

# Introduction to Shell-based Data Processing

**Lecture 13: GIT (versioning control)**  
**Practice 2: GIT, python plotting, (R)**

**Richard Veale**  
**Kyoto University, Fall 2024**


# Video link

- Lecture 2025/01/10: <https://youtu.be/Klcf9-HeGXk>

# Today: git

- GIT

- Versioning control system initially created for linux kernel.
- Now widely used, e.g. <https://github.com/>

 Git

Article

Talk

Read

Edit

View history

Tools

From Wikipedia, the free encyclopedia

*For other uses, see [Git \(disambiguation\)](#).*


*Not to be confused with [GitHub](#), [GitLab](#), or [Gitea](#).*

**Git** (/git/<sup>[8]</sup> is a [distributed version control system](#)<sup>[9]</sup> that tracks versions of [files](#). It is often used to control [source code](#) by [programmers](#) who are [developing](#) software collaboratively.

Design goals of Git include speed, [data integrity](#), and support for [distributed](#), non-linear workflows — thousands of parallel [branches](#) running on different computers.<sup>[10][11][12]</sup>

As with most other distributed version control systems, and unlike most [client–server](#) systems, Git maintains a local copy of the entire [repository](#), also known as

Git

 **git**

```
$ git init
Initialized empty Git repository in /tmp/tmp.IMBYSY7RBY/.git/
$ cat > README << 'EOF'
> Git is a distributed revision control system.
> EOF
$ git add README
$ git commit
[master (root-commit) e4dcc69] You can edit locally and push
to any remote.
1 file changed, 1 insertion(+)
 create mode 100644 README
$ git remote add origin git@github.com:cdown/thats.git
$ git push -u origin master
```

A command-line session showing repository

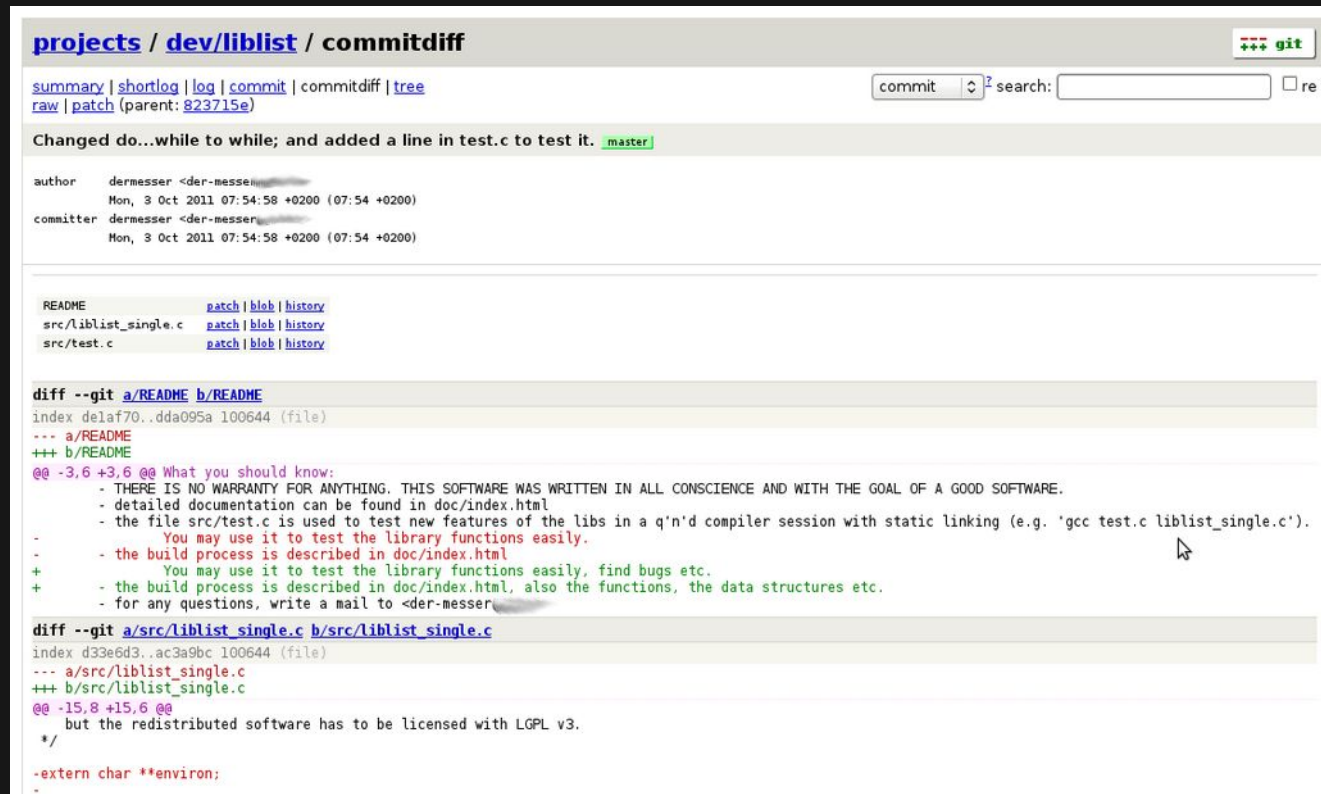
# Other versioning systems

- CVS (old!)
- SVN (subversion) – commonly used
- mercurial (newer)

# Git tracks directories/files

- When you make changes and *commit* them, they will be saved as a **version**.
- You can go back to old versions.
- You can merge versions (by same or different people)

This shows the difference between two versions of the files, at time of commit.



The screenshot displays a web interface for a Git commit diff. The browser address bar shows the URL: `projects / dev/liblist / commitdiff`. The page title is "commitdiff". Below the title, there are links for "summary", "shortlog", "log", "commit", "commitdiff", and "tree". A search bar is present with the text "commit" and a search icon. The commit message is "Changed do...while to while; and added a line in test.c to test it. master". The commit details show the author as "dermesser <der-messer@...>" and the committer as "dermesser <der-messer@...>", both dated "Mon, 3 Oct 2011 07:54:58 +0200 (07:54 +0200)". The files changed are listed as "README", "src/liblist\_single.c", and "src/test.c". The diff output shows changes between "a/README" and "b/README", and between "a/src/liblist\_single.c" and "b/src/liblist\_single.c". The diff for README shows a change in the "What you should know:" section, adding a line about the build process. The diff for src/liblist\_single.c shows a change in the license text, adding a line about the LGPL v3 license.

```
projects / dev/liblist / commitdiff

summary | shortlog | log | commit | commitdiff | tree
raw | patch (parent: 823715e)

commit 823715e
Changed do...while to while; and added a line in test.c to test it. master

author    dermesser <der-messer@...>
date      Mon, 3 Oct 2011 07:54:58 +0200 (07:54 +0200)
committer dermesser <der-messer@...>
date      Mon, 3 Oct 2011 07:54:58 +0200 (07:54 +0200)

diff --git a/README b/README
index d41af70..dda095a 100644 (file)
--- a/README
+++ b/README
@@ -3,6 +3,6 @@ What you should know:
- THERE IS NO WARRANTY FOR ANYTHING. THIS SOFTWARE WAS WRITTEN IN ALL CONSCIENCE AND WITH THE GOAL OF A GOOD SOFTWARE.
- detailed documentation can be found in doc/index.html
- the file src/test.c is used to test new features of the libs in a q'n'd compiler session with static linking (e.g. 'gcc test.c liblist_single.c').
+ You may use it to test the library functions easily, find bugs etc.
- the build process is described in doc/index.html
+ You may use it to test the library functions easily, find bugs etc.
+ the build process is described in doc/index.html, also the functions, the data structures etc.
- for any questions, write a mail to <der-messer@...>

diff --git a/src/liblist_single.c b/src/liblist_single.c
index d33e6d3..ac3a9bc 100644 (file)
--- a/src/liblist_single.c
+++ b/src/liblist_single.c
@@ -15,8 +15,6 @@
but the redistributed software has to be licensed with LGPL v3.
*/

extern char **environ;
```

# Simple git

- 1) Create a new repository and add your files
  - `git init .`
  - `git add FILE1 FILE2 DIR1` or `git commit *`
  - `git commit FILE1 FILE2 DIR1` or `git commit -a`
- 2) `.gitignore` tells which files to ignore (not track)
- 3) Git "information" is stored in hidden `".git"` directory at root of repository.
- 4) "Check out" (clone) from *remote* URLs
  - `git clone URL`
- 5) "pull" to update (`git pull`) from remote

# Create a new git repo 1

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ tree -a
```

```
.
```

```
0 directories, 0 files
```

# Create a new git repo 2

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git init .
hint: Using 'master' as the name for the initial branch. This default br
nch name
hint: is subject to change. To configure the initial branch name to use
n all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:     git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this com
and:
hint:
hint:     git branch -m <name>
```

Some warnings about naming the branches...



# Create a new git repo 3

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ tree -a
.
├── .git
│   ├── branches
│   ├── config
│   ├── description
│   ├── HEAD
│   ├── hooks
│   │   ├── applypatch-msg.sample
│   │   ├── commit-msg.sample
│   │   └── fsmonitor-watchman.sample
└──
```

It created hidden .git folder with lots of stuff in it.

# Create a new git repo 4

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

We're not tracking anything yet though!

m:~/dataproc2024/labs/13git\$ emacs myfile.py

```
import sys
```

```
for a in range(10):  
    print(a);
```

```
    pass;
```

```
exit(0)
```

# Create a new git repo 5

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git add myfile.py
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git status
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: myfile.py

We "added" the file, but nothing  
committed yet...

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git log
fatal: your current branch 'master' does not have any ucommits yet
```

# Create new git repo 6

- Try to commit (oh, we have not configured our "git identity" yet)

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git commit myfile.py
Author identity unknown
```

```
*** Please tell me who you are.
```

Run

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

# Configure git identity...

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git config --global user.email "richard.e.veale@gmail.com"
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git config --global user.name "Richard Veale"
```

# git commit (type in "commit message")

```
/home/riveale99/dataproc2024/labs/13git/.git/COMMIT_EDITMSG *
```

```
Added new python file to print numbers 1-10
```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   myfile.py
#
```

```
Save modified buffer?
```

```
Y Yes
```

```
N No      ^C Cancel
```



# Commit the file

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git commit myfile.py
[master (root-commit) 5ee900a] Added new file to print numbers 1-10
1 file changed, 8 insertions(+)
create mode 100644 myfile.py
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$
```



# See the log

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git commit myfile.py
[master (root-commit) 5ee900a] Added new file to print numbers 1-10
1 file changed, 8 insertions(+)
create mode 100644 myfile.py
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git status
On branch master
nothing to commit, working tree clean
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git log
commit 5ee900aa4c1c9ae4a48b0432e9ebee0478ae9ab5 (HEAD -> master)
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:13:41 2025 +0000

    Added new file to print numbers 1-10
```

---

# Change something...

- Print 0-19 instead...

File Edit Options Buffers Tools Pyth

```
import sys
```

```
for a in range(20):  
    print(a);
```

```
    pass;
```

```
exit(0)
```

-UU-:---

F1

myfile.py

All

L

# Commit changes

- `git commit -a` (or `git commit FILENAME`)

```
/home/riveale99/dataproc2024/labs/13git/.git/COMMIT_EDITMSG *
REV: changed to print 0-19
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   myfile.py
#
```

```
^G Help
^X Exit
```

```
^O Write Out
^R Read File
```

```
^W Where Is
^_ Replace
```

```
^K Cut
^U Paste
```

```
^T Execute
^J Justify
```

# See history of revisions...

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git log
commit 487b225d5c4064e777b0d1237d098bbd8bf372d8 (HEAD -> master)
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:15:47 2025 +0000
```

REV: changed to print 0-19

```
commit 5ee900aa4c1c9ae4a48b0432e9ebee0478ae9ab5
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:13:41 2025 +0000
```

Added new file to print numbers 1-10

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$
```

# Oops, we want to go back!

- git checkout <ID>
  - IDs are the commit hashes

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git log
commit 487b225d5c4064e777b0d1237d098bbd8bf372d8 (HEAD -> master)
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:15:47 2025 +0000

    REV: changed to print 0-19

commit 5ee900aa4c1c9ae4a48b0432e9ebee0478ae9ab5
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:13:41 2025 +0000

    Added new file to print numbers 1-10
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$
```



# Git reverting

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git checkout 5ee900aa4c1c9ae4a48b0432e9eb0478ae9ab5 .  
Updated 1 path from 3c95963
```

Note the DOT!

- Our file is back to how it was before!

```
File Edit Options Buffers Tools Pyt
import sys

for a in range(10):
    print(a);

    pass;

exit(0)
```

-UU-:--- F1 myfile.py All

# Check in with the old code...

- The -m replaces opening an editor (faster sometimes)

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git commit -m "Restoring old source code"
[master e6200a9] Restoring old source code
1 file changed, 1 insertion(+), 1 deletion(-)
```



# Now we're back.

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git log
commit e6200a97003ee89f6dbb4e92043bdb9b185b9985 (HEAD -> master)
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:20:36 2025 +0000
```

Restoring old source code

```
commit 487b225d5c4064e777b0d1237d098bbd8bf372d8
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:15:47 2025 +0000
```

REV: changed to print 0-19

```
commit 5ee900aa4c1c9ae4a48b0432e9eb0478ae9ab5
Author: Richard Veale <richard.e.veale@gmail.com>
Date:   Fri Jan 10 04:13:41 2025 +0000
```

Added new file to print numbers 1-10

# Cloning remote repositories

- Repositories can be available via SSH or HTTPS

The screenshot shows the GitHub interface for the repository `dbarnett / python-helloworld`. The repository is public and has 411 forks and 22 stars. The main branch is `main`, and there are 2 branches and 0 tags. The repository description is "No description, website, or topics provided." The file list includes `helloworld`, `.gitignore`, `LICENSE`, `MANIFEST.in`, `README.md`, `VERSION.txt`, `helloworld.py`, `pyproject.toml`, and `setup.py`. The commit history shows a recent commit by `dbarnett` titled "Move package metadata from setup.py into pyproject.toml" 5 months ago, with 15 commits in total.

Product ▾ Solutions ▾ Resources ▾ Open Source ▾ Enterprise ▾ Pricing

Search or jump to... / Sign in Sign up

dbarnett / python-helloworld Public

Notifications Fork 411 Star 22

<> Code Issues Pull requests 2 Actions Projects Wiki Security Insights

main 2 Branches 0 Tags Go to file <> Code ▾

**About**  
No description, website, or topics provided.  
Readme  
Apache-2.0 license  
Activity  
22 stars  
6 watching  
411 forks  
Report repository

**Releases**  
No releases published

**Packages**

File	Description	Time
helloworld	Add --version and --help arguments using argparse (#11)	5 years ago
.gitignore	.gitignore: Add a few missing ignore patterns for build o...	6 years ago
LICENSE	License under Apache 2.0	5 years ago
MANIFEST.in	Move package metadata from setup.py into pyproject.to...	5 months ago
README.md	Expand README with more basic info and examples	5 months ago
VERSION.txt	Add --version and --help arguments using argparse (#11)	5 years ago
helloworld.py	Add --version and --help arguments using argparse (#11)	5 years ago
pyproject.toml	Move package metadata from setup.py into pyproject.to...	5 months ago
setup.py	Move package metadata from setup.py into pyproject.to...	5 months ago

Q Go to file

<> Code ▾

About

Clone



HTTPS GitHub CLI

<https://github.com/dbarnett/python-helloworld>



Clone using the web URL.

Download ZIP

etup.py into pyproject.toml

Add --version and

.gitignore: Add a fe

License under Apa

Move package met

Expand README with more basic info and examples

5 months ago

No descrip  
provided.

Readm

Apache

Activity

☆ 22 star

👁 6 watc

🔗 411 fo

Report rep

# Remote repo

- `git clone [URL]`

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git$ git clone https://github.com/dbarnett/python-helloworld.git
Cloning into 'python-helloworld'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 59 (delta 14), reused 8 (delta 8), pack-reused 37 (from 1)
Receiving objects: 100% (59/59), 13.96 KiB | 6.98 MiB/s, done.
Resolving deltas: 100% (20/20), done.
```



# Then you can modify it...

```
riveale99@dataproc2023vm:~/dataproc2024/labs/13git/python-helloworld$ ls
helloworld      LICENSE        pyproject.toml  setup.py
helloworld.py   MANIFEST.in    README.md       VERSION.txt
riveale99@dataproc2023vm:~/dataproc2024/labs/13git/python-helloworld$ git
log
```

```
commit 288d7ced1b971fd1b3b0c36002b96e1c3f91542e (HEAD -> main, origin/main, origin/HEAD)
```

```
Author: David Barnett <david@mumind.me>
```

```
Date:   Fri Aug 16 17:45:24 2024 -0600
```

Move package metadata from setup.py into pyproject.toml

Details on pyproject.toml: PEP 621.

```
commit aed348ececb1860aa32e12ca9767b02e6dce8d21
```

```
Author: David Barnett <david@mumind.me>
```

```
Date:   Fri Aug 16 17:57:57 2024 -0600
```

setup.py: genericize author name/email (to avoid copy/paste of actual

# Git LAB (13)

- Create a new folder  
/home/USER/dataproc2024/labs/13git
- Make it into a git repository.
- Write a python script to print numbers 0-9.
- Check it in.
- Change the script to print numbers 0-19.
- Check it in.
- Revert to the first version.
- Check in the (reverted) first version.
- Paste your git LOG into PANDAS as submission.