P&O - git

Marco Patrignani

KU Leuven

Outline

These slides

http://people.cs.kuleuven.be/~marco.patrignani/Research.html
Teaching (scroll down) → 2014-2015 → slides on git

Who uses git?

- Linux http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git

Who uses git?

- Linux http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git
- Google https://github.com/google
- Facebook https://github.com/facebook
- Microsoft http://aspnetwebstack.codeplex.com/
- Twitter https://github.com/twitter
- linkedin https://github.com/linkedin
- Android https://android-review.googlesource.com/#/q/status:open,n,z
- ...

What is git?

git [wikipedia]

Git is a distributed revision control and source code management (SCM) system.

What is git?

git [wikipedia]

Git is a distributed revision control and source code management (SCM) system.

Distributed revision control system [wikipedia]

[a program that] keeps track of software revisions and allows many developers to work on a given project without requiring that they maintain a connection to a common network.

git in a nutshell

- create your repo (where the code is located)

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)
- manage the repo by adding files/directories/binaries etc

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)
- manage the repo by adding files/directories/binaries etc
- when a file is added, its history is recorded by git (this is *versioning*)

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)
- manage the repo by adding files/directories/binaries etc
- when a file is added, its history is recorded by git (this is *versioning*)
- commit changes

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)
- manage the repo by adding files/directories/binaries etc
- when a file is added, its history is recorded by git (this is *versioning*)
- commit changes
- commits describe the changes to the repo and who did them

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)
- manage the repo by adding files/directories/binaries etc
- when a file is added, its history is recorded by git (this is *versioning*)
- commit changes
- commits describe the changes to the repo and who did them
- create new features in branches that do not affect the main project

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)
- manage the repo by adding files/directories/binaries etc
- when a file is added, its history is recorded by git (this is *versioning*)
- commit changes
- commits describe the changes to the repo and who did them
- create new features in branches that do not affect the main project
- branches represent independent lines of development that can be merged in the main line

git in a nutshell

- create your repo (where the code is located)
- a repo is a folder (with hidden files for git to manage)
- manage the repo by adding files/directories/binaries etc
- when a file is added, its history is recorded by git (this is *versioning*)
- commit changes
- commits describe the changes to the repo and who did them
- create new features in branches that do not affect the main project
- branches represent independent lines of development that can be merged in the main line
- maintain a *remote* repo to share access to it with *multiple* programmers

Why using git?

- easy to share code among developers

Why using git?

- easy to share code among developers
- cannot lose code by accident

Why using git?

- easy to share code among developers
- cannot lose code by accident
- keeps the history of code

Why using git?

- easy to share code among developers
- cannot lose code by accident
- keeps the history of code
- allows one to maintain different versions of code, to compare and merge them

Why using git?

- easy to share code among developers
- cannot lose code by accident
- keeps the history of code
- allows one to maintain different versions of code, to compare and merge them
- that's what **good** software developers do

How to use git

- https://www.atlassian.com/git/tutorial *(expected in the first report)*
- https://www.atlassian.com/git/workflows (centralised and the feature branch workflows are recommended)

Useful software

- http://sourcetreeapp.com/ for Mac and Win
- http://www.syntevo.com/smartgit/download for Mac, Win and Linux
- command line
- list of all laptop apps http://git-scm.com/downloads/guis
- mobile apps https://play.google.com/store/apps/details?id=com.saibotd.bitbeaker

git commands

- `git init` – turns the current directory into a repo

git commands

- `git init` – turns the current directory into a repo
- `git clone <repo>` – Clone the repository located at `<repo>` in the current directory (this is what you'll do)

git commands

- `git add <f>` – Adds file `f` (or directory `f`) to the repo (not added until commit is done)

git commands

- `git add <f>` – Adds file `f` (or directory `f`) to the repo (not added until commit is done)
- `git commit -m "<message>"` – Commits changes to the repo

git commands

- `git add <f>` – Adds file `f` (or directory `f`) to the repo (not added until commit is done)
- `git commit -m "<message>"` – Commits changes to the repo
- `git revert <commit>` – Undo all of the changes introduced in `<commit>`

git commands

- `git branch <branch>` – Create a new branch called `<branch>`

git commands

- `git branch <branch>` – Create a new branch called `<branch>`
- `git checkout <existing-branch>` – This makes `<existing-branch>` the current branch (watch out for **detached HEADs**)

git commands

- `git branch <branch>` – Create a new branch called `<branch>`
- `git checkout <existing-branch>` – This makes `<existing-branch>` the current branch (watch out for **detached HEADs**)
- `git merge <branch>` – Merge the specified branch into the current branch

git ignore

- create a file called `.gitignore`
- add regular expressions for files **NOT** to include in the repo
- for example `*.o *.a`

The workflow

The workflow

centralised Central repo. Each contributor clones it, edits her local copy and pushes his local changes remotely. Before pushing, she pulls for other people's pushes and merges eventual conflicts.

The workflow

centralised Central repo. Each contributor clones it, edits her local copy and pushes his local changes remotely. Before pushing, she pulls for other people's pushes and merges eventual conflicts.

feature-branch Like above, but a new feature is started in a different branch (to avoid conflicts). Once completed, it is merged in the master. Other contributors see the branch but can ignore it.

P&O specifics

- register on https://bitbucket.org/ with your **`student.kuleuven`** account
- check your plan in
  `Manage account` → `plan details` → `change plan`
- upgrade to academic license
  https://www.atlassian.com/software/views/bitbucket-academic-license.jsp
- create a **Team** called `KULPn014-15_`*thenameofyourteam*.
  E.g. `KULPn014-15_zilver`
- add your group supervisors to the team *with Administrator privileges*