Software Configuration Management

Introduction

Workflows

Software Configuration Management

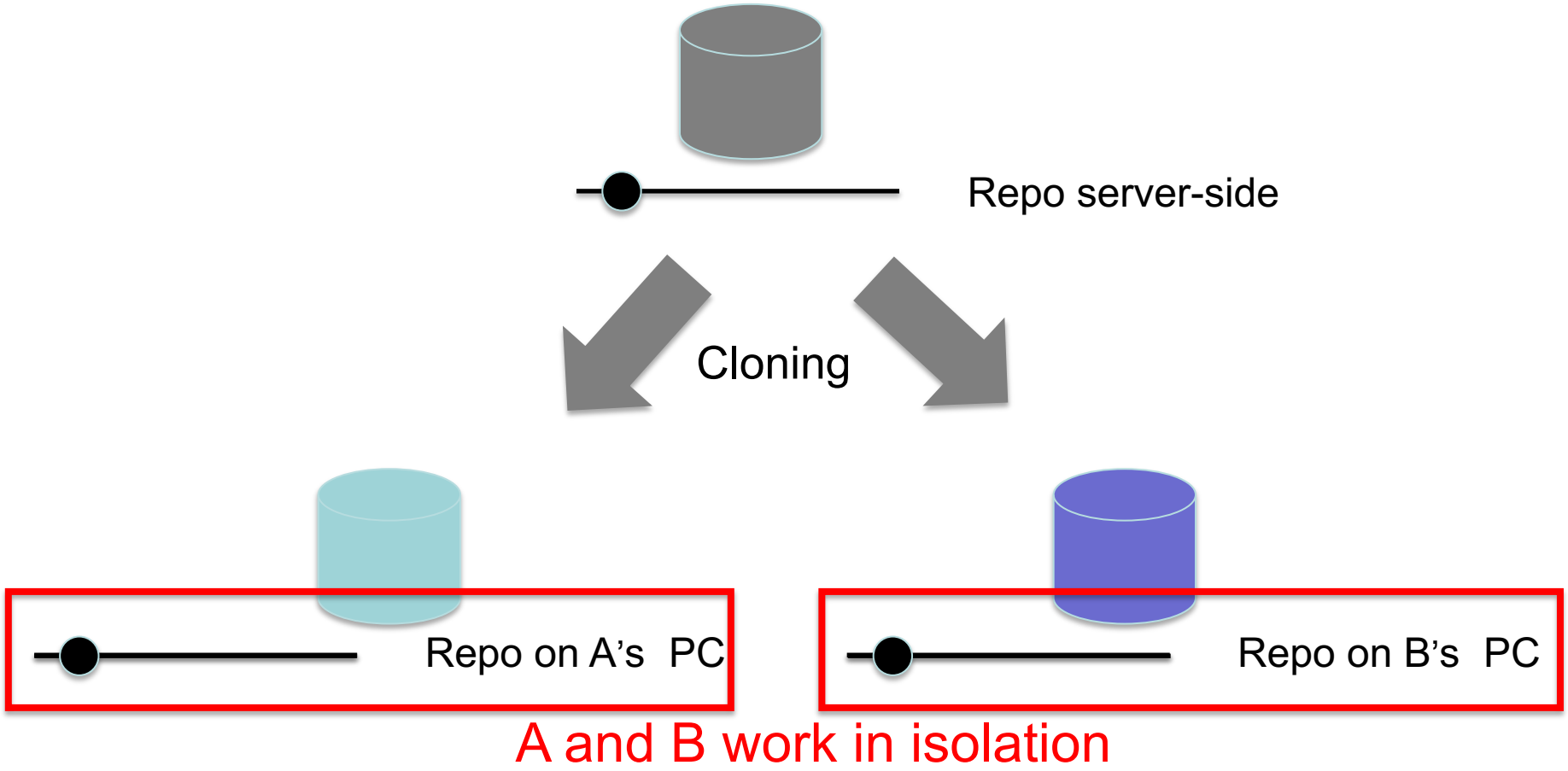
POLITECNICO MILANO 1863

- Roles:
  - Manage change in a controlled manner
  - Sharing software artifacts
- Centralized
  - Central repository, selective access to project tree
  - Tools: CVS, SVN, Team Foundation Server, Rational ClearCase, Rational TeamConcert...
- Decentralized
  - Distributed repositories, replicated repositories
  - Tools: Git, Mercurial

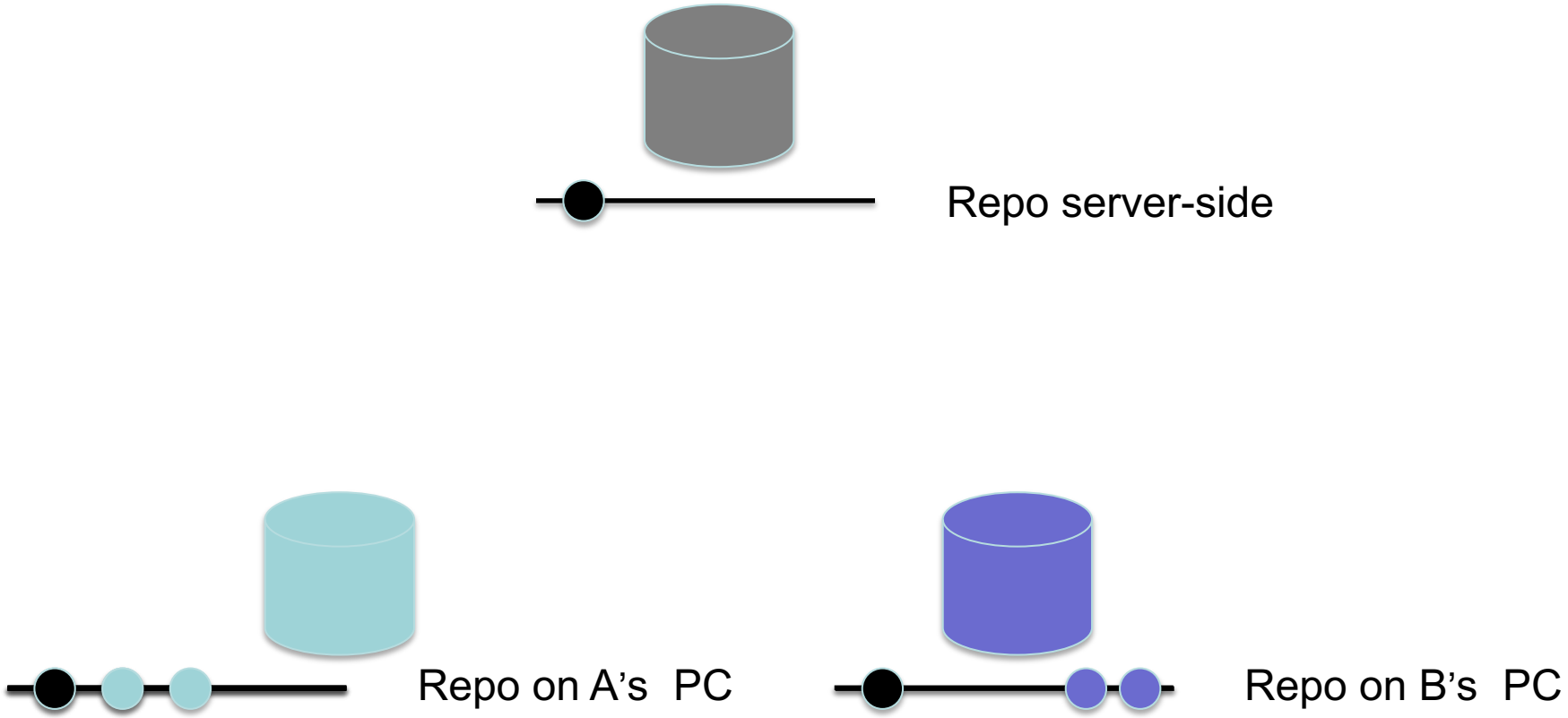
Basic approach to use a decentralized CM

Repo server-side

Cloning to local repositories

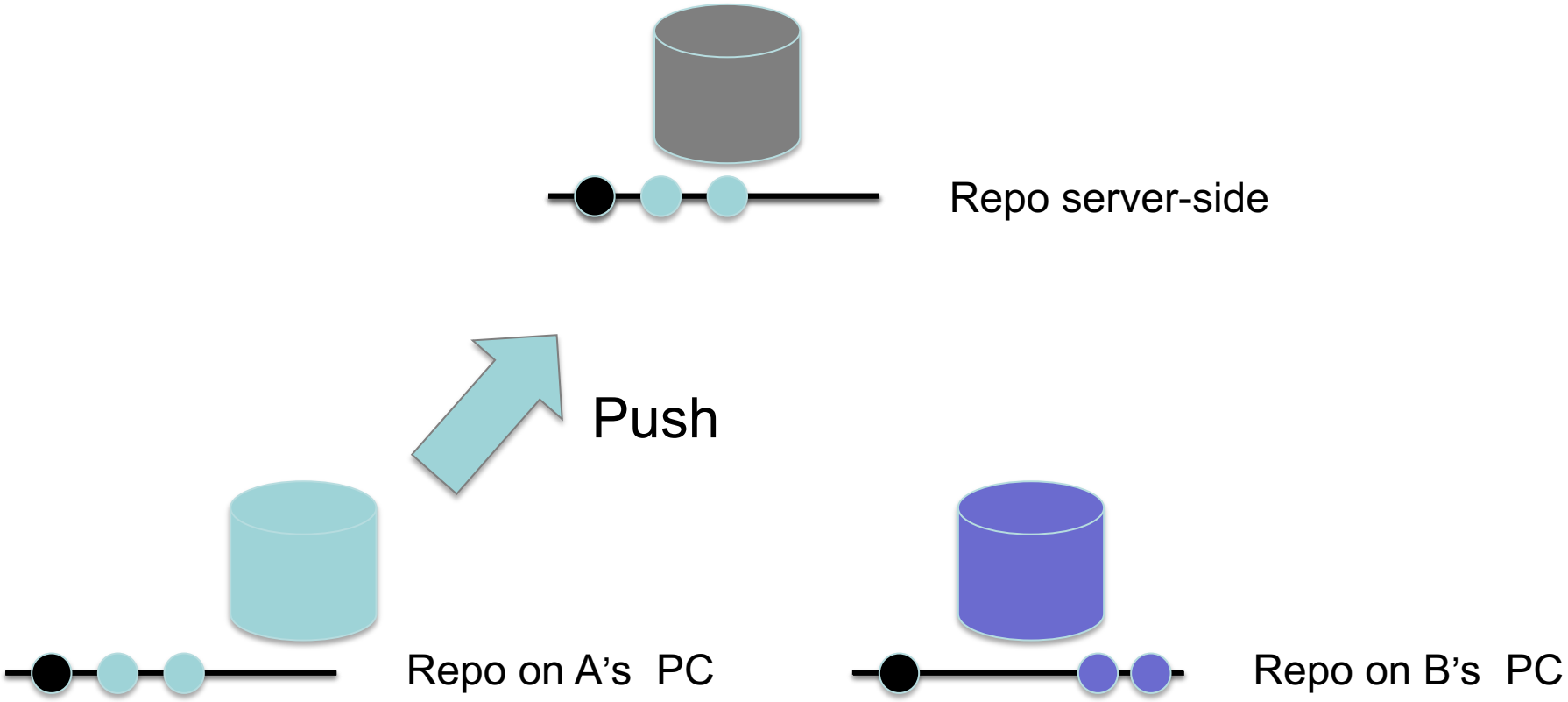


Local work



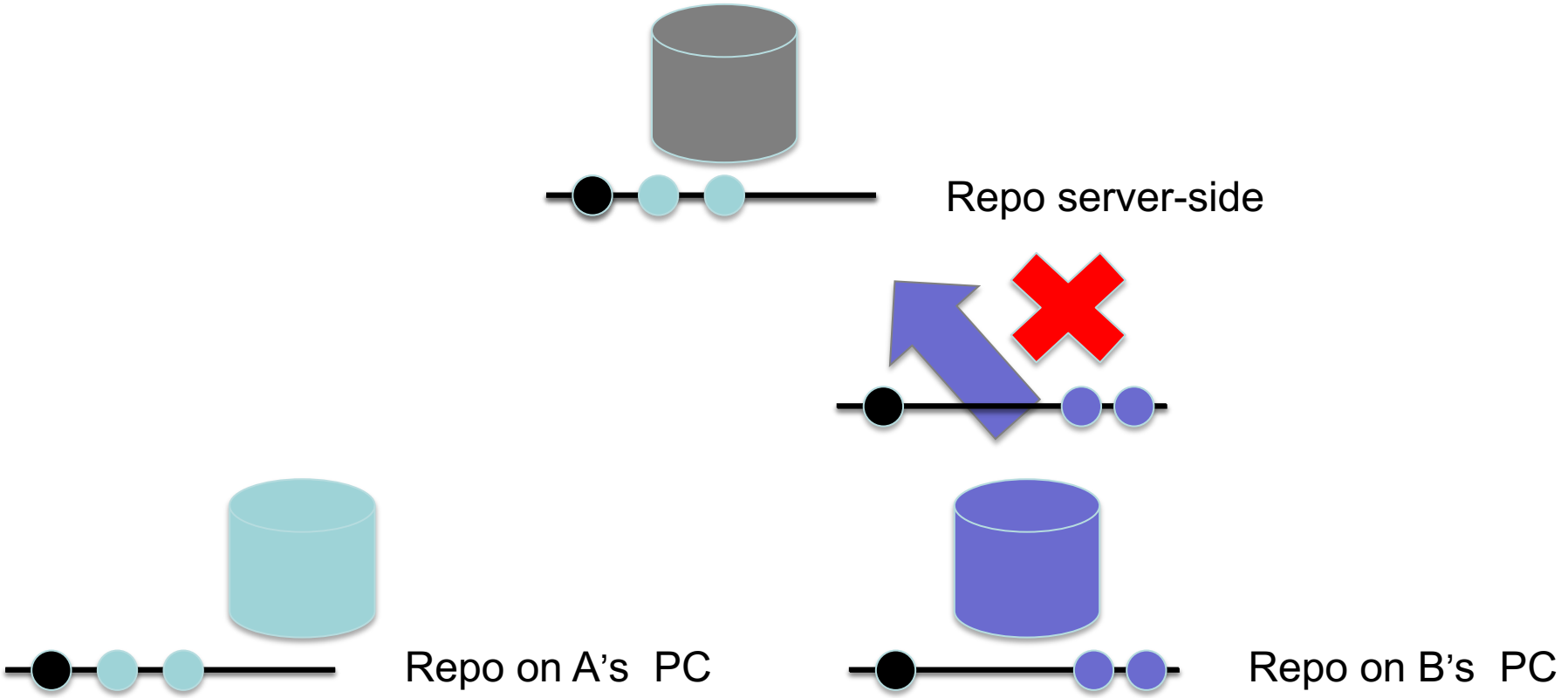
Locally code, compile, test, commit

Integrating changes to central repository



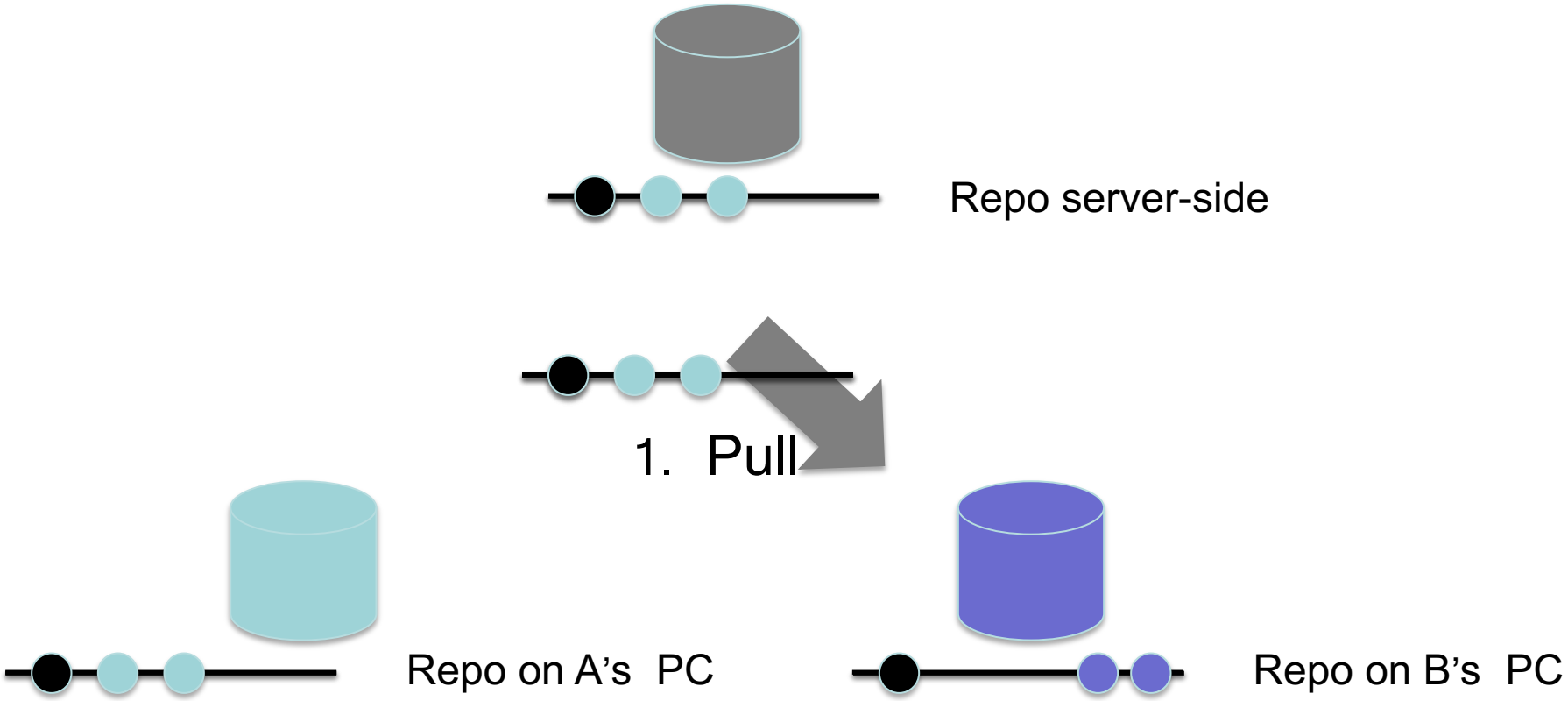
Push changes to central repository master branch

Integrating changes to central repository



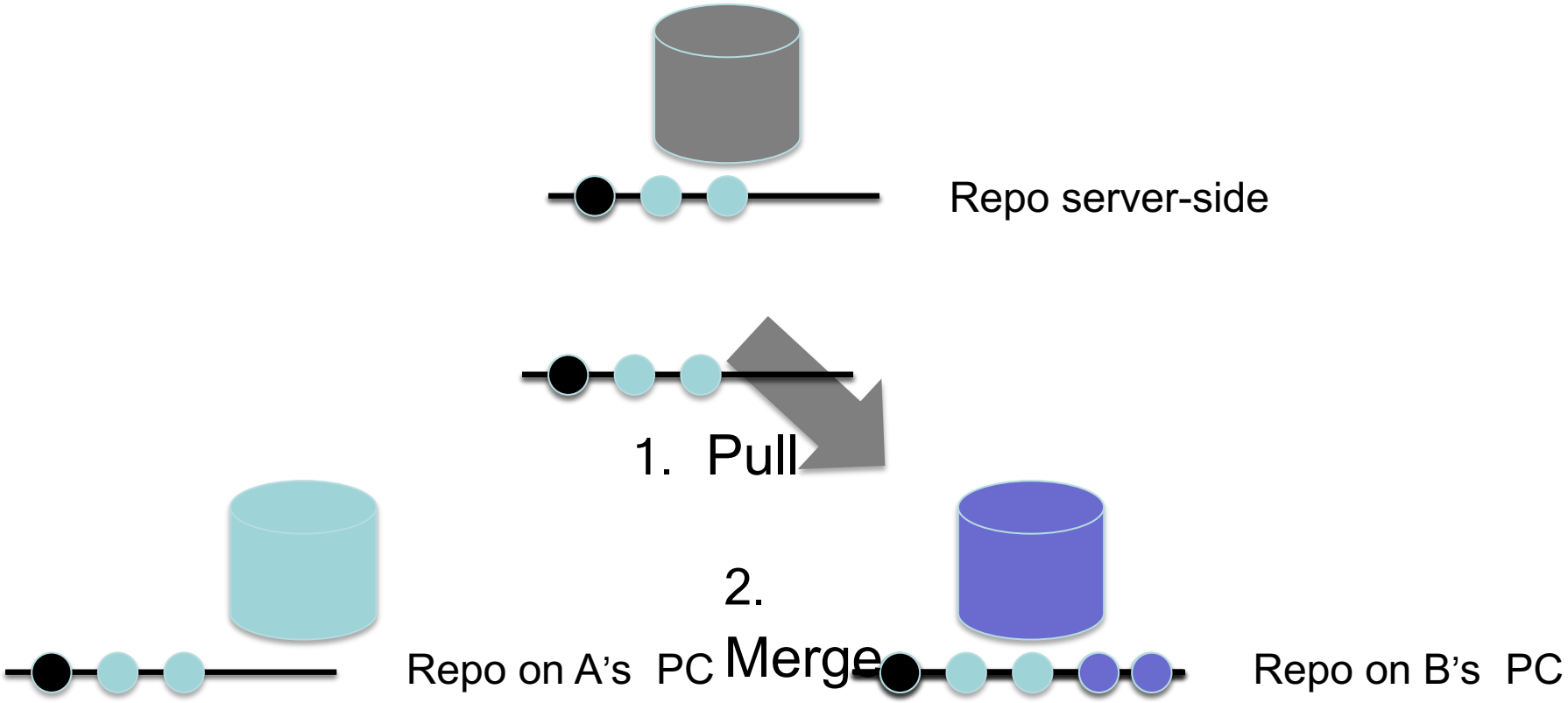
Push cannot be completed due to changes in central repository master thread

Integrating changes to central repository



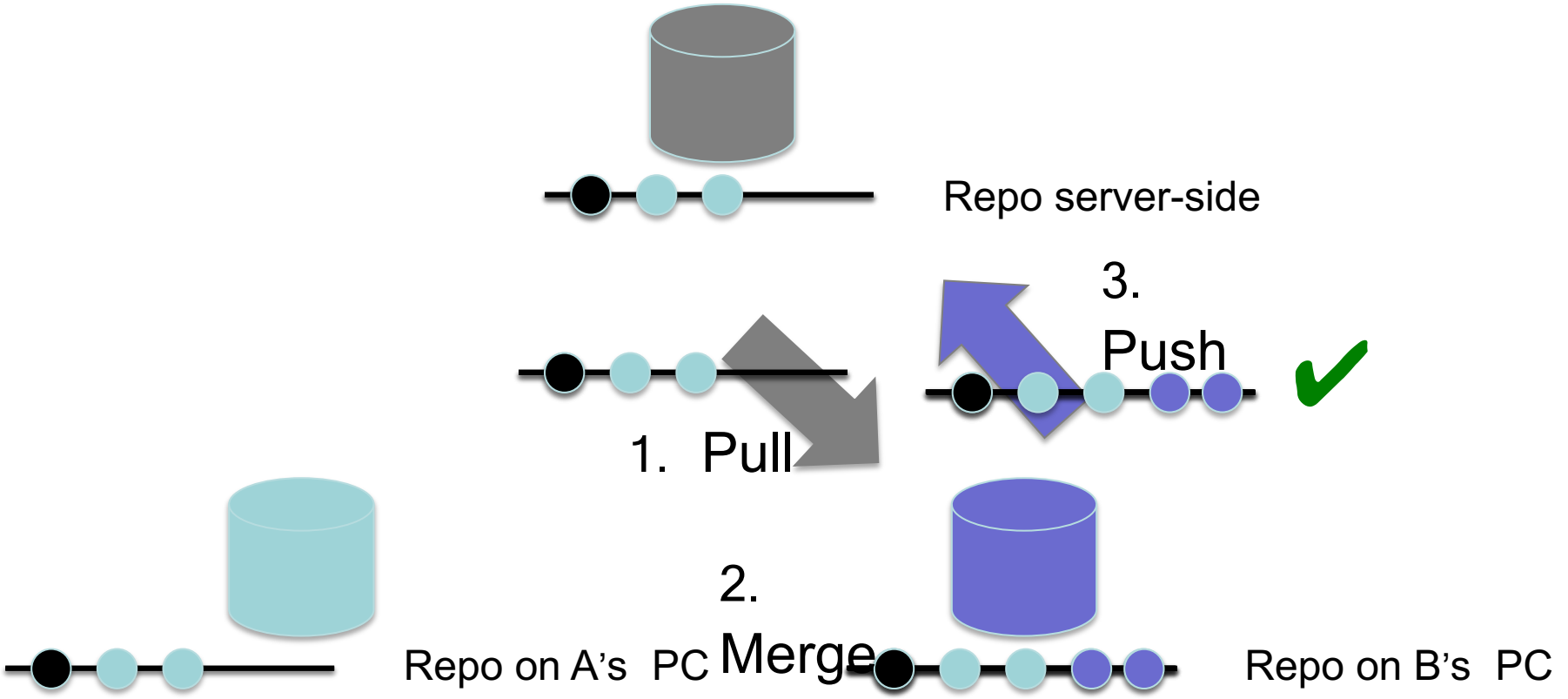
Push cannot be completed due to changes in central repository master thread

Integrating changes to central repository



Push cannot be completed due to changes in central repository master thread

Integrating changes to central repository



Push cannot be completed due to changes in central repository master thread

Main Git commands

POLITECNICO MILANO 1863

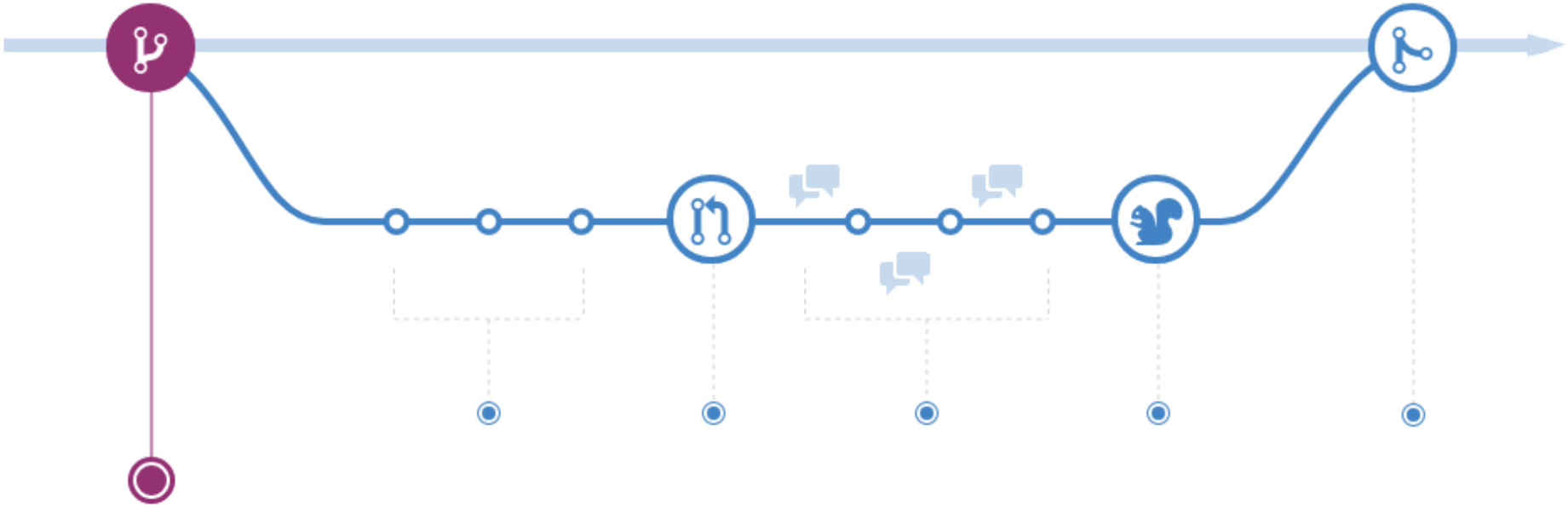
- git clone <repository URL>: create a local copy of the repository on the server
- git pull: get new updates from the server and merges them with the local changes
- git add: add modified files to the local repository
- git commit: finalizes the changes in the local repository
- git push: transfers local commits to the server
- git status: it shows the status of the local repository (if there are files not included in the repository, if it not aligned with the server-side one)
- References
  - https://education.github.com/git-cheat-sheet-education.pdf
  - https://git-scm.com/doc

Workflows

POLITECNICO MILANO 1863

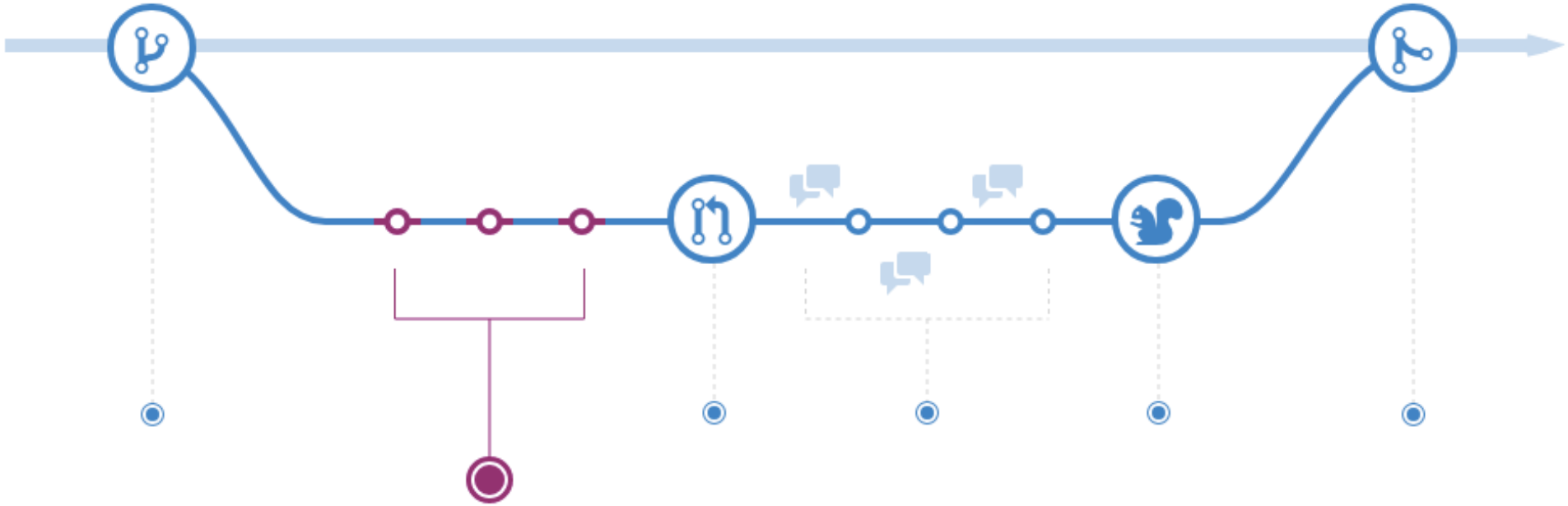
- Define the way an organization uses the CM for a specific project
- Main principles
  - The *master* repository must be kept as clean as possible
  - The distinction between local and central repository is not enough in case of multiple contributors
  - On the central repository we can create one *master* repository and as many *branches* as we want
  - A new contribution becomes part of the master only if the whole team agrees

GitFlow https://guides.github.com/introduction/flow/



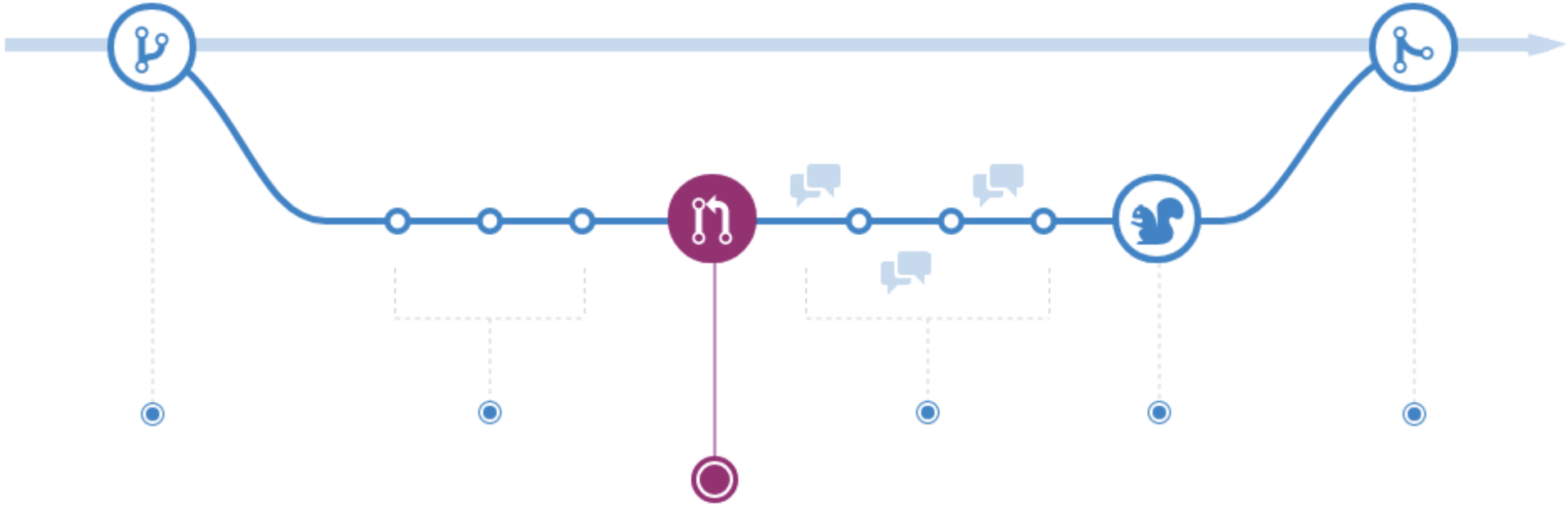
- When you need to develop a new feature or a new idea, create a new *branch*
  - A new independent sub-repository where you can experiment without impacting on what is in the master

GitFlow https://guides.github.com/introduction/flow/



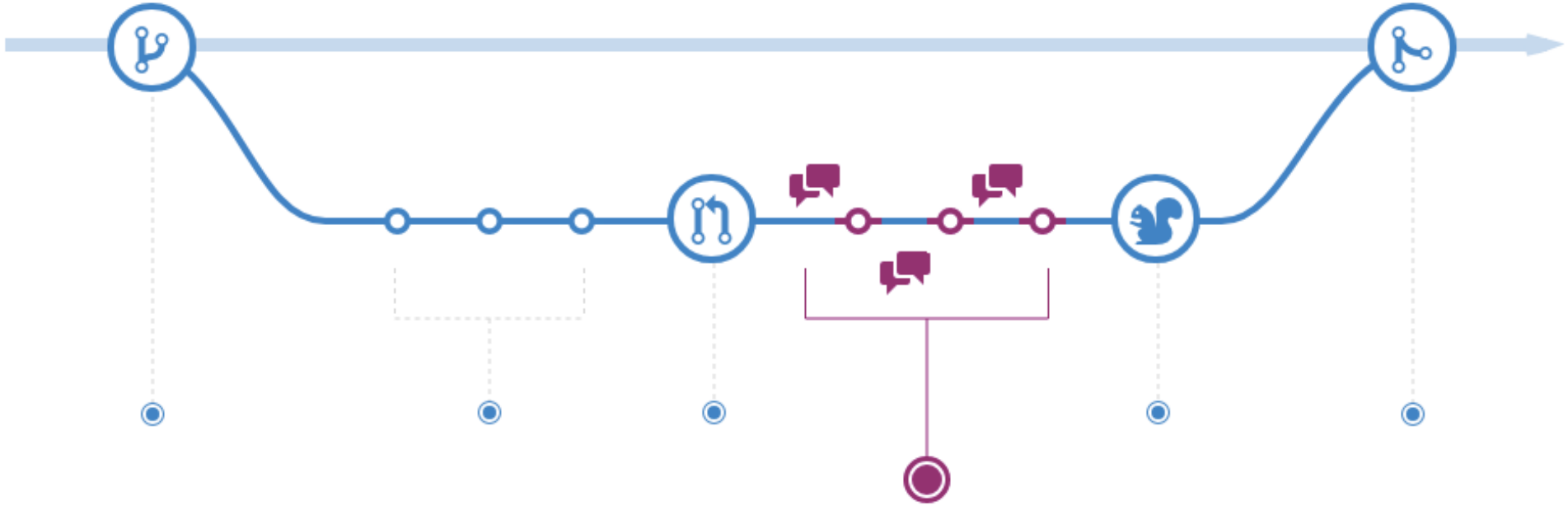
- New versions of software are produced in the branch

GitFlow https://guides.github.com/introduction/flow/



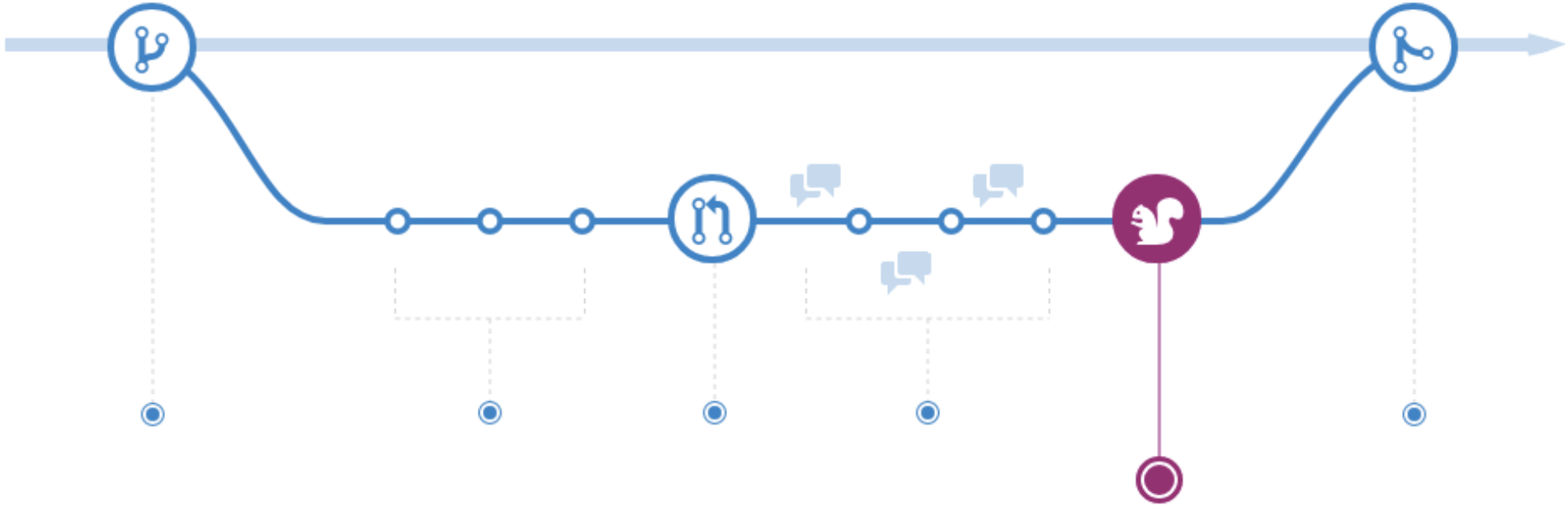
- When the developers are happy, they can issue a *pull request*
  - The sub-team is asking the whole team to review the changes and decide whether to accept them or not
  - The sub-team has already merged any local change in the branch server-side

GitFlow https://guides.github.com/introduction/flow/



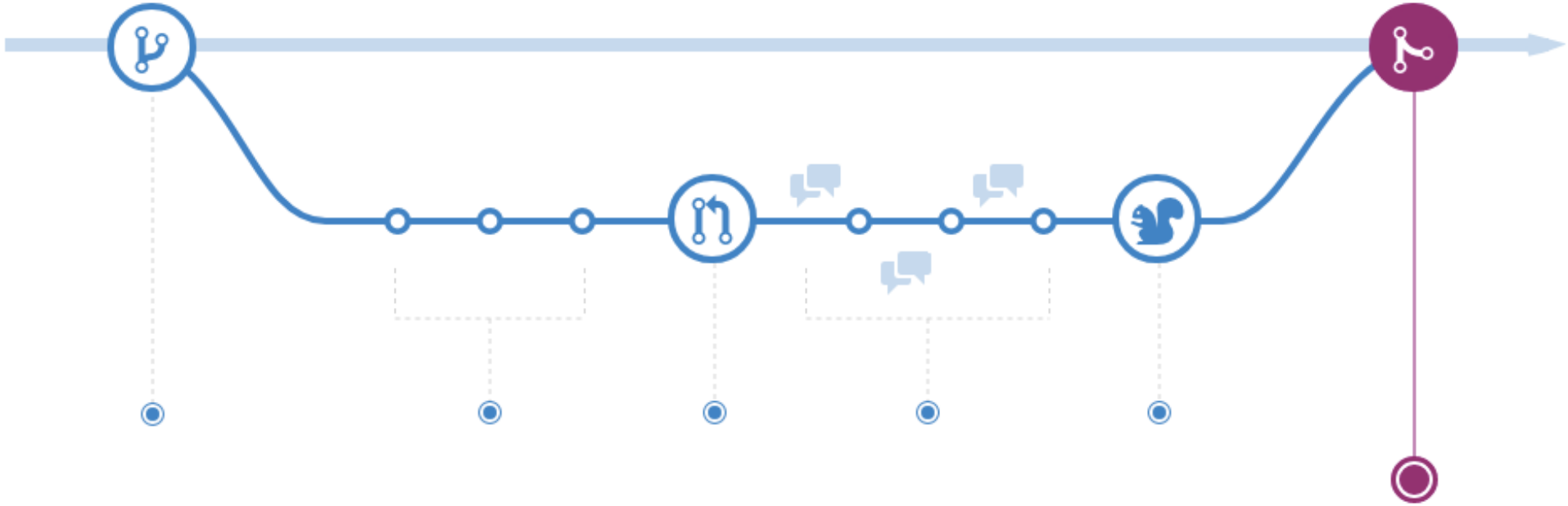
- A discussion starts
- During the discussion, new versions can be created in the branch

GitFlow https://guides.github.com/introduction/flow/



- The code is deployed and tested

GitFlow https://guides.github.com/introduction/flow/



- Then finally, the branch is merged in the master

Some examples of free of charge clients

POLITECNICO MILANO 1863

- Atlassian SourceTree
  - OS X, Windows
  - Integrate with any git repository
- Github desktop
  - OS X, Windows
  - Specific for Github
  - Simple to use
- SmartGit (free for non-commercial use)
  - OS X, Windows, Linux
  - Integrate with Github, GitLab, Bitbucket, or Stash
  - Advanced features