POLITÉCNICO DE LEIRIA

Cofinanciado por:

CENTRO 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

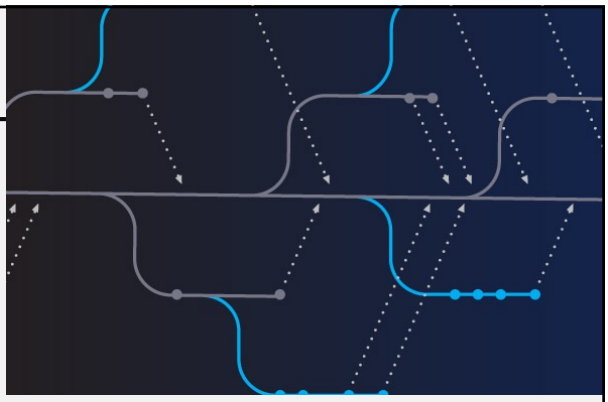
INTRODUÇÃO À ANÁLISE E PROJETO DE SISTEMAS DE INFORMAÇÃO (IAPSI)

Cap. 1

Version Control Systems

Version Control Systems **(VCS)**

O que são VCS?



Version Control Systems (VCS) consistem numa categoria de ferramentas de *software* que permite a uma equipa de software gerir alterações no código fonte ao longo do tempo. Este tipo de *software* controla todas as modificações no código num tipo especial de Base de Dados (BD). Se um erro for cometido, os programadores podem voltar atrás e comparar com as versões anteriores do código para ajudar a corrigir o erro e minimizar a perturbação a todos os membros da equipa.

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

Version Control Systems (VCS)

Benefícios

1. **Equipas trabalham mais rapidamente** e preservam a **eficiência e agilidade** à medida que a equipa cresce
2. **Histórico** completo de todos os ficheiros ao longo do tempo
3. ***Branching*** e ***merging*** – equipas trabalham em fluxos independentes de alterações
4. ***Traceability*** – rastrear cada alteração através de notas claras e objetivas

3

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

Version Control Systems (VCS)

Tipos

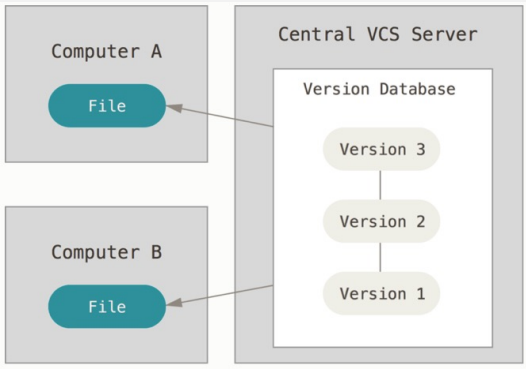
- **Perforce Version Control** – para equipas globalmente distribuídas e equipas a trabalhar em produtos complexos. Modelo centralizado.
- **Git** – para equipas mais pequenas, especialmente as que trabalham em aplicações web e móveis. Modelo distribuído e extremamente rápido.
- **Concurrent Versions System (CVS)** – existe desde os anos 90! Tem um ótimo suporte *cross-platform*.
- **Apache SubVersion (SVN)** – opção mais popular de CVS e simples de usar.
- **Team Foundation Server** – da Microsoft, baseado num modelo cliente-servidor distribuído.

4

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

Version Control Systems (VCS)

Modelo Centralizado



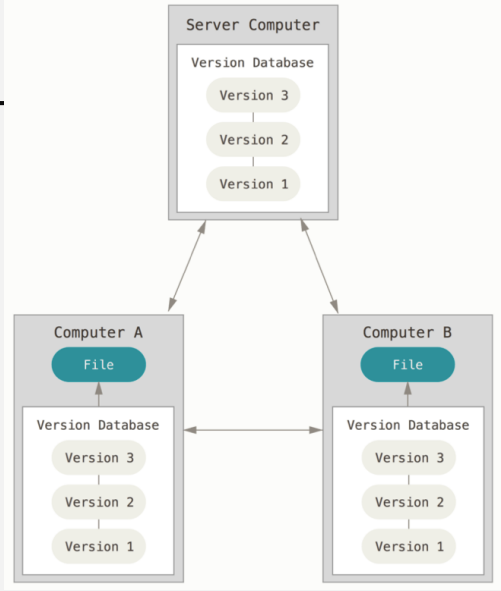
- **Centralized VCS (CVCS)** surgiram da necessidade dos programadores colaborarem com outros programadores de outros projetos
  - Exemplos: CVS, Subversion, Perforce
- Contêm um único servidor com todas as versões dos ficheiros e clientes que efetuam *check out* do servidor central
- **Vantagens:**
  - A equipa sabe sempre o que todos os elementos estão a fazer
  - Os administradores têm um grau elevado de controlo sobre quem pode fazer o quê
  - É mais fácil gerir um CVCS do que lidar com BD locais em cada cliente
- **Desvantagens:**
  - Como o servidor é central e único, corre o risco de falhar (servidor em baixo, BD corrompida, etc)

5

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

Version Control Systems (VCS)

Modelo Distribuído



- **Distributed VCS (DVCS)** surgiram da necessidade de descentralizar as versões dos ficheiros
  - Exemplos: Git, Mercurial, Bazaar, Darcs
- Os clientes não efetuam apenas o *check out* da última *snapshot* dos ficheiros. Efetuam uma cópia integral do repositório incluindo o histórico completo
- Estes sistemas lidam muito bem com facto de haver múltiplos repositórios remotos, por isso várias equipas de diferentes grupos podem colaborar dentro do mesmo projeto
- É então possível configurar diversos tipos de *workflow* que não é possível em CVCS

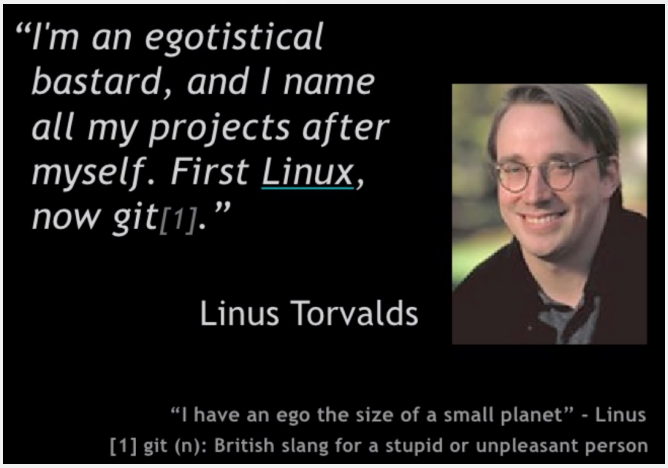
6

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Origem

- Criado em 2005 por Linus Torvalds
- O Git teve como objectivo consistir na base do controlo de versões do kernel do Linux
- O Git é um software *free* e *open-source*
- Não está ligado a nenhuma linguagem ou *framework* específica. Apenas armazena ficheiros!



7

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

O que é o Git?

- **Version Control System (VCS)** para fazer o *tracking* de alterações em ficheiros
    1. Versão de controlo distribuída
    2. Coordena o trabalho entre múltiplos programadores
    3. Regista quem fez alterações e quando
    4. Reverte as versões em qualquer altura
    5. Usa um repositório local (na máquina local) e um remoto (e.g., GitHub ou Bitbucket)

8

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Conceitos do Git

- Mantém o controlo do histórico de código
- Tira *snapshots* dos ficheiros
- O programador decide quando tirar uma *snapshot* ao fazer um *commit*
- É possível visitar uma *snapshot* em qualquer altura
- É possível preparar os ficheiros antes de efectuar *commit*

9

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Gitflow Workflow

- Define o **modelo de *strict branching*** desenhado à volta do projeto
  - Fornece uma *framework* robusta para gestão de projetos de grande dimensão
  - Ideal para projetos com ciclos de *releases*
- **Vantagens**:
  - Atribui papéis específicos a diferentes *branches* e como e quando devem interagir
  - Usa *branches* individuais para preparar, manter e gravar *releases*
  - *Pull requests*, experiências isoladas e colaboração eficiente
- Gitflow determina que tipo de ***branches*** se devem criar e como se devem juntar (***merging***)
- O Gitflow *toolset* não é mais que uma ferramenta de **linha de comandos**

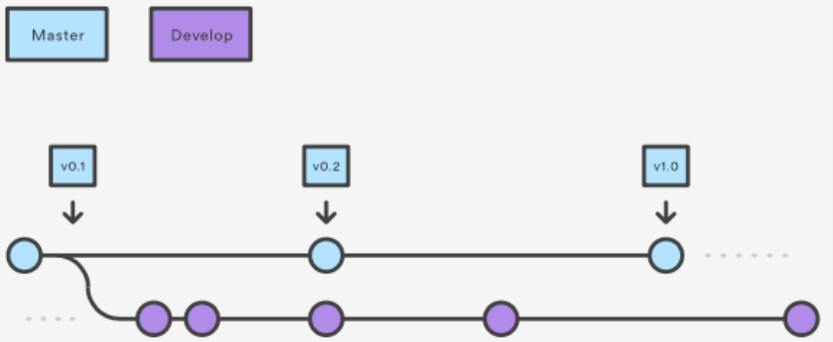
10

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Gitflow: Develop & Master Branches

- 2 *branches* para guardar o histórico do projeto:
  - **Master** – *releases* oficiais
  - **Develop** – *branch* de integração de *features*



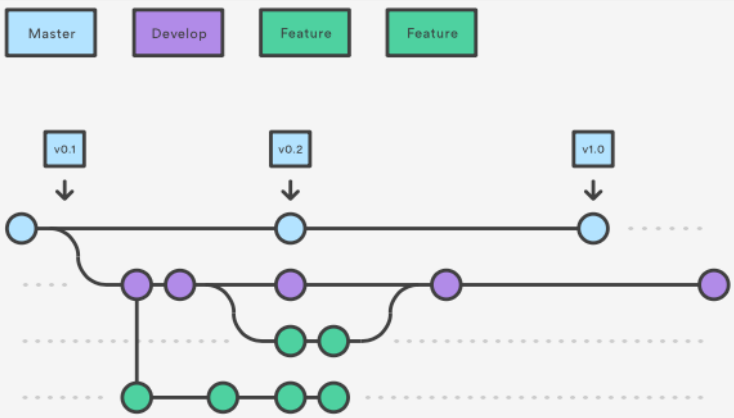
11

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Gitflow: Feature Branches

- Cada *feature* deve ser ter o seu próprio *branch* e deve ser feito o *push* para o repositório central
- O *branch* pai deve ser o `develop`
- Quando termina deve ser feito o *merge* com o *branch* `develop`



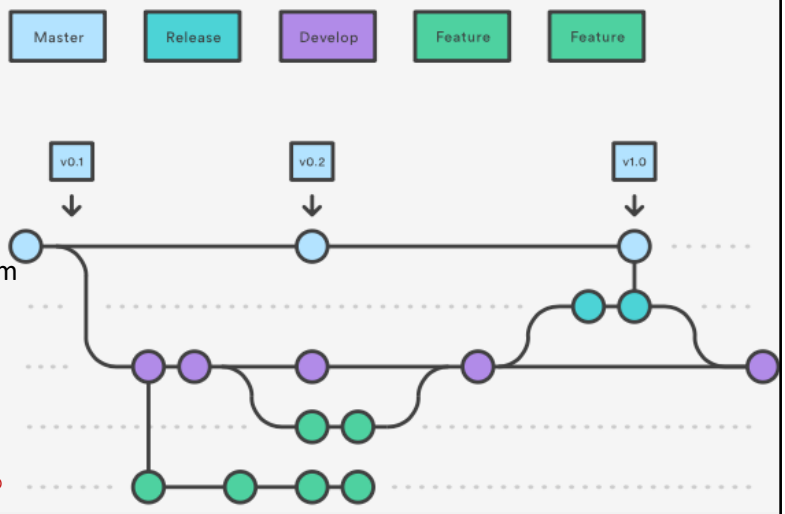
12

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Gitflow: Release Branches

- O *branch* `develop` já contém as funcionalidades todas da *release*
- Faz-se um ***fork*** (uma separação) do *branch* `develop`
- No *branch* `release` apenas podem ser feitos: bug fixes, criação de documentação, e pouco mais
- *Feature is ready to ship!!!*
- *Merge* com o `master` e `develop`



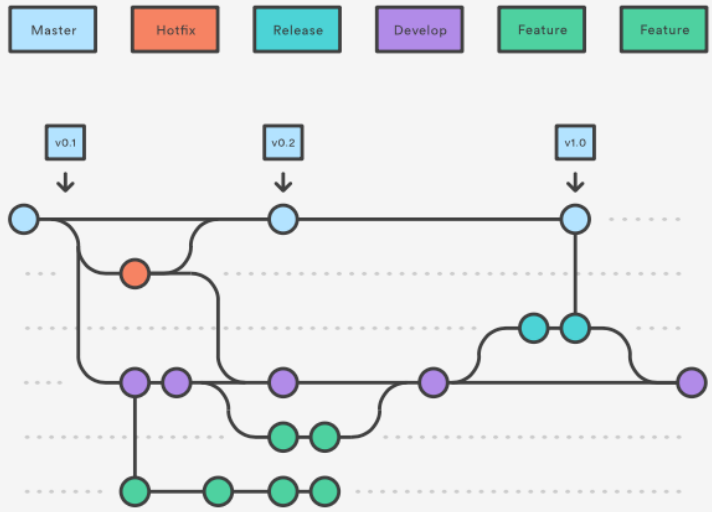
13

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Gitflow: Hotfix Branches

- *Branches* de manutenção/hotfix são usados para corrigir rapidamente releases em produção
- Os *branches* `hotfix` são baseados no `master`
- Assim que o *fix* esteja pronto deve ser feito o *merge* com o `master` e com o `develop` (ou o actual *branch* `release`) e ser etiquetado com um novo número de versão



14

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Gitflow: *Summary*

1. *Branch* `develop` criado a partir do `master`
2. *Branch* `release` criado a partir do `develop`
3. Os *branches* `feature` são criados do *branch* `develop`
4. Quando um `feature` está completo é feito o *merge* com o *branch* `develop`
5. Quando o *branch* `release` está pronto é feito o *merge* com o *branch* `develop` e `master`
6. Se é detectada uma *issue* no `master`, é criado um *branch* `hotfix` a partir dos `master`
7. Assim que o `hotfix` está completo, é feito o *merge* com os *branches* `master` e `develop`

15

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Comando básicos do Git

Comando	Descrição
`$ git init`	Inicializa o repositório local do Git
`$ git add <file>`	Adiciona ficheiro(s) ao *index* (*staging area*)
`$ git status`	Verifica o estado da árvore de trabalho (*working tree*)
`$ git commit`	Efectua o *commit* dos ficheiros do *index*
`$ git push`	*Push* para o repositório remoto
`$ git pull`	*Pull* das últimas alterações do repositório remoto
`$ git clone`	Cria um clone do repositório remoto para uma nova directoria
`$ git branch`	Cria um novo *branch*
`$ git checkout <branch>`	Muda para o *branch* referido
`$ git merge <branch>`	Merge do *branch* actual e do *branch* referido

16

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

DEMO: Visualização da abstracção do git flow

- http://git-school.github.io/visualizing-git/

DEMO

17

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

O que é o GitHub?

O **GitHub** é um website e um ***cloud-based Git repository hosting service*** que ajuda os programadores a armazenar e gerir o seu código, além de monitorizar e controlar alterações em código.

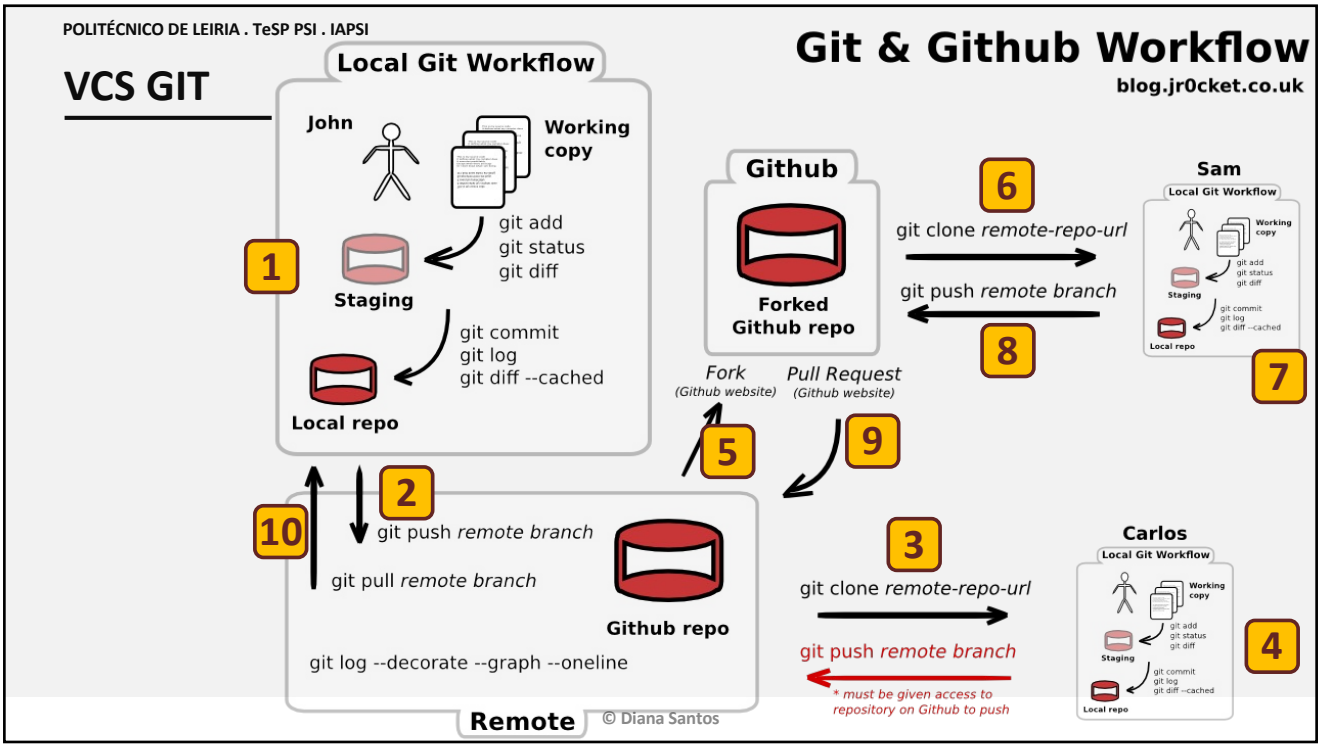
- ✓ ***Version control*** – os programadores trabalham de forma segura através de ***branching*** e ***merging***
- ✓ **Git** – é um VCS distribuído e *open-source*

- Facilita o uso do Git para versão de controlo e colaboração entre equipas
- É possível criar uma conta gratuita e fazer *host* de um repositório público de código
- Tutorial Hello World: https://guides.github.com/activities/hello-world/

18

POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT



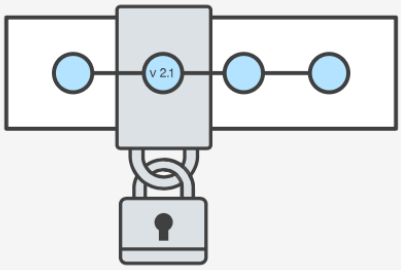
POLITÉCNICO DE LEIRIA . TeSP PSI . IAPSI

VCS GIT

Instalação do Git

- **Linux** (Debian)
  ```
  $ sudo apt-get install git
  ```
- **Linux** (Fedora)
  ```
  $ sudo yum install git
  ```
- **Mac**
  https://git-scm.com/download/mac
- **Windows**
  https://git-scm.com/download/win

Bibliografia



- Git
  - https://git-scm.com
- GitHub
  - https://github.com
- Livro Pro Git
  - https://git-scm.com/book/en/v2
- Git Tutorial Manual Page (comandos)
  - http://schacon.github.io/git/gittutorial.html
  - https://gist.github.com/leocomelli/2545add34e4fec21ec16
- Git & GitHub Crash Course for Begginers
  - https://www.youtube.com/watch?v=SWYqp7iY_Tc
- Git Tutorial for Begginers: Command-Line Fundamentals
  - https://www.youtube.com/watch?v=HVsySz-h9r4&t=1s
- Atlassian Bitbucket Beginner Tutorial
  - https://www.atlassian.com/git/tutorials/what-is-version-control