MANRS

Git for network engineers

Philip Paeps

philip@trouble.is

PacNOG 32 — Nuku‘alofa, Tonga

27 November 2023

Agenda

1. **Revision control essentials**
2. **Git survival kit for network engineers**
3. **Using GitHub or GitLab to collaborate**

Revision control essentials

Computers are better at remembering things than you are.

Revision control for network engineers

Revision control systems remember changes you make to your network.

With good revision control hygiene, you can easily:

- **Revert configurations to a known working state**
- **Review changes before deploying them to production**
- **Recover configurations when network equipment breaks**
- **Collaborate on projects with others without conflicts**

Not only for source code and configuration

Revision control systems don’t care about the data they control.

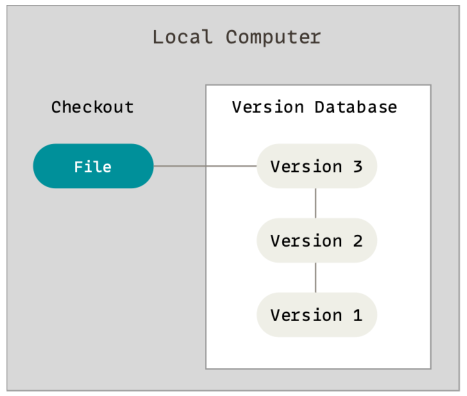
Use them to track changes and collaborate on all sorts of things:

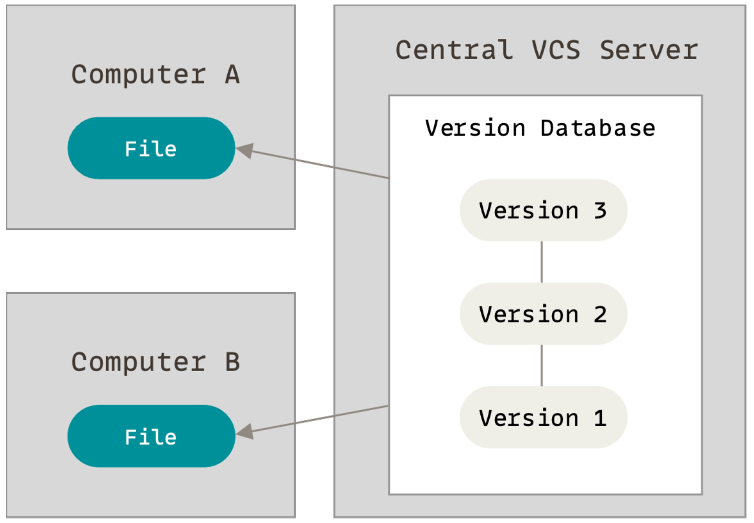
- **Internet drafts**
- **Network policy documents**
- **Training materials**
- **Presentations**

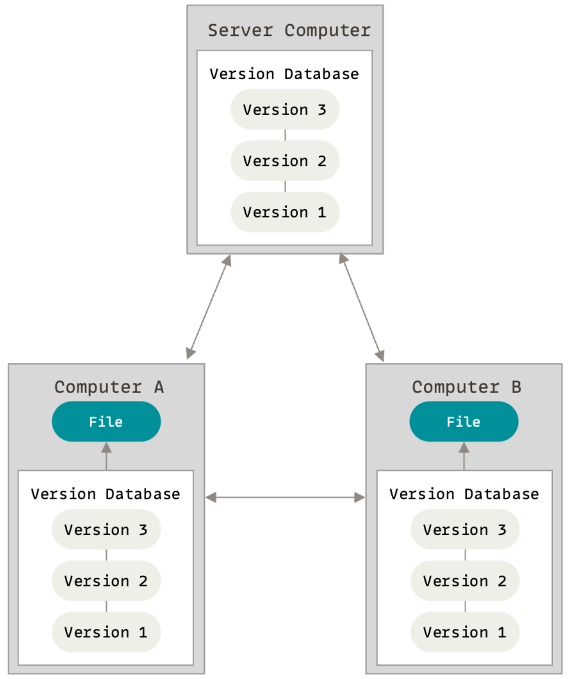
Revision control options

Amount of control	Pros and cons
Chaos reigns **Loose files all over the place**	✓ **Easy to learn** ✗ **Impossible to undo changes**
Archiving for posterity **NFS, SMB, OneDrive, Dropbox,…**	✓ **Audit and roll back previous versions** ✗ **Concurrent access nightmares**
Revision control **CVS, Subversion, Git, etc.**	✓ **Full control and low-friction collaboration** ✗ **Learning curve**

Basics of revision control







Git survival kit for network engineers

Revision control system? Content addressable filesystem?
Something software people use? A synonym for software people?
Why should network engineers care?

What is Git anyway?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

From git-scm.com

GitHub is a company providing a cloud service built around Git.

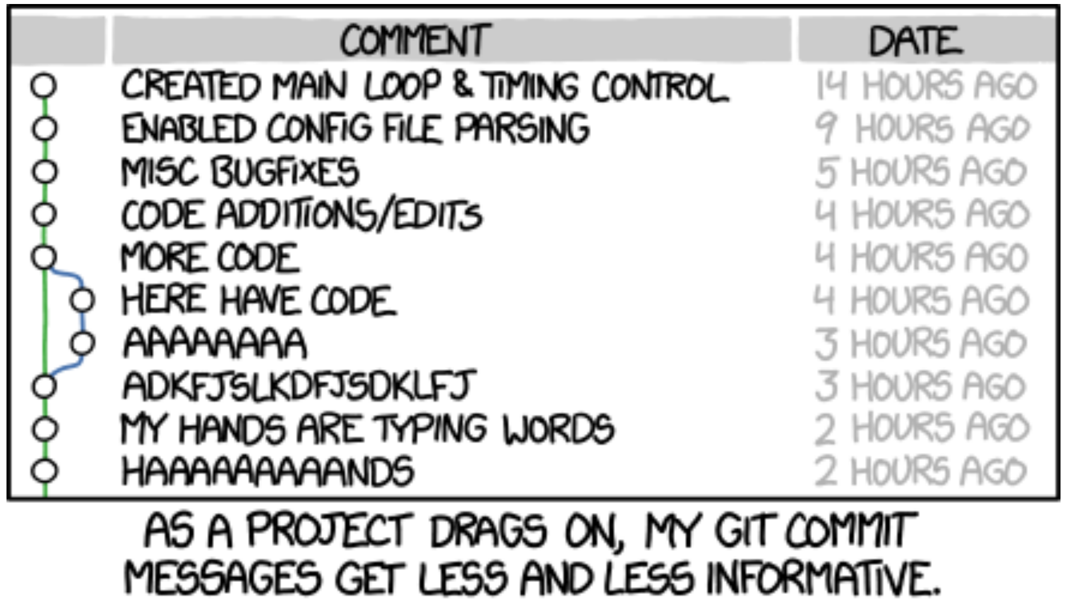
THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

https://xkcd.com/1597/

Be nice to your future self

The Git commit **command writes staged changes to the repository. The** *commit message* **should explain what the changes are intended to do.**

The log of a repository are notes to your future self. When things break, you will want to read them.



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

https://xkcd.com/1296/

Git commands for everyday use

Get a repository
```
git init
git clone
```

Manipulate the index
```
git add
git rm
```

Commit changes
```
git commit
```

Review logs
```
git log
git show
```

Figure out what's happening
```
git status
git diff
```

Undo changes
```
git reset
git checkout
```

Work with others
```
git fetch
git rebase
```

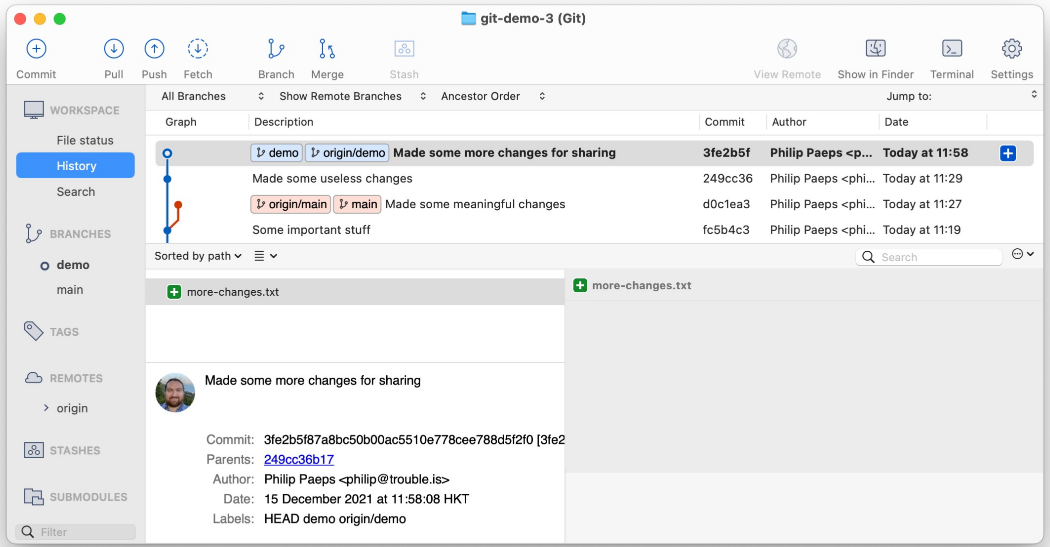
GUI Git tools

Git comes with two GUIs: gitk for browsing branches and git-gui for preparing/staging commits. Neither of them is particularly useful.

Atlassian Sourcetree (free) is pretty and works well.

GitHub has desktop clients (also free).

Sublime Merge (US$99) is also very pretty, and also works well.



Graph	Description	Commit	Author	Date
	demo origin/demo Made some more changes for sharing	3fe2b5f	Philip Paeps <p...	Today at 11:58
	Made some useless changes	249cc36	Philip Paeps <phi...	Today at 11:29
	origin/main main Made some meaningful changes	d0c1ea3	Philip Paeps <phi...	Today at 11:27
	Some important stuff	fc5b4c3	Philip Paeps <phi...	Today at 11:19

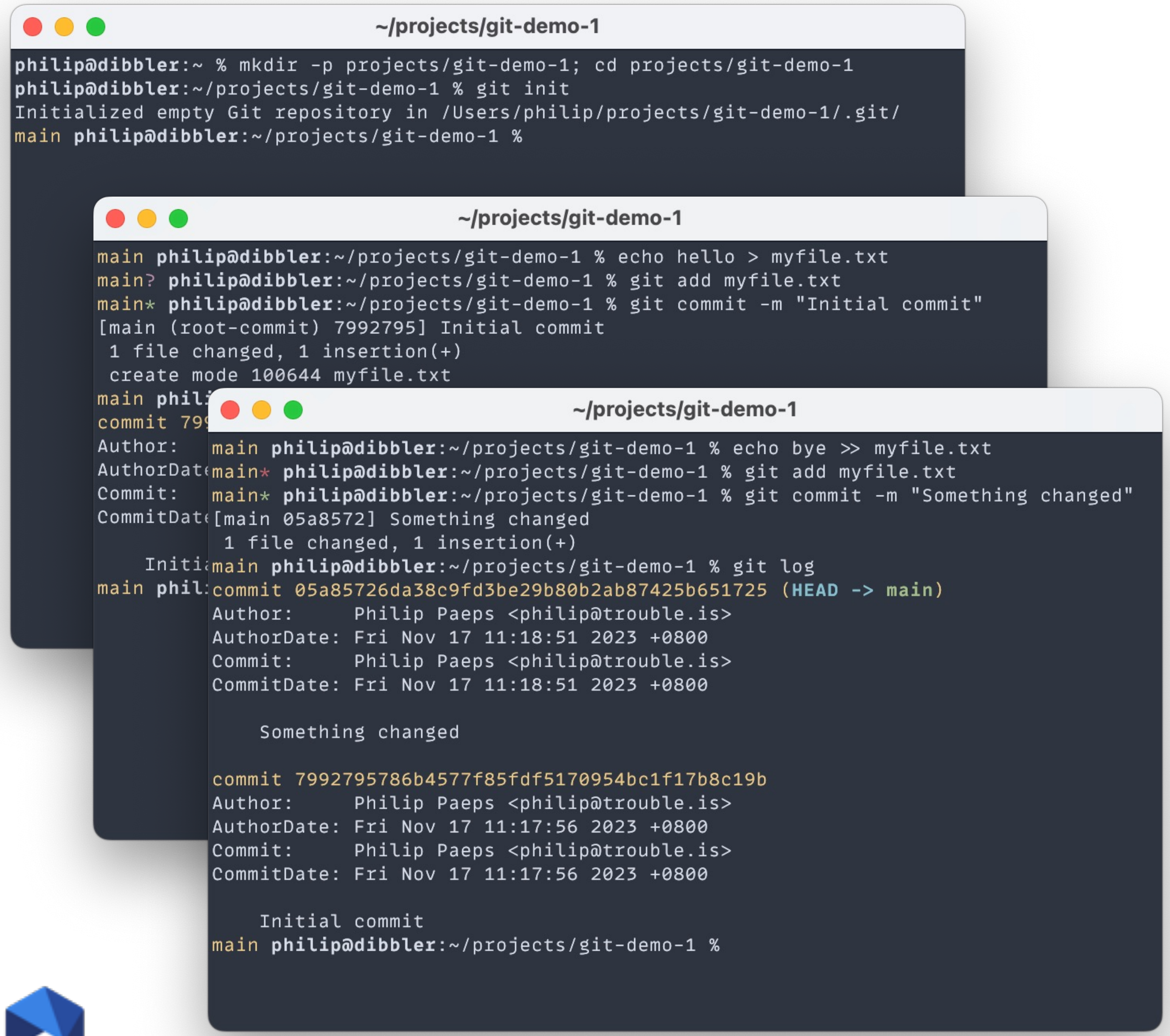
more-changes.txt

more-changes.txt

Made some more changes for sharing

Commit: 3fe2b5f87a8bc50b00ac5510e778cee788d5f2f0 [3fe2
Parents: 249cc36b17
Author: Philip Paeps <philip@trouble.is>
Date: 15 December 2021 at 11:58:08 HKT
Labels: HEAD demo origin/demo

Five-minute intro to Git (demo)



~/projects/git-demo-1

```
philip@dibbler:~ % mkdir -p projects/git-demo-1; cd projects/git-demo-1
philip@dibbler:~/projects/git-demo-1 % git init
Initialized empty Git repository in /Users/philip/projects/git-demo-1/.git/
main philip@dibbler:~/projects/git-demo-1 %
```

~/projects/git-demo-1

```
main philip@dibbler:~/projects/git-demo-1 % echo hello > myfile.txt
main? philip@dibbler:~/projects/git-demo-1 % git add myfile.txt
main* philip@dibbler:~/projects/git-demo-1 % git commit -m "Initial commit"
[main (root-commit) 7992795] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 myfile.txt
main phil
commit 79
Author:
AuthorDat
Commit:
CommitDat

    Initi
main phil
```

~/projects/git-demo-1

```
main philip@dibbler:~/projects/git-demo-1 % echo bye >> myfile.txt
main* philip@dibbler:~/projects/git-demo-1 % git add myfile.txt
main* philip@dibbler:~/projects/git-demo-1 % git commit -m "Something changed"
[main 05a8572] Something changed
 1 file changed, 1 insertion(+)
main philip@dibbler:~/projects/git-demo-1 % git log
commit 05a85726da38c9fd3be29b80b2ab87425b651725 (HEAD -> main)
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Fri Nov 17 11:18:51 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Fri Nov 17 11:18:51 2023 +0800

    Something changed

commit 7992795786b4577f85fdf5170954bc1f17b8c19b
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Fri Nov 17 11:17:56 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Fri Nov 17 11:17:56 2023 +0800

    Initial commit
main philip@dibbler:~/projects/git-demo-1 %
```

Create a new repository
`git init`

Add a file to the staging area
`git add`

Commit changes to the repository
`git commit`

Show history
`git log`

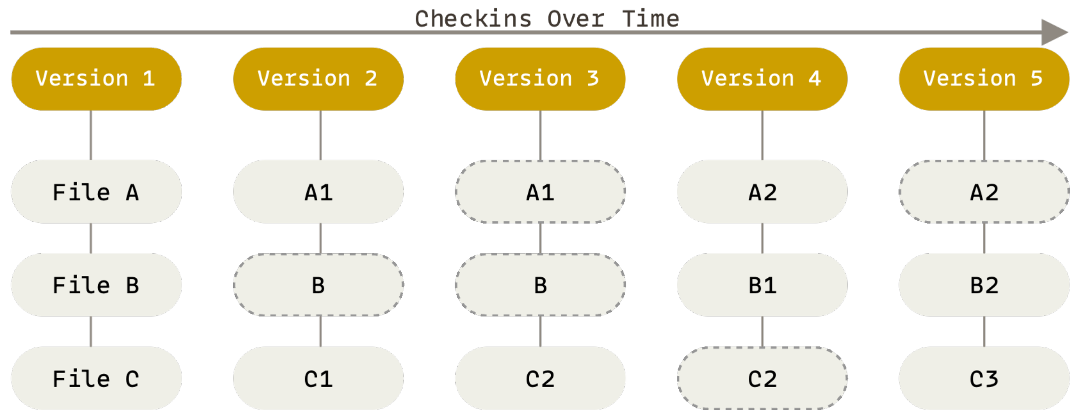
A series of snapshots

Each commit **is a snapshot of the repository at that point in time.**

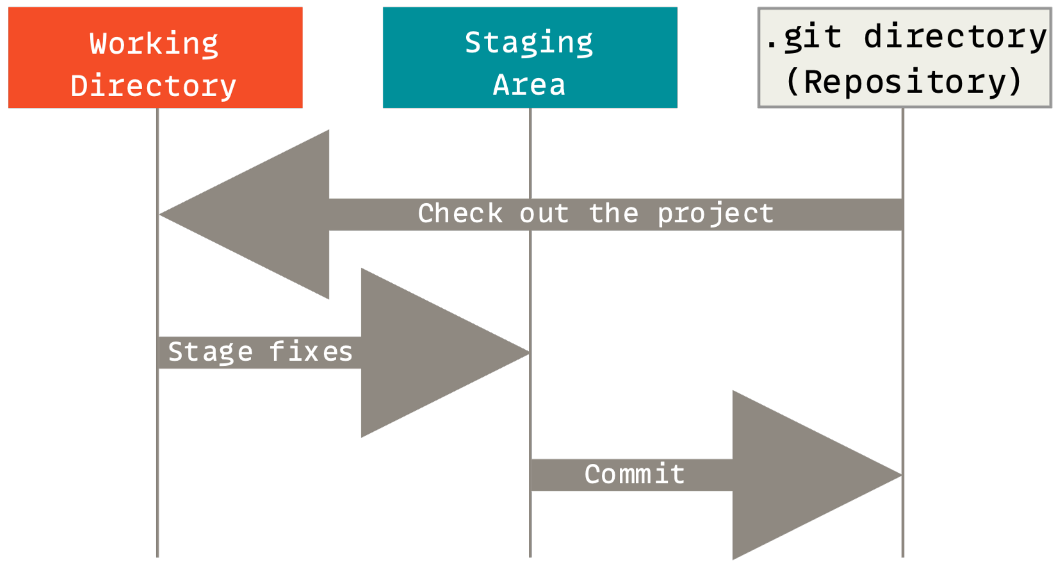
Git references snapshots by the SHA-1 hash **of their contents.**

Most Git operations are local**.**

Git generally only adds **data. It is difficult to** *lose data* **once committed.**



Git terminology: states and the index



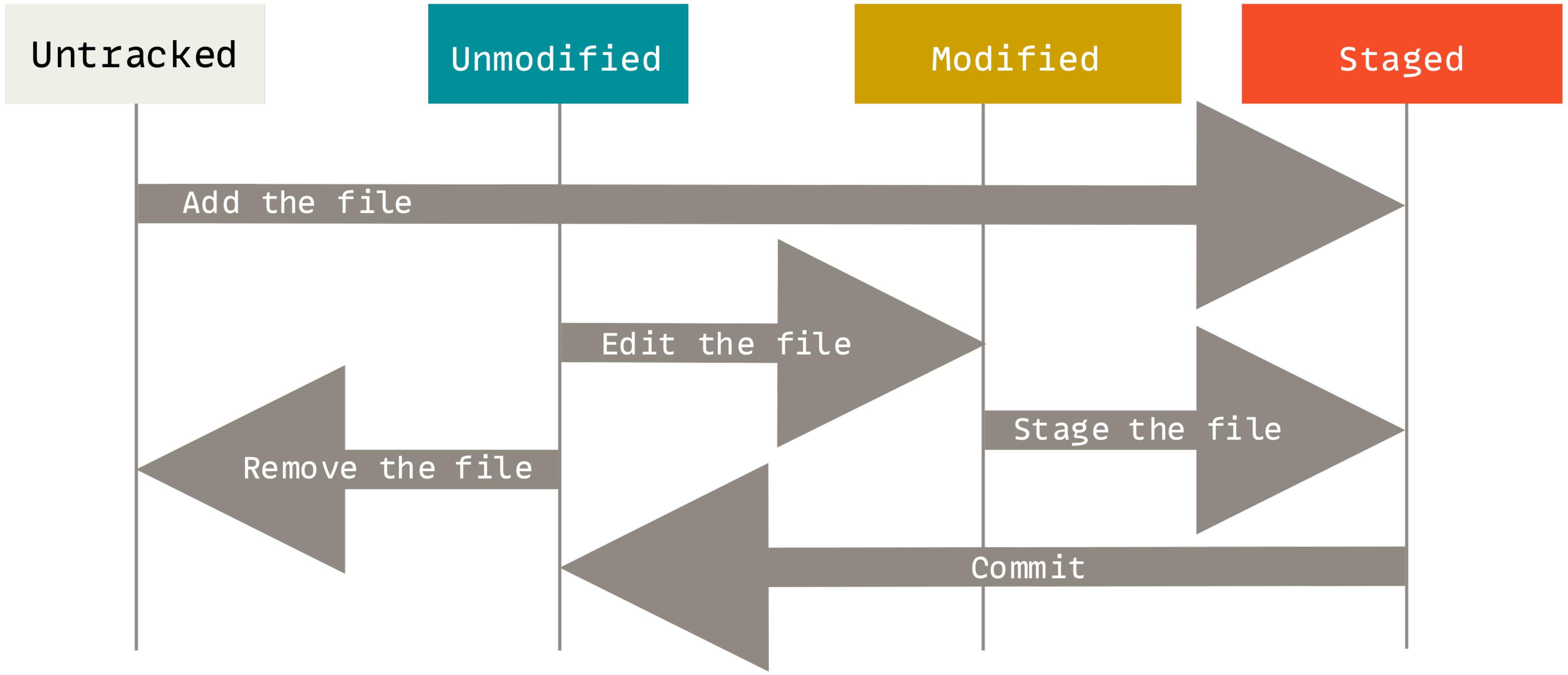
Three main states of Git:

- **Modified files have uncommitted changes**
- **Staged changes will be written to the repository in the next commit ("index")**
- **Committed changes are safely stored**

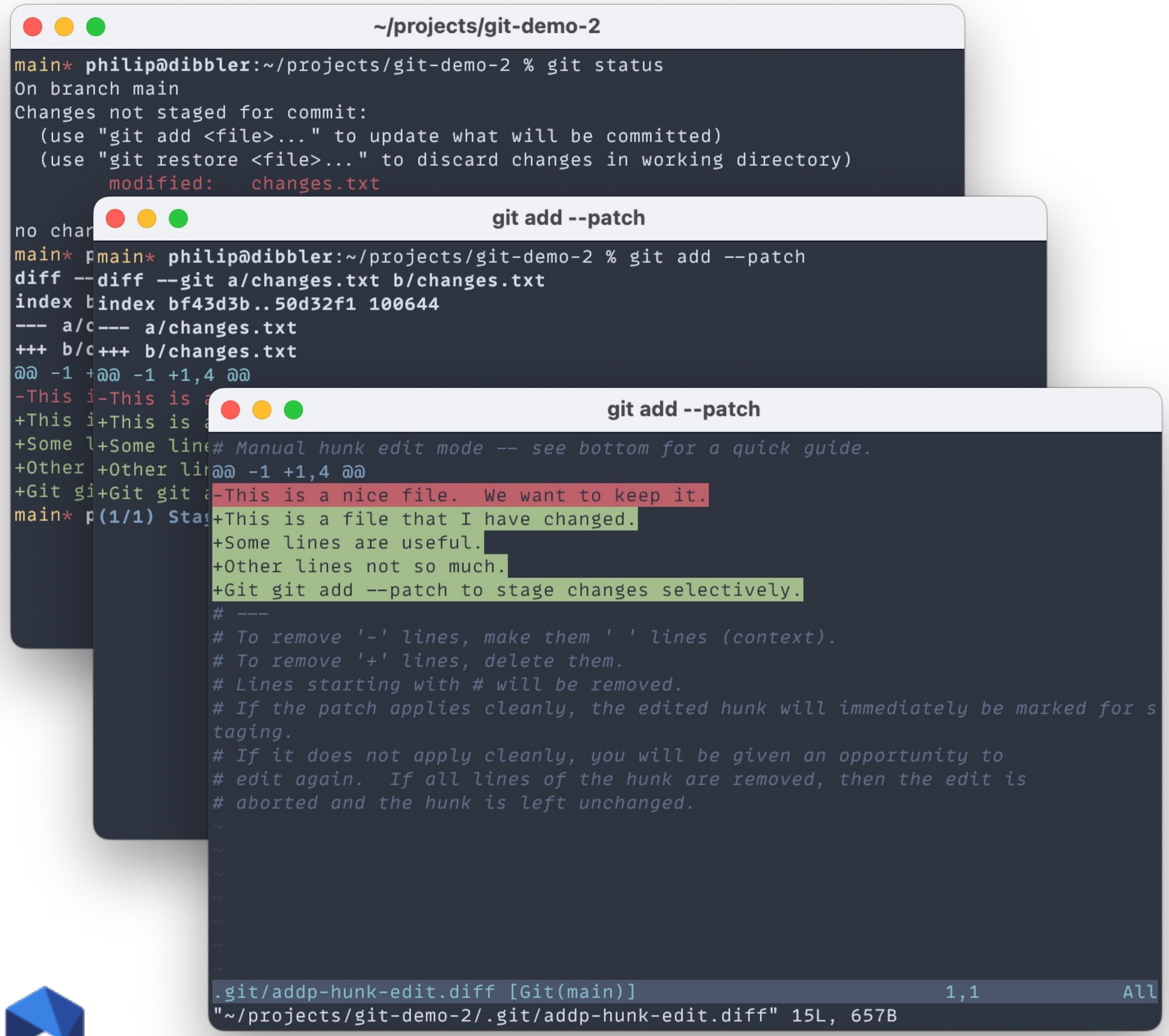
Not really a state:

- **Untracked files are unknown to Git**

Git workflow: recording changes



Using the index effectively (demo)



~/projects/git-demo-2

```
main* philip@dibbler:~/projects/git-demo-2 % git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   changes.txt

no char
main* p
diff --
index b
--- a/c
+++ b/c
@@ -1 +
-This i
+This i
+Some l
+Other
+Git gi
main* p
```

git add --patch

```
main* philip@dibbler:~/projects/git-demo-2 % git add --patch
diff --git a/changes.txt b/changes.txt
index bf43d3b..50d32f1 100644
--- a/changes.txt
+++ b/changes.txt
@@ -1 +1,4 @@
-This is a
+This is a
+Some line
+Other lin
+Git git a
(1/1) Stag
```

git add --patch

```
# Manual hunk edit mode -- see bottom for a quick guide.
@@ -1 +1,4 @@
-This is a nice file.  We want to keep it.
+This is a file that I have changed.
+Some lines are useful.
+Other lines not so much.
+Git git add --patch to stage changes selectively.
# ---
# To remove '-' lines, make them ' ' lines (context).
# To remove '+' lines, delete them.
# Lines starting with # will be removed.
# If the patch applies cleanly, the edited hunk will immediately be marked for s
taging.
# If it does not apply cleanly, you will be given an opportunity to
# edit again.  If all lines of the hunk are removed, then the edit is
# aborted and the hunk is left unchanged.
~
~
~
~
~
~
~
.git/addp-hunk-edit.diff [Git(main)]                    1,1            All
"~/projects/git-demo-2/.git/addp-hunk-edit.diff" 15L, 657B
```

Stage changes before committing

`git add --patch`

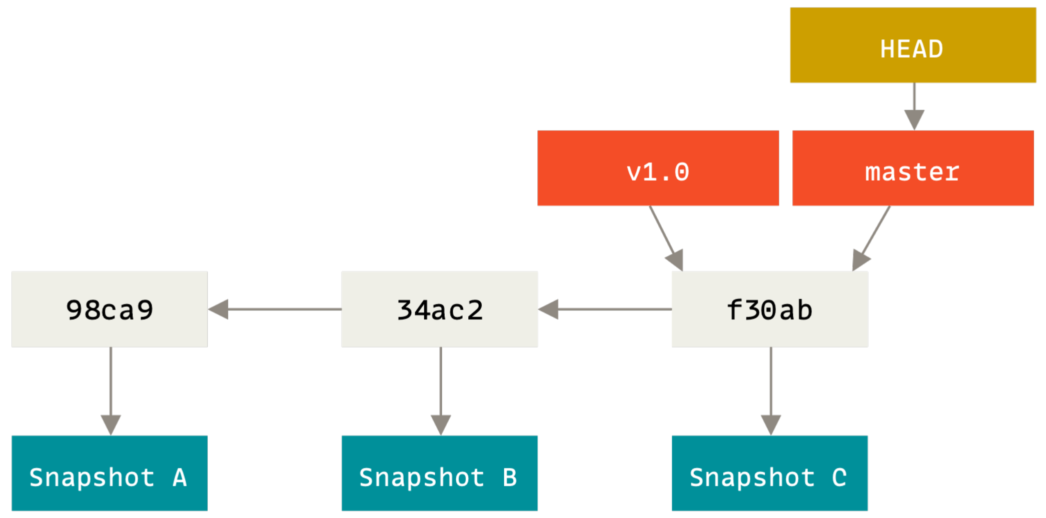
Undo local changes

`git restore`

Keeping track of local changes

`git status`
`git diff`

Basics of Git branches

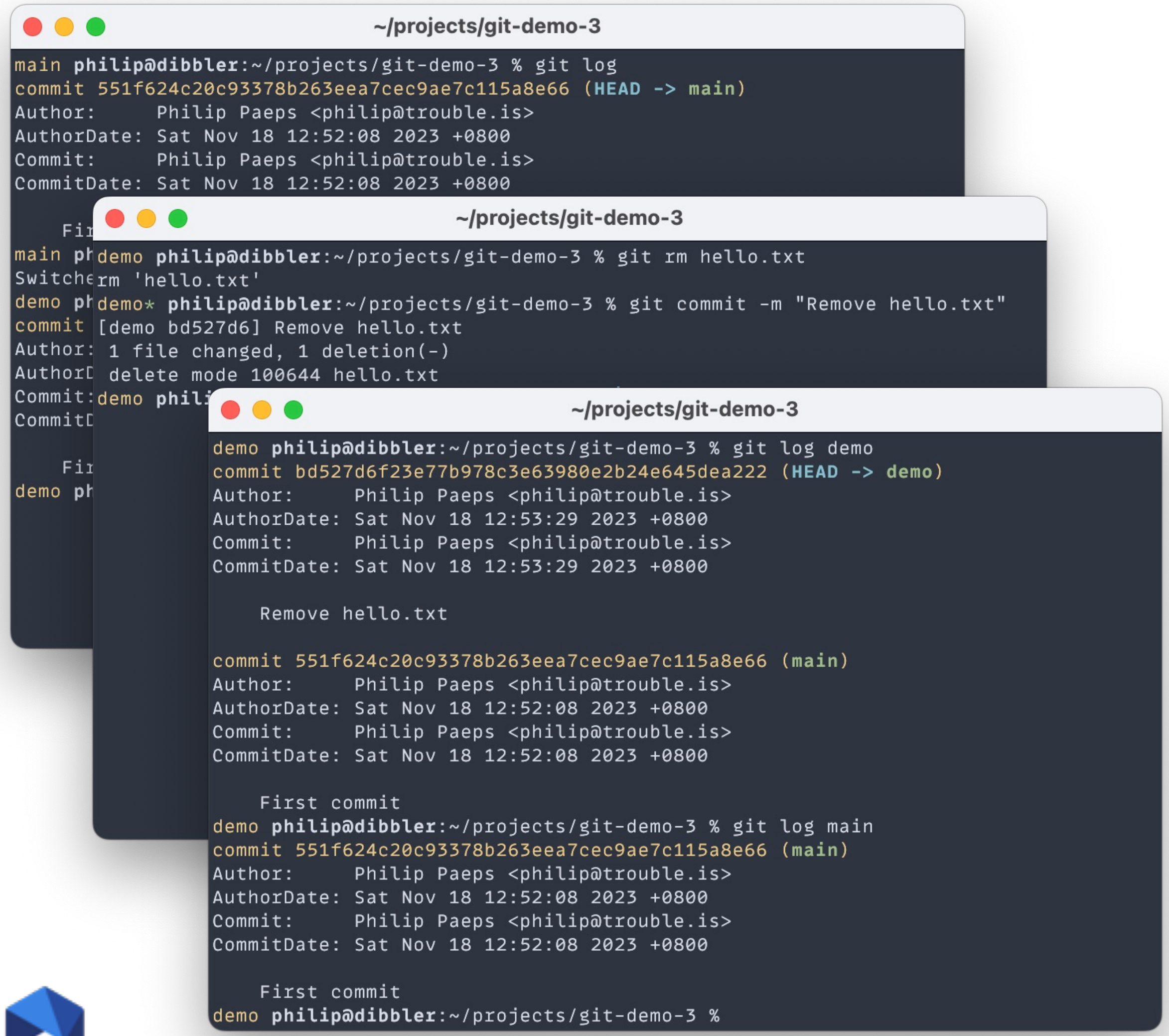


A branch is a named pointer to a snapshot (commit) known to Git.

Git makes it easy to switch between branches and record distinct histories.

The HEAD points to the currently checked out branch (commit).

Branching essentials (demo)



```
~/projects/git-demo-3
main philip@dibbler:~/projects/git-demo-3 % git log
commit 551f624c20c93378b263eea7cec9ae7c115a8e66 (HEAD -> main)
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Sat Nov 18 12:52:08 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Sat Nov 18 12:52:08 2023 +0800

    Fi
main ph
Switche
demo ph
commit
Author:
AuthorD
Commit:
CommitD

    Fi
demo ph
```

```
~/projects/git-demo-3
demo philip@dibbler:~/projects/git-demo-3 % git rm hello.txt
rm 'hello.txt'
demo* philip@dibbler:~/projects/git-demo-3 % git commit -m "Remove hello.txt"
[demo bd527d6] Remove hello.txt
 1 file changed, 1 deletion(-)
 delete mode 100644 hello.txt
demo phili
```

```
~/projects/git-demo-3
demo philip@dibbler:~/projects/git-demo-3 % git log demo
commit bd527d6f23e77b978c3e63980e2b24e645dea222 (HEAD -> demo)
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Sat Nov 18 12:53:29 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Sat Nov 18 12:53:29 2023 +0800

    Remove hello.txt

commit 551f624c20c93378b263eea7cec9ae7c115a8e66 (main)
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Sat Nov 18 12:52:08 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Sat Nov 18 12:52:08 2023 +0800

    First commit
demo philip@dibbler:~/projects/git-demo-3 % git log main
commit 551f624c20c93378b263eea7cec9ae7c115a8e66 (main)
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Sat Nov 18 12:52:08 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Sat Nov 18 12:52:08 2023 +0800

    First commit
demo philip@dibbler:~/projects/git-demo-3 %
```

Create a new branch

```
git branch <branch>
git checkout –b <branch>
```

Switching between branches

```
git checkout <branch>
```

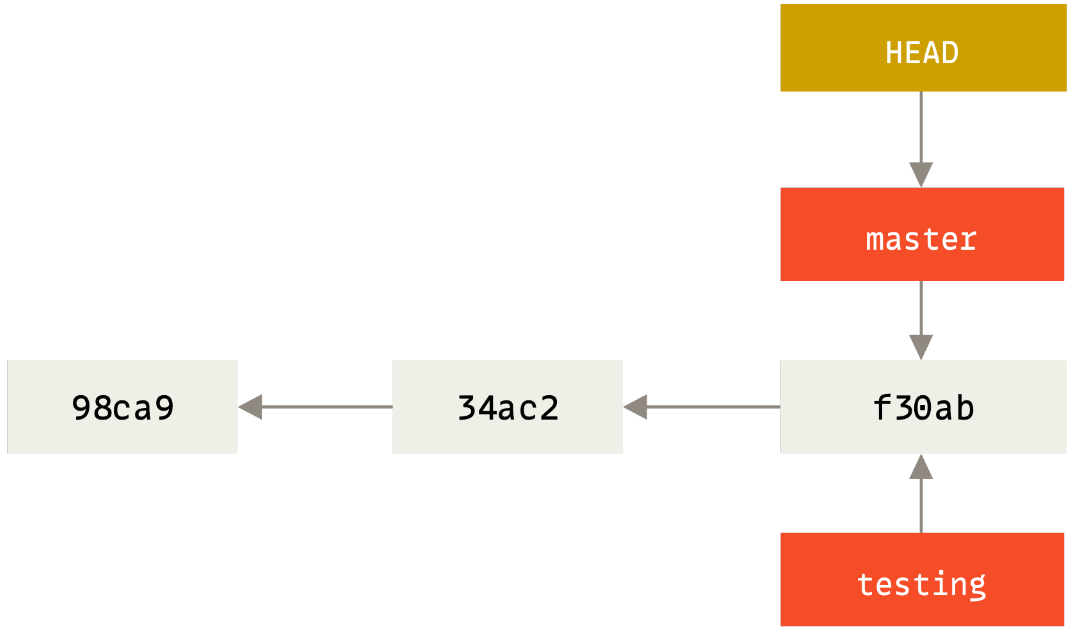
Keeping track of changes on branches

```
git log <branch>
git diff <branch>
```

Branches: creating a branch

Creating a branch adds a new pointer.
The HEAD does not move.

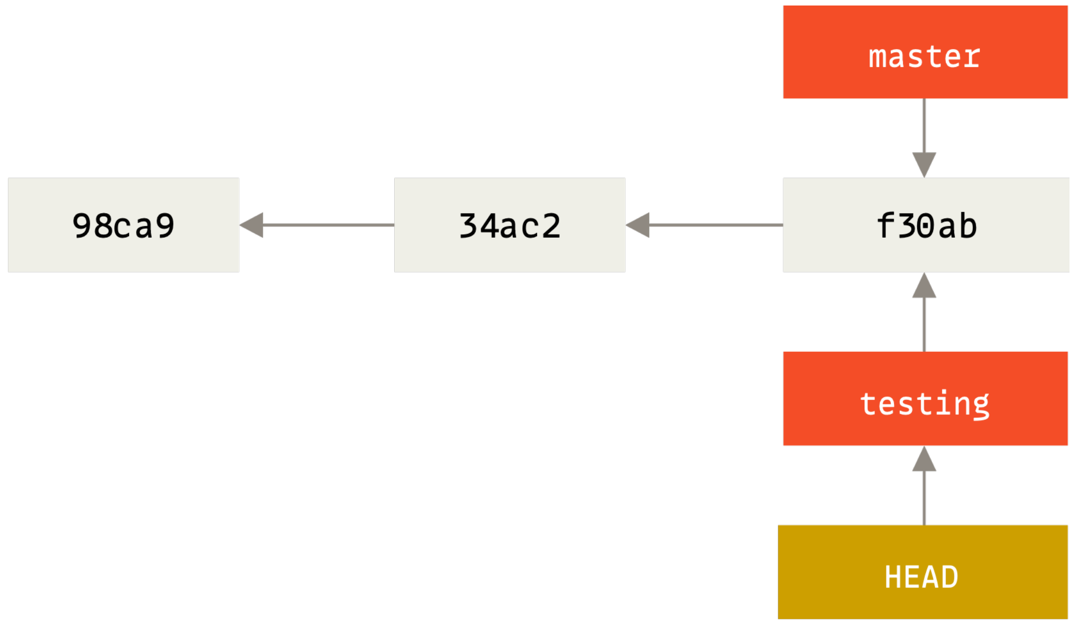
```
git branch testing
```



Branches: switching to another branch (1)

Switching to a branch moves the HEAD.

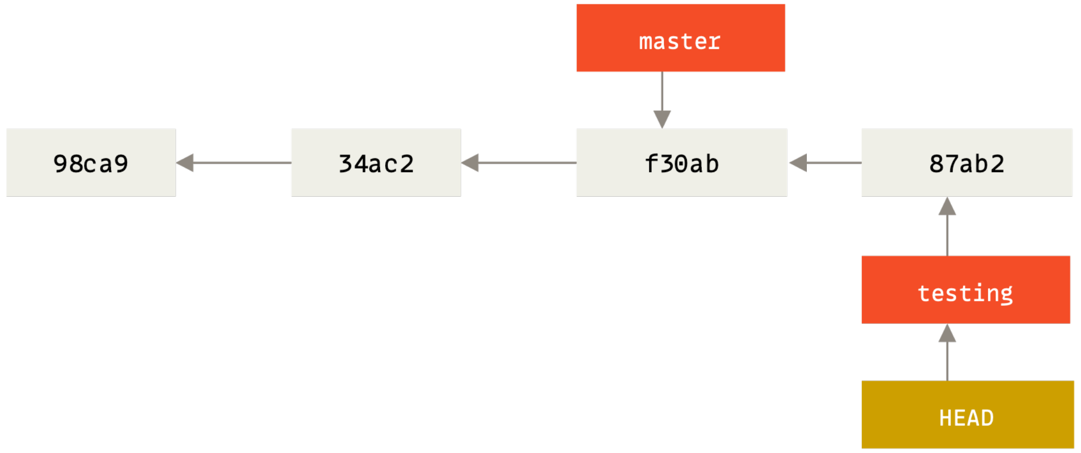
```
git checkout testing
```



Branches: committing to a branch

Committing a change moves the current branch and the HEAD.

```
$EDITOR file.txt
git commit -m "change made"
```

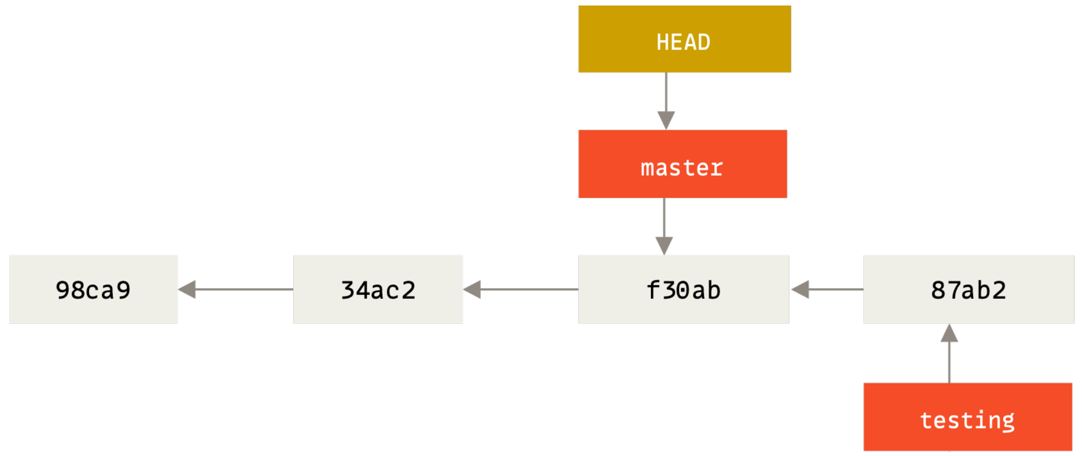


Branches: switching to another branch (2)

Switching to a branch moves the HEAD.

```
git checkout master
```

The commit only exists on the testing branch.

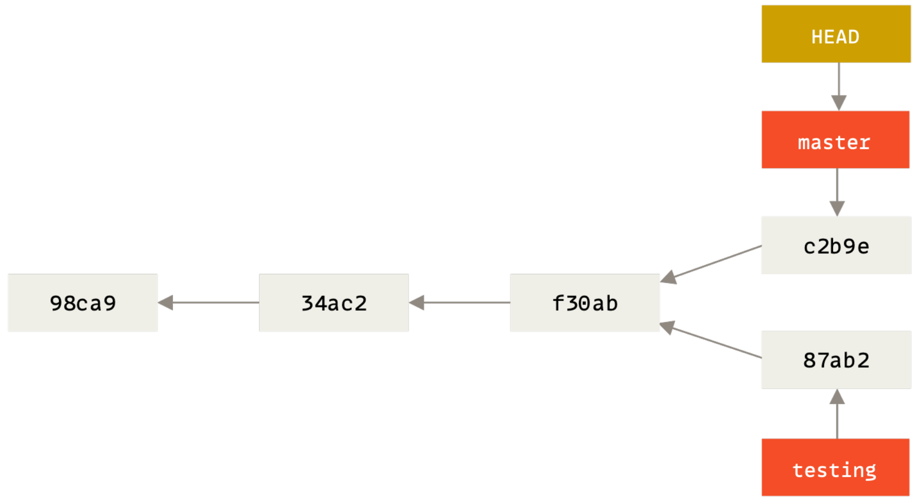


Branches: divergent histories

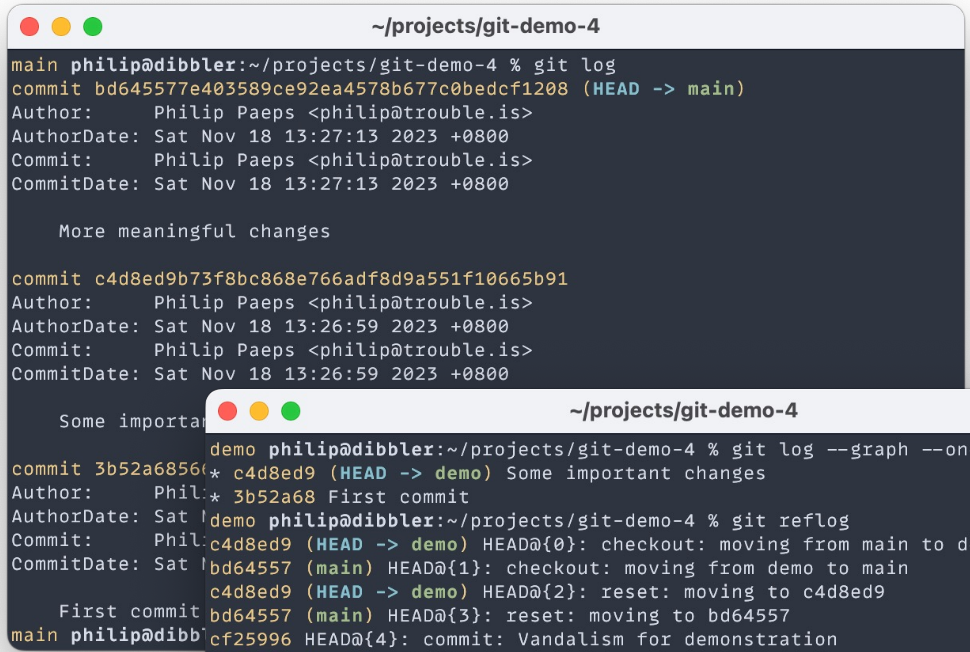
Committing a change moves the current branch and the HEAD.

```
$EDITOR file.txt
git commit -m "change made"
```

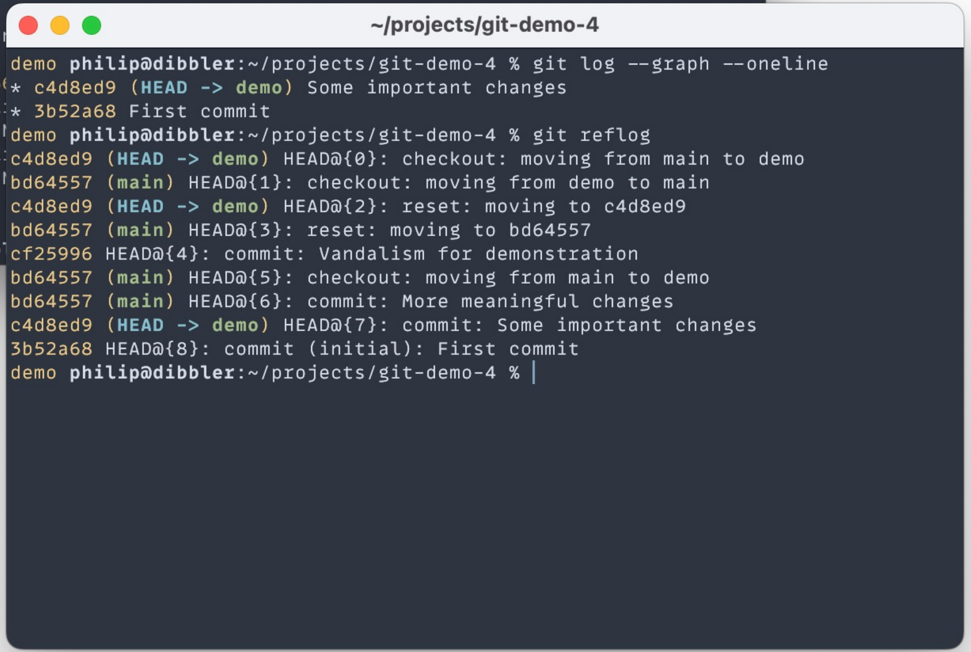
The histories have diverged. Switching between master and testing will show their respective histories.



Using branches to track changes (demo)



```
main philip@dibbler:~/projects/git-demo-4 % git log
commit bd645577e403589ce92ea4578b677c0bedcf1208 (HEAD -> main)
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Sat Nov 18 13:27:13 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Sat Nov 18 13:27:13 2023 +0800

    More meaningful changes

commit c4d8ed9b73f8bc868e766adf8d9a551f10665b91
Author:     Philip Paeps <philip@trouble.is>
AuthorDate: Sat Nov 18 13:26:59 2023 +0800
Commit:     Philip Paeps <philip@trouble.is>
CommitDate: Sat Nov 18 13:26:59 2023 +0800

    Some importa

commit 3b52a6856
Author:     Phil
AuthorDate: Sat N
Commit:     Phil
CommitDate: Sat N

    First commit
main philip@dibb
```



```
demo philip@dibbler:~/projects/git-demo-4 % git log --graph --oneline
* c4d8ed9 (HEAD -> demo) Some important changes
* 3b52a68 First commit
demo philip@dibbler:~/projects/git-demo-4 % git reflog
c4d8ed9 (HEAD -> demo) HEAD@{0}: checkout: moving from main to demo
bd64557 (main) HEAD@{1}: checkout: moving from demo to main
c4d8ed9 (HEAD -> demo) HEAD@{2}: reset: moving to c4d8ed9
bd64557 (main) HEAD@{3}: reset: moving to bd64557
cf25996 HEAD@{4}: commit: Vandalism for demonstration
bd64557 (main) HEAD@{5}: checkout: moving from main to demo
bd64557 (main) HEAD@{6}: commit: More meaningful changes
c4d8ed9 (HEAD -> demo) HEAD@{7}: commit: Some important changes
3b52a68 HEAD@{8}: commit (initial): First commit
demo philip@dibbler:~/projects/git-demo-4 %
```

Remembering where you've been

`git reflog`

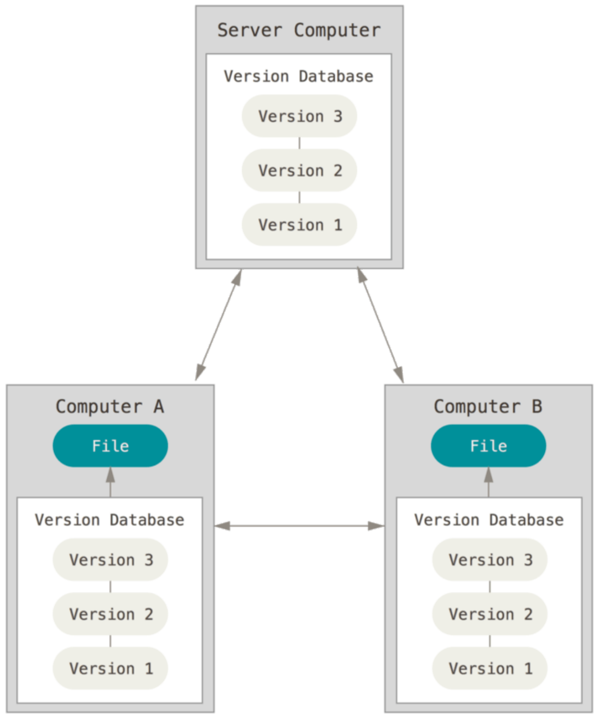
Moving branches

`git reset`

Keeping track of changes on branches

```
git log --graph <branch>
git diff <branch>
```

Remote repositories



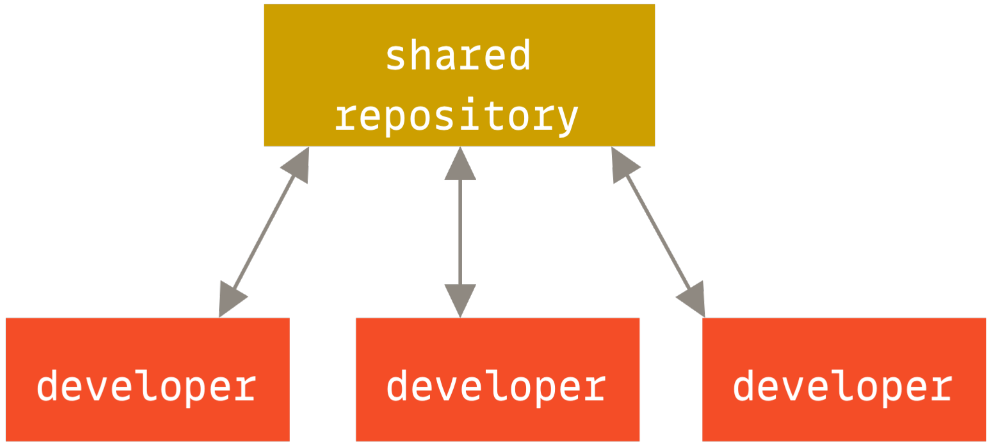
Git is a distributed **revision control system. Adding** remote **repositories enables sharing changes with others.**

Notes that "remote" repositories can be elsewhere on the "local" machine too.

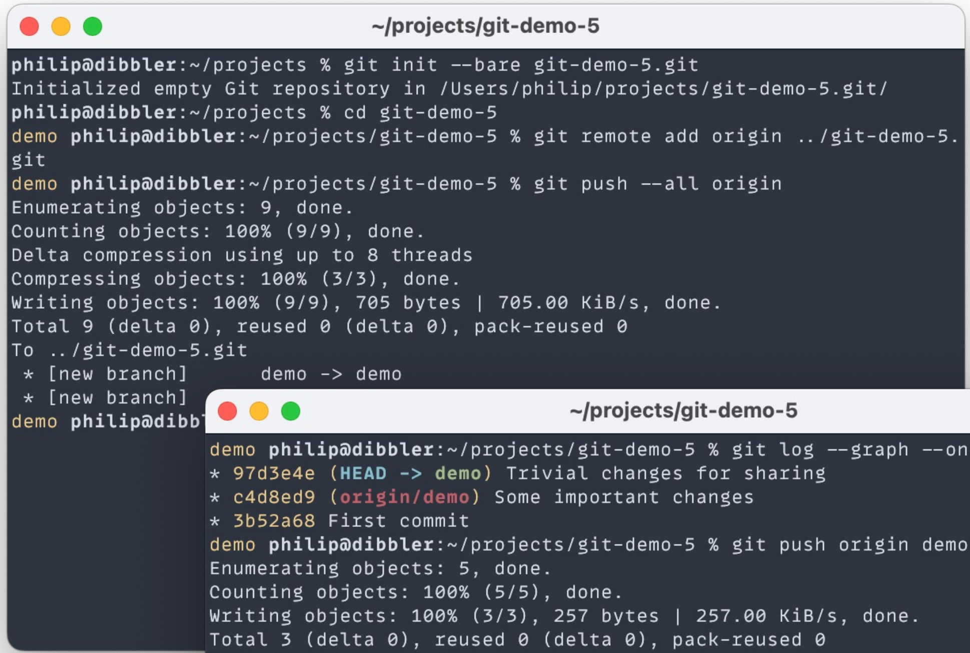
Working with repositories

A remote is a complete clone of the repository including all history. This makes collaborating with others easy.

There are several possible workflows of differing complexity. Most of these are irrelevant to network engineers.

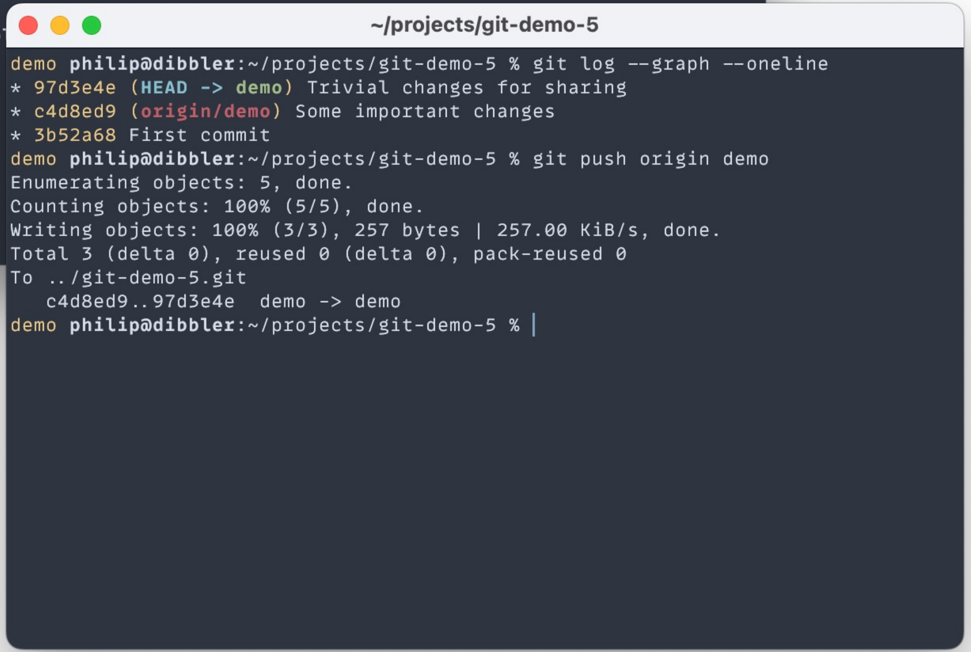


Using remote repositories (demo)



~/projects/git-demo-5

```
philip@dibbler:~/projects % git init --bare git-demo-5.git
Initialized empty Git repository in /Users/philip/projects/git-demo-5.git/
philip@dibbler:~/projects % cd git-demo-5
demo philip@dibbler:~/projects/git-demo-5 % git remote add origin ../git-demo-5.
git
demo philip@dibbler:~/projects/git-demo-5 % git push --all origin
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (9/9), 705 bytes | 705.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To ../git-demo-5.git
 * [new branch]      demo -> demo
 * [new branch]
demo philip@dibb
```



~/projects/git-demo-5

```
demo philip@dibbler:~/projects/git-demo-5 % git log --graph --oneline
* 97d3e4e (HEAD -> demo) Trivial changes for sharing
* c4d8ed9 (origin/demo) Some important changes
* 3b52a68 First commit
demo philip@dibbler:~/projects/git-demo-5 % git push origin demo
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 257 bytes | 257.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To ../git-demo-5.git
   c4d8ed9..97d3e4e  demo -> demo
demo philip@dibbler:~/projects/git-demo-5 % 
```

Adding remote repositories
`git remote add <name> <URL>`

Sharing changes with remotes
`git push <remote> <branch>`

Getting changes from others
`git fetch <remote>`
`git fetch --all`

Merging changes from others
`git rebase <branch>`

GitHub, GitLab, etc

Collaboration tools and Git repository hosting.

Tools for collaboration

GitHub provides hosting for Git repositories.

Superficially targeted at software projects but great for any Git repository.

Issue tracker. Pull requests. Wiki.

The GitHub workflow

1. Fork a repository from a project
2. Clone your fork and make changes on a branch
3. Push the branch to your namespace
4. Create a Pull Request in the project repository
5. Discuss changes and push updates to your branch
6. Project owner merges the accepted pull request

GitLab

Very popular implementation of the GitHub workflow. Developed as an open source project with a premium/hosted business model.

Self-hosted option with convenient integrations for enterprises.

Bitbucket

Variant on the theme. Integrates well with other Atlassian tools. Also has a very credible offline GUI client.

GitHub tour (demo)

Credits and further reading

Most of the images in this presentation are from the excellent "Pro Git" book by Scott Chacon and Ben Straub. (CC BY-NC-SA 3.0)

Book: https://git-scm.com/book/en/v2/
Source code: https://github.com/progit/progit2

GitHub cheat sheet
https://training.github.com/downloads/github-git-cheat-sheet/

Escaping a Git mess (Justin Hileman)
http://justinhileman.info/article/git-pretty/

Thank you.

Philip Paeps

philip@trouble.is

manrs.org