# Git & GitHub

## QUICK INTRODUCTION

Introduction to Git as a version control system:
concepts, main features and practical aspects.

Luigi De Russis and Fulvio Corno

POLITECNICO
DI TORINO

e-Lite

# Goal

- What is Revision Control?
- What is Git?
- What is GitHub?
- How to access Revision Control with Git and GitHub from within Eclipse?
- What are the Eclipse workflows useful in this course?

# Version Control Systems

Record changes to a file or a set of files over time so that you can recall specific versions later

Three generations:

1. Local (RCS, SCCS)
2. Centralized (CVS, Subversion, Team Foundation Server)
3. Distributed (Git, Mercurial)

NOW

# Basic Concepts


**Repository**

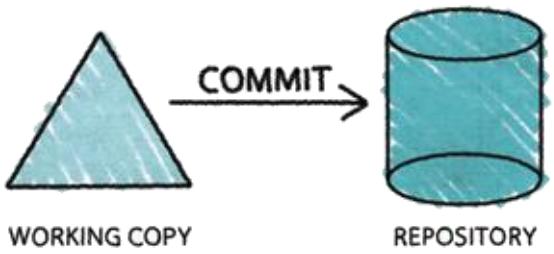
- place where you store all your work
- contains every version of your work that has ever existed
  - files
  - directories layout
  - history
- can be shared with the whole team

# Basic Concepts


**Working copy**

- a snapshot of the repository used for… working
- the place where changes happens
- private, not shared with the team
- it also contains some metadata so that it can keep track of the state of things
  - has a file been modified?
  - is this file new?
  - has a file been deleted?

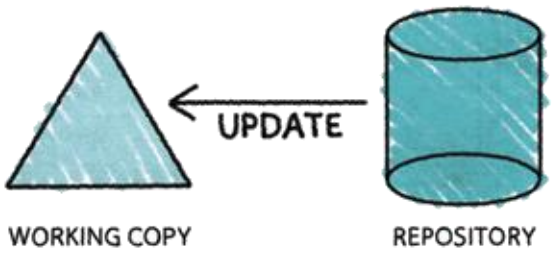
# Basic Concepts




**Commit**

- the operation that modifies the repository
- atomically performed by modern version control tools
  - the integrity of the repository is ensured
- it is typical to provide a log message (or comment) when you commit
  - to explain the changes you have made
  - the message becomes part of the history of the repository

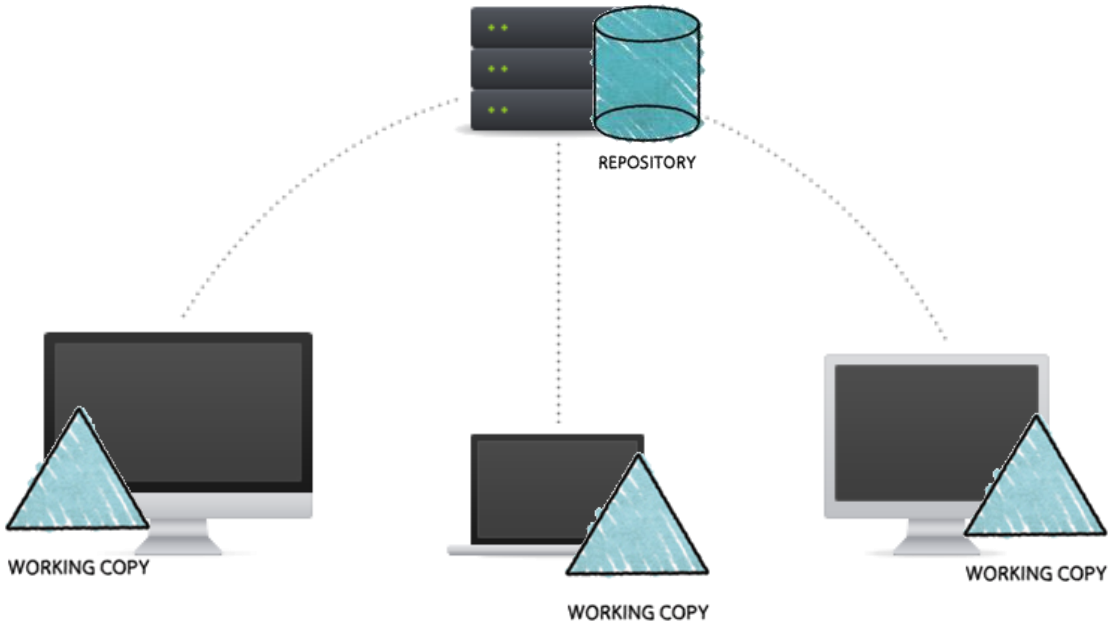
# Basic Concepts




**Update**

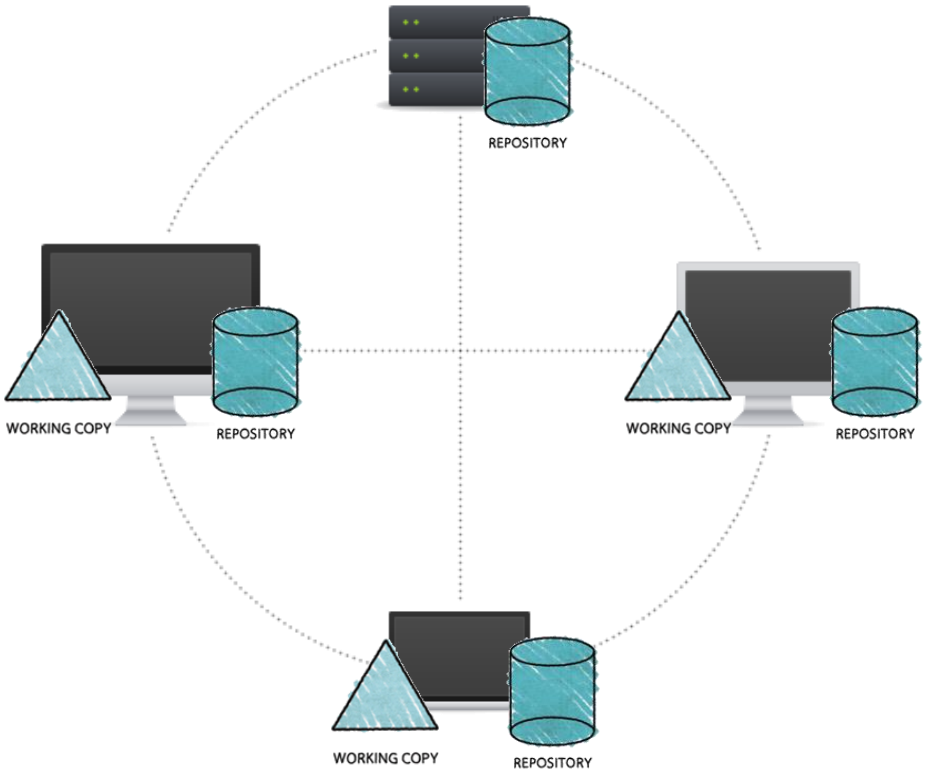
- update the working copy with respect to the repository
  - apply changes from the repository
  - merge such changes with the ones you have made to your working copy, if necessary

# Centralized Version Control




- one central repository
- client-server relationship

# Distributed Version Control

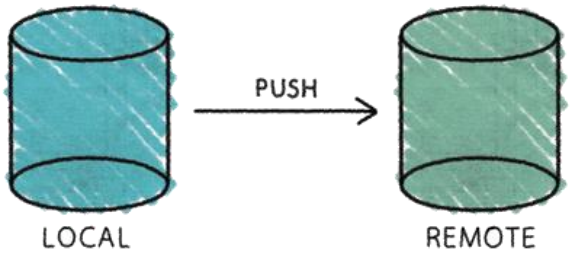



- clients and server have the full copy of the repository
  - local repositories ‘clone’ a remote repository
- it is possible to have more than one server

# More Basic Concepts

**Push**

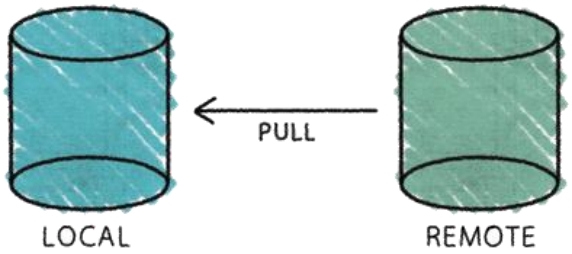
- copy changesets from a local repository instance to a remote one
  - synchronization between two repository instances



# More Basic Concepts

**Pull**

- copy changesets from a remote repository instance to a local one
  - synchronization between two repository instances



# Introducing… Git

- Distributed Version Control System
- Born
  - on 2005 for the Linux kernel project
  - to be used via command line
- Website: http://git-scm.com
- Highlights:
  - free and open source
  - strong support for non-linear development
  - fully distributed
  - efficient handling of large projects
  - cryptographic authentication of history

# Who uses Git?


Google

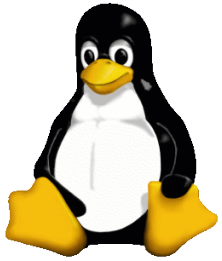

facebook


Microsoft


LinkedIn


NETFLIX


GNOME




RAILS


Qt


eclipse

# Getting started with Git

- Standard installations
  - http://git-scm.com/downloads
- Available for all the platform
- Git Graphical Applications
  - http://git-scm.com/downloads/guis
  - Suggestion: GitExtensions, SourceTree
- For this course, Git is
  - integrated in Eclipse (plugin “EGit”)

# Installing Git (outside Eclipse)

- Windows
  - download and install Git from http://git-scm.com/downloads
- Linux
  - check if it is already installed
    - open a terminal and type “git”
  - otherwise, install it from your package manager or via http://git-scm.com/downloads
- Mac
  - check if it is already installed
    - open a terminal and type “git”
  - otherwise, install it from http://git-scm.com/downloads

# Hosted Git

- To have (at least) one remote repository
  - alternative: set up your own Git server!
- Most popular:
  - GitHub, https://github.com/
  - Bitbucket, https://bitbucket.org/
  - GitLab, https://about.gitlab.com/gitlab-com/
  - Sourceforge, http://sourceforge.net/
  - CodePlex (by Microsoft), https://www.codeplex.com/

# GitHub

- Slightly different than other code-hosting sites
  - instead of being primarily based on the project, it is user-centric
  - social coding
- A commercial company
  - charges for accounts that maintain private repository
  - free account to host as many open source project as you want
  - free Micro plan for students
    - 5 private repositories, unlimited public repositories
    - https://education.github.com

# Bitbucket


- Similar to GitHub
- Less used than GitHub, right now
- Mercurial support
- A commercial company
  - free private and public repositories for small team (up to 5 private collaborators)
  - charges for project involving bigger team
  - free for academia (also for students)
    - unlimited public and private repositories
    - unlimited users for single projects

# GitHub Pages

- Website for your (GitHub) repository
  - https://pages.github.com/
- FAQ
  - https://help.github.com/categories/github-pages-basics/

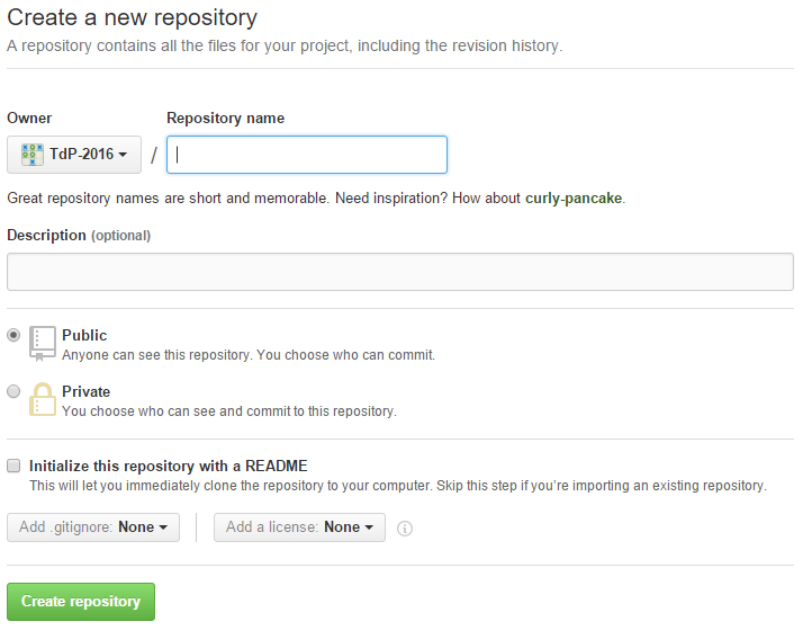
# For Labs

- Create a personal GitHub account
  - you can also ask for a “student discount” at https://education.github.com
- Try Git!
  - http://try.github.io/
  - 15 minutes tutorial

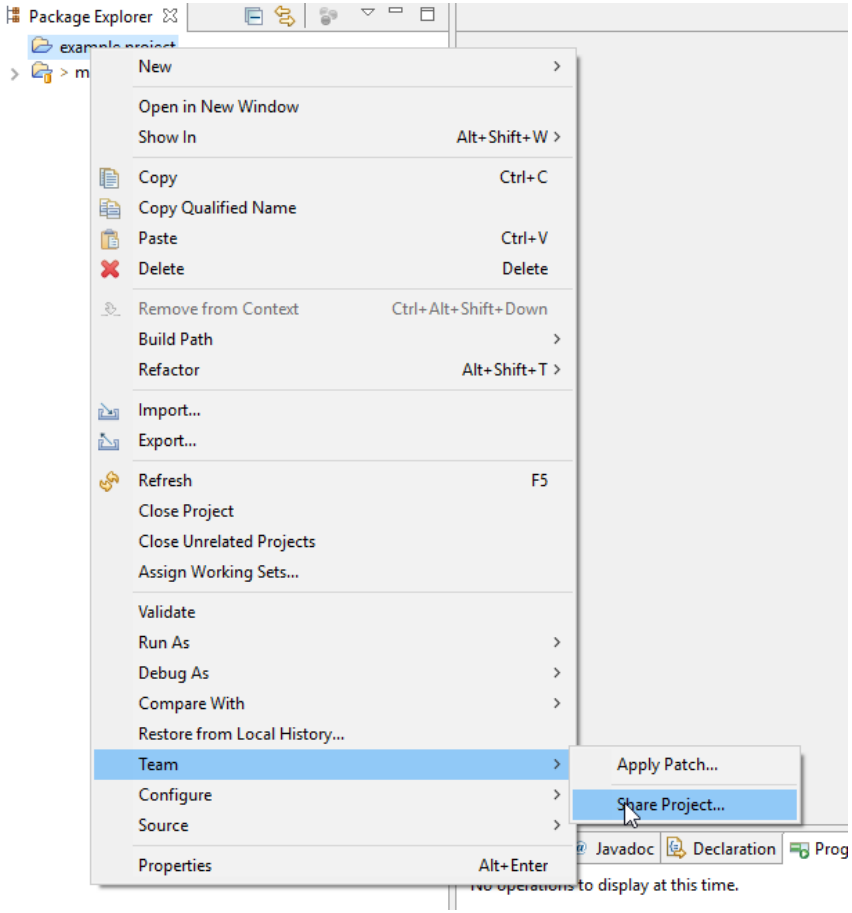
# Workflow 1: “Create new project”

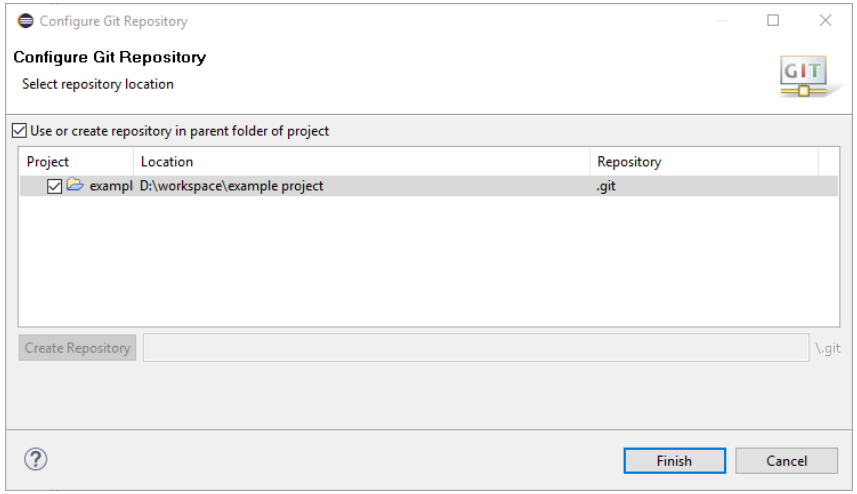
- Create a project in Eclipse (normally, or using Maven Archetypes)
- Create the local repository in Eclipse (Team|Share)
- Create a new project in GitHub
- Push changes (Team|Commit&push)

# New Project on GitHub

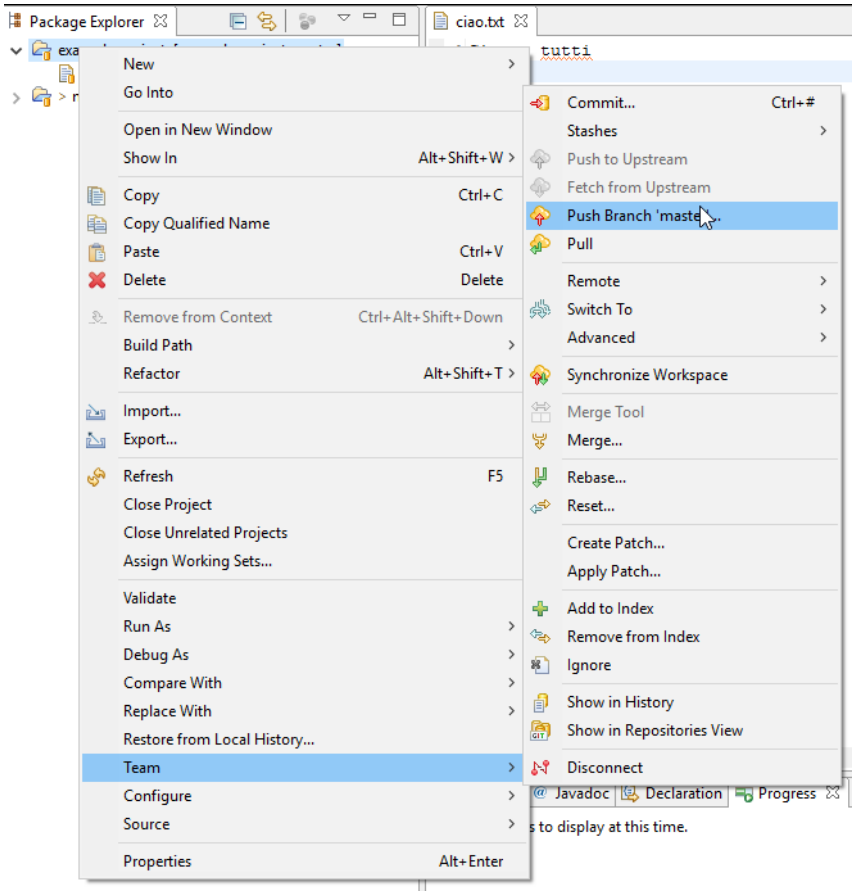



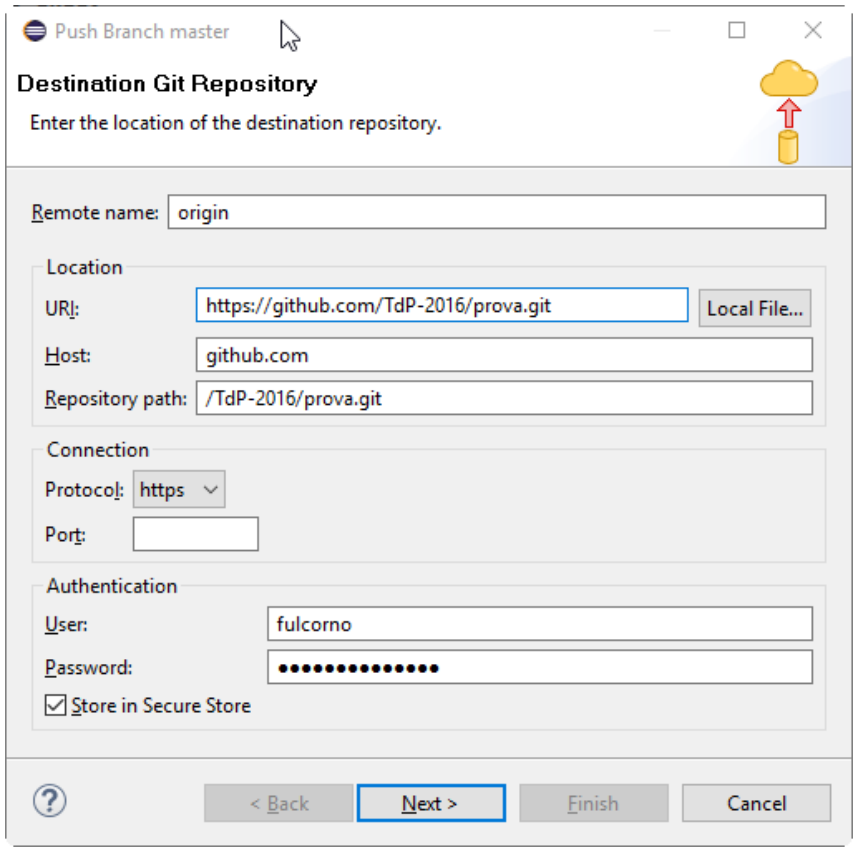
# New repository in Eclipse



# Add remote & push in Eclipse



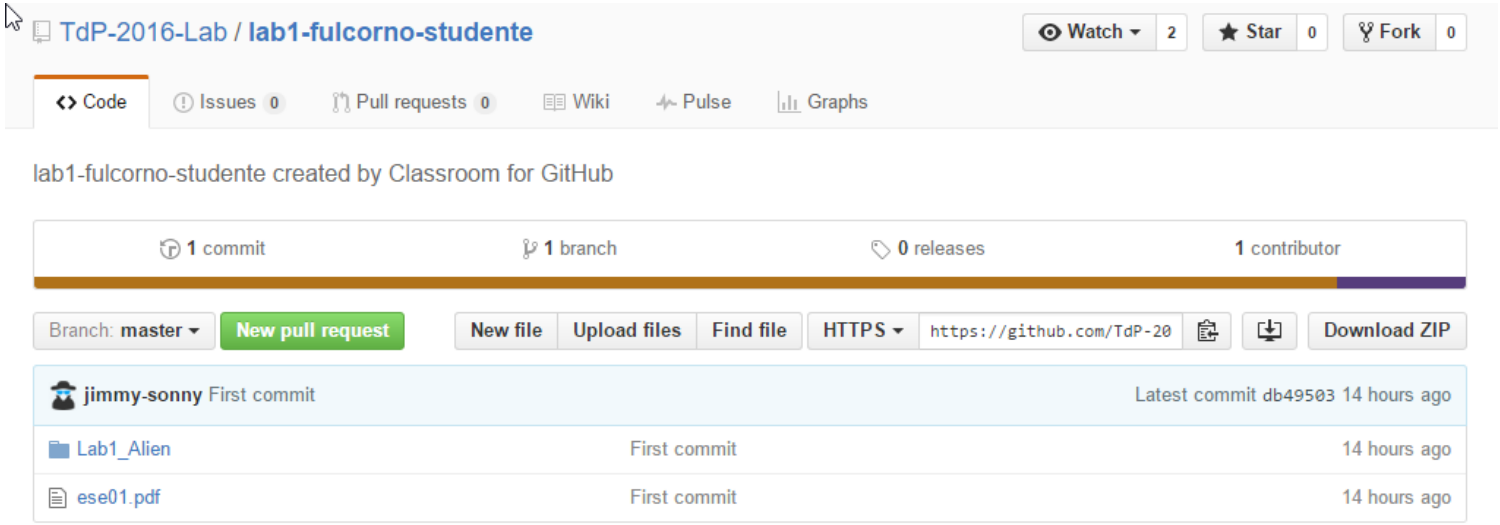



# Workflow 2: “Work on a project”

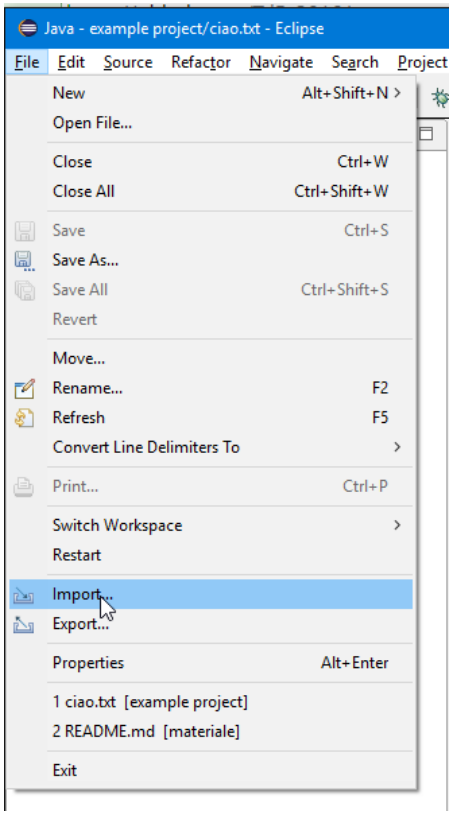
- Fork the project in GitHub (you make a copy in your repository)
- Clone your project in Eclipse
- Work on the project
- Commit and Push the changes

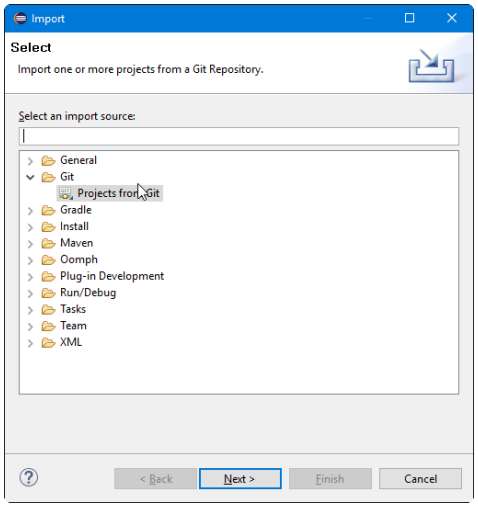
# Forking

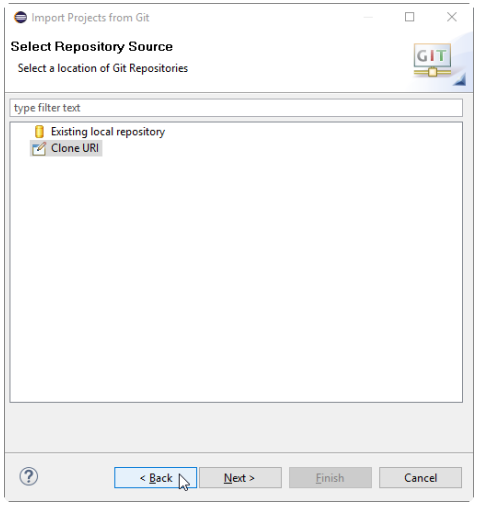
- "Fork" makes a private copy of some else's repository
- You may clone, work, and commit on this repository

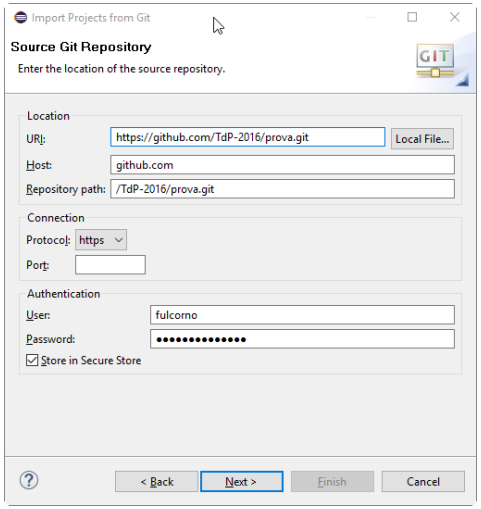


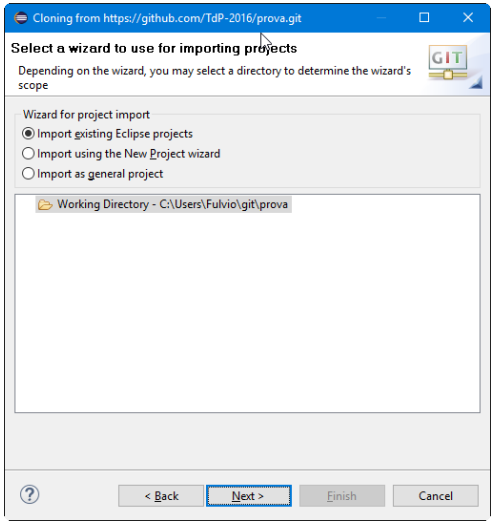
# Cloning in Eclipse



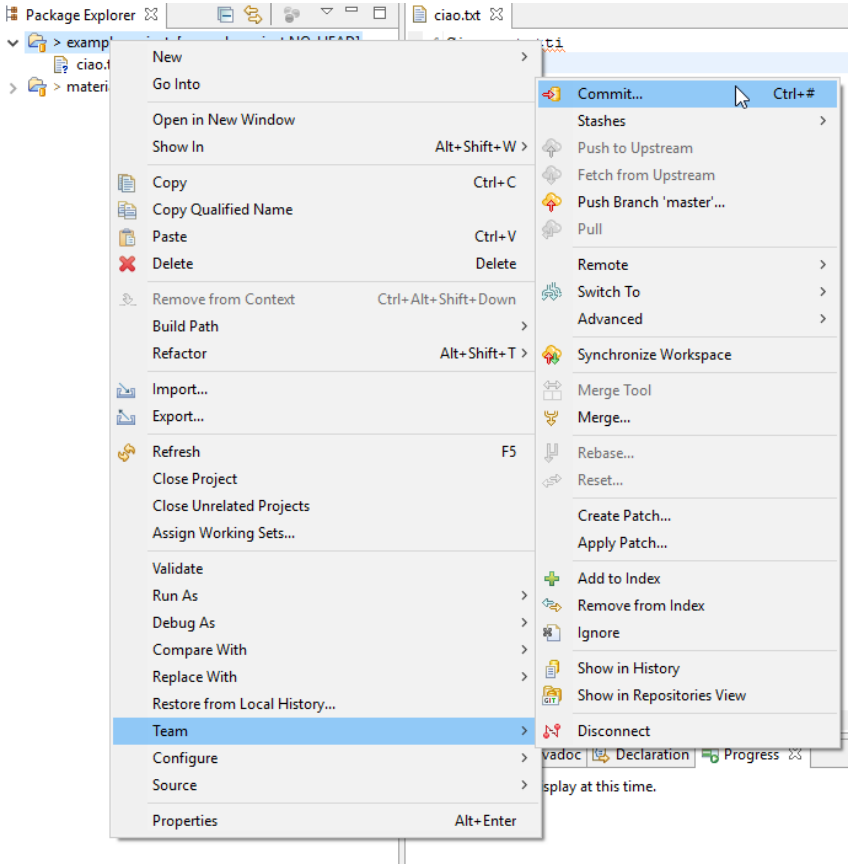



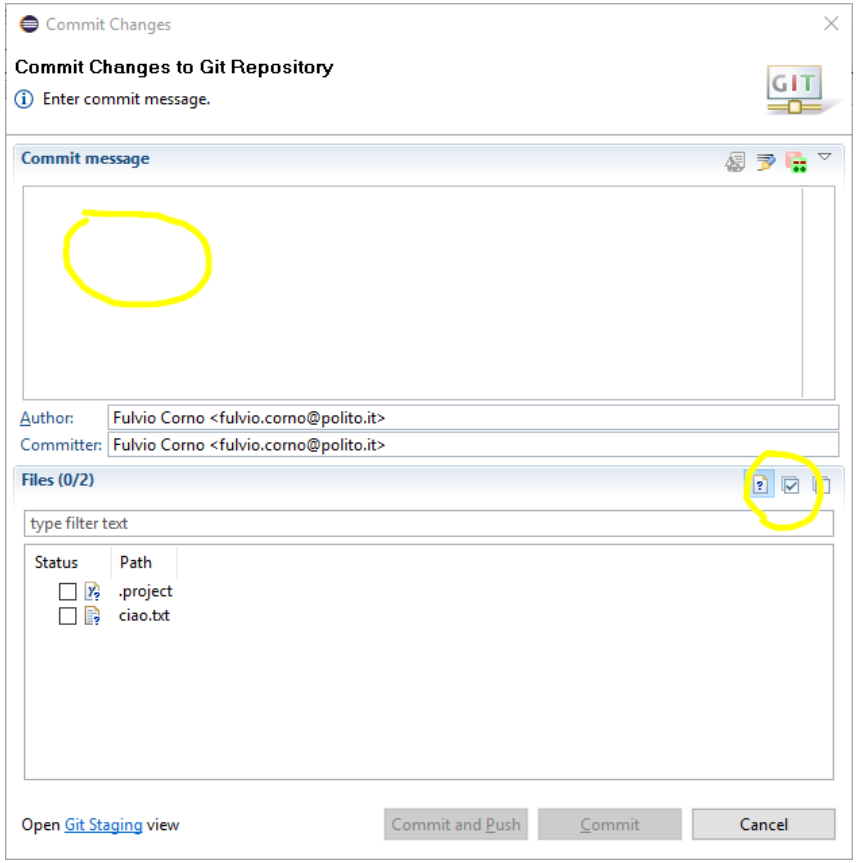



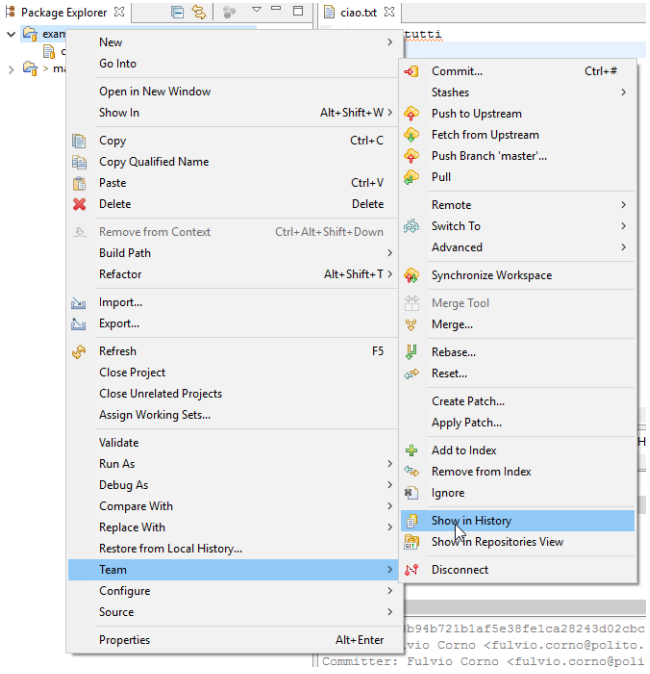



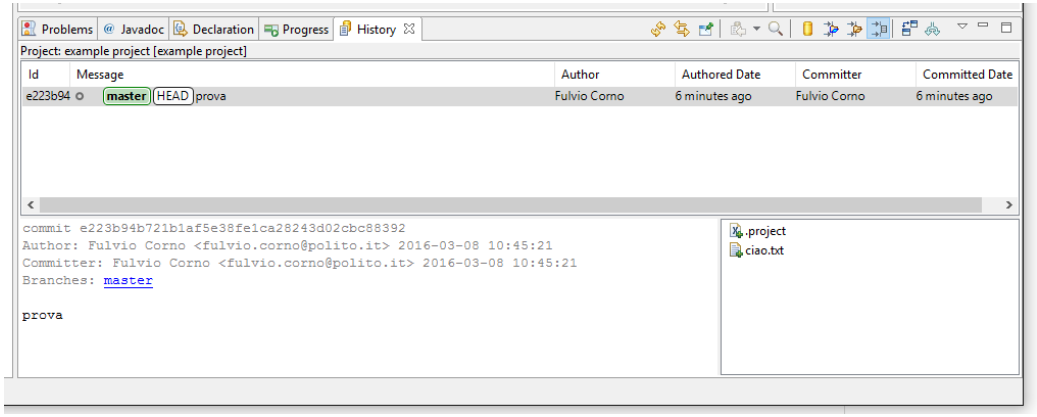
# Commit in Eclipse



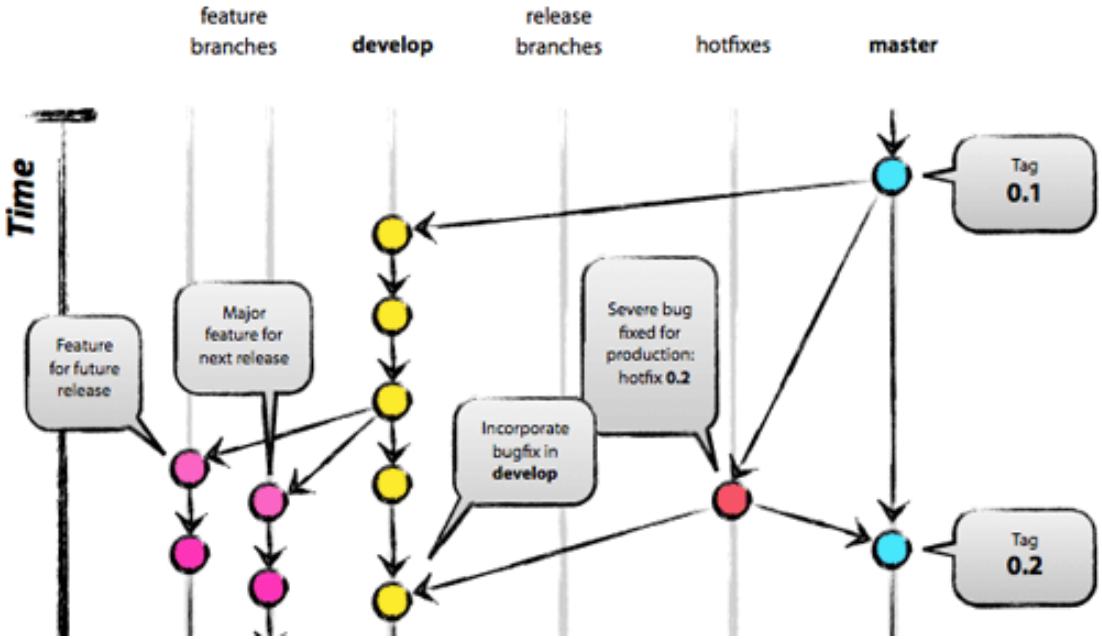



# History in Eclipse



# Tags and Branches in a Nutshell




- Local and remote
- Do not push automatically

[Image from http://nvie.com/posts/a-successful-git-branching-model/]

# Branches… in brief

- used to develop features isolated from each other
- the *master* branch is the “default” branch when you create a repository
  - you should use other branches for development and merge them back to the master branch upon completion
- really lightweight in Git
- commands:
  - *git branch [branch-name]*, create a new branch
  - *git branch*, lists all existing branches
  - *git checkout [branch-name]*, switches to the selected branch
  - *git branch -d [branch-name]*, removes the selected branch

# Tags… in brief

- useful to mark **release points**
- two types:
  - lightweight
  - annotated (more complete)
- commands:
  - *git tag*, shows the available existing tags
  - *git tag [tag-name]*, creates a lightweight tag
  - *git tag -a [tag-name] -m [message]*, creates an annotated tag
  - *tag show [tag-name]*, shows the tag data

# References

- Git Reference
  - http://gitref.org/
- Git - the simple guide
  - http://rogerdudler.github.io/git-guide/
- Git Documentation
  - http://git-scm.com/docs
- Pro Git (online book)
  - http://git-scm.com/book
- Version Control by Example (online book)
  - http://www.ericsink.com/vcbe/

# References

- Try Git!
  - http://try.github.io/
- Various Git resources
  - https://help.github.com/articles/what-are-other-good-resources-for-learning-git-and-github
- A successful Git branching model
  - http://nvie.com/posts/a-successful-git-branching-model/
- Some Git (graphical) clients
  - http://git-scm.com/downloads/guis

# License

- This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3,0)” License.
- You are free:
  - to **Share** - to copy, distribute and transmit the work
  - to **Remix** - to adapt the work
- Under the following conditions:

  - **Attribution** - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

  - **Noncommercial** - You may not use this work for commercial purposes.

  - **Share Alike** - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
- To view a copy of this license, visit http://creativecommons.org/license/by-nc-sa/3.0/