Git Book

/

What is version control?

-

A system that records changes to files over time.

/

What type of version control system is CVS, Subversion, and Perforce.

-

Centralized

/

What type of version control is Git, Mercurial, or Bazaar?

-

Distributed

/

In distributed version control, what does each client receive?

-

A full version of the repository. Thus every checkout is a backup.

/

How does Git think about the data it stores differently from most version control systems?

-

Most VCSs think of data as changes in files, Git thinks of data as a snapshot of the entire filesystem at given point.

/

Does Git work offline?

-

Yes.

/

What mechanism does Git use for checksums?

-

SHA-1 hash

/

How does Git reference files in its database?

-

Via the hash

/

What are the three states files in Git can be in?

What do they mean?

-

1) Committed – Data is saved in local database (Unmodified)

2) Modified – Changed file but not committed to database

3) Staged – Modified file is marked to go into next commit (snapshot)

/

What are the three main sections of a Git project?

-

1) Working Directory – Single checkout of one version of a project. Pulled from compressed database and on local disk to work with.

2) Staging Area – File containing data about what will go into next commit. Sometime called the index.

3) Git Directory – Where metadata and objects are stored. Part that is cloned.

/

What tool can be used to change configuration settings?

Where are the three places these configuration settings can be stored?

-

git config

1) /etc/gitconfig – for all users, pass option --system to git config

2) ~/.gitconfig – for current user, pass option --global to git config

3) .git/config – local to project, trumps others

/

What should you always set when you install Git? Why?

-

Identity info:

git config --global user.name “Super Fresh”

git config --global user.email [funky@fresh.com](mailto:funky@fresh.com)

Identity info is included with every commit.

/

What two other things can you set?

-

Your text editor and diff tool

/

What are the three ways to get help.

-

1) git help <verb>

2) git <verb> --help

3) man git-<verb>

/

What are the two ways to start a Git project?

-

1) Start new one with git init, which creates a .git file to store the skeleton.

2) Clone an existing Git project repository

/

Why does Git use clone, while other VCSs use checkout?

-

Because Git copies the whole database, and doesn’t just checkout files.

/

What two states can a file in your project be in?

-

Tracked and Untracked

/

What command is used to determine the state of files?

Does it show untracked files?

-

git status

yes

/

What command tells git to track a file?

-

git add <file name>

/

Is it possible for a file to be both staged and not staged?

-

Yes, if it was modified, added, and then modified again prior to commit.

/

What file holds the name of files to ignore?

What patterns does it recognize?

-

.gitignore

# for comments

glob patterns

end with / for directory

start with ! to negate a pattern

/

What are glob patterns?

-

Simplified regular expressions used by the shell.

\* - matches zero or more characters

[abc] - matches any character inside the brackets

? – matches a single character

[0-9] – matches characters in the range.

\*\* - ignore in subdirectories too (doc/\*\*/\*.txt)

/

What does git diff with no arguments do?

-

Shows you what you’ve changed but not yet staged. Diff of files in working directory and staging area. Important: doesn’t show all changes, only unstaged ones.

/

How do you compare your staged changes to your last commit?

-

git diff --cached or git diff --staged.

/

What does the –v commit option do?

-m ?

-a?

-

Shows the diff of the file/s. This diff and the comments are stripped out when the commit message is saved.

Inline commit message.

Skip staging area by adding all files.

/

How do you remove a file?

-

git rm <file name>

just deleting a file doesn’t stage the file. rm is needed just like add.

Use –f option if the file is already staged.

/

How do you remove a file from being tracked, while still keeping it on disk (i.e. when you forgot to add something to .gitignore.

-

git rm --cached <file name>

/

What can be passed to git rm?

-

Files, directories, and glob patterns.

Need a \ in front of \* pattern

/

What is mv?

-

A rename convenience: git mv file\_name\_old.txt file\_name\_new.txt. Same as mv on local, rm original and add new.

/

How do you list commits made in reverse chronological order?

What 4 things are shown?

-

git log

commit’s SHA-1 hash, author’s name and email, and the commit message

/

How do you show the diffs?

How do you limit the entries?

How do you show word-diffs?

How do you limit the normal 3 line context?

Show abbreviated stats?

-

-p to show diffs

-# (i.e. -2 to limit to 2 entries)

--word-diff

-U1 to show 1 line context

--stat

/

Which two ways can you control output?

-

--pretty has options oneline for viewing many commits, along with short, full and fuller.

It also has format opton (--pretty=format:”%h - %an, %ar : %s” so you can totally specify the output for parsing.

/

What shows an ascii drawing of merge history?

And what format options are usefully to use with it?

-

--graph

--online or --format (i.e. git log --pretty=format:”%h - %s” --graph)

/

How do you limit the output to a directory or file in git log?

How do you limit it to a date?

How do you limit it to an author?

How do you limit it to specific text?

-

pass -- path after all the options. ‘--' is used to separate the path from the options, a space is between it and the path, and the path is always last.

--since or --until to limit it to dates

--author to limit it to an author

--grep text to search for text

(--all-match is necessary with grep and author to limit it to cases where both are matched)

/

How do you launch the Tcl/Tk git viewer?

-

gitk and pass almost all the filtering options of git log

/

How do you change your last commit?

-

git commit --ammend

Will launch the editor with your last commit. You can add new files prior to executing this command if you forgot to add a file with the commit.

/

How do you unstage a file?

-

git reset HEAD <file>

(git status reminds you)

/

How do you unmodify a modified file (re-checkout the file from the git database)?

-

git checkout -- <file>

(git status also reminds you. This command can loose info since the file changes where never commited)

/

What are remote repositories?

What name does git give to the repository the project was cloned from?

How do you list remotes?

How do you show the urls?

-

Repositories on the internet or network.

Origin.

git remote

-v option

/

How can you add a remote?

What does the shortname do?

-

git add [shortname] [url]

the shortname is an easy reference for fetching/pulling/pushing

/

How do you get all the data from a remote repository without merging?

How do you get the data from a repository you are tracking and merge automatically?

-

git fetch <remote-name>

git pull <remote-name>

(git pull alone will generally pull and merge the origin)

/

How to you send changes you’ve made to a remote?

-

git push [remote-name] [branch-name]

/

How do you inspect a remote?

How do you rename a remote?

How do you remove a remote?

-

git remote show [remote-name]

git remote rename [old-name] [new-name]

git remote rm [remote-name]

/

How do you mark a specific commit?

What are the two types of marks?

How do you show them?

Are they pushed automatically to remotes?

How do you mark an old commit?

-

With git tag -a <label> -m “message”

Where label is something like v1.4.

Anotated (checksum, tagger, email, message, Gnu privacy guard if desired.)

and lightweight (only the label).

git tag

No, must push with git push remote [tag-name] or --tags to push all.

Same as normal but add short sha or sha of commit to the end

/

Can you make aliases and enable autocomplete?

-

Yes.

/

When Git performs a commit, what 3 things does it store?

-

A commit object containing:

1) a pointer to the snapshot of the data you staged : a tree (changed directories) with blobs (changed files)

2) the author and message metadata

3) zero or more pointers to the parent commits

(0 for the first commit, 1 for normal, 2 for a merge)

/

What is a branch?

What is the master branch?

How does Git know which branch you are on?

-

A branch is a lightweight pointer to a commit.

The master points to the last commit and moves forward automatically.

Git keeps a pointer HEAD, the points to the current branch

/

How do you create a branch?

How do you switch to a branch?

How do you do both?

-

git branch [name]

git checkout [name]

git checkout –b [name]

/

How can you cause two branches to diverge?

-

Create a new branch, switch to it, make a commit. Switch back to the old branch, make a commit, and now the two branches have diverged.

/

Why are branches one of Git’s advantages.

-

Because they are simply light weight points, whereas in most VCSs they are copies of the repository.

/

How do you merge a branch?

What is a fast forward?

How do you delete a branch?

-

git merge [branch] (so finish commits, go to master, and then merge other branch)

A fast forward is when the branch head only needs to move forward to a new commit since there is no divergent work to merge.

git branch –d [name] to delete (branch must be merged into current unless use -D)

/

With two divergent branches: master and hotfix, what are the two ways the work can be merged?

-

Changes in master can be pulled into hotfix, or hotfix’s changes can be pulled into master.

/

What three snapshots does Git use for merging divergent branches?

-

The snapshots of the commit to merge into, the commit to merge in, and the common ancestor of the two.

/

What happens if two branches have modified the same part of the same file?

How do you determine the file?

How does git change the file?

How do you launch a merge tool?

How do you know the file is ready to be merged?

How do you finalize the merge

-

A merge conflict happens and Git pauses the merge.

git status will show the file as unmerged

Git places the diff inside the file. You must choose one version or the other, or merge the conflicts. Above ===== is the merged into file, below the merge in file.

git mergetool

run git status again to verify file has changed to modified

git commit (you can add a message describing the merge)

/

How do you list branches?

What indicates the current branch?

What option shows which branches are merged with the current branch?

Un-merged?

-

git branch

\*

--merged

--no-merged

/

What are two types of common git workflows?

-

Long running branches – where master is the most stable, new work is done on develop and merged in later once stable.

Topic branches – where new branches are used for new features or bug fixes.

(can use both together – master, develop, topic1 off of develop, bugfix off of master)

/

What are remote branches?

-

Branches in your local repository that refer to branches on your remote repository. You can’t move them, they only change via network communication.

/

Can you work on remote branches that you fetch?

-

No you must merge them into your local branch. So if you fetch origin/hotfix you must merge that into you local hotfix branch.

/

How do you share work?

-

git push (remote) (branch)

/

What are tracking branches?

-

Local branches with a direct relationship to a remote branch (i.e. when you clone a repository your local master refers to origin/master and thus pull and push automatically refer to it. When you checkout a remote branch it is set up as tracking: git checkout –b [branch] [remotename]/[branch]. You can also use shorthand --track.)

/

How do you delete a remote?

-

git push origin :[branch]

/

What are the two main ways to combine branches?

-

Merge and Rebase

/

What does rebase do?

Why would you use it?

-

It replays changes in one branch onto another so there is only one linear branch.

It achieves the same results as merge, but creates a linear history, so when you push to a remote repository you don’t own, the person merging the branch only needs to do a fast forward merge.

/

What is the peril of rebasing?

-

Do not rebase commits you’ve pushed to a public repository. Others will need to merge your new rebased work in their own if they had pulled your original commits. So only rebase local work prior to pushing.

/

What command is used to create a bare repository for a server from an existing project?

-

git clone --bare my\_project my\_project.git

/

What are three types of distributed workflows?

-

1) Centralized – everyone pulls and pushes

2) Integration-Manager

3) Dictator and Lieutenants – like integration-manager with multiple masters and one main master

/

What are the steps of the Integration-Manager workflow (such as that used on Github)?

-

1) Project maintainer pushes to public repository

2) Contributor clones the repository (fork on Github)

3) Contributor makes changes and pushes to their copy

4) Contributor emails maintainer with a pull request

5) The maintainer adds the contributor’s repository as a remote and pulls/merges changes to his local

6) The maintainer pushes to the main public remote

/

What are some commit guidelines?

-

Check for white space errors with git diff --check

Try to make each commit logically separate changes (use --patch if necessary)

Write commit messages with 1 line summary, space, additional details include – bullet points if necessary. Mention motivation for change and contrast it with previous implementation. Use present imperative tense: “Add tests for…”

/

When push up changes to a fork should you use master?

-

No. If the maintainers cherry-pick changes or don’t merge anything, pushing a fork for the pull request is best since it is easier to merge the public master.

/

How do you check which commits are in a branch but not in the master?

-

git log (branch) --not master

/

When will git diff master give funny results?

-

When you are on a branch and comparing it with master if master has moved forward. The diff will seem like all newly added lines are being deleted from master. You want to compare the branch with it’s common ancestor with master. Use git diff master…(branch)

/

How do you pull in one commit from a branch?

-

git cherry-pick (commit hash)

/

How can you create a release of your code for people who don’t use Git?

-

git archive

/

If you’ve done work on a branch that your not ready to commit, but you need to switch branches, what command can you use?

-

git stash

retrieve with git stash apply

remove with git stash drop

(can have multiple stashes)

/

How do you interactively stage files?

-

git add –i

/

How do you change the last commit?

Older ones?

-

git commit --amend (don’t do if already pushed since changes the sha-1)

git rebase –i HEAD~3 to edit last 3 commits and current in interactive editor

(this will change current commit and last three, so don’t do if already pushed)

switch the word pick to edit for the commits you wish to change and hit enter

type git commit--amend to change the commit and git rebase --continue when done

You can reorder, delete, and squash commits from the interactive editor.

/

What is the nuclear option to remove a file from every commit?

-

git filter-branch --tree-filter ‘rm –f password.txt’ HEAD

/

How can you trouble shoot bugs?

-

git blame to see changes to part of a file

git bisect to isolate the middle point between a good state and a bad state.

/

How can you interact programmatically with Git events?

-

Through client side and server side hooks. Stored in .git/hooks directory. Can use any scripting language.

/

What are the four main parts of .git?

1) HEAD – a file pointing to the current branch

2) index – a file storing staging information

3) ojbects – directory storing all content for the database (key-value store)

4) refs – directory storing pointers into commit objects in the data (branches)

/

What are the three main types of objects in Git?

-

1) commit objects – details of commits, point to a tree and parent commit/s

2) tree objects – details of a directory

3) blob object – a compressed file

/

What 3 things are stored in ref (references)?

-

1) branches

2) tags – (another type of object similar to commits)

3) remotes

/

What are packfiles?

-

Git initially stores objects as their own blob for each file, but if only a single line is changed this can be expensive. So Git combines loose objects into a single binary packfile that tracks changes.