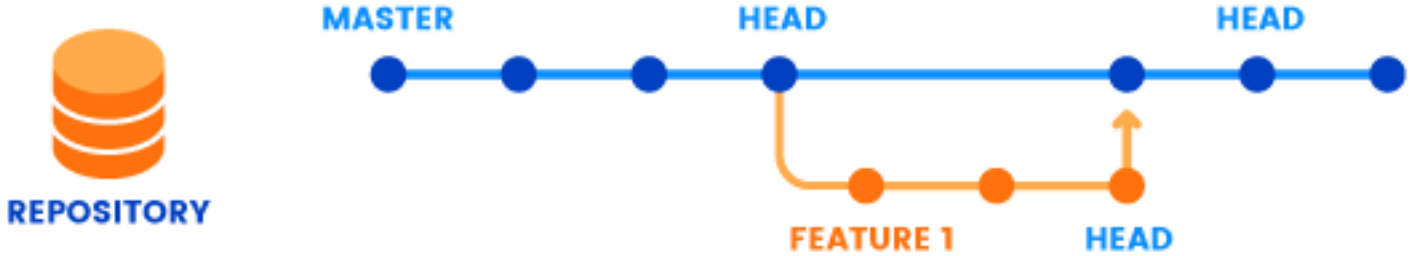
```
$ git commit
```

Ver o histórico das submissões

```
$ git log
```

Ramos

- Um ramo no Git é simplesmente um ponteiro para um objeto " commit". Por defeito, existe um único ramo, chamado "master".
- HEAD é um ponteiro especial para o ramo para o qual estamos a trabalhar atualmente (extraída do repositório)



Trabalho em equipa

Comandos para os ramos

Criar um ramo

```
$ git branch <ramo>
```

Ver os ramos do repositório local

```
$ git branch
```

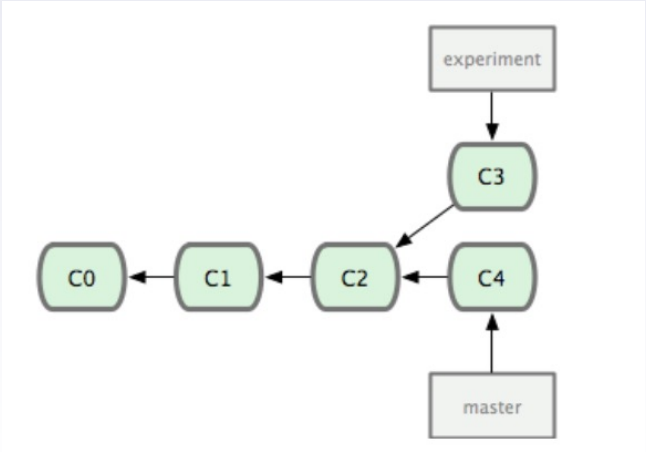
Eliminar um ramo

```
$ git branch -d <ramo>
```

Saltar par um determinado ramo

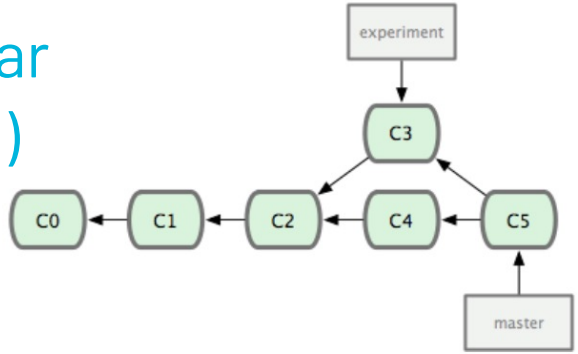
```
$ git checkout <ramo>
```

Git - ramos

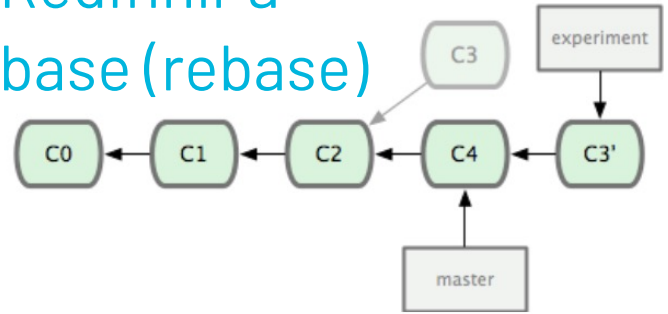


Duas formas de incluir as modificações de uma ramo num outro ramo corrente.

Fusionar (merge)



Redifinir a base (rebase)

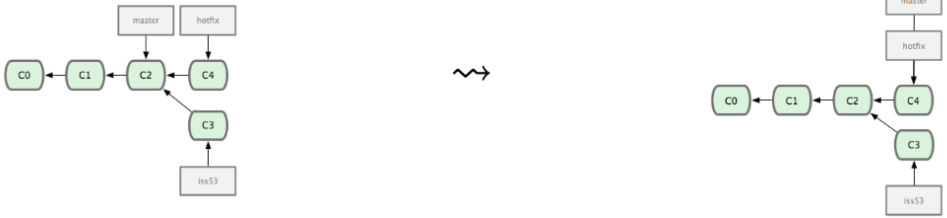


Git - ramos

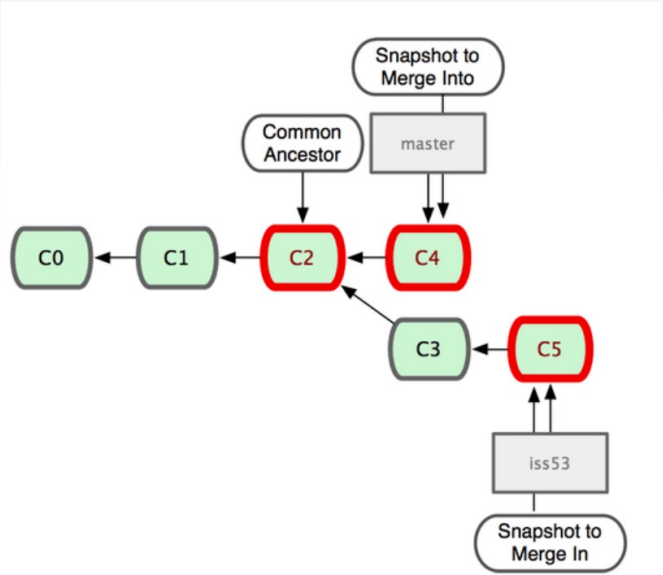
- Fusionar as modificações de uma dado ramo no ramo corrente (HEAD)

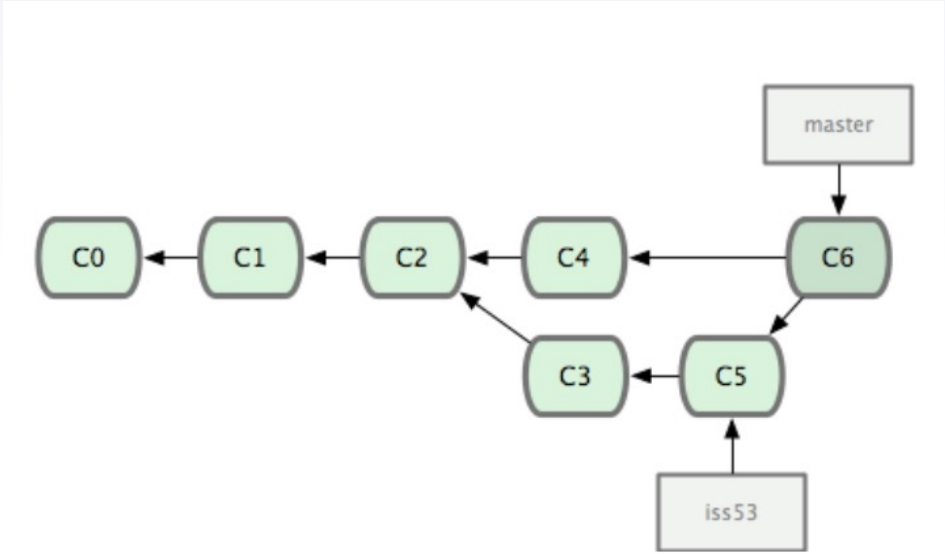
```
$ git merge <ramo>
```

  - Se o objeto "commit" apontado por <ramo> já é um **antepassado** do objeto "commit" corrente (HEAD), então nada é feito.
  - Se o objeto "commit" apontado por <ramo> já é um **descendente** do objeto "commit" corrente, só o apontador do ramo corrente é movido para o objeto "commit" abrangido pela fusão ("fast-forward").



Git - ramos


Snapshot to Merge Into
Common Ancestor
master
C0
C1
C2
C4
C3
C5
iss53
Snapshot to Merge In


master
C0
C1
C2
C4
C6
C3
C5
iss53

Git - ramos

Em caso de conflito

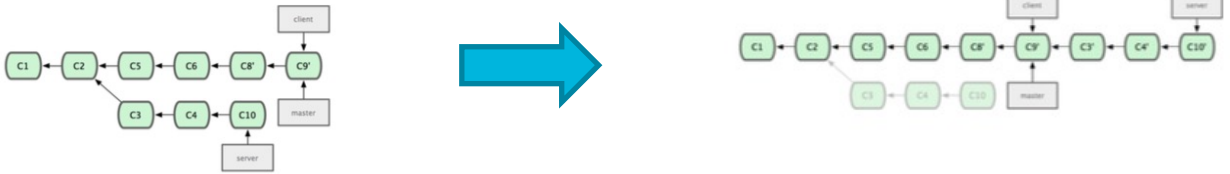
- Nenhum objeto "commit" de fusão é criado, mas o processo entra em pausa.
- **`git status`** dá os ficheiros que não puderam ser fusionados (listados como "unmerged").
- Git adiciona marcadores de resolução de conflitos à cada ficheiro sujeito a conflitos afim que possam ser resolvidos manualmente.
- Para marcar os conflitos num ficheiro **`<ficheiro>`** como resolvido, tem-se de se fazer **`git add <ficheiro>`**. Podemos, após resolução de todos os conflitos, submeter as modificações sob a forma de objeto "commit" de fusão com **`git commit`** e terminar assim o processo de fusão.

Git - ramos

Rebasear as modificações do ramo corrente (HEAD) num dado ramo

```
$ git rebase <ramo>
```

- Ir ao objeto "commit" correspondente ao antepassado mais novo comum aos dois objetos "commit" apontados pelo ramo corrente e <ramo> .
- Obter e salvaguardar as mudanças introduzidas desde este ponto para cada objeto "commit" do ramo corrente.
- Fazer apontar o ramo corrente para o mesmo objeto "commit" que <ramo> e reaplicar todas as modificações uma a uma.



Git – trabalho com repositórios remotos

Para colaborar, é necessário comunicar com um ou mais repositórios remotos alojando o mesmo projeto (tipicamente repositórios públicos associados à uma pessoa, uma equipa ou todo o projeto).

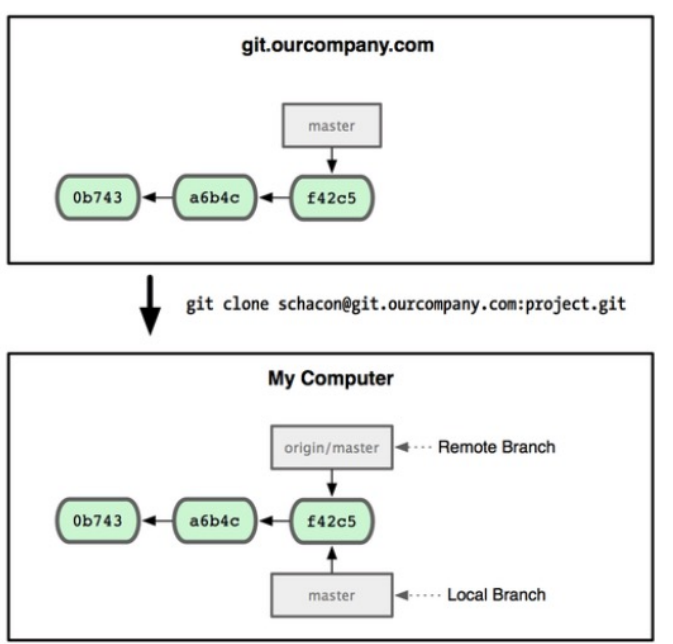
Os dados dos repositórios remotos (objetos "commits" e instantâneos) são inteiramente copiados no repositório local, e para cada ramo <ramo> de um repositório remoto <repositório> é mantida um ramo local <repositório>/<ramo> não modificável, permitindo de seguir a posição de <ramo> no <repositório> localmente.

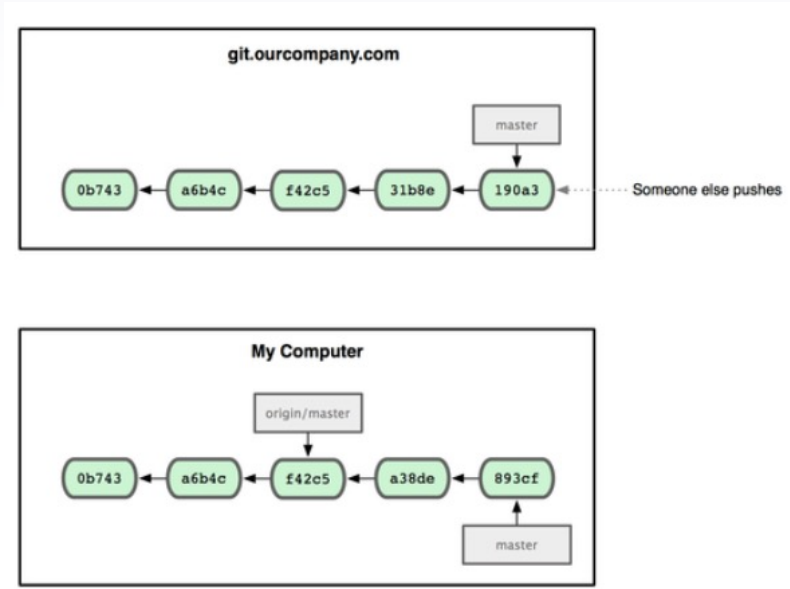
- Mostrar a lista de todos os repositórios remotos do projeto

```
$ git remote
```

Git – trabalho com repositórios remotos

- Clonar um repositório remoto `$ git clone <URL> [<repositório>]`

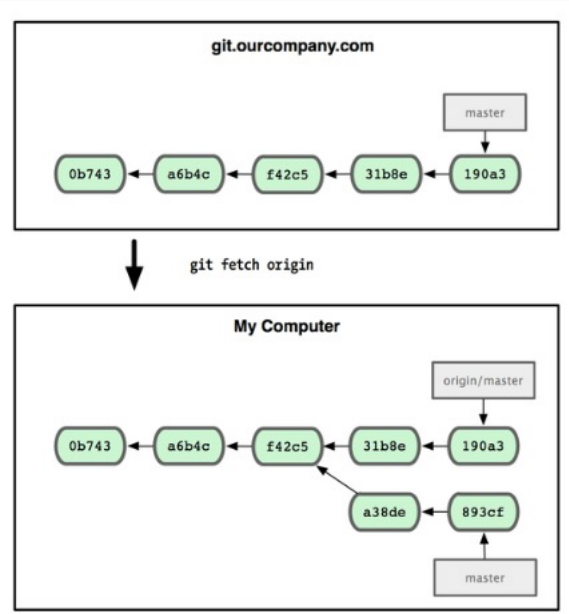




Git – trabalho com repositórios remotos

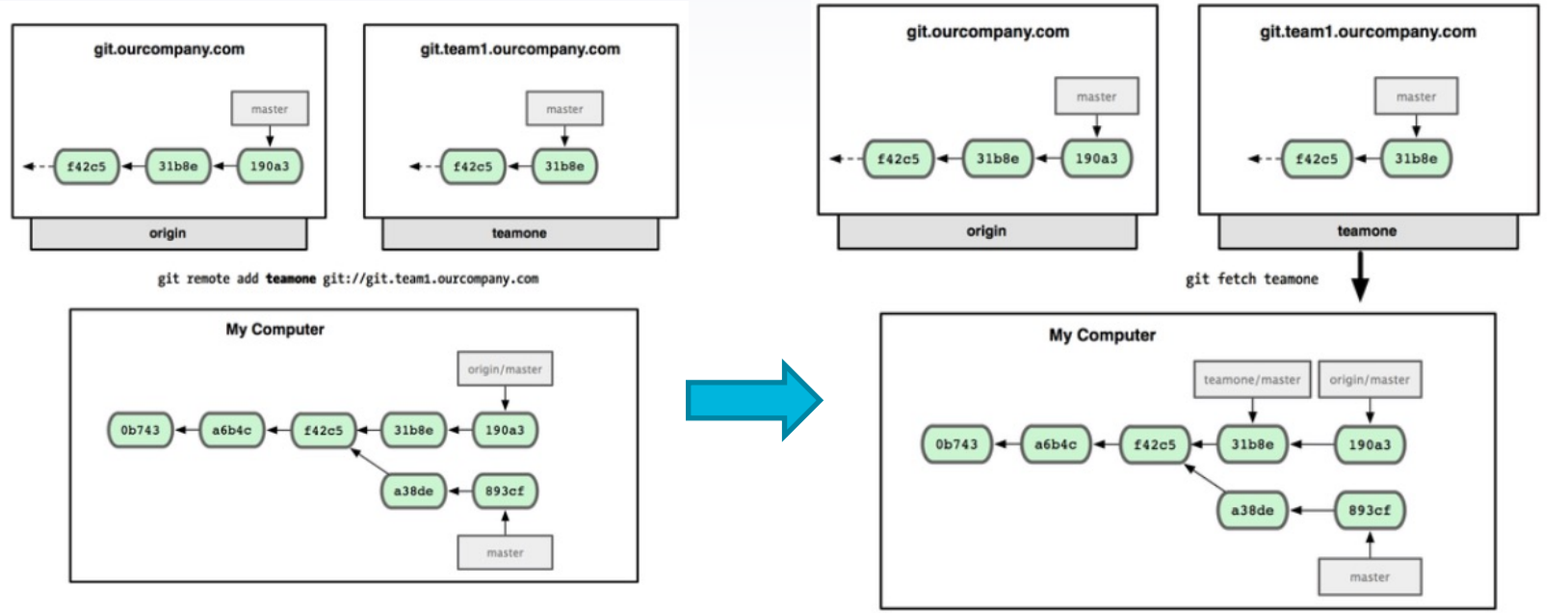
- Ir buscar as modificações de um repositório remoto

```
$ git fetch <repositório>
```



Git – trabalho com repositórios remotos

- Adicionar um repositório remoto `$ git remote add <nome> <URL>`



Git – trabalho com repositórios remotos

- Atualizar um dado repositório remoto com um dado ramo local

```
$ git push <repositorio> <ramo>
```

Cuidado: só funciona se foi-se buscar o último objeto commit do ramo abrangido do repositório remoto e tiver sido integrado no ramo do repositório local, i.e., se a atualização do ramo do repositório remoto pode ser feita pelo "fast-foward".

- Combinar git fetch e git merge

```
$ git pull <repositorio> <ramo>
```

- Combinar git fetch e git rebase

```
$ git pull --rebase <repositorio> <ramo>
```

Git – Conselhos e boas práticas

Conselhos

- Usar e abusar dos ramos
- Evitar de fazer sistematicamente um **`git pull`**, mas pensar no que é pertinente fazer (fusionar ou alterar a base)

Boas Práticas

- Nunca rebasear um ramo num repositório público
- Respeitar as convenções de formatação da mensagem de submissão
  - Título de 50 caracteres no máximo, seguido de uma linha vazia e de uma descrição detalhada com linhas de 72 caracteres no máximo, usando o presente: https://tbaggery.com/2008/04/19/a-note

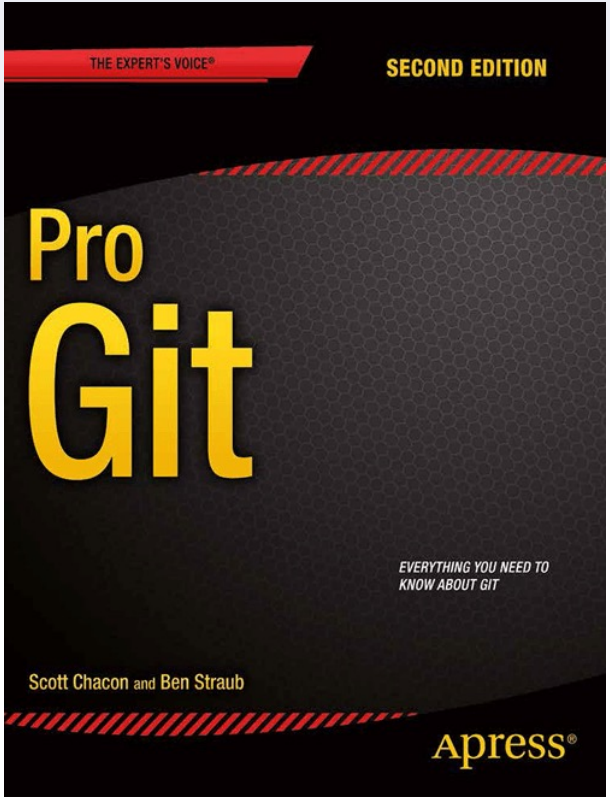
Demo

Resumo

- O que é controlo de versões e porque devemos usá-lo.
- Compreender os benefícios de um sistema de controle de versão automatizado.
- Compreender os princípios básicos de como funcionam os sistemas automatizados de controle de versão.

Bibliografia

- Pro Git, Scott Chacon & Ben Straub Apress, 2014
  - Git - Book (git-scm.com)



Glossário

- changeset / conjunto de mudanças
  - Um grupo de mudanças num ou mais ficheiros que são ou serão adicionados à um único commit num repositório de controle de versão.
- commit / confirmação
  - Como verbo, significa registar o estado atual de um conjunto de ficheiros (um changeset) num repositório de controle de versão. Como substantivo, designa o resultado da confirmação, ou seja, um conjunto de alterações registado num repositório. Se um commit contém mudanças em vários ficheiros, todas as mudanças são registadas juntas.
- conflict / conflito
  - Uma alteração feita por um utilizador de um sistema de controle de versão que é incompatível com as alterações feitas por outros utilizadores. Ajudar os utilizadores a resolver conflitos é uma das principais tarefas do controle de versão.

Glossário

- HTTP
  - O protocolo de transferência de hipertexto usado para partilhar páginas da web e outros dados na World Wide Web.
- merge / fusão
  - (aplicado à um repositório): Para reconciliar dois conjuntos de alterações num repositório.
- protocol / protocolo
  - Um conjunto de regras que definem como um computador comunica com outro. Os protocolos comuns na Internet incluem HTTP e SSH.
- remote / remoto
  - (aplicado à um repositório) Um repositório de controle de versão conectado a outro, de forma que ambos possam ser mantidos sincronizados trocando commits.

Glossário

- repository / repositório
  - Uma área de armazenamento onde um sistema de controle de versão armazena o histórico completo de commits de um projeto e informações sobre quem mudou o quê e quando.
- resolve / resolver
  - Para eliminar os conflitos entre duas ou mais alterações incompatíveis num ficheiro ou conjunto de ficheiros geridos por um sistema de controle de versão.
- revision / revisão
  - Um sinônimo para commit.

Glossário

- SHA-1
  - SHA-1 Hashes é o método que o Git usa para processar identificadores, inclusive para commits. Para processá-los, Git usa não apenas a mudança real de um commit, mas também os seus metadados (como data, autor, mensagem), incluindo os identificadores de todos os commits de mudanças anteriores. Isso torna os IDs de commit do Git virtualmente únicos. Ou seja, a probabilidade de que dois commits feitos independentemente, mesmo com a mesma mudança, recebam o mesmo ID é extremamente pequena.
- SSH
  - O protocolo **Secure Shell** usado para uma comunicação segura entre computadores.
- timestamp
  - Um registo de quando um determinado evento ocorreu.
- version control / controle de versão
  - Uma ferramenta para gerir mudanças num conjunto de ficheiros. Cada conjunto de mudanças cria um novo commit dos ficheiros; o sistema de controle de versão permite que os utilizadores recuperem commits antigos de forma