

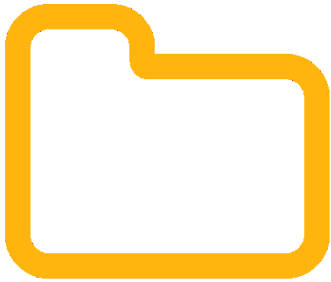
Git:

A Distributed
Versioning Control
System

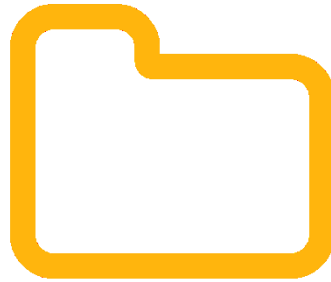


Version Control

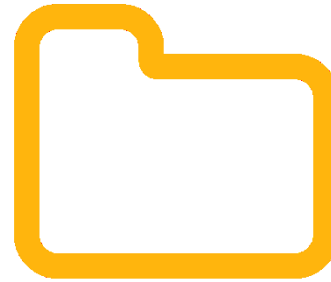
Version Control



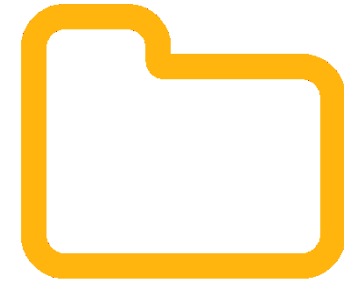
Project



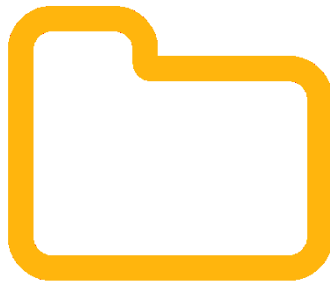
Project V2



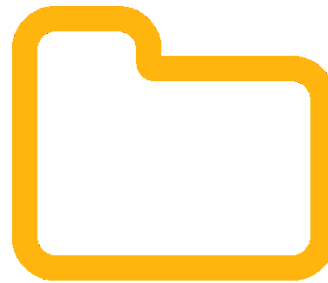
Project V3



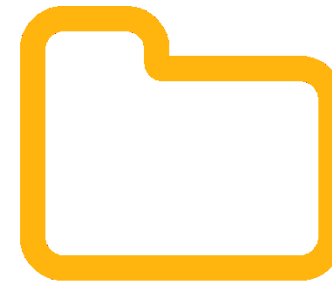
Project Final Version



Project Final Version 2

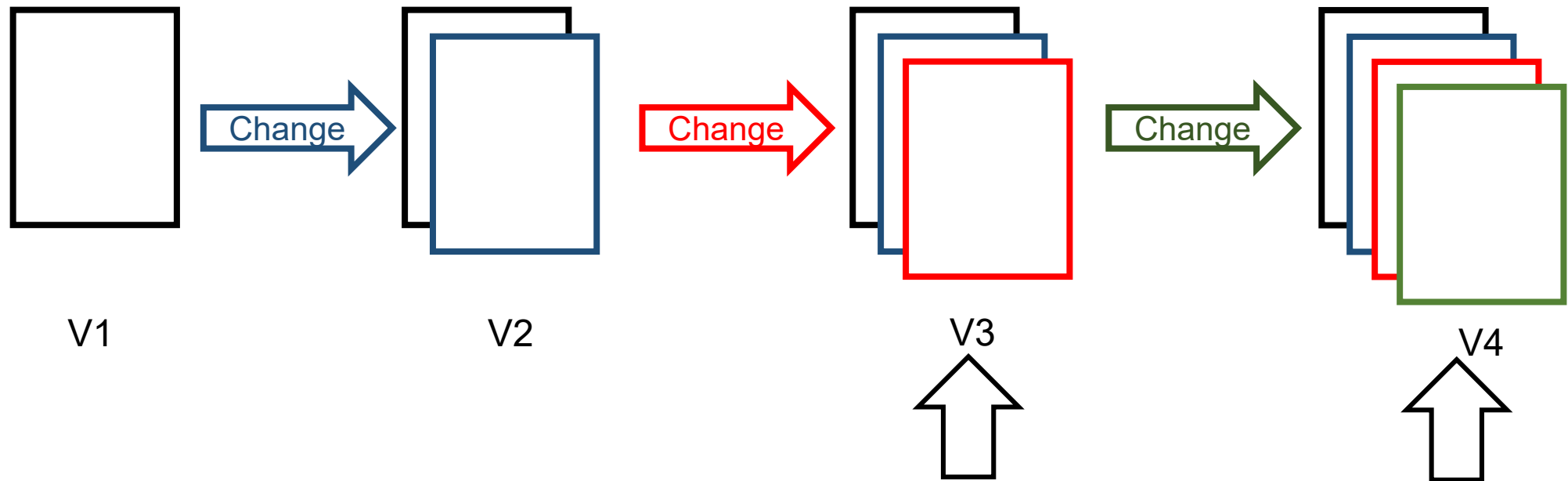


Project this is really the
final final version ever



Project Final Version 4

Version Control



(image credit: Herval on Flickr)

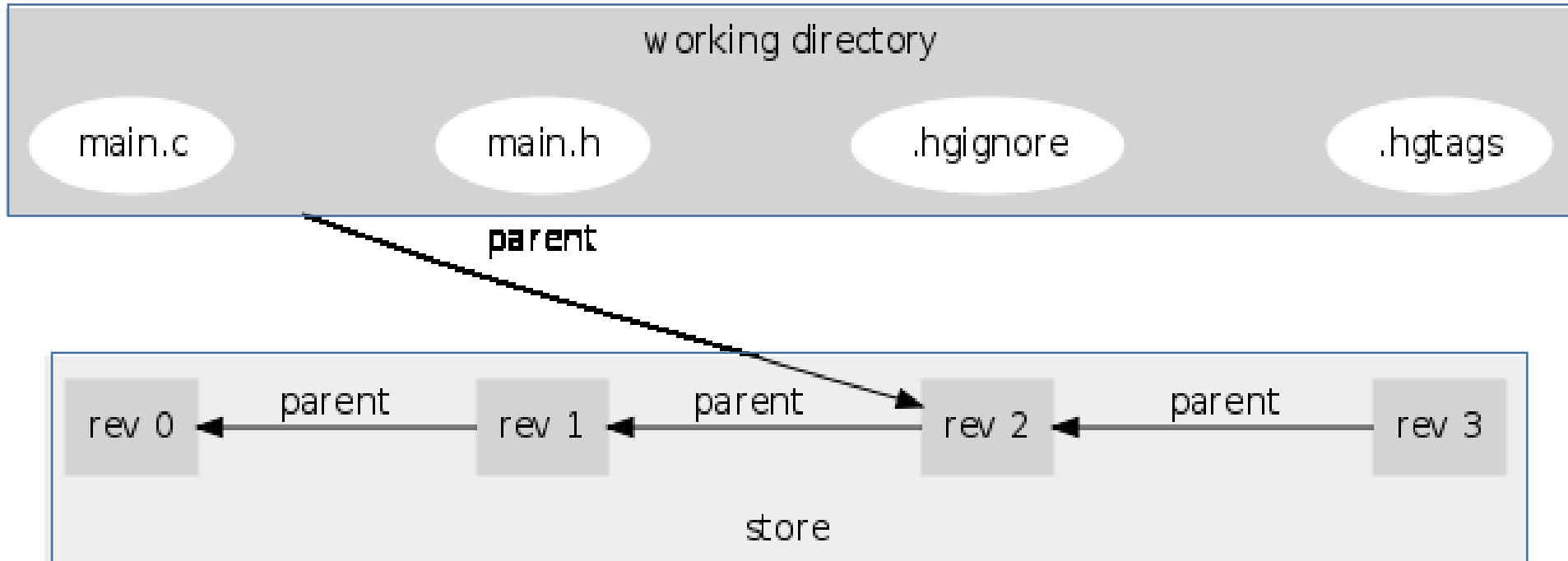
Version Control

A version control system allows you to **track the history of a collection of files** and **revert to previous** or more recent versions of the files. Each version captures a snapshot of the files at a certain point in time.

- ✓ Track changes (What, who, how, when)
- ✓ Revert errors/mistakes
- ✓ Coordinate team efforts
- ✓ Work in parallel
- ✓ Backup of your code

Repository

A repository = **working directory** + **a store**

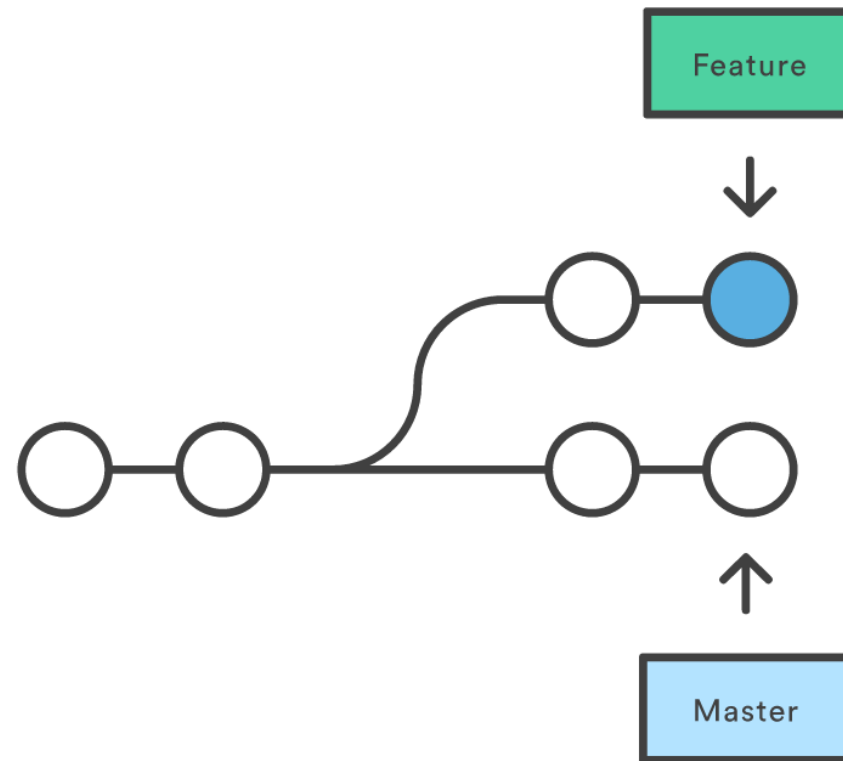


Working Directory









- The **working directory** contains a copy of the project's files at a given point in time (e.g.. rev 2), ***ready for editing***.
- It is the private space of the developer.

Store

The **store** contains the ***complete*** history of the project!



Ricardo > PhD > Multimedia > android

Name	Date modified	Type	Size
 .git	6/23/2016 2:23 PM	File folder	
 .settings	6/22/2016 10:48 AM	File folder	
 src	6/23/2016 12:08 PM	File folder	
 target	10/19/2015 8:37 AM	File folder	
 .classpath	6/22/2016 10:48 AM	CLASSPATH File	2 KB
 .gitignore	10/19/2015 8:00 AM	Text Document	1 KB
 .project	10/19/2015 8:00 AM	PROJECT File	1 KB
 pom.xml	6/23/2016 2:23 PM	XML Document	4 KB

Ricardo > PhD > Multimedia > android

Name

Date modified

Type

Size



.git

Store

6/23/2016 2:23 PM

File folder



.settings

6/22/2016 10:48 AM

File folder



src

6/23/2016 12:08 PM

File folder



target

10/19/2015 8:37 AM

File folder



.classpath

6/22/2016 10:48 AM

CLASSPATH File

2 KB



.gitignore

10/19/2015 8:00 AM

Text Document

1 KB



.project

10/19/2015 8:00 AM

PROJECT File

1 KB



pom.xml

6/23/2016 2:23 PM

XML Document

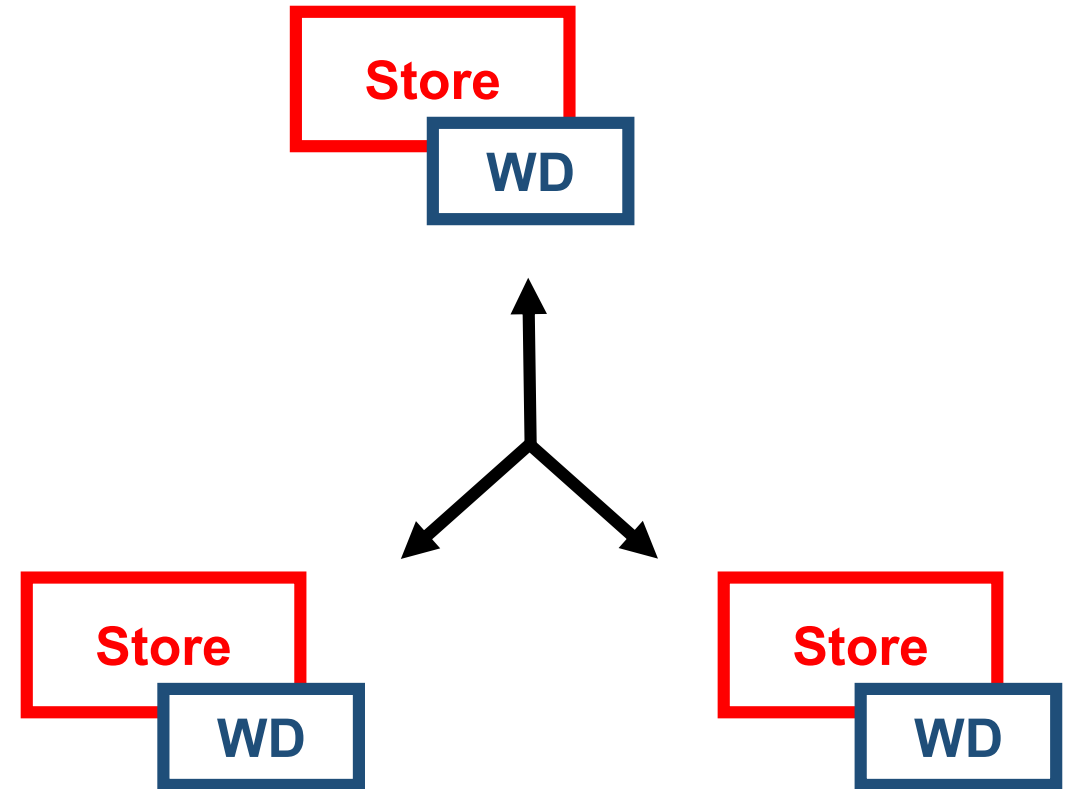
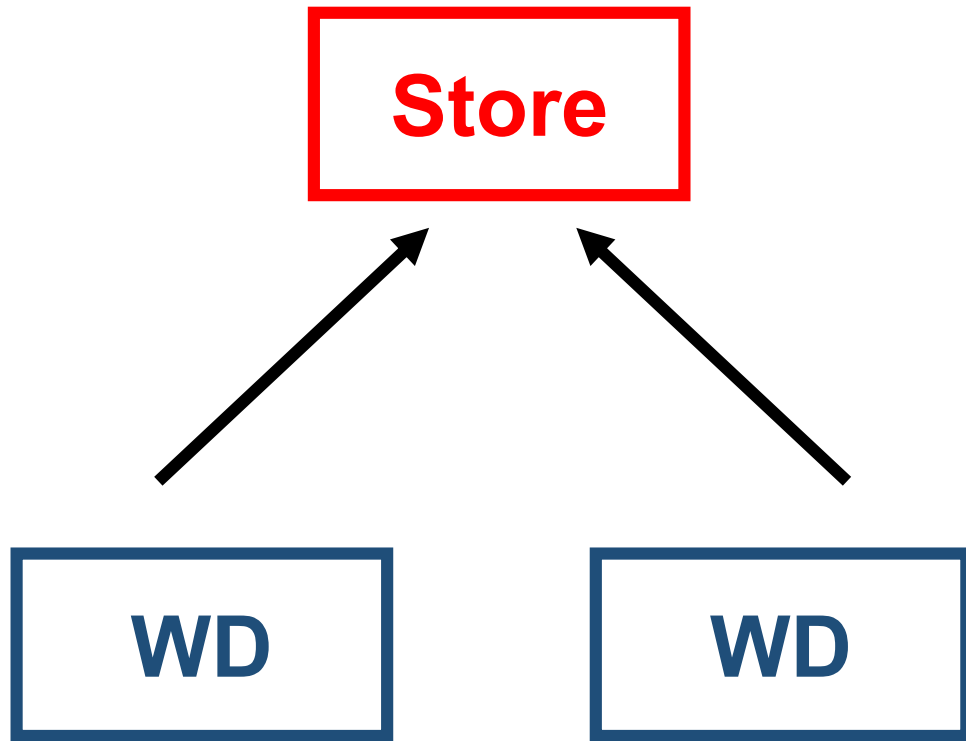
4 KB

Ricardo > PhD > Multimedia > android

Name	Date modified	Type	Size
.git	6/23/2016 2:23 PM	File folder	
.settings	6/22/2016 10:48 AM	File folder	
src	6/23/2016 12:08 PM	File folder	
target	10/19/2015 8:37 AM	File folder	
.classpath	6/22/2016 10:48 AM	CLASSPATH File	2 KB
.gitignore	10/19/2015 8:00 AM	Text Document	1 KB
.project	10/19/2015 8:00 AM	PROJECT File	1 KB
pom.xml	6/23/2016 2:23 PM	XML Document	4 KB

Working
Directory

[CVS] Centralized vs Distributed [DCVS]



[CVS] Centralized vs Distributed [DCVS]

- Single point of failure
- Linear process
- Slow
- One error affects everyone
- Network connection mandatory for everything
- Easier to administrate and control backups, access and progress

- Network connection required for collaboration purposes
- Each developer can have their own repository
- Integration is often delegated

Resources on CVS and DCVS

- https://www.youtube.com/watch?v=_yQIKEq-Ueg
- <https://www.youtube.com/watch?v=eDRt9wl15ml>
- https://www.youtube.com/watch?v=gnz60_N5kZQ
- An article on more details on version control systems [here](#)
- An article on migration from CVS to DVCS [here](#)

Git



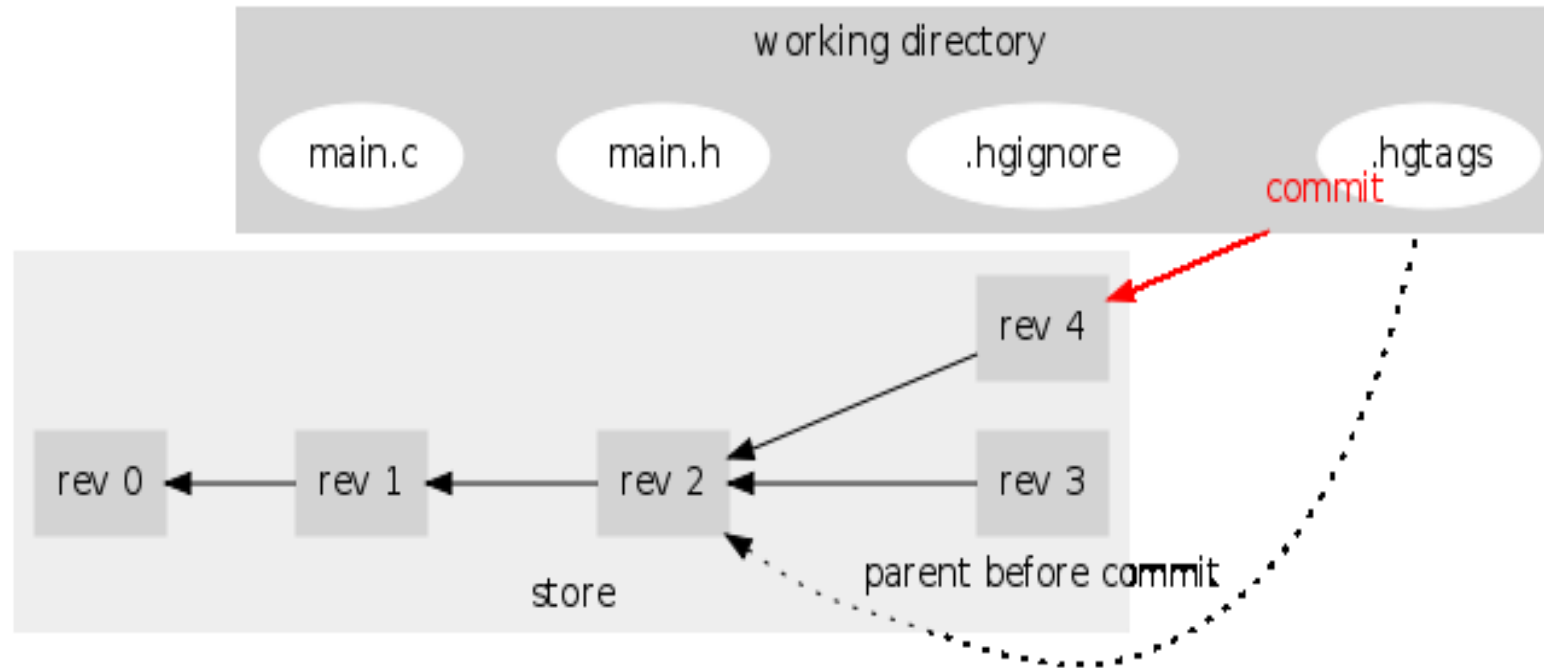
- A **Distributed Version Control System** developed by **Linus Torvalds** for Linux kernel development in 2005.
- Focuses on: speed, simple design, strong support for non-linear development, fully distributed, ability to handle large projects efficiently.
- Git is released under the [GNU GPL V 2.0](#)

Commit

- A **commit** is an **atomic collection of changes** to files in a repository. It contains all recorded local modification that lead to a **new revision of the repository**.
 - ✓ Commit ID
 - ✓ Revision number
 - ✓ List of changes
 - ✓ Committer, comments, date
 - ✓ Name of the branch
 - ✓ Other information

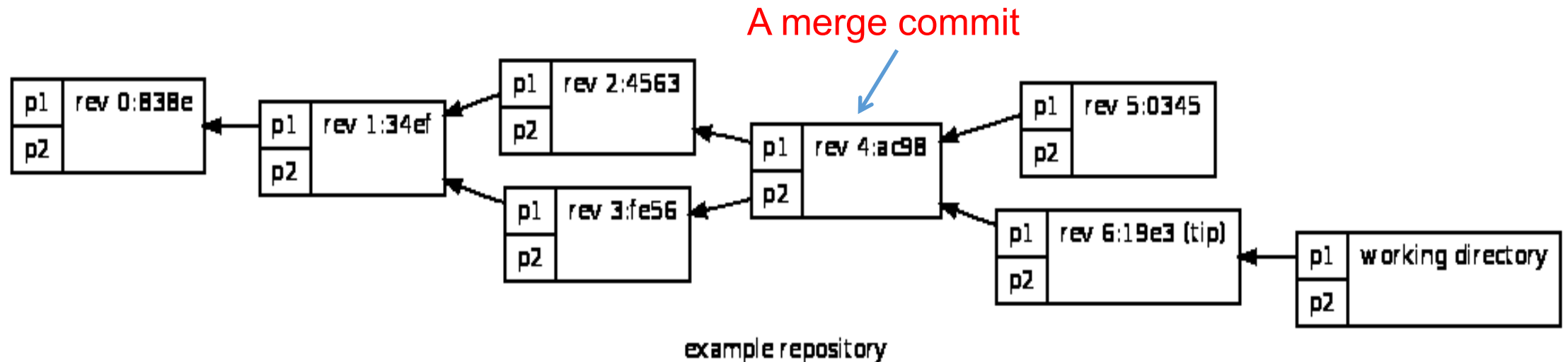
Revision

The state of the working directory relative to its parent is recorded as a new **revision**.



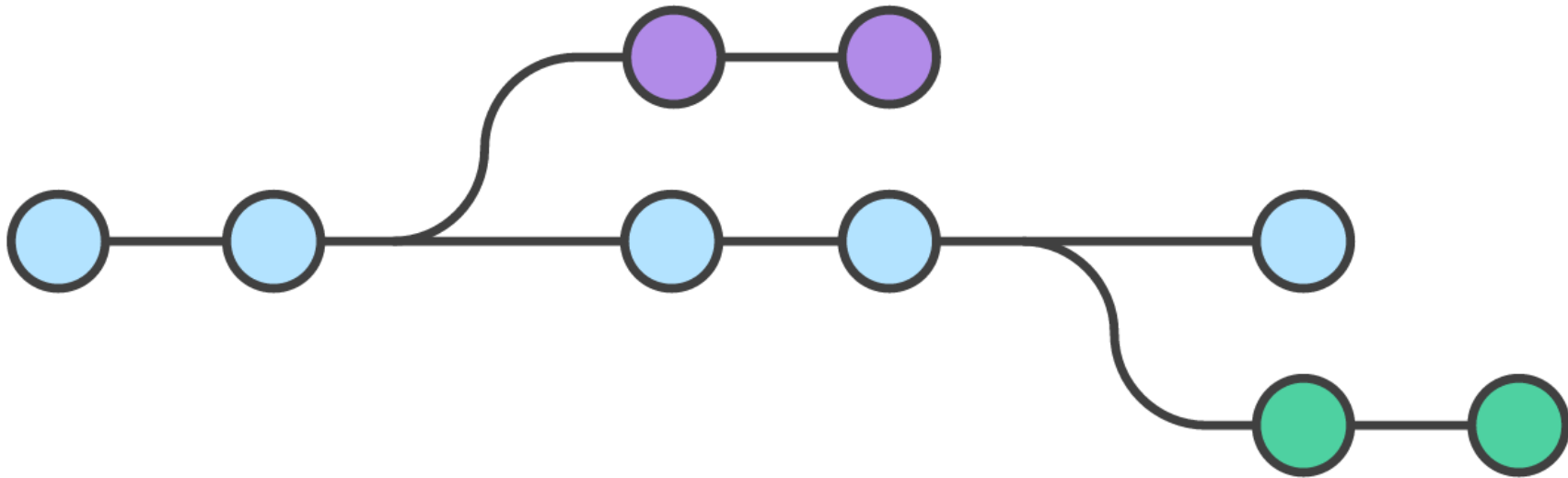
Commit Graph

Revisions of a whole project can be thought as a graph, where new nodes (commits) are added when a developer commits, merges, pulls, etc.



Branch

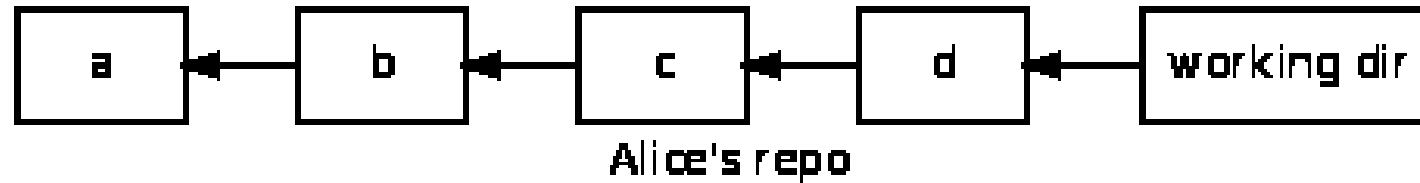
A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process.



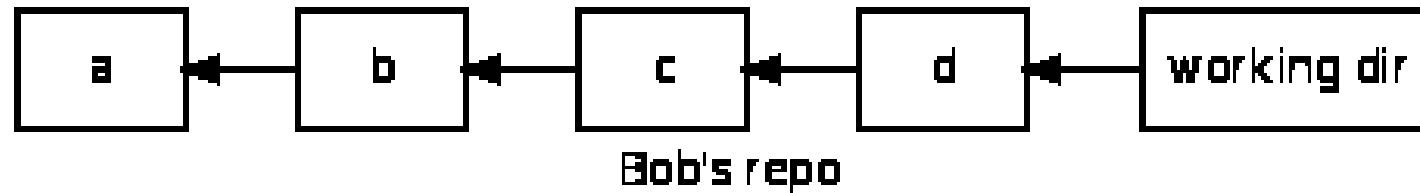
From: <https://www.atlassian.com/git/images/tutorials/collaborating/using-branches/hero.svg>

git clone

Let's suppose that a user *Alice* has a repository that looks like:

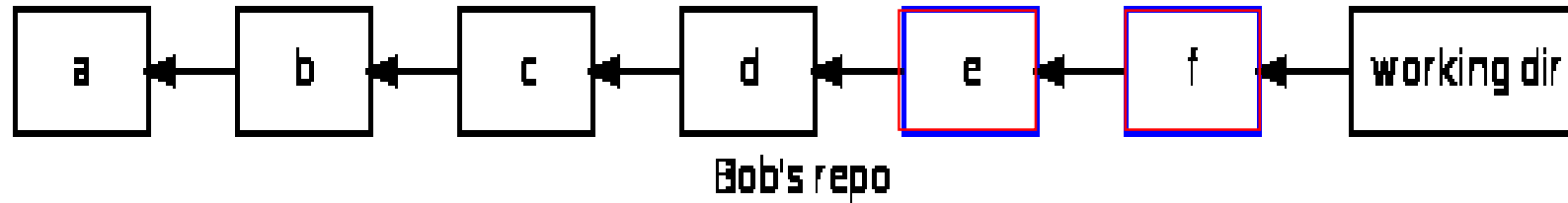


Bob, another developer, **clones** this repository, and ends up with a complete, **independent, local** copy of Alice's store :

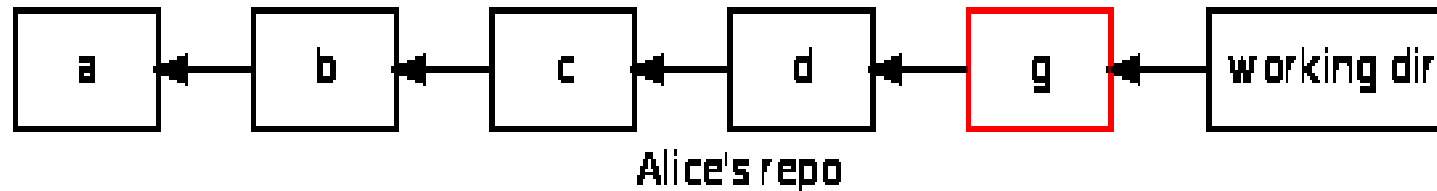


git commit

Bob can now work independently of *Alice*. He then **commits** two changes **e** and **f**:



Alice then can make her own change **g** in parallel, which causes her repository store to diverge from Bob's.



Committing changes

- When you perform **git commit** a new revision is created in the repository, based on the state of the working directory
- During the commit, *a description for the change made must be provided*

The Commit Message

- You should treat the first line of the comment like the **subject line** of an email message.
- The subject should:
 - **(i)** be descriptive,
 - **(ii)** be less than 65 characters long, and
 - **(iii)** give a quick idea of the content of the commit.

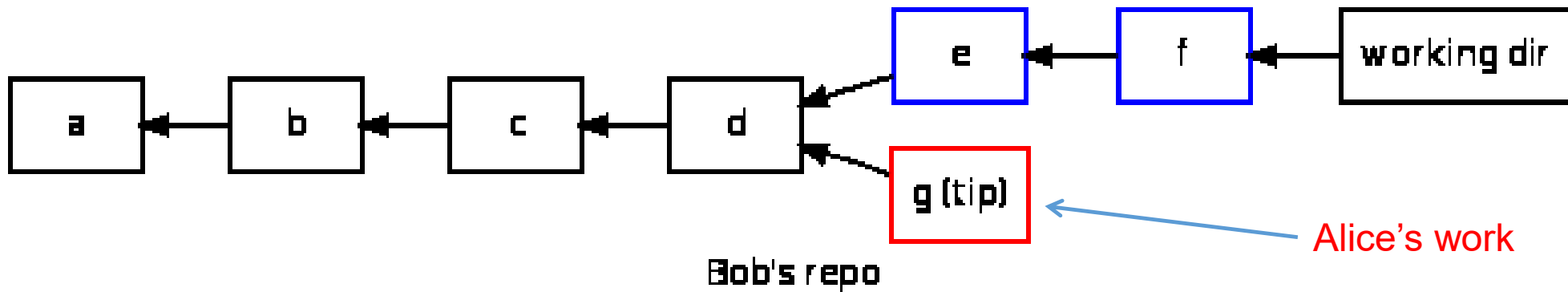
How to write a good commit message

- Separate subject from body with a blank line
- Limit the subject line to 50 characters
- Capitalize the subject line
- Do not end the subject line with a period
- Use the imperative mood in the subject line
- Wrap the body at 72 characters
- Use the body to explain *what* and *why* vs. *how*

<http://chris.beams.io/posts/git-commit/>

git pull

Allows you to merge upstream changes into your local repository.

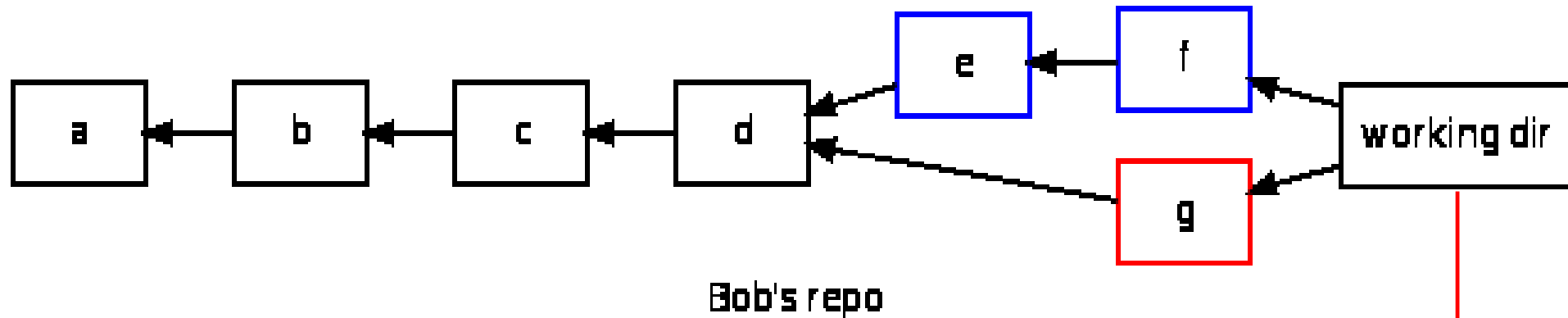


Pulling Rules

- **Finish your work and commit it before you start pulling in stuff from the outside world.**
 - What you are doing might be inconsistent with things that other people have done.
- **It is better to know what other people have done, before pulling their work.**

git merge

- *Bob* then does a **merge**, which combines the last change he was working on (**f**) with Alice's change (**g**). Now, his working directory has two parent revisions (**f** and **g**):

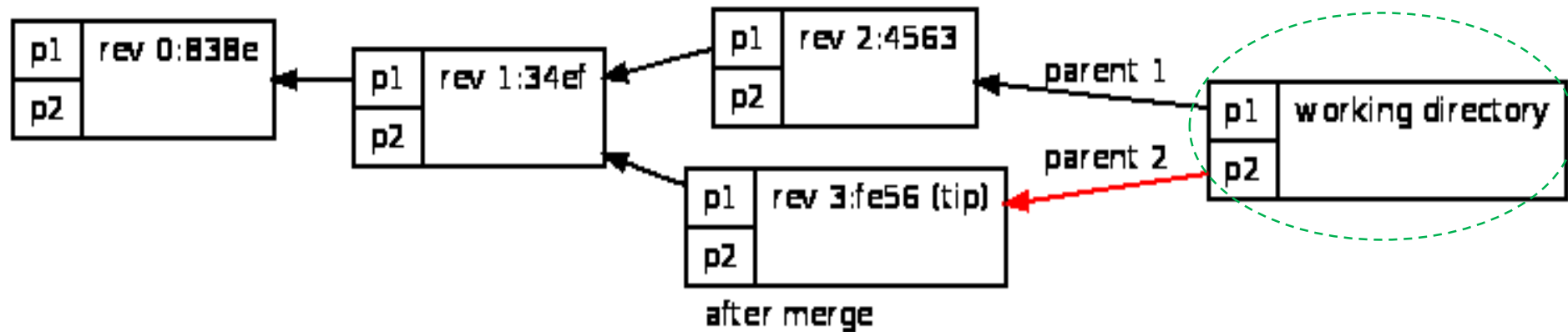
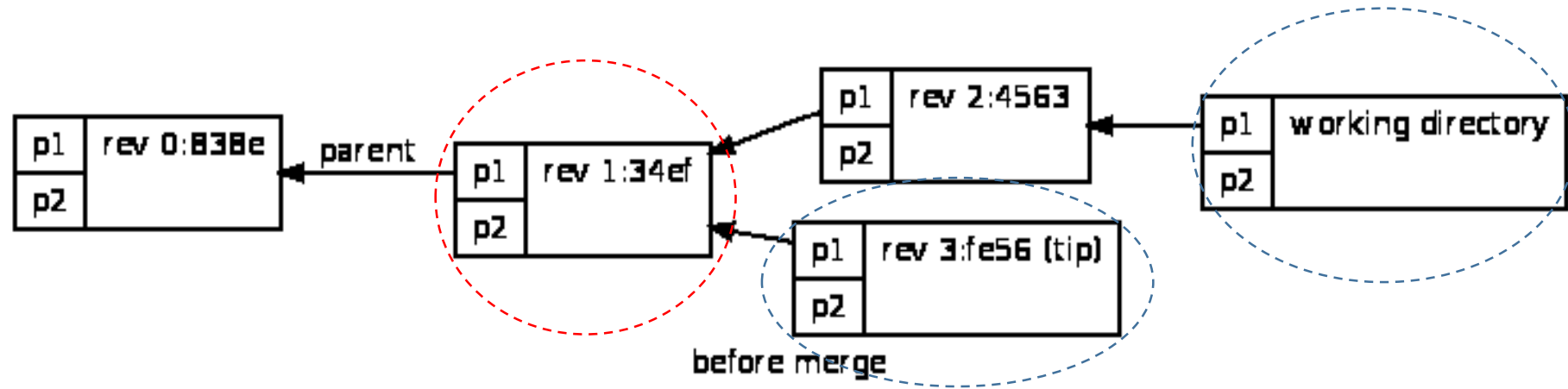


Now his working directory reflects Alice's work

Merging Actions

- In general, Merging is the process of joining points on two branches into one.
- Usually you will want to merge **the last changes** into your **working directory**.

Merge Example



Merge (Stage A)

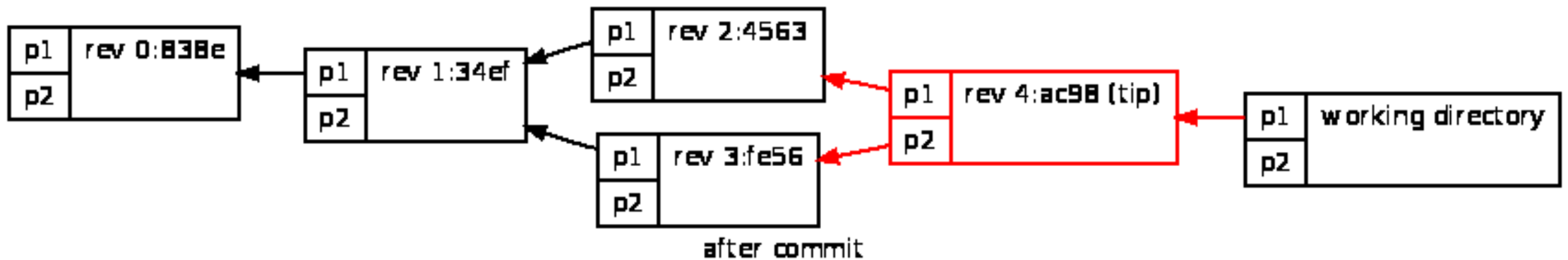
- The first step of this process is tracing back through the history of changesets and finding the 'common ancestor' of the two versions that are being merged.
- This is done on a repository-wide and a file-by-file basis.

Merge (Stage B)

- For files that have been changed in both changesets, **a three-way merge is attempted** using a merge program to add the changes made remotely into the changes made locally.
- If there are **conflicts** between these changes, the user is usually prompted to interactively resolve them.

Merge (Stage C)

After you've completed the merge, you still need to **commit your changes**



git push

A **push** propagates changes from a local repository to a remote one.

A (simple) routine

1. git clone <repository>
2. <make changes>
3. git commit
4. git pull
5. <Check everything works OK>
6. git push
7. Go to 2

A cheat sheet on git commands

<http://rogerdudler.github.io/git-guide/>

git - the simple guide

just a simple guide for getting started with git. no deep shit ;)

 Tweet 4,747

by Roger Dudler

credits to @tfnico, @fhd and Namics

this guide in deutsch, español, français, indonesian, italiano, nederlands, polski, português, русский, türkçe,

မြန်မာ, 日本語, 中文, 한국어 Vietnamese

please report issues on github



download the
cheat sheet
now. it's free!



want a simple
but powerful
git client for
your mac?



Checkout a repository

- `git clone /path/to/repository`
- `git clone username@host:/path/to/repository`

Add files and commit changes

- `git add <filename>` (the file must already exist)
- `git add *`
- `git commit -m "Commit message"`

Pushing changes

- `git push origin master` (master is a branch)
- `git remote add origin <remote location>` (adds a link to a remote repository)

Branching

- `git checkout -b name_of_branch`
- `git checkout master` (go back to master)
- `git branch --d name_of_branch` (delete branch)
- `git push origin <branch>` (push changes to a particular branch)

Update and merge

- git pull (get recent changes from remote repository)
- git merge <branch> (merge another branch into your active branch)

Log

- `git log` (get info about last commits)
- `git log --author=bob` (get info about commits done by bob)
- `git log --pretty=oneline` (pretty print)
- `git log --help` (for more options)

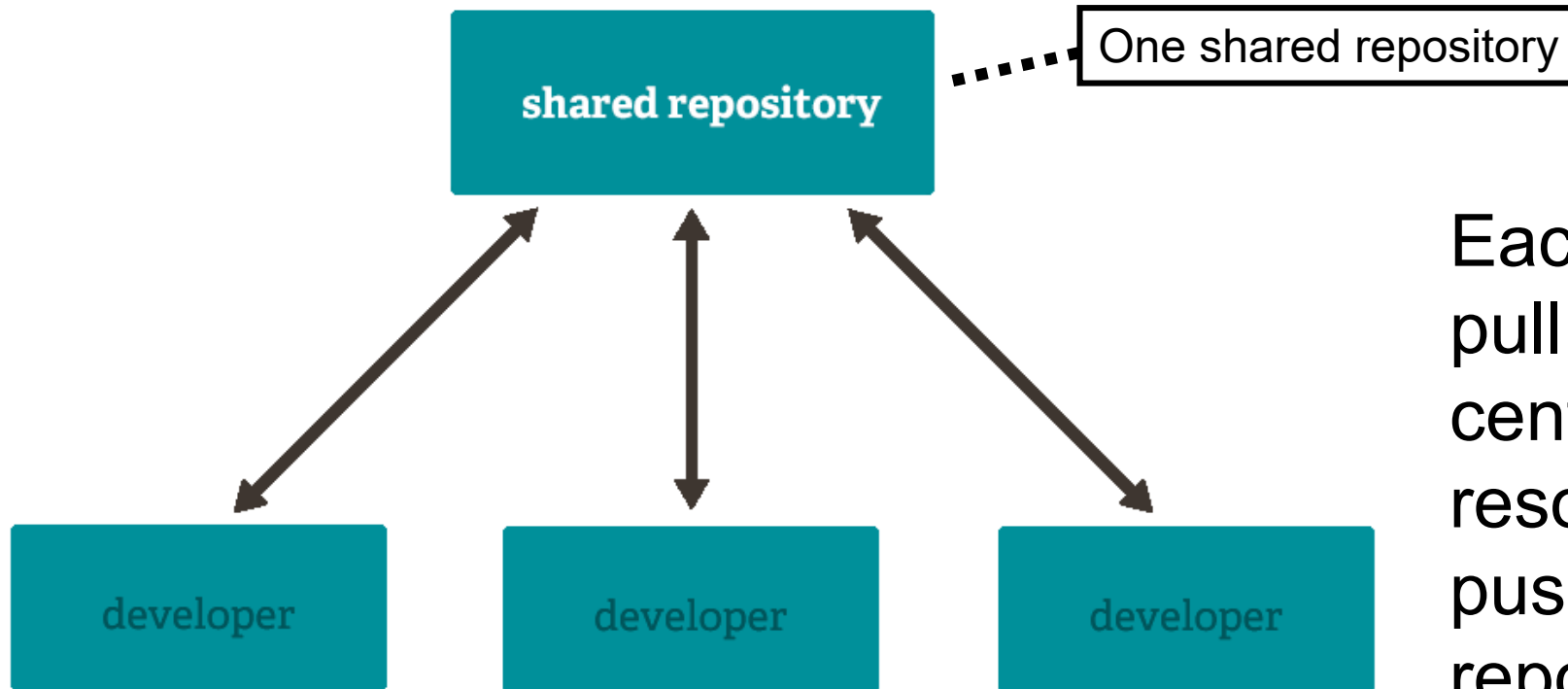
Log

- `git log --follow <file path>` (listing the history of a file beyond renames)
- `git status` (Show the working tree status)
- `git status --help`
- `git log --grep="term"` (Retrieves the commits with messages containing the word "term")

Workflows

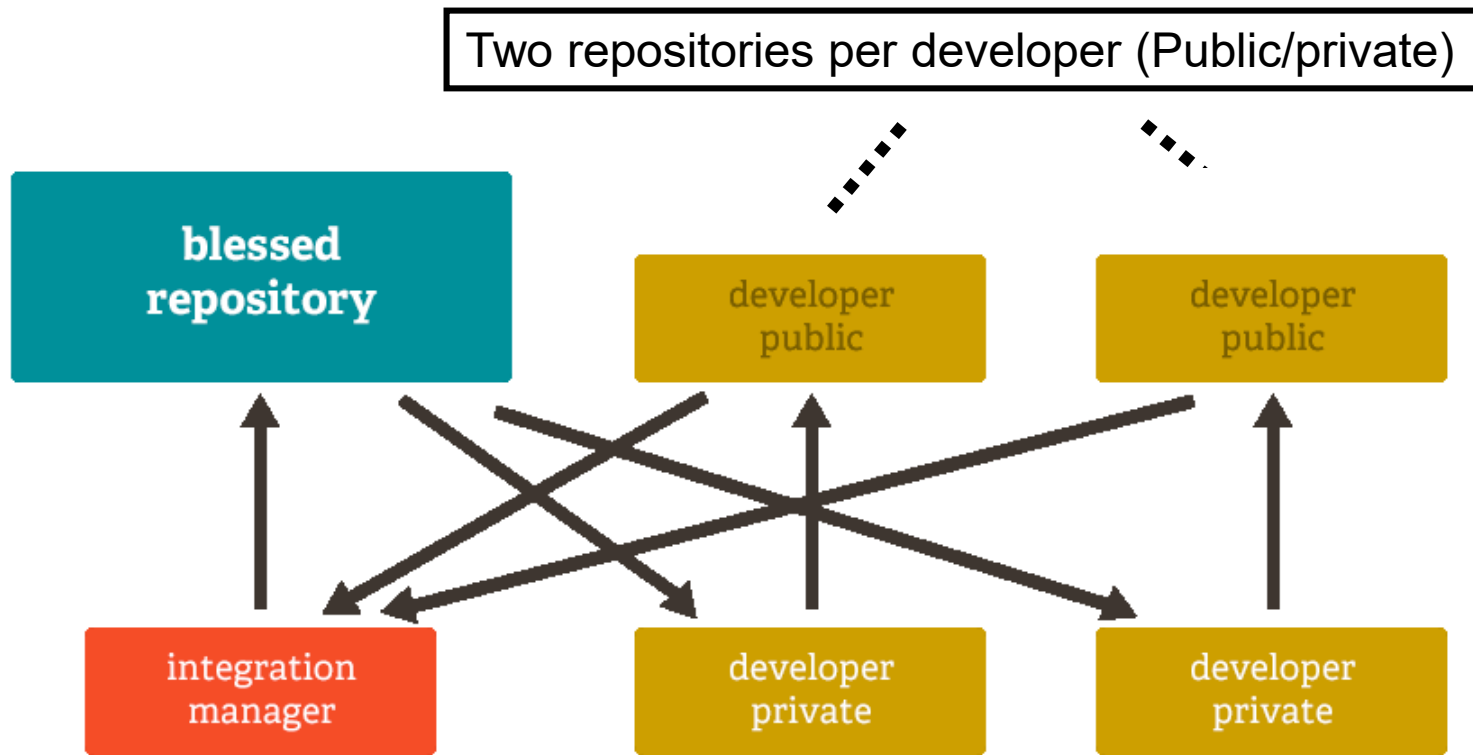
- A **workflow** is the model followed by a development team in order to support collaborative work.
- It outlines the organization and structure of the team.
- Describes the steps required to carry out, synchronize and integrate development activities in a software project.

Centralized workflow



Each developer has to pull any changes in the central repository and resolve conflicts before pushing to the central repository.

Integration Manager Workflow

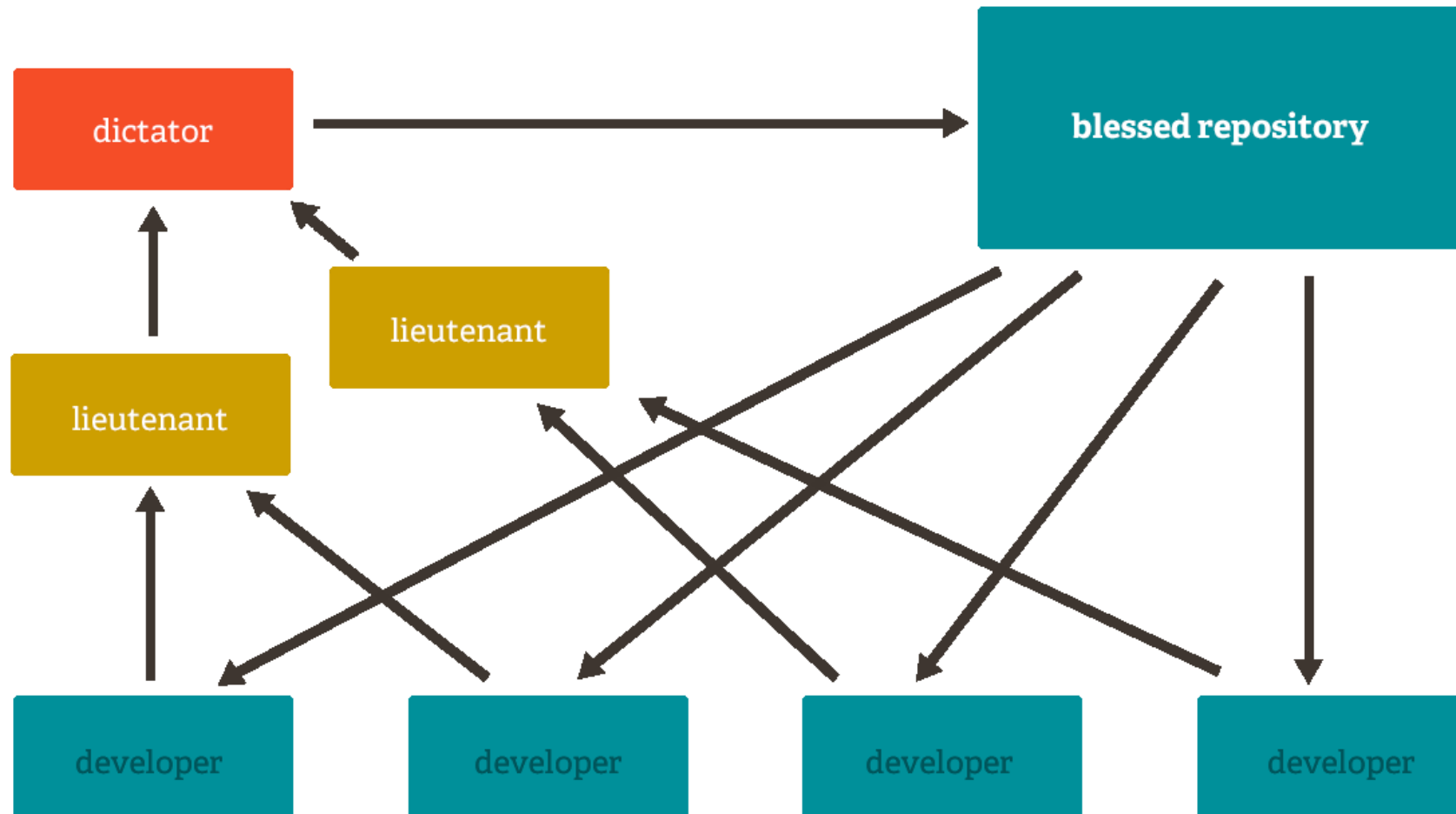


Each developer makes push to his/her own public repository, and push to the blessed/central repository.

Conflicts are solved by the integration manager.
(Centralized integration)

Each developer makes pull and push to his/her own public repository.

Dictator and Lieutenants Workflow



From: <https://git-scm.com/about/distributed>

Dictator and Lieutenants Workflow

- Distributed development and integration.
- One central/blessed repository: Everyone makes pull from this repository. Only the dictator can push to it.
- Dictator repository: Merges changes in the lieutenant repositories and solves any arising conflicts.
- Lieutenant repositories: Merges changes for some (not all) developers and solves conflicts. A developer pushes changes to the assigned lieutenant repository.

Got 15 minutes and want to learn Git?

- <https://try.github.io/levels/1/challenges/1>

1.1 · Got 15 minutes and want to learn Git?

Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.

Our terminal prompt below is currently in a directory we decided to name "octobox". To initialize a Git repository here, type the following command:

```
➔ git init
```



-
- <http://www.vogella.com/tutorials/Git/article.html#firstgit>
 - <https://es.atlassian.com/git/tutorials/>

Remotes

- *Remotes* are URLs in a Git repository to other repositories that are hosted on the Internet, locally or on the network.

GitHub

- A provider of hosting services for Git repositories
- Provides a web-based interface
- Provides additional features such as a issue tracker and a wiki for the project

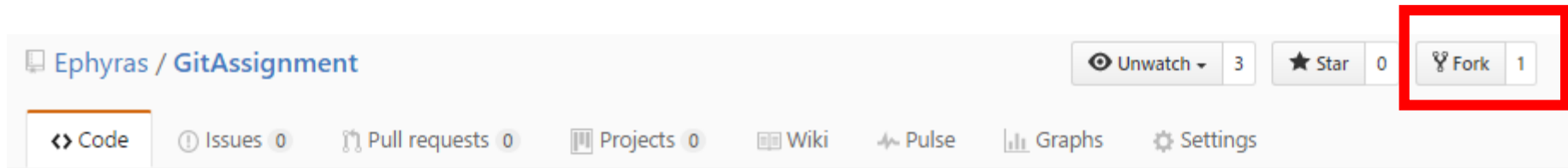
Fork (in GitHub)

Will allow you to copy all the contents of another GitHub repository into your GitHub account.

Useful to contribute to other software projects in GitHub.

You must issue a “pull request” if you want to integrate your changes into the repository you forked from.

Fork (in GitHub)



Clients

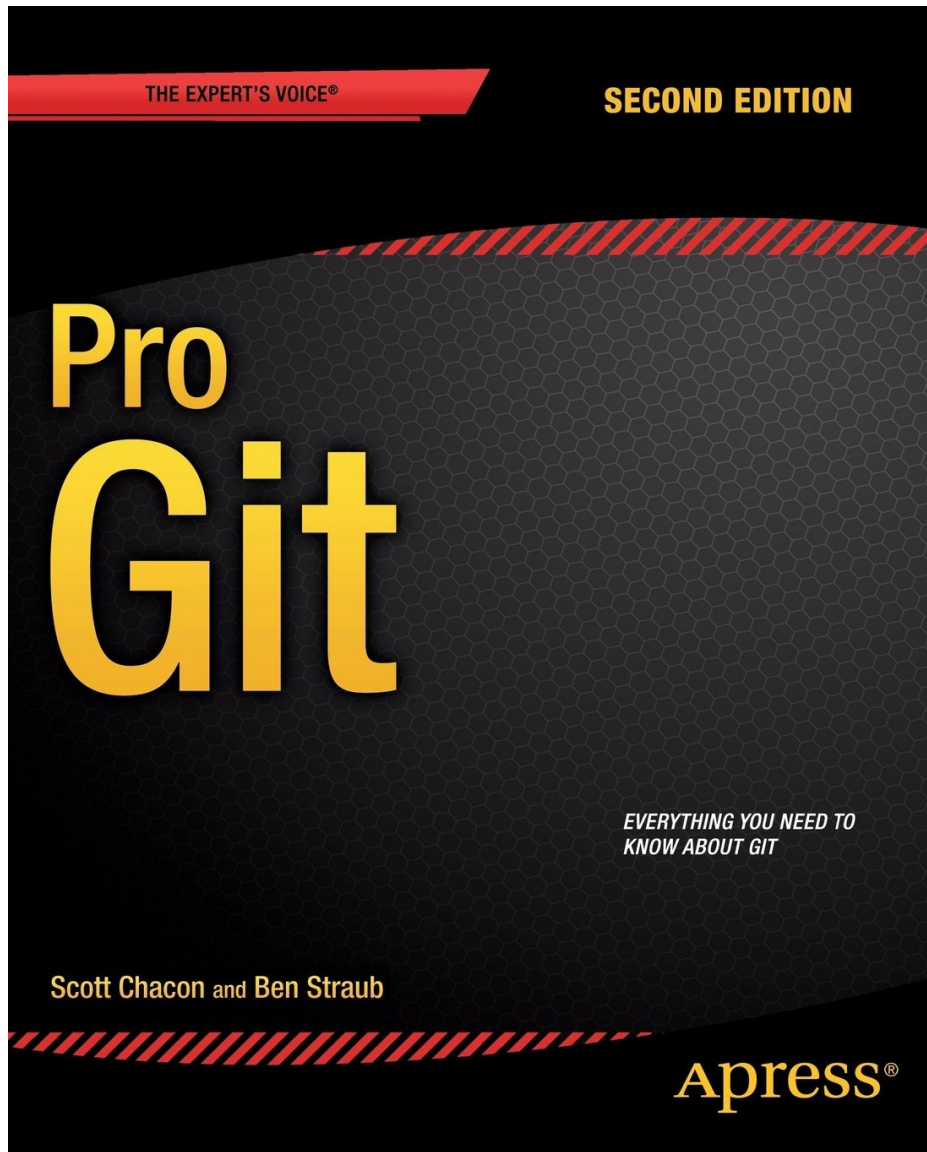
Provide an interface to browse and perform operations on Git repositories.

- <https://windows.github.com/>
- <https://mac.github.com/>
- <https://git-scm.com/download/gui/linux>
- <https://www.sourcetreeapp.com/>
- <http://www.collab.net/products/giteye>

More clients [here](#)

Tutorials and resources

- <https://guides.github.com/activities/hello-world/>
- <https://www.youtube.com/watch?v=Yq32Ifx0bXw>
- <http://www.vogella.com/tutorials/JUnit/article.html>



From: <http://ecx.images-amazon.com/images/I/71AgQkGoyoL.jpg>

Pro Git

This book contains an in depth introduction to Git, basic Git terminology and usage. It also includes a section dedicated entirely to GitHub.

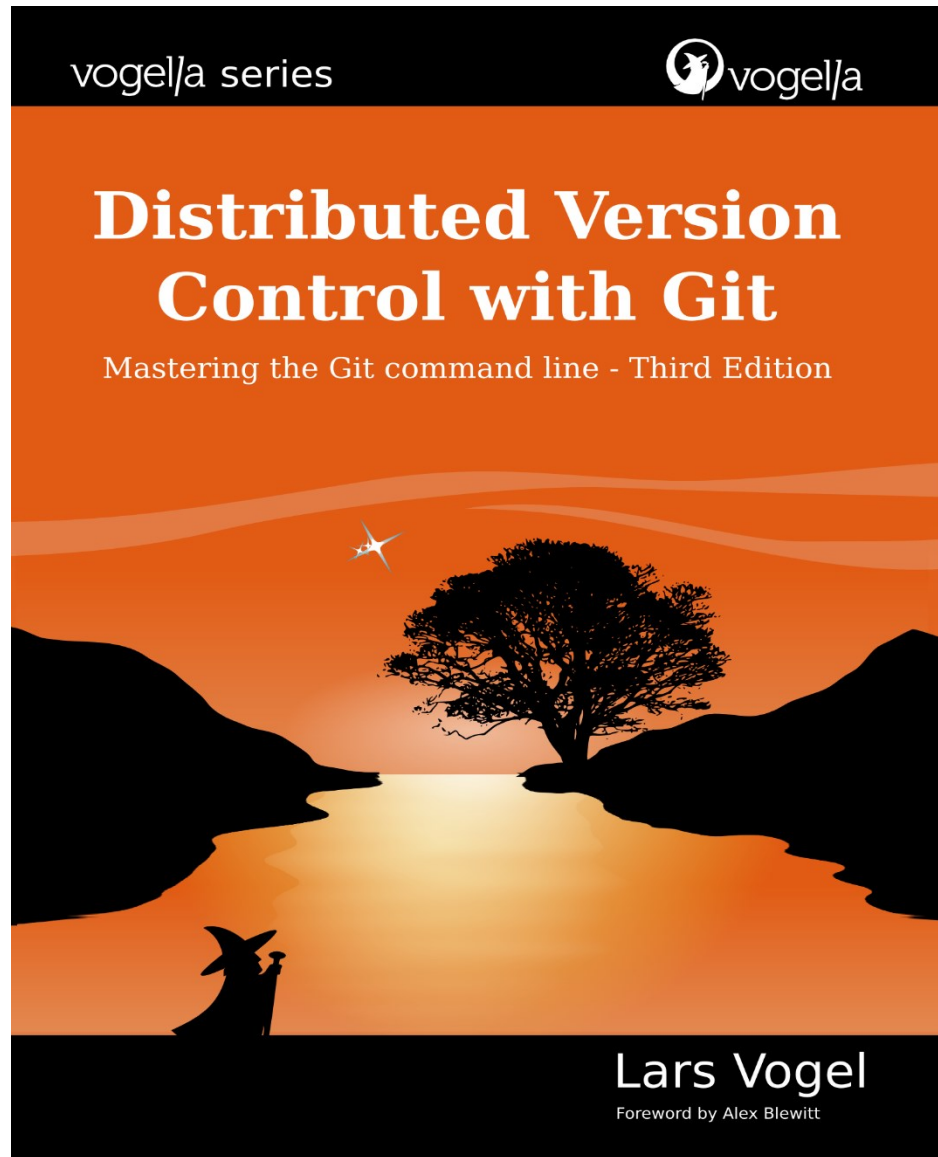
Authors: [Scott Chacon](#) and [Ben Straub](#)

Year 2014

Second Edition

Free available at

<https://git-scm.com/book/en/v2>
<https://progit.org/>



Distributed Version Control with Git: Mastering the Git command line

This book contains an introduction to DVCSs, basic Git terminology and usage. As well as tips on Git. It also includes how to use remote repositories (i.e. GitHub, Bitbucket)

Author [Lars Vogel](#)

Year 2014

Third Edition

[Paperback edition](#)

[Ebook](#)

From: http://blog.vogella.com/wp-content/uploads/2014/12/Git_CoverKindle.png