Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

GIT for Beginners

Anthony Baire

Universit´e de Rennes 1 / UMR IRISA

June 2, 2022



This tutorial is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 France License 1 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Objectives

*•* Understand the basics about version control systems

*•* Getting started with GIT

*•* working with a local repository

*•* synchronising with a remote repository

*•* setting up a server

2 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Summary

1. About Version Control Tools

2. Overview of GIT

3. Working locally

4. Branching & merging

5. Interacting with a remote repository

6. Administrating a server

7. Working with third-party contributors

8. Extras

3 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Part 1.

About Version Control Tools

*•* Definition

*•* Use cases

*•* Base concepts

*•* History

4 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras What is a version control system ?

From: http://en.wikipedia.org/wiki/Revision\_control

*Revision control* [...] *is the management of changes to documents, computer programs, large web sites, and other collections of information.*

*Changes are usually identified by a number or letter code, termed the ”revision number”* [...]. *For example, an initial set of files is ”revision 1”. When the first change is made, the resulting set is ”revision 2”, and so on.*

*Each revision is associated with a timestamp and the person making the change.*

*Revisions can be compared, restored, and with some types of files, merged.*

5 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Use case 1: keeping an history

The life of your software/article is recorded from the beginning *•* at any moment you can revert to a previous revision 1

*•* the history is browseable, you can inspect any revision 2 *•* when was it done ?

*•* who wrote it ?

*•* what was changed ?

*•* why ?

*•* in which context ?

*•* all the deleted content remains accessible in the history

1let’s say your not happy with your latest changes

2this is useful for understanding and fixing bugs

6 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Use case 2: working with others

VC tools help you to:

*•* share a collection of files with your team

*•* merge changes done by other users

*•* ensure that nothing is accidentally overwritten

*•* ~~know who you must blame when something is broken~~

7 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Use case 3: branching

You may have multiple variants of the same software, materialised as **branches**, for example:

*•* a main branch

*•* a maintainance branch *(to provide bugfixes in older releases) •* a development branch *(to make disruptive changes)*

*•* a release branch *(to freeze code before a new release)*

VC tools will help you to:

*•* handle multiple branches concurrently

*•* merge changes from a branch into another one

8 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Use case 4: working with external contributors

VC tools help working with third-party contributors:

*•* it gives them visibility of what is happening in the project

*•* it helps them to submit changes (patches) and

it helps you to integrate these patches

*•* forking the development of a software and merging it back into mainline3

3decentralised tools only

9 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Use case 5: scaling

Some metrics4about the Linux kernel (developed with GIT):

*•* about 10000 changesets in each new version

(every 2 or 3 months)

*•* 1000+ unique contributors

4source: the Linux Foundation

10 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Some illustrations

11 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Taxinomy

Architecture:

*•* **centralised** *→* everyone works on the same unique repository *•* **decentralised** *→* everyone works on his own repository

Concurrency model:

*•* **lock before edit** (mutual exclusion)

*•* **merge after edit** (may have conflicts)

History layout:

*•* **tree** (merges are not recorded)

*•* **direct acyclic graph**

Atomicity scope: **file** vs **whole tree GIT** 12 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Other technical aspects

**Space efficiency**: storing the whole history of a project requires storage space *(storing every revision of every file)*

*→* most VC tools use delta compression to optimise the space *(except Git which uses object packing instead)*

**Access method**: A repository is identified with a URL. VC tools offer multiple ways of interacting with remote repositories.

*•* dedicated protocol (*svn:// git://*)

*•* direct access to a local repository (*file://path* or just *path*) *•* direct access over SSH (*ssh:// git+ssh:// svn+ssh://*) *•* over http (*http:// https://*)

13 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Creating new revisions

14 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Creating new revisions

14 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Creating new revisions

14 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Creating new revisions

14 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Creating new revisions

14 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Creating new revisions

14 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

What shall be stored into the repository ?

You should store all files that are not generated by a tool:

*•* source files (.c .cpp .java .y .l .tex . . . )

*•* build scripts / project files (Makefile configure.in

Makefile.am CMakefile.txt wscript .sln)

*•* documentation files (.txt README . . . )

*•* resource files (images, audio, . . . )

You should not store generated files

(*or you will experience many unneccessary conflicts*)

*•* .o .a .so .dll .class .jar .exe .dvi .ps .pdf *•* source files / build scripts when generated by a tool

(like autoconf, cmake, lex, yacc)

15 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Guidelines for committing

*•* commit often

*•* commit independent changes in separate revisions

*•* in commit messages, describe the rationale behind of your changes (*it is often more important than the change itself*)

16 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras History (Centralised Tools)

*•* 1*st* generation *(single-file, local-only, lock-before-edit)*

*•* 1972: **SCCS**

*•* 1982: **RCS**

*•* 1985: PVCS

*•* 2*nd* generation *(multiple-files, client-server, merge-before-commit) •* 1986: **CVS**

*•* 1992: Rational ClearCase

*•* 1994: Visual SourceSafe

*•* 3*rd* generation *(+ repository-level atomicity)*

*•* 1995: Perforce

*•* 2000: **Subversion**

*•* + many others

17 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras History (Decentralised tools)

18 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Part 2.

Overview of GIT

*•* History

*•* Git’s design & features

*•* User interfaces

19 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras History

*•* before 2005: Linux sources were managed with Bitkeeper (proprietary DVCS tool) 5

*•* April 2005: revocation of the free-use licence

(because of some reverse engineering)

*•* No other tools were enough mature to meet Linux’s dev constraints (distributed workflow, integrity, performance). *⇒* Linus Torvald started developing Git

*•* June 2005: first Linux release managed with Git

*•* December 2005: Git 1.0 released

5now open source! (since 2016)

20 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Git Design objectives

*•* distributed workflow (decentralised)

*•* easy merging (merge deemed more frequent than commit) *•* integrity (protection against accidental/malicious corruptions) *•* speed & scalability

*•* ~~ease of use~~

21 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Git Design choices

*•* Easily hackable

*•* simple data structures (blobs, trees, commits, tags)

*•* no formal branch history

(a branch is just a pointer to the last commit)

*•* low-level commands exposed to the user

*•* Integrity

*•* cryptographic tracking of history (SHA-1 hashes)

*•* tag signatures (GPG)

*•* Merging

*•* pluggable merge strategies

*•* staging area (index)

*•* Performance

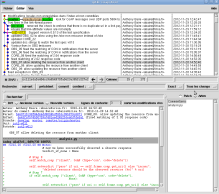
*•* no delta encoding

22 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Git Commands

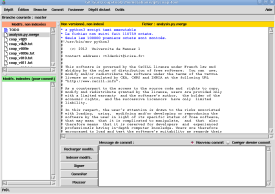
23 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Git GUIs: gitk *→* browsing the history



24 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Git GUIs: git gui *→* preparing commits

25 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras 3rd party GUIs

*•* Turtoise git (Windows)

*•* GitUp, Gitx (MacOS-X)

*•* Smartgit (java, multiplatform)

*•* Eclipse git plugin

26 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Part 3.

Working locally

*•* creating a repository

*•* adding & committing files

*•* the staging area (or index)

27 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Create a new repository

git init *myrepository*

This command creates the directory *myrepository*.

*•* the repository is located in *myrepository*/.git

*•* the (initially empty) working copy is located in *myrepository*/ ✞ ☎ **$ pwd**

/tmp

**$ git init helloworld**

Initialized empty Git repository in /tmp/helloworld/.git/

**$ ls -a helloworld/**

. .. .git

**$ ls helloworld/.git/**

branches config description HEAD hooks info objects refs

✝ ✆

**Note:** The /.git/ directory contains your whole history, **do not delete it**6

6unless your history is merged into another repository

28 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Commit your first files

git add *file*

git commit [ -m *message* ]

✞ ☎ **$ cd helloworld**

**$ echo 'Hello World!' > hello**

**$ git add hello**

**$ git commit -m "added file 'hello'"**

[master (root-commit) e75df61] added file 'hello'

1 files changed, 1 insertions(+), 0 deletions(-)

create mode 100644 hello

✝ ✆

**Note:** “master” is the name of the default branch created by git init

29 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras The staging area (aka the “index”)

Usual version control systems provide two spaces:

*•* the **repository**

*(the whole history of your project)*

*•* the **working tree** (or **local copy**)

*(the files you are editing and that will be in the next commit)*

Git introduces an intermediate space : the **staging area**

(also called **index**)

The index stores the files scheduled for the next commit:

*•* git add *files →* copy files into the index

*•* git commit *→* commits the content of the index

30 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras The staging area (aka the “index”)

31 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Update a file

✞ ☎ **$ echo 'blah blah blah' >> hello**

**$ git commit**

# On branch master

# Changed but not updated:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: hello

#

no changes added to commit (use "git add" and/or "git commit -a")

✝ ✆ Git complains because the index is unchanged (nothing to commit)

*→* We need to run git add to copy the file into the index ✞ ☎ **$ git add hello**

**$ git commit -m "some changes"**

[master f37f2cf] some changes

1 files changed, 1 insertions(+), 0 deletions(-)

✝ ✆ 32 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Bypassing the index7

Running git add & git commit for every iteration is tedious. GIT provides a way to bypass the index.

git commit *file1* [ *file2 . . .* ]

This command commits files (or dirs) directly from the working tree

**Note:** when bypassing the index, GIT ignores new files:

*•* “git commit .” commits only files that were present in the last commit (updated files)

*•* “git add . && git commit” commits everything in the working tree

(including new files)

7also named “partial commit”

33 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Bypassing the index

34 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Deleting files

git rm *file*

*→* remove the file from the index and from the working copy git commit

*→* commit the index

✞ ☎ **$ git rm hello**

rm 'hello'

**$ git commit -m "removed hello"**

[master 848d8be] removed hello

1 files changed, 0 insertions(+), 3 deletions(-)

delete mode 100644 hello

✝ ✆ 35 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Showing differences

git diff [ *rev a* [ *rev b* ] ] [ -- *path . . .* ]

*→* shows the differences between two revisions *rev a* and *rev b* (*in a format suitable for the* patch *utility*)

*•* by default *rev a* is the **index**

*•* by default *rev b* is the **working copy**

git diff --staged [ *rev a* ] [ -- *path . . .* ]

*→* shows the differences between *rev a* and the index

*•* by default *rev a* is HEAD *(a symbolic references pointing to the last commit)*

36 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras About git diff and the index

37 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

✞ Diff example ☎ **$ echo foo >> hello**

**$ git add hello**

**$ echo bar >> hello**

**$ git diff**

--- a/hello

+++ b/hello

@@ -1,2 +1,3 @@

Hello World!

foo

+bar

**$ git diff --staged**

--- a/hello

+++ b/hello

@@ -1 +1,2 @@

Hello World!

+foo

**$ git diff HEAD**

--- a/hello

+++ b/hello

@@ -1 +1,3 @@

Hello World!

+foo

+bar

✝ ✆38 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Resetting changes

git reset [ --hard ] [ -- *path . . .* ]

git reset cancels the changes in the index (and possibly in the working copy)

*•* git reset drops the changes staged into the index8, but the working copy is left intact

*•* git reset --hard drops all the changes in the index **and** in the working copy

8it restores the files as they were in the last commit

39 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Resetting changes in the working copy

git checkout -- *path*

This command restores a file (or directory) as it appears in the index (thus it drops all unstaged changes)

✞ ☎ **$ git diff HEAD**

--- a/hello

+++ b/hello

@@ -1 +1,3 @@

Hello World!

+foo

+bar

**$ git checkout -- .**

**$ git diff HEAD**

--- a/hello

+++ b/hello

@@ -1 +1,2 @@

Hello World!

+foo

✝ ✆ 40 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Other local commands

*•* git status *→* show the status of the index and working copy

*•* git show *→* show the details of a commit (metadata + diff) *•* git log *→* show the history

*•* git mv *→* move/rename a file9

*•* git tag *→* creating/deleting tags (to identify a particular revision)

9note that git mv is strictly equivalent to: “cp *src dst* && git rm *src* && git add *dst*” (file renaming is not handled formally, but heuristically) 41 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Exercises

1. create a new repository

2. create a new file, add it to the index and commit it

3. launch gitk to display it. Keep the window open and hit F5 after each

command (to visualise the results of your commands)

4. modify the file and make a new commit

5. rename the file (either with git mv or git add+git rm), do a git status

before committing (to ensure the renaming is correctly handled)

6. delete the file and commit it

7. create two new files and commit them. Then modify their content in the

working copy and display the changes with git diff

8. add one file into the index but keep the other one. Display the changes

between:

*•* the index and the working copy

*•* the last commit and the index

*•* the last commit and the working copy

9. run git reset to reset the index

10. run git reset --hard to reset the index and the working copy

42 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras

Part 4.

Branching & merging

*•* How GIT handles its history

*•* Creating new branches

*•* Merging & resolving conflicts

43 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras How GIT handles its history

Each **commit** object has a list of **parent**

**commits**:

*•* 0 parents *→* initial commit

*•* 1 parent *→* ordinary commit

*•* 2+ parents *→* result of a merge

*→* This is a Direct Acyclic Graph

44 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras How GIT handles its history

*•* There is no formal “branch history”

*→* a **branch** is just a pointer on the latest commit.

(*git handles branches and tags in the same way internally*)

*•* Commits are identified with **SHA-1 hash** (160 bits)

computed from:

*•* the commited files

*•* the meta data (commit message, author name, . . . )

*•* the hashes of the parent commits

*→* A commit id (hash) identifies **securely** and **reliably** its content and all the previous revisions.

45 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Creating a new branch

git checkout -b *new branch* [ *starting point* ]

*• new branch* is the name of the new branch

*• starting point* is the starting location of the branch (possibly a commit id, a tag, a branch, . . . ). If not present, git will use the current location.

✞ ☎ **$ git status**

# On branch master

nothing to commit (working directory clean)

**$ git checkout -b develop**

Switched to a new branch 'develop'

**$ git status**

# On branch develop

nothing to commit (working directory clean)

✝ ✆ 46 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Switching between branches

git checkout [-m] *branch name*

✞ ☎ **$ git status**

# On branch develop

nothing to commit (working directory clean)

**$ git checkout master**

Switched to branch 'master'

✝ ✆

**Note:** it may fail when the working copy is not clean. Add -m to request merging your local changes into the destination branch. ✞ ☎ **$ git checkout master**

error: Your local changes to the following files would be overwritten by checkout: hello Please, commit your changes or stash them before you can switch branches.

Aborting

**$ git checkout -m master**

M hello

Switched to branch 'master'

✝ ✆ 47 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Merging a branch

git merge *other branch*

This will merge the changes in *other branch* into the current branch.

✞ ☎ **$ git status**

# On branch master

nothing to commit (working directory clean)

**$ git merge develop**

Merge made by recursive.

dev | 1 +

hello | 4 +++-

2 files changed, 4 insertions(+), 1 deletions(-)

create mode 100644 dev

✝ ✆ 48 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Notes about merging

*•* The result of git merge is immediately committed

(unless there is a conflict)

*•* The new commit object has **two parents**.

*→* the merge history is recorded

*•* git merge applies only the changes since the last common ancestor in the other branch.

*→* if the branch was already merged previously, then only the changes since the last merge will be merged.

49 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96

Version Control GIT Intro Local GIT Branches Remote GIT Server Bazar Extras Branching example

50 / 96