EN
English

Universidad de Oviedo

School of Computer Science

SOFTWARE ARCHITECTURE

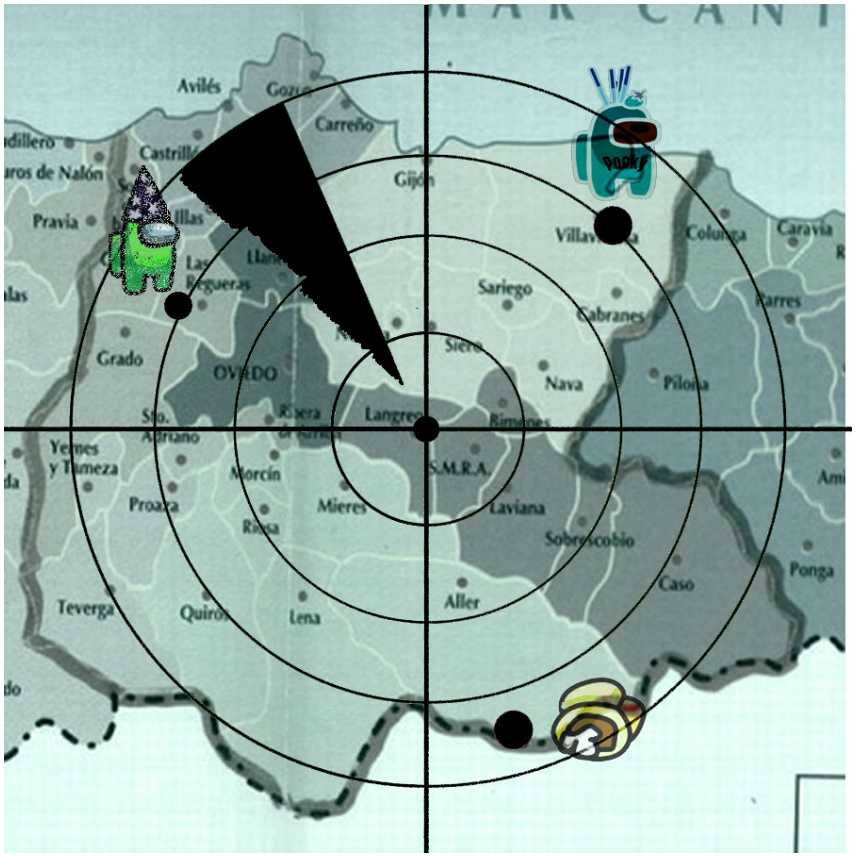
Software Architecture

- ✓ Lab. 1
- ✓ Introduction to Labs
- ✓ Teams Organization
- ✓ Git

2020-21

Jose Emilio Labra Gayo
Pablo González
Irene Cid
Paulino Álvarez

Introduction to Labs- Radarin



- What are we going to do in these sessions?

Radarin

- Grades:
  - 70% - Group work
  - 30% - Individual work

Ressources

- Software Architecture course website
- Campus virtual.
- Radarin. Specification
- Github repositories for the project

Teams organization

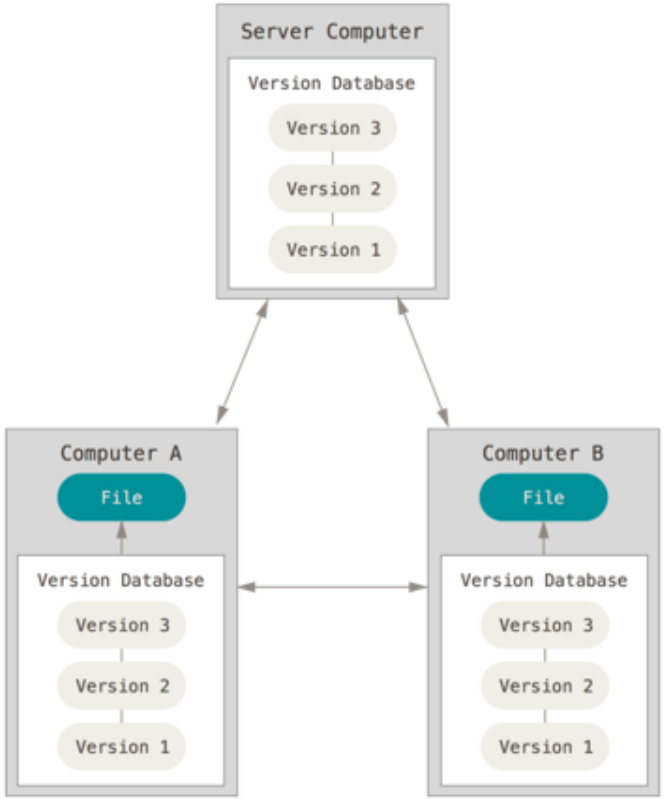
- Github accounts
- Meetings, wiki and issues in Github
  - Lab session = meeting
  - It is mandatory to write the minutes for each meeting
    - What to include
      - List of assistants
      - Revision of assigments of previous session (links to issues and pull requests)
      - Agreements
      - Work assigment for next session

What is Git

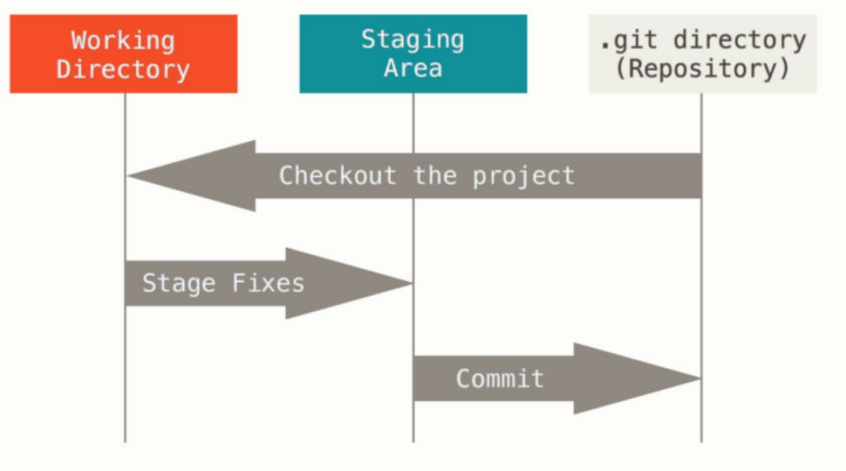
Why do we need a version control system like Git?

- We are working in a team
- We need to keep track of changes
- Security

Git structure



Git local work



SOURCE https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git

We get repositories files in local: checkout o pull

We add changes to our stage repository: add

We confirm changes and generate a snapshot : commit

Initial steps with Git & Github (1)

- Each team will have a Git repository:
  - Let's assign each one of you to your repository
- For this session you will clone the repository to your local machine
  - Step 1: Let's init our local repository
    - >>git init
  - Step 2: We'll retrieve content from remote
    - >> git clone https://github.com/Arquisoft/radarin_xxx.git
  - Step 3:We make changes
  - Step 4:We add changes in our stage repository

    >> git add .

Initial steps with Git & Github(2)

- Step 5:We confirm changes

  >> git commit –m “my first commit”

- Step 6: We sent changes to remote

  >> git push origin master

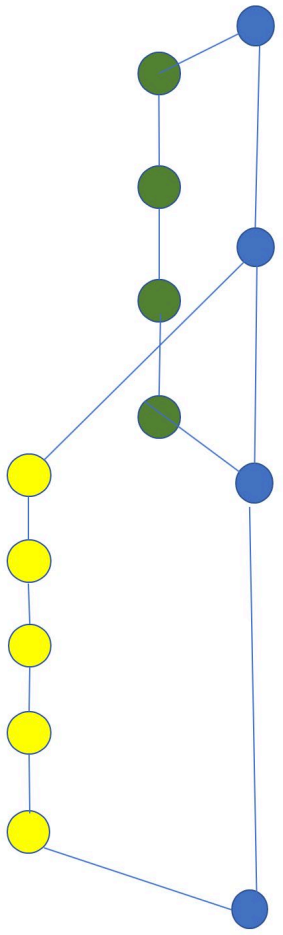
- More info:
  - Git cheatsheet with the most used commands
  - Slides: Introduction to git

    https://www.slideshare.net/jelabra/introduction-to-git-44244608

Working with branches in Git

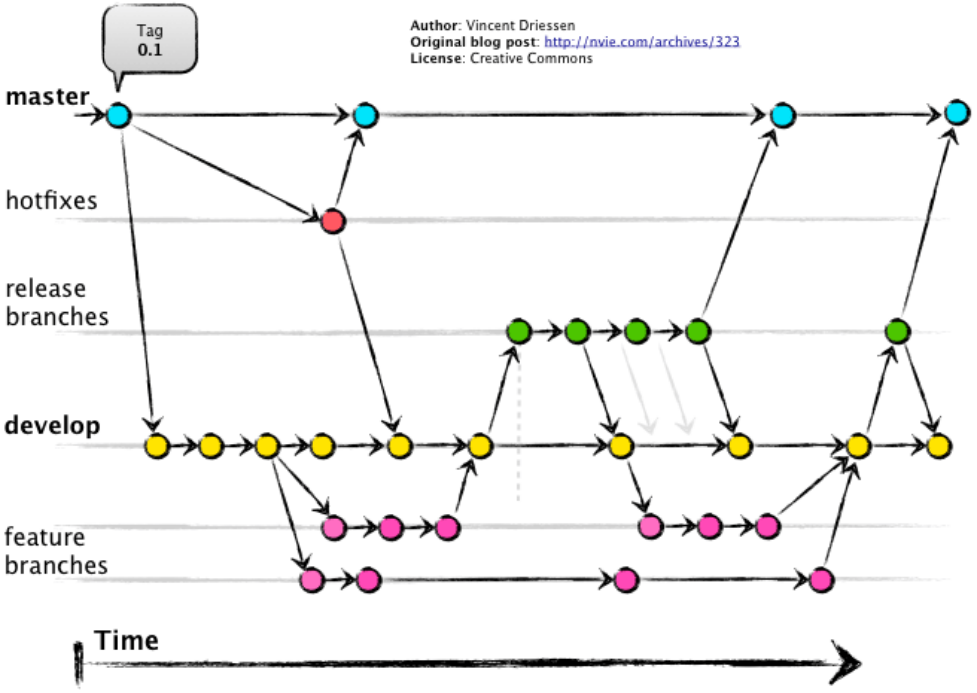
- Create a branch:
  ```
  $ git checkout -b branch1
  ```
- Check our current branch:
  ```
  $ git branch
  ```
- Change to another branch:
  ```
  $ git checkout master
  ```
- See differences from branchs
  ```
  $ git diff --stat master branch1
  ```
- Merge branch:
  ```
  $ git checkout master
  $ git merge --no-ff branch1
  ```
- Remove branch:
  ```
  $ git branch -d branch1
  ```

- Step 1: Let's create remote develop branch
  - >> git checkout -b develop
  - >> git push origin develop

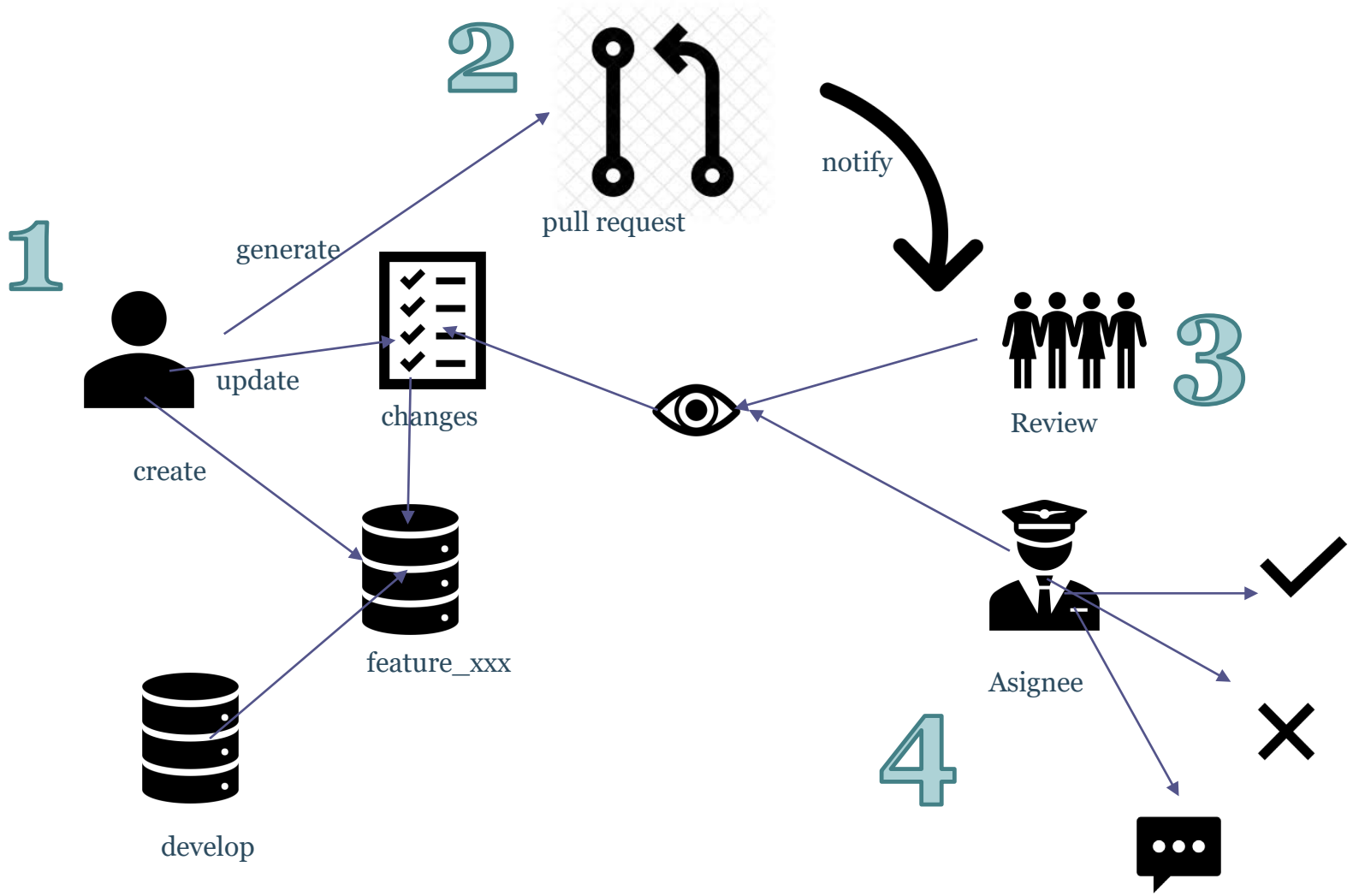


Git-Flow

- How is daily work in a development Team: iteration, evolutive, solving critical issues.
- Our repository should be configured to work in this way. So that , Git-flow stablish this branch hierarchy :

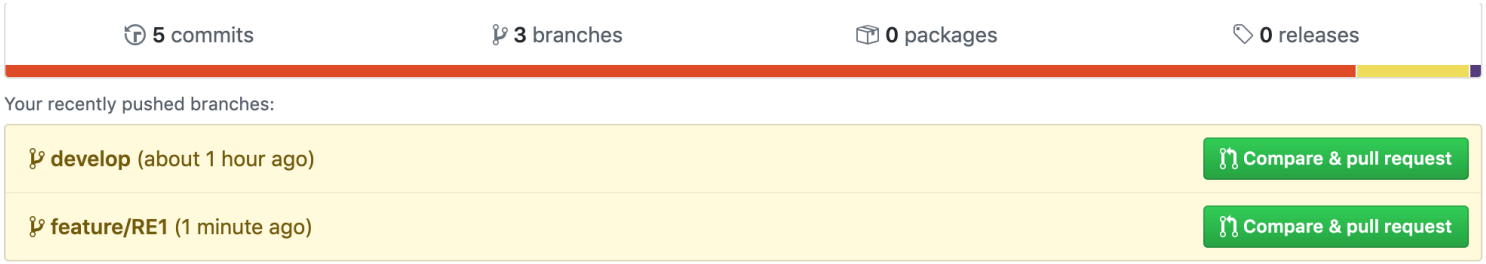


Pull request



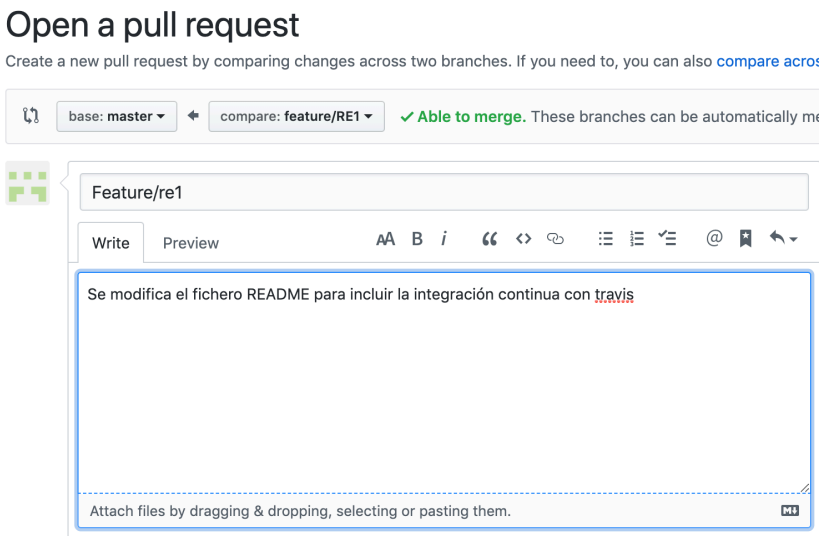
Pull request - Steps

- New branch
  - `$ git-flow feature start RE1 develop`
  - `$ git checkout -b feature-RE1 develop`
- Add your name inside the *Collaborator* section in your *README.md* file
- Send your local changes
  - `$ git add .`
  - `$ git commit`
- Send your change to remote
  - `$ git push --set-upstream origin feature-RE1`
- Go to github and ask for a pull request



Pull request - Steps

- Add comments: (tambien se pueden añadir reviewers)



- Code reviewers can add comments, accept changes, reject