



02

Chapter

Distributed Version Control

The GitLab example

2I1AC3 : Génie logiciel et Patrons de conception

Régis Clouard, ENSICAEN - GREYC

“Il est facile se tirer une balle dans le pied avec Git, mais il est également facile de revenir à un pied précédent et de le fusionner avec sa jambe actuelle.”

Jack William Bell

Recall: Git objectives

2

- 1) For tracking and managing changes to files (lecture 1)
 - `add + commit + checkout`
- 2) For managing multiple versions of the same project (lecture 1)
 - `branch+checkout+merge`
- 3) For sharing files (lecture 2)
- 4) For organizing collaborative work (lecture 2)

Outline

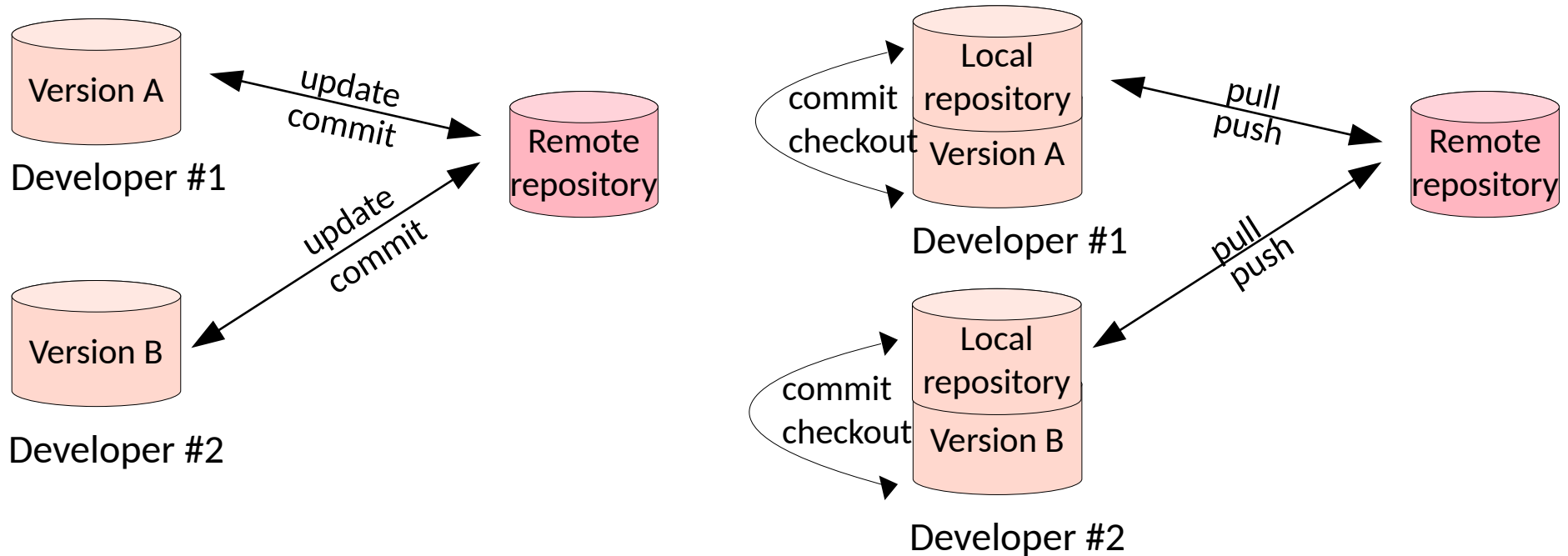
3

1
Centralized
Versus
Distributed
Version Control

Distributed vs Centralized Version Control

4

- Git is a distributed Version Control System
 - Each contributor has a complete and distinct copy of the repository
 - Can be used offline



Setting Up the Communication Protocol

5

■ https protocol

- Needs to enter the password for each writing transaction
- **Hint:** cache the password (in memory)

```
$ git config --global credential.helper 'cache --timeout=7200' # 2 hours
```

■ Git protocol

- Needs SSH keys

Outline

6

1

Centralized
Versus
Distributed
Version Control

2

Remote Repository
for sharing files (3)

Create Remote Repository

7

- On a server accessible by way of URL
(e.g. *https://www.ecole.ensicaen.fr/~rclouard/*)
 - Create a remote repository (*bare*: without working directory)

```
$ git init --bare my_repo.git  
$ cd my_repo.git  
$ git update-server-info  
$ cd hooks  
$ mv post-update.sample post-update
```

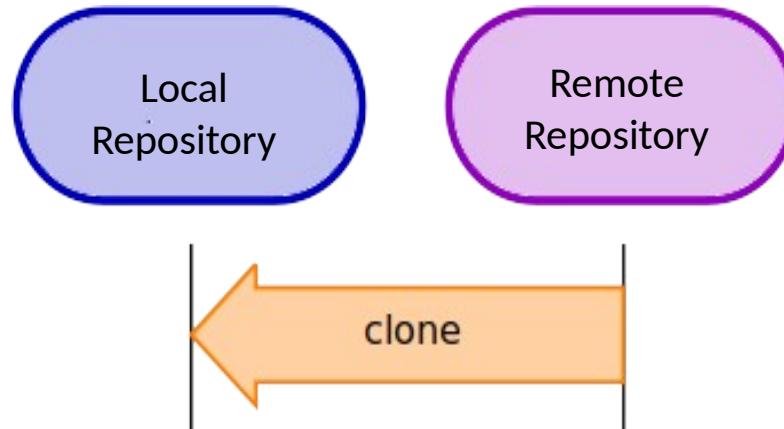
Local Repository

8

- Get a copy of the remote repository

```
$ git clone https://www.ecole.ensicaen.fr/~rclouard/my_repo.git  
$ cd my_repo
```

- Remote repository is named “*origin*” by default



Local Repository ↔ Remote Repository

9

■ Fetch the repository:

- update local repository from remote repository (all branches) but not working directory

```
$ git fetch
```

■ Pull the version from the remote repository:

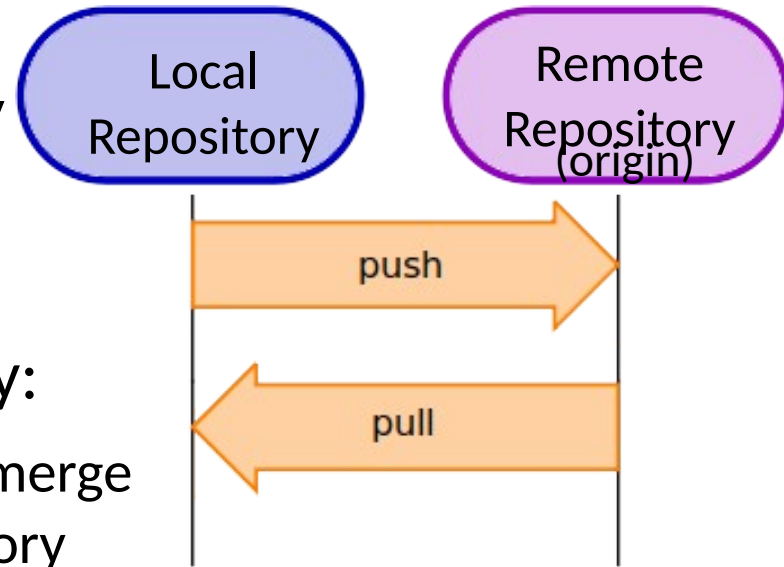
- Update the local repository (all branches) and merge the main branch into the current branch directory

```
$ git pull origin main
```

- *pull is equivalent to fetch + merge main branch into the current branch*

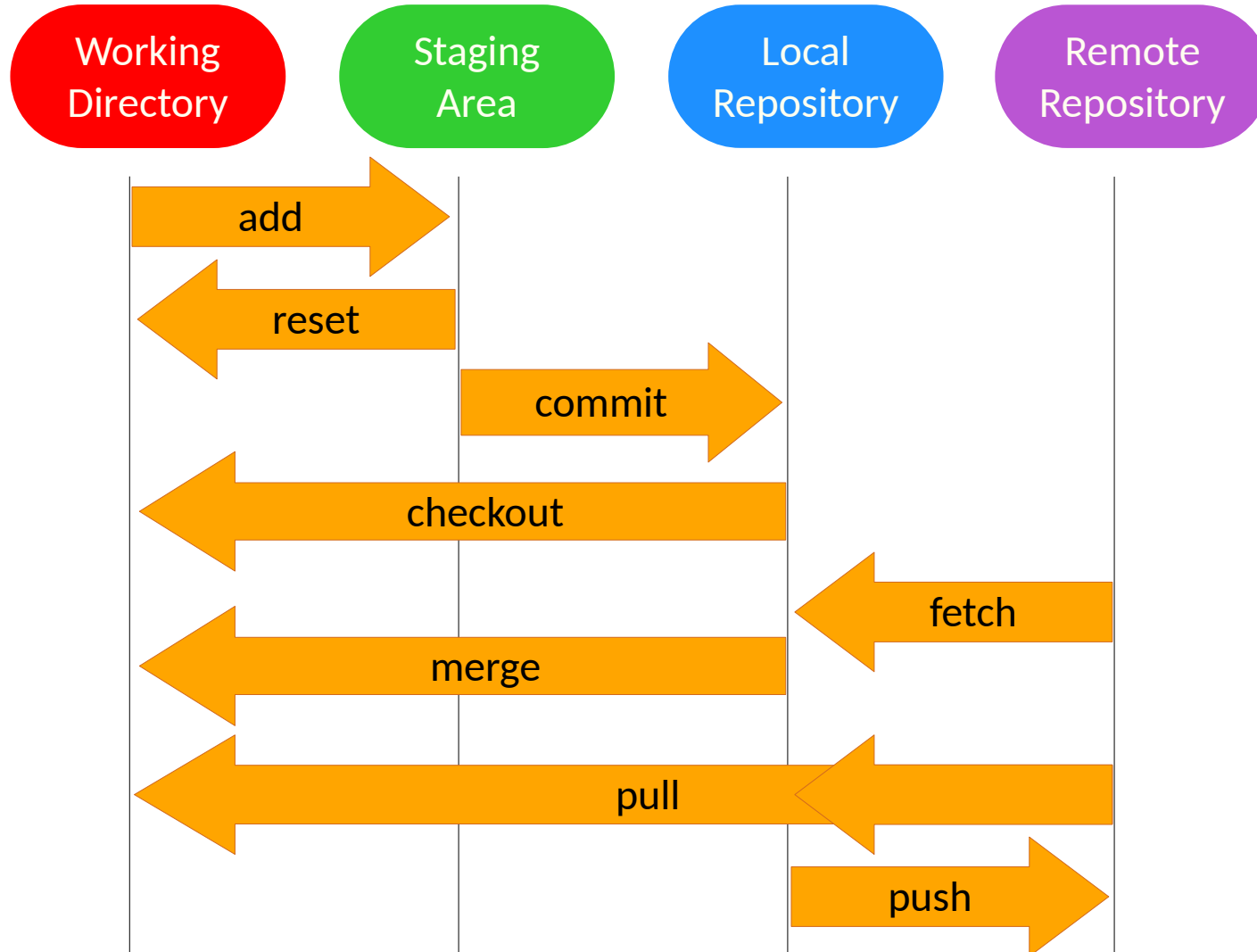
■ Push the current branch of the local repository to the remote repository

```
$ git push
```



Decentralized Version Control with Git

10



Outline

11

1
Centralized
Versus
Distributed
Version Control

2
Remote
Repository
for sharing files (3)

3
The GitLab
Example

Remote Repository with GitLab

12

- GitLab is a free web-based collaborative software platform (*aka forge*) built on top of Git
 - ENSICAEN hosts a local version
 - ▶ `https://gitlab.ecole.ensicaen.fr`

GitLab Features: (1) Repository

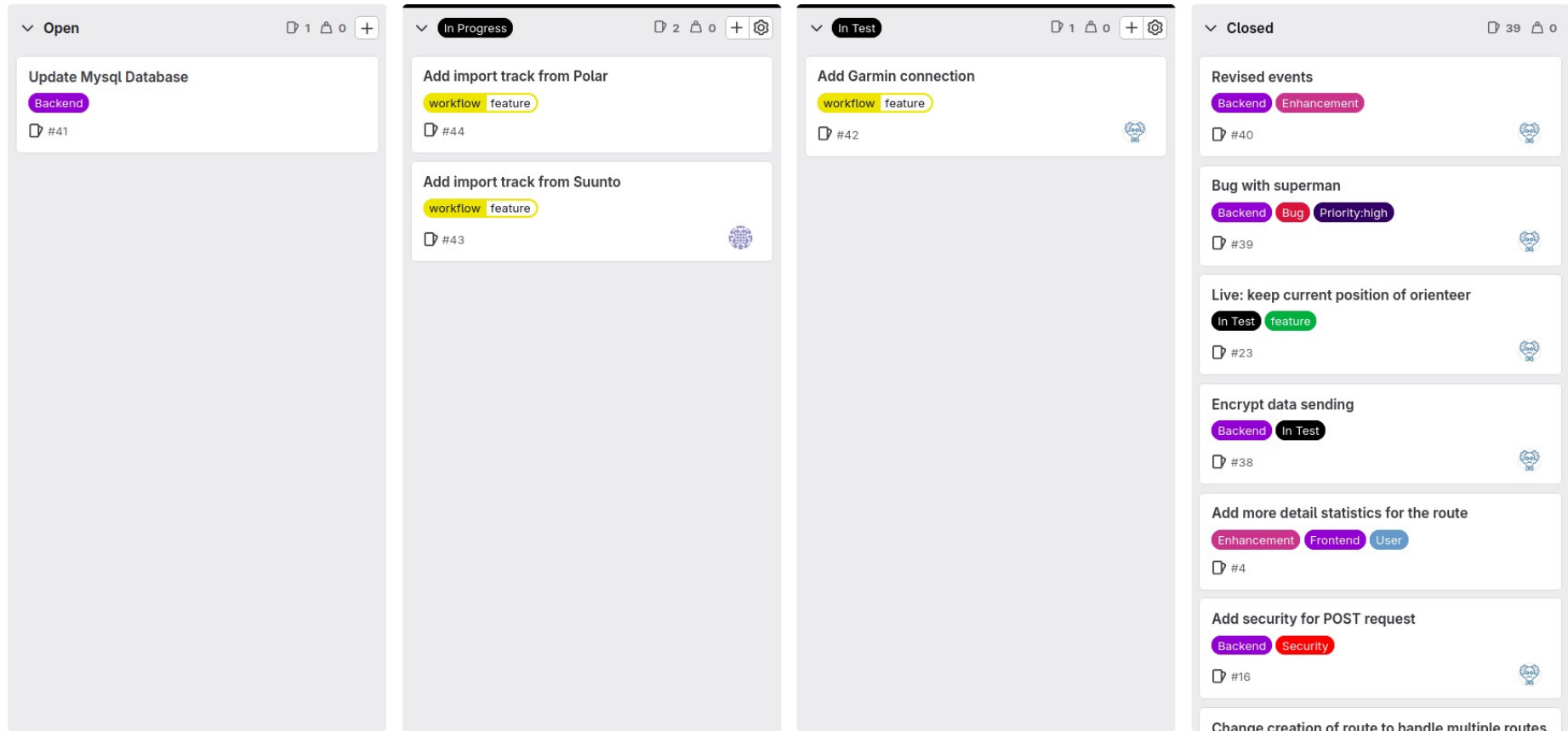
13

- Git Repository
 - Managed by “maintainer” (*aka “version manager”*)
 - ▶ modify the “main” branch
 - ▶ add / remove developers from the project

GitLab Features: (2) Kanban Board

14

- Kanban board based on issues
 - Issue: number + name + labels + assignment + comments



Merge GitLab Features: (3) Request

15

- Merge branches
- Run CI/CD

The screenshot displays a GitLab merge request interface. The left sidebar contains a navigation menu with options like 'Project', 'Merge requests', 'Repository', and 'What's new'. The main content area shows the merge request details for 'release 0.11.1', including a status bar indicating it is 'Open' and 'Regis requested to merge production into main'. Below this, there are tabs for 'Overview', 'Commits', 'Pipelines', and 'Changes'. The 'Overview' tab is active, showing a 'Pipeline #1991645328 passed' status and a 'Ready to merge!' section with options to 'Squash commits' or 'Edit commit message'. At the bottom, there is an 'Activity' section with a text input for comments.

release 0.11.1

Open Regis requested to merge `production` into `main` 3 days ago

Overview 0 Commits 1 Pipelines 1 Changes 1

0 0

✓ Pipeline #1991645328 passed
Pipeline passed for `bc95660a` on `production` 3 days ago
Test coverage 50.00% (0.00%) from 1 job

8v Approve Approval is optional

✓ Ready to merge!

☐ Squash commits ☐ Edit commit message
• 1 commit and 1 merge commit will be added to `main`.

Merge

Activity

Preview B I S L E < > P L R T M E U P

Write a comment or drag your files here...

GitLab Features: (4) CI/CD tool

16

- CI/CD: continuously release changes to end environment
 - **CI: Continuous Integration**
 - ▶ Build + Test verification
 - **CD:**
 - ▶ **Continuous Deployment:** deploy automatically (deployment)
 - ▶ **Continuous Delivery:** deploy manually: someone decides when to release
- Pipelines
 - **Pipelines** are made up of jobs
 - **Jobs** are code executed with **runner**. For example, a job can compile or test code
 - **Runners** are docker containers
 - ▶ Use dedicated image (Docker container)

GitLab Features: (4) CI/CD tool

17

■ Script file: *.gitlab-ci.yml*

```
image: gradle:jdk16

variables:
  # Using a fresh runtime for each build to isolate each runtime from any previous builds.
  GRADLE_OPTS: "-Dorg.gradle.daemon=false"
  # Variable for the runner at ENSICAEN
  http_proxy: http://193.49.200.22:3128

before_script:
  - chmod +x gradlew
  - mv gradle.properties.ci gradle.properties
  - apt-get install findutils

unit_test:
  tags :
    - gitlab-runner-ensicaen
  stage: test
  coverage: '/Coverage Total: ([0-9]{1,3})%/'
  only:
    - main
  script:
    - ./gradlew -Pci --console=plain test jacocoTestReport
    - cat build/jacocoHtml/index.html | grep -o 'Total[^%]*%' | sed 's/<.*>/ /; s/Total/Coverage Total:/'

Build: ...

Deploy: ...
```