upcomillas es

upcomillas es

Advanced Computing Tools for Applied Research

Chapter 4. Version control

Jaime Boal Martín-Larrauri
Rafael Palacios Hielscher

Academic year 2014/2015

1

Version control fundamentals

What you probably do now

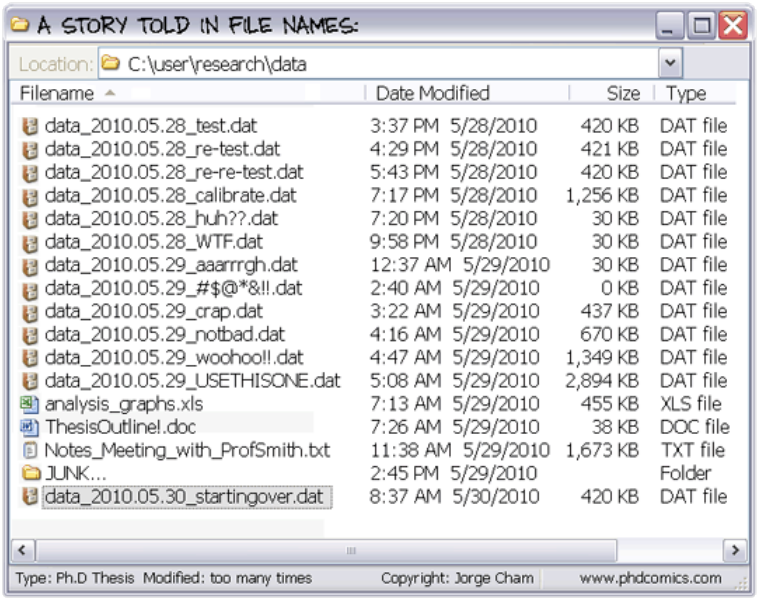
- Manually save copies of files or folders (into a backup directory?)

file-v1.6-copy2-works.m

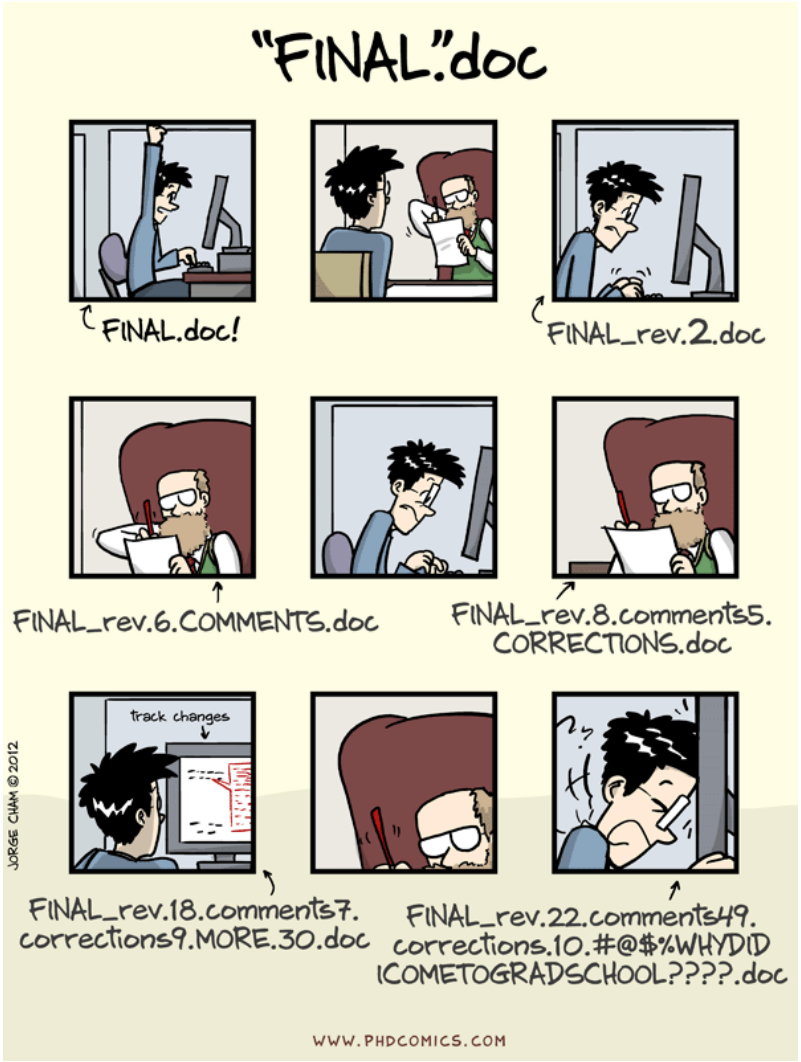
- If you are smart, the files or folders might be time-stamped

2015-02-23-file-v1.6-copy2-works.m

- Even better, you might also save your documents in a cloud storage service
  - Keep them in sync throughout all your devices
  - Limited file history



How do you get to this situation?



Pros and cons of your current approach

✓ Extremely simple

✗ Error prone ➡ Accidental file overwriting

- The original version may never be retrieved again

✗ Difficult to find the appropriate version

- Need to keep a revision history file ➡ Never up-to-date!

✗ Communicate updates via email to your team

- Difficult to keep track of who made a change and when it happened

What is version control and why should you care?

Version control (also known as *revision control* or *source control*) is a system that records and manages changes to a file or set of files over time so specific versions can be recalled later.

- ✓ Roll back code to previous states
- ✓ Identify when and how bugs were introduced
- ✓ Keep multiple versions of the same code in sync across computers
- ✓ Work concurrently on the same file an then merge all changes
- You should use version control for almost everything
  - Software development
  - Document writing (papers, PhD thesis…)

Types of version control systems

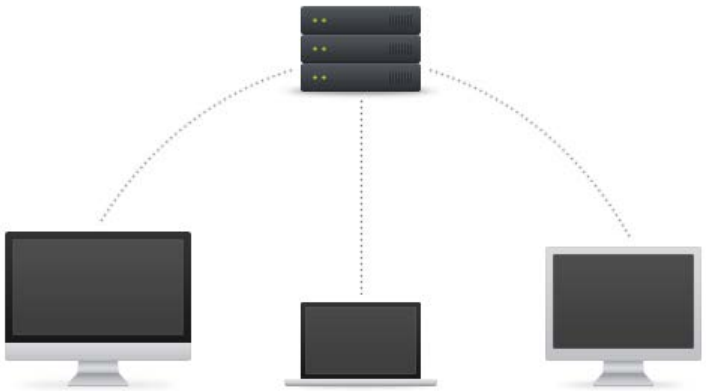
- Local version control systems ➡ Obsolete
  - Revision control system (RCS)

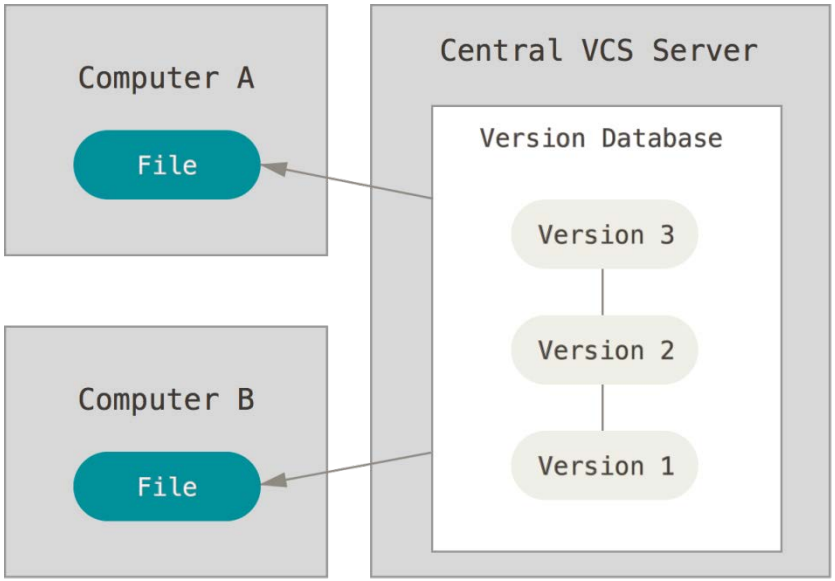
- Centralized version control systems (CVCS)
  - Concurrent versions system (CVS)
  - Subversion (SVN)

- Distributed version control systems (DVCS)
  - Git
  - Mercurial (Hg)
  - Plastic SCM

Centralized version control systems

- The repository is located in one place and provides access to many clients
- Everything must be set and received from this central repository





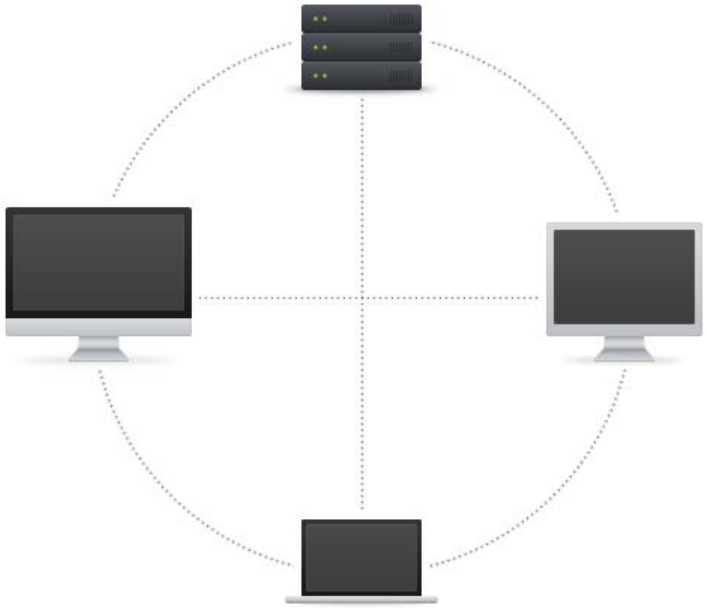
Centralized version control systems

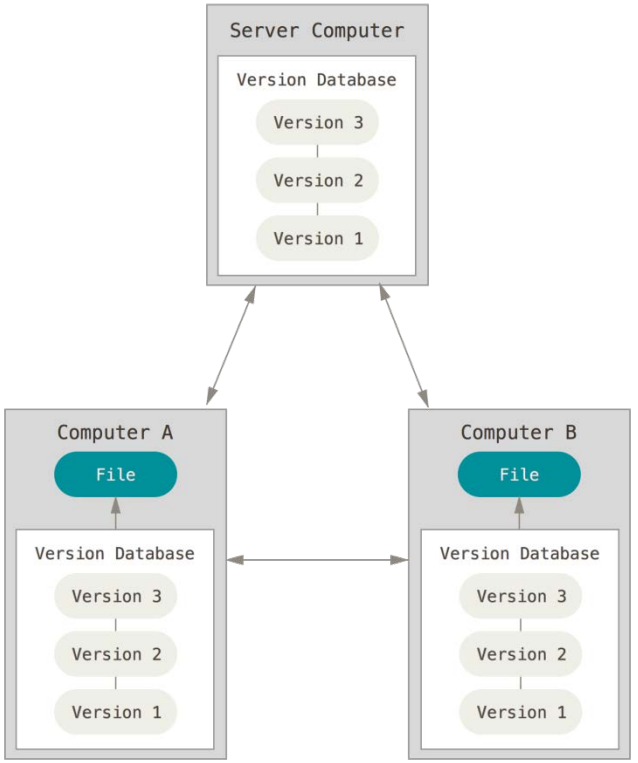
- Advantages
  - Easy to understand
  - More control over users and access (served from a single place)
  - Simple to get started

- Disadvantages
  - Dependent on access to the server
  - It is hard to manage a server and backups
  - It can be slower because every command connects to the server
  - Branching and merging tools are difficult to use

Distributed version control systems

- Each user has its own copy of the entire repository, not just the files but the history as well
- It can be regarded as a network of individual repositories





Distributed version control systems

- Advantages
  - More powerful and detailed change tracking ➡ Less conflicts
  - Server not required (everything except syncing repositories is local)
  - Branching and merging are more reliable
  - It is fast

- Disadvantages
  - The distributed model is harder to understand
  - It is easier to make mistakes until you are familiar with the model
  - Steeper learning curve

DVCS | Project hosting services

- GitHub
  - Unlimited users for free
  - Charges for private repositories
    - You can claim free private repositories if you are in academia
  - Supports only Git
  - Mac and Windows client
- Bitbucket
  - Limited to 5 users for free
  - Private repositories
  - Supports Git and Mercurial
- Other alternatives
  - Beanstalk, Gitorious, GitLab, Google code, Kiln…

2

Git | Facts

- It has seen a rapid growth over the past few years thanks to the open-source community
- Used by major companies

- Heavily terminal-based ➡ Thank Linus Torvalds for it
- Fortunately there are many GUI clients available

Git | Basic terminology

- **Repository:** Files or directories that are under version control
  - Metadata in hidden directory ➡ .git
- **Clone:** Create a local copy of a repository
- **Branch:** An independent line of development
  - Default development branch: “master” ➡ Production code
- **Commit:** Collection of actions that are submitted together to the version control system
  - It is a snapshot of every file in your repository
- **Merge:** Integrate changes from divergent branches

Git | File states

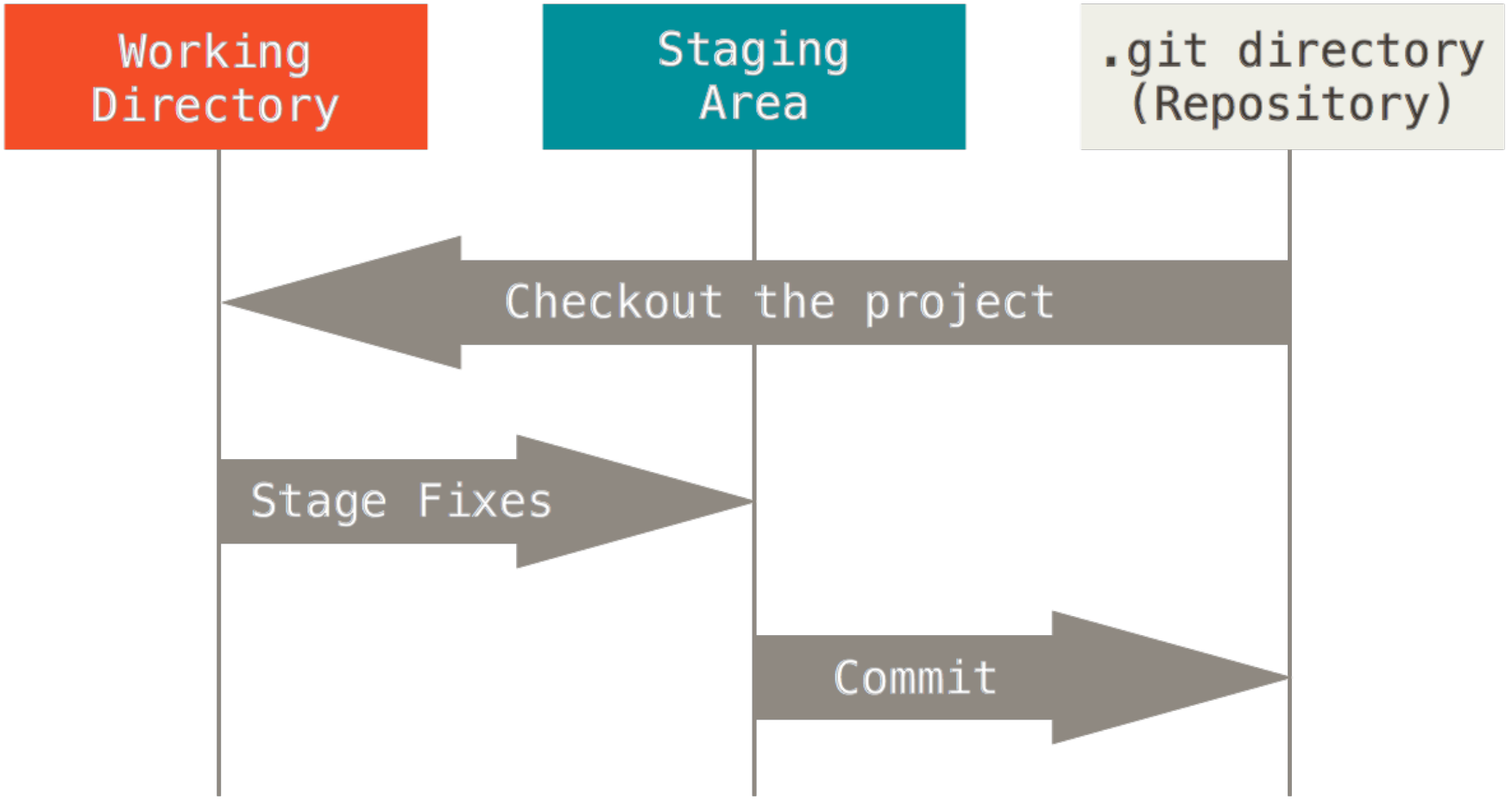


Image source: Git

Git | File status

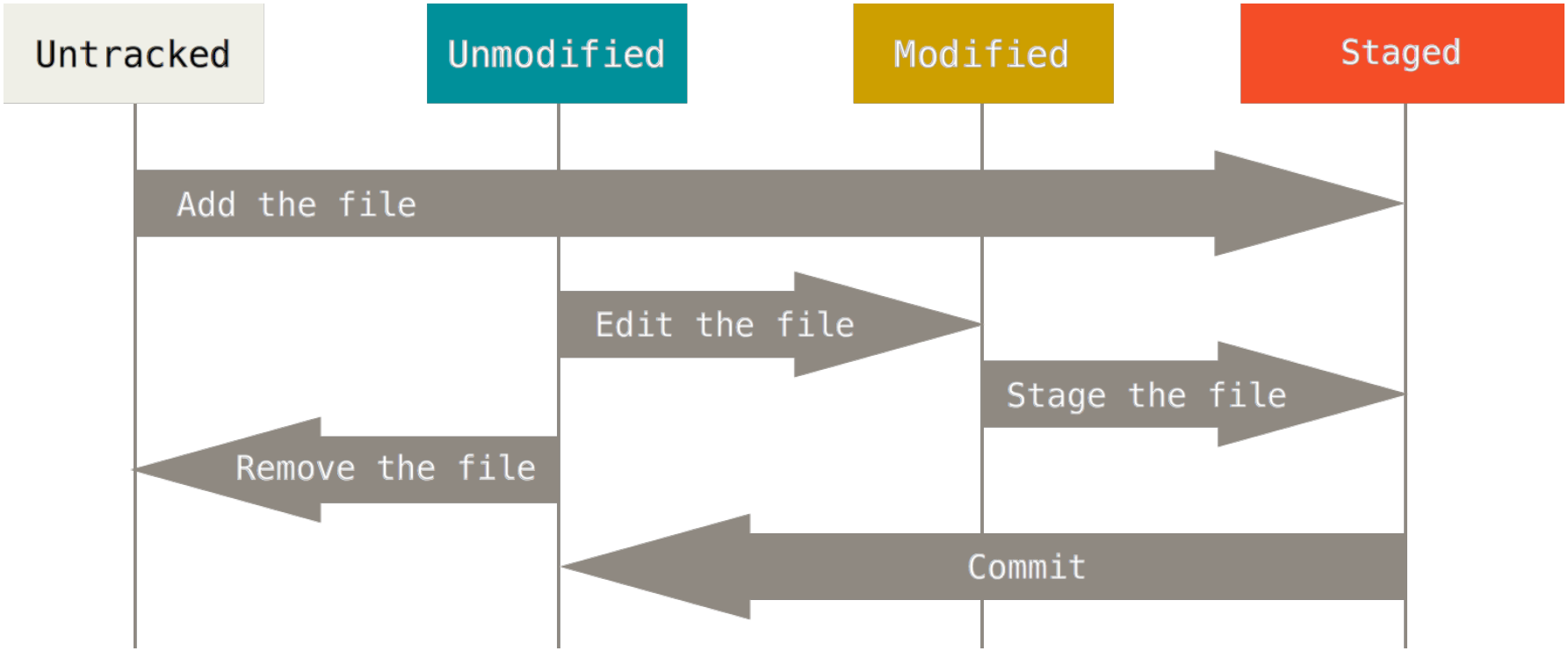
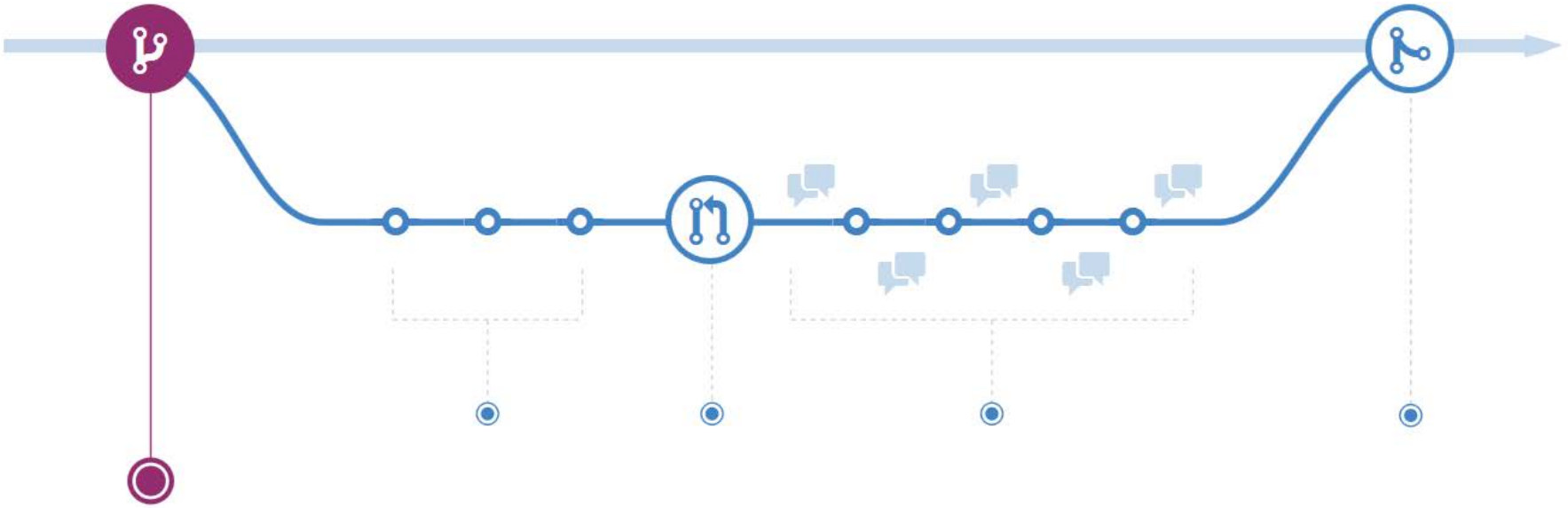


Image source: Git

3

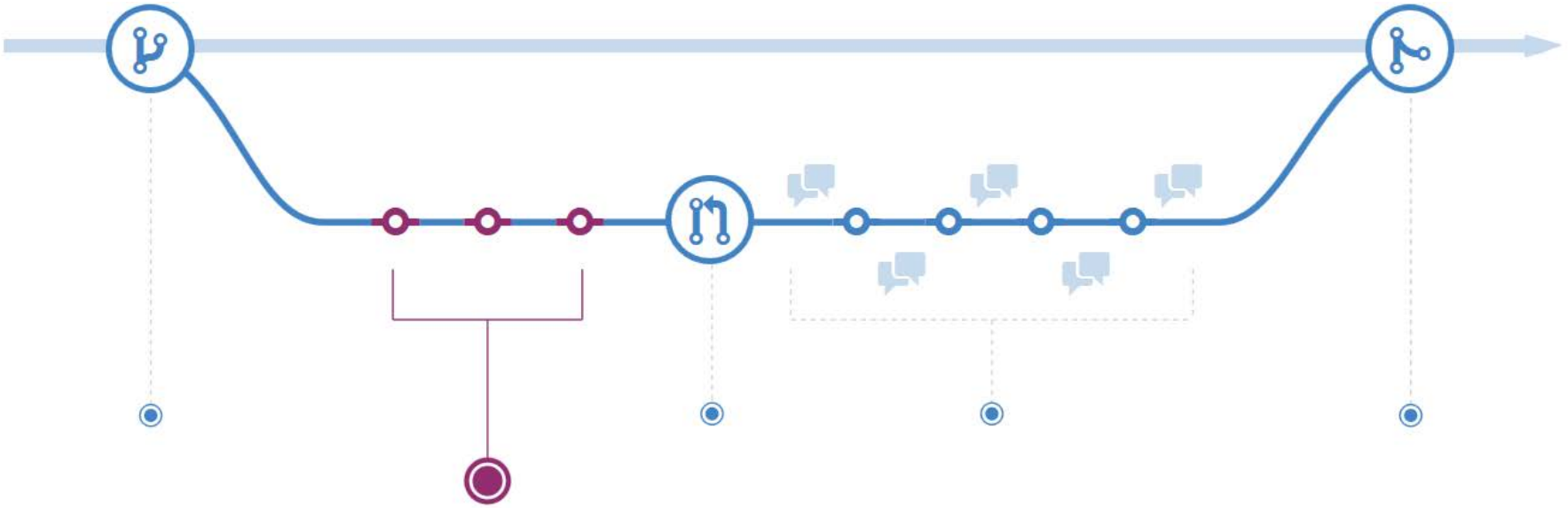
GitHub flow | Create a branch



- Never modify the master branch directly
  - Anything on the master branch is always deployable
- Always create a new branch to work on a feature or fix
  - Branch names should be descriptive
- Check https://guides.github.com/introduction/flow/ for more details

Image source: GitHub

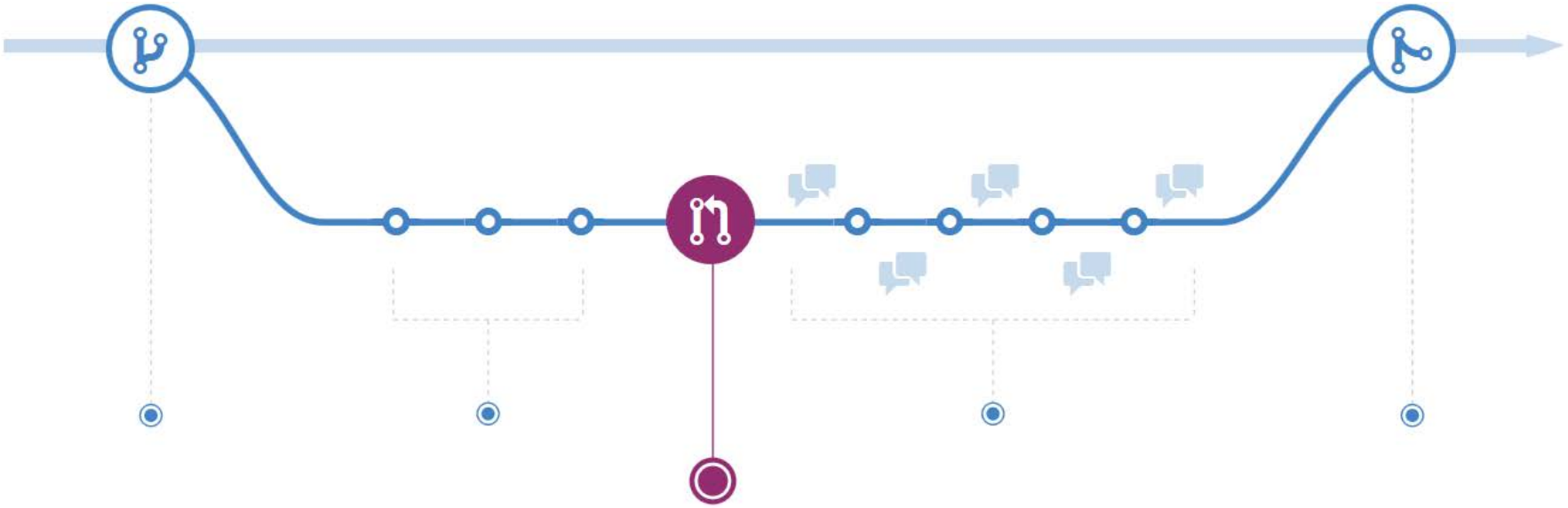
GitHub flow | Add commits



- Commits naturally create a transparent development history to keep track of your work
  - Write messages in present tense as if they were commands ➡ Fix bug #1
- Keep commits small
  - One commit per logical change: fix a typo, fix a bug, add a function…
  - This way commit messages make sense

Image source: GitHub

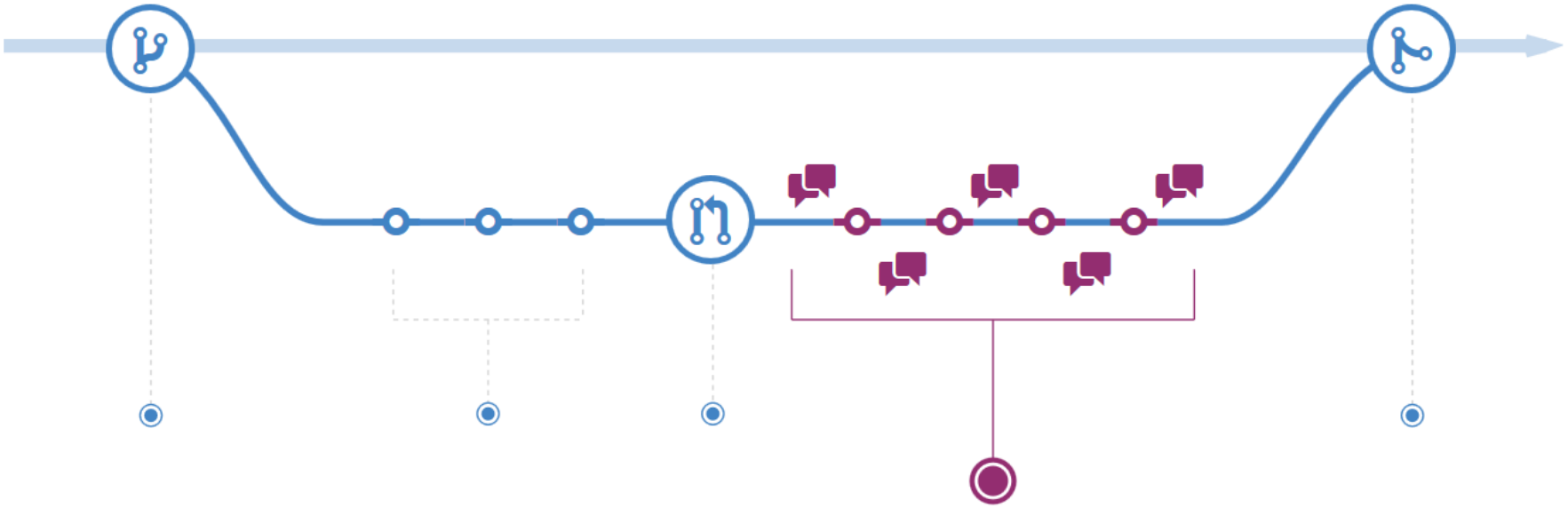
GitHub flow | Open a pull request



- Pull requests help start code review and conversation about proposed changes before merging them into the master branch

Image source: GitHub

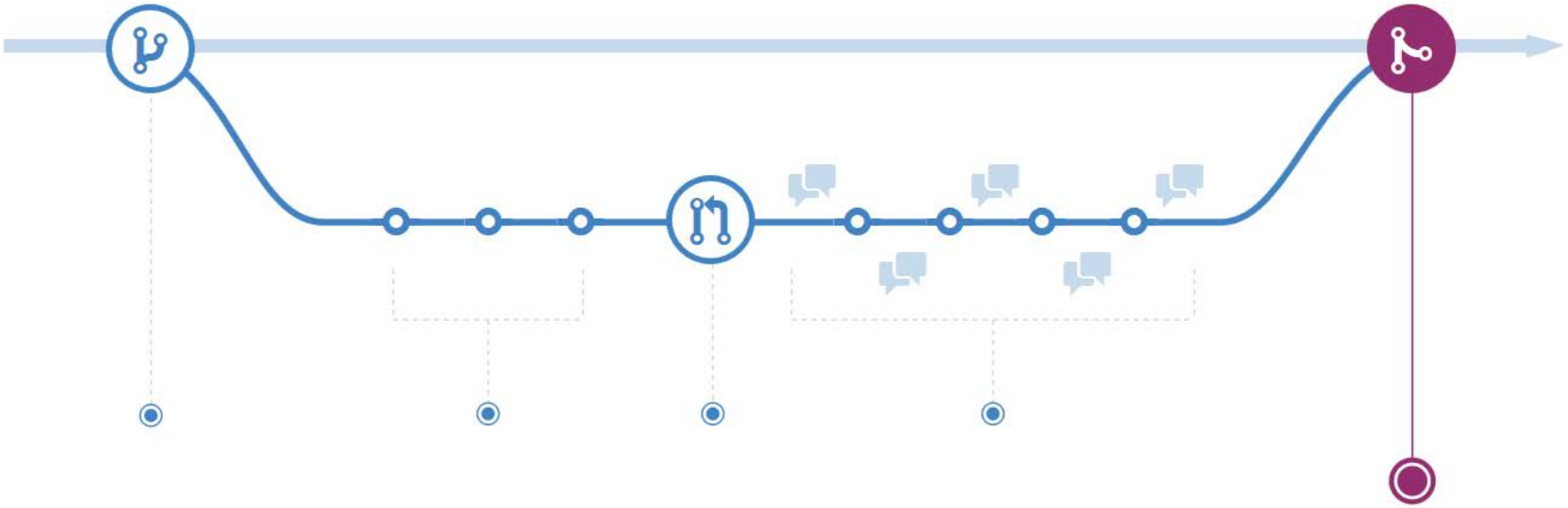
GitHub flow | Discuss and review your code



- The person or team reviewing your request will give you feedback
- You can continue making commits to correct any issues detected

Image source: GitHub

GitHub flow | Merge and deploy



- Once the pull request has been accepted it can be safely merged
  - If done locally you will have to sync the changes with the server afterwards
- Conflicts may occur during merges
  - Single user makes modifications in multiple machines
  - Single user changes directly on GitHub
  - Several users work on the same files simultaneously

Image source: GitHub

References

- Git, *Git documentation,* 2015.
  http://git-scm.com/doc/
- Atlassian, *Getting Git right,* 2015.
  https://www.atlassian.com/git/
- Wildbit, LLC, *Beanstalk guides*, 2007 – 2015.
  http://guides.beanstalkapp.com
- GitHub, *GitHub guides*, 2015.
  https://guides.github.com
- Git Tower, *Learn version control with Git*, 2015.
  http://git-tower.com/learn/
- S. Mutch, *Version control tutorial*, 2013.
  http://smutch.github.io/VersionControlTutorial/index.html