



Git Smarter

20 Tips From Beginner to Power User

A Bit About Me

My Name is Jacob Beers

- I've been a developer for a little over 10 years now
- I started at Ortus about 2 years ago as a Senior Software Developer
- I live in KCMO with my wife and 3 boys



Follow Along

This QR code provides a link to my site where you can download these slides and follow along if you want to click links and whatnot.

I don't blame you if you don't want to scan random QR codes. Here's the link.

<https://jbeers.github.io/talks/git-tips/>



Expectation Management

I don't know what I was thinking when I suggested a talk on 20 tips.

20 tips is too many for a 50 minute talk. (That's like 2.5 minutes per tip)

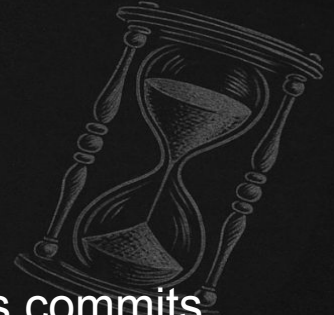
20 tips is not nearly enough to cover everything Git has to offer.



“Say as many of them as you can as fast as you can. Don’t stop for any reason.”

- Zapp Brannigan

What is Git?



Git is a version control system where changes are recorded as commits. Each commit is a snapshot of the project at a moment in time, identified by a unique SHA hash.

Commits are immutable: if anything changes — even metadata like time or author — Git creates a completely new commit. Commits can be referenced not only by their hash but also by other refs, such as branches and tags, which are just names pointing to commits.

I like to think of git like a time machine. More like Avengers: Endgame less like Back To the Future.

Tip #1 - Check the Release Notes

Believe it or not git has releases!

Initially released on April 7th, 2005 git is still under active development 20 years later!

Github blog has a great series where they review each release and share highlights
<https://github.blog/open-source/git/>

Here is the latest release notes:
<https://github.blog/open-source/git/highlights-from-git-2-51/>



```
$ git init
Initialized empty Git repository in /tmp/tmp.I8BYSY7R8Y/.git/
$ cat > README << 'EOF'
> Git is a distributed revision control system.
> EOF
$ git add README
$ git commit
[master (root-commit) e4dccc69] 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:cdowen/chaos.git
$ git push -u origin master
```

A command-line session showing repository creation, addition of a file, and remote synchronization

Original author(s) Linus Torvalds^[1]

Developer(s) Junio Hamano and others^[2]

Initial release 7 April 2005; 20 years ago

Stable release 2.51.0^[3] / 18 August 2025

Tip #2 - Get a Cheat Sheet

Git has over 140 commands! Use a cheat sheet.

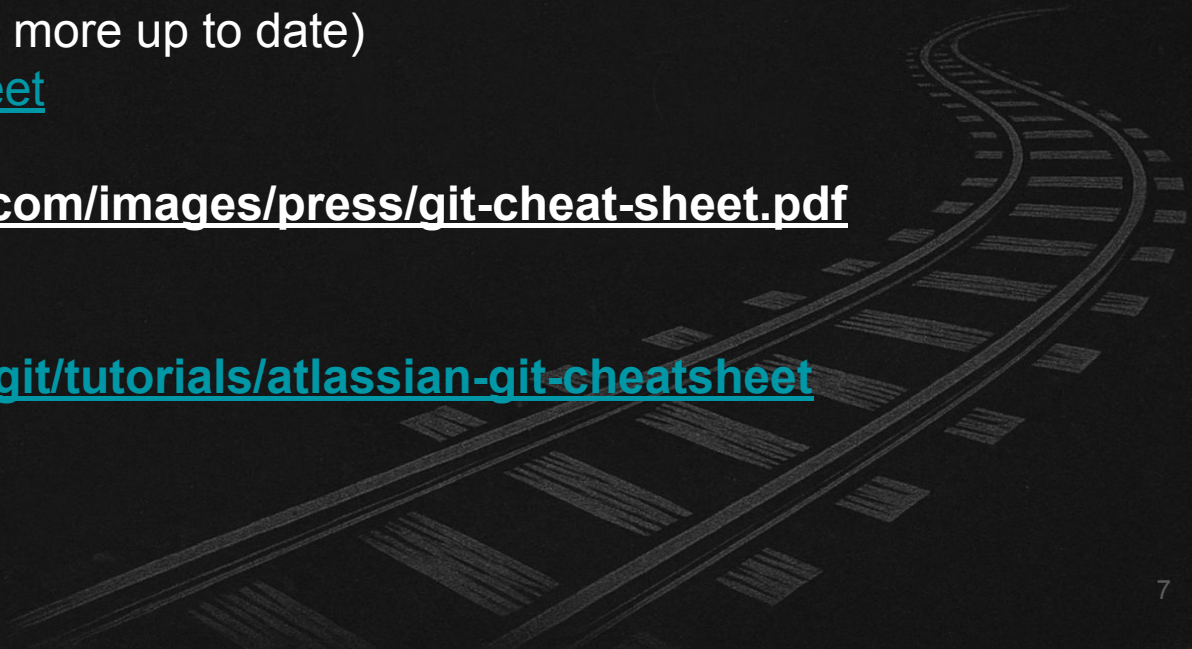
Official Git: (recommended - more up to date)

<https://git-scm.com/cheat-sheet>

Gitlab: <https://about.gitlab.com/images/press/git-cheat-sheet.pdf>

Atlassian:

<https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet>



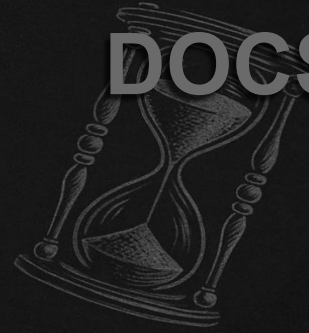
Tip #3 - Read “Pro Git”

“Pro Git” is a book written by Scott Chacon and Ben Straub is available for free as a PDF.

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

I have seen this book referenced ALL OVER the place.

I haven't read it yet but I look forward to it!



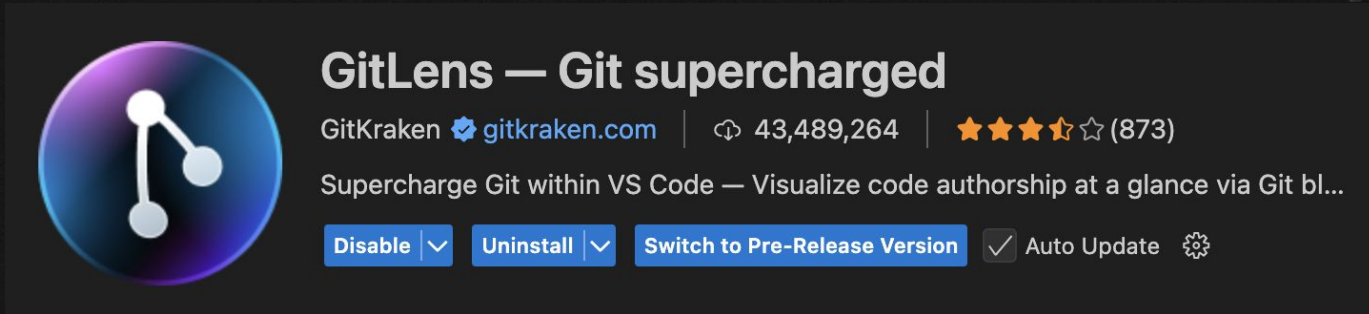
Tip #4 - GUI vs CLI - Why Not Both?

WORKFLOW



My favorite GUI is VSCode + GitLens

GitLens is owned by GitKraken so you can get both under the same license.



GitLens — Git supercharged

GitKraken gitkraken.com | 43,489,264 | 4.5 (873)

Supercharge Git within VS Code — Visualize code authorship at a glance via Git bl...

[Disable](#) [Uninstall](#) [Switch to Pre-Release Version](#) ☒ Auto Update [Settings](#)

Don't neglect the CLI! It is a pain to get started using but it pays for itself!

Tip #5 - Design Your Workflow

Develop a ritual - learn from Gilbert Arenas. Guess his free throw percentage!

My workflow

- `git stash -p` (interactive stash)
- `git reset --hard`
- `git switch -c wip/my-feature`
- Write test, code, run test, commit, repeat
- `git rebase -i main`
- `git push` and create PR





Tip #6 - Quality Commits (part 1)

Use message and summary.

✗ fix bug

Don't slack! Write something useful!



PRJ-85 Add caching to API

Commit messages can be multiline. 1st line is the header. 2nd line is the message body.

Chester and I found an issue where the API ran slower than we expected. We couldn't improve DB performance due to the table architecture. Adding caching was a simple fix that did the job.



Tip #7 - Quality Commits (part 2)

Writing a good commit message takes time and effort.
What do we do when things are hard? Procrastinate!

Commit early and commit often - and don't worry about your commit messages.

Once you are done working use ``git rebase -i`` to start an interactive rebase and clean everything up.

Make sure to read these guides!

[StackOverflow - Use Git Tactically](#)

[Hackernoon - Interactive Rebase Guide](#)

Tip #8 - Alias and Document Helpful Commands



Learn a lesson from Hansel and Gretel - leave yourself crumbs!

Makes aliases for lengthy commands that will help you remember them.

Make aliases/functions for steps that streamline your workflow.

Make aliases for obscure commands that take 30 minutes of google searching for. Your future self will thank you!

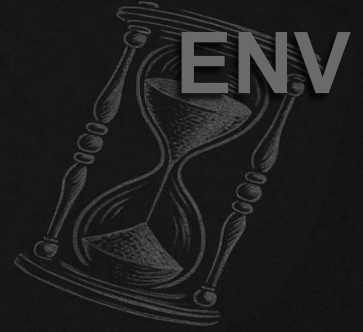
Tip #9 - Use Git For More Than Projects

If you don't have one make a repo for personal use.
(Seriously do it right now!)

Put everything (except for secrets!) in there

- command aliases
- one-off scripts
- notes and documentation
- env configuration
- Templates
- Video game saves

<https://github.com/jbeers/jtb-env>





Tip #10 - Git Static Hosting

Several companies provide static site hosting for git repos.
My favorites are github pages and netlify.

Lots of benefits

- Free
- No database
- Change tracking
- Low commitment
- Serverless

[Github Docs](#)

Check out my tools at <https://jbeers.github.io/terminal>

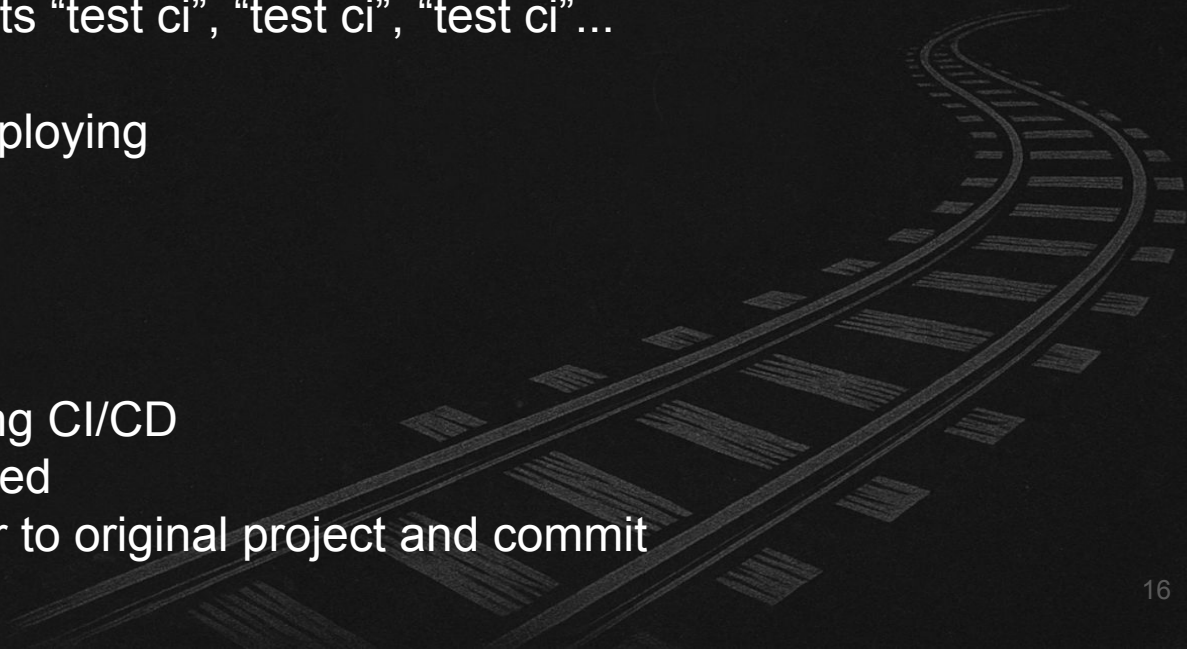
Tip #11 - Use a Fork for CI/CD Development

Oftentimes working on CI/CD automation can be painful.

- Tons of useless commits “test ci”, “test ci”, “test ci”...
- Messy CI/CD history
- Fear of accidentally deploying

Try using a fork instead.

- Fork the project
- Remove/disable existing CI/CD
- Write only what you need
- Copy final product over to original project and commit



Tip #12 - Git Commands in CI/CD

It is common to use `git pull` in CI/CD. Did you know you can do a shallow clone? Usually controlled with `--depth` cli arg or a depth command.

Can make large repos download quicker.

Get creative with more than just `pull`

- Generate tags
- Conditional logic based on commit message
- Detect which files changed and selectively deploy
- Generate changelog

Tip #13 - New vs Old Skool

Many Git commands are being updated in favor of newer syntax.

`git checkout <branch> -> git switch <branch>`

`git checkout <file> -> git restore <file>`

`git filter-branch -> git filter-repo`

Usually a command is not fully deprecated, just a particular usage.

GIT COMMANDS



Tip #14 - Stash Like a Pro

Create stashes for things you want to keep but don't want to commit.

You can now share stashes with others!

<https://github.blog/open-source/git/highlights-from-git-2-51/#h-stash-interchange-format>

Often GUIs stash all or nothing - on the CLI you can stash selectively and/or interactively.



Macintosh HD — Git Smarter — -zsh — 86x9

```
git-tps / > git stash -p
```

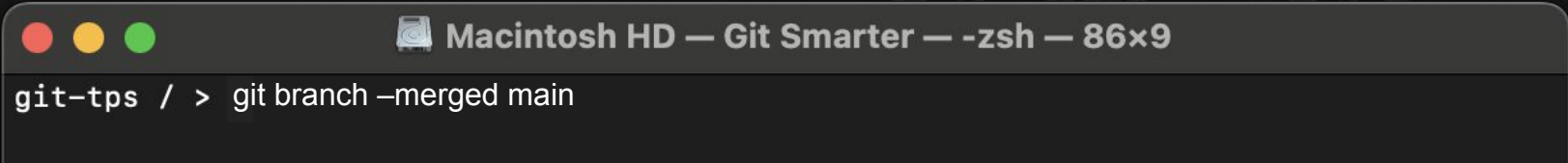
Tip #15 - Prune Your Branches

Old feature branches that have been fully merged do not serve a purpose. You can safely remove them.

If you think you might want to “recover” that at some point in the future then you should use a tag.

Use ``git branch --merged main`` to list branches that can be deleted.

Use ``git branch --merged main | grep -vE '^*|main' | xargs -r git branch -d`` to safely remove merged branches.



```
git-tps / > git branch --merged main
```


Tip #16 - Use Tags

Tags are a great way to create “save points” to reference later on.

Often used for releases but can be used for anything.

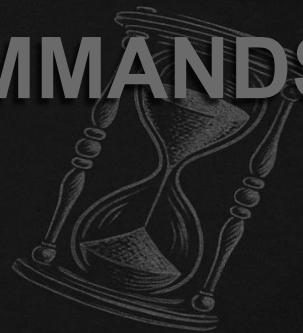
Things to tag

- Releases
- Branch points
- Milestones

Bonus: tags can have messages



```
git-tps / > git tag -a v1.0.0 -m "Release version 1.0.0 with new features"
```

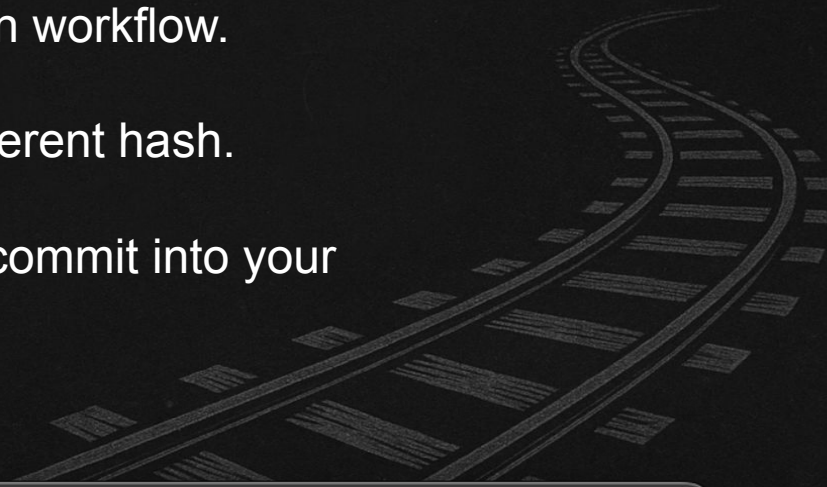


Tip #17 - Use Cherry Pick Wisely

Be careful when cherry picking
Sometimes it is your only option.
Sometimes it means you have a broken workflow.

Creates a totally different commit with a different hash.

Use `--no-commit` to get the contents of the commit into your working files.



```
Macintosh HD — Git Smarter — -zsh — 86x9  
git-tps / > git cherry-pick <ref>
```

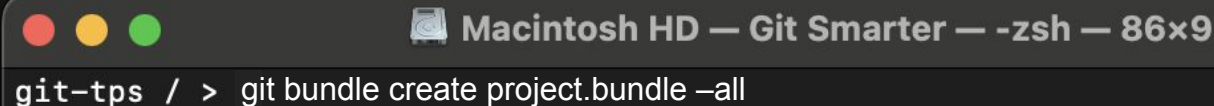
Tip #18 - Backups with Git Bundle

Git bundles can be a good way to archive and share repos if you aren't using a service like github/gitlab.

Use “git bundle create project.bundle --all”

A bundle is a single file that can be used as a read-only remote.

```
git clone repo.bundle my-repo  
git remote add bundleRemote /path/to/project.bundle
```

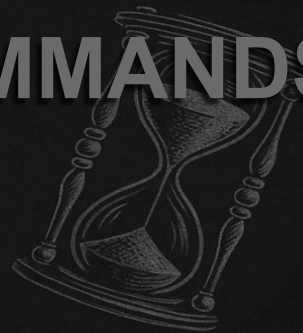


Macintosh HD — Git Smarter — -zsh — 86x9

```
git-tps / > git bundle create project.bundle --all
```


Tip #19 - SVN + Git

GIT COMMANDS

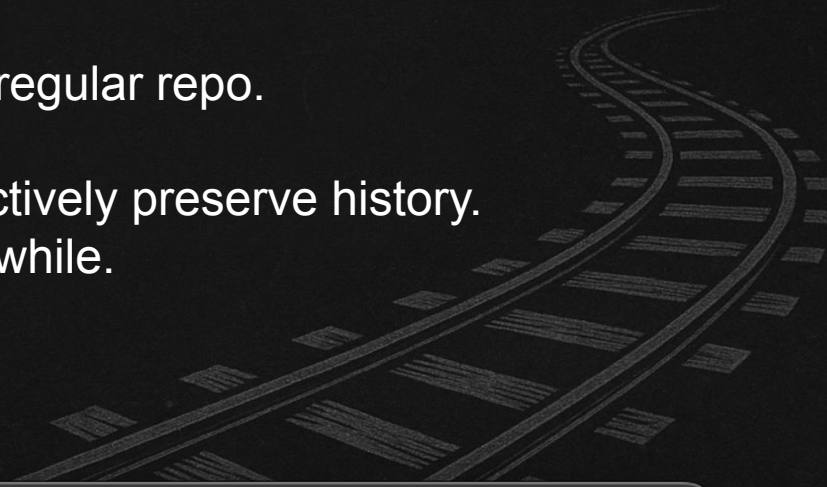


Git has tools for acting as an SVN client as well as importing/converting an SVN repository.

You can push and pull from SVN just like a regular repo.

When converting a repo to git you can selectively preserve history. Great for projects that have been around a while.

[Git docs for the svn command](#)



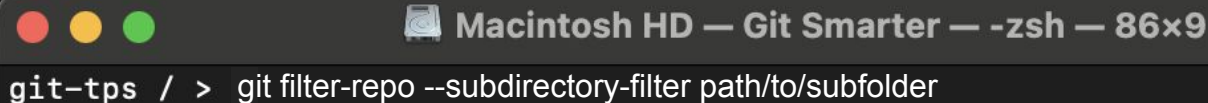
```
Macintosh HD — Git Smarter — -zsh — 86x9
git-tps / > git svn clone <SVN_URL> <target-directory>
```

Tip #20 - Break It Up

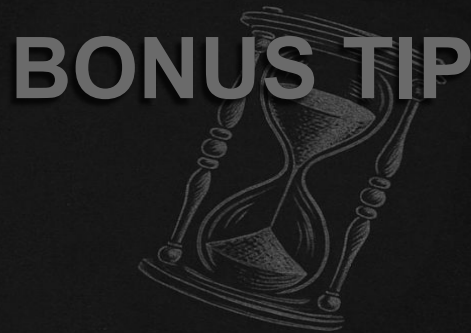
Git is so flexible that it gives you the ability to separate out subfolders into completely new repos. Have a sub project that has grown into its own library? No problem!

Start with a fresh clone and then use filter-repo.

The subfolder contents will now be in the root directory and your git history will be focused on that folder's history.



```
git-tps / > git filter-repo --subdirectory-filter path/to/subfolder
```



Tip #21 - Start With Git Log

Start your week off with ``git log``.

You can use this to review all the work that has been done since a certain time or would be pulled into your branch.

Use ``--since="2 weeks ago"`` for a time based log.

Use ``HEAD..origin/main`` to see what would be pulled into your branch.

A screenshot of a terminal window on a Mac. The title bar shows "Macintosh HD — Git Smarter — -zsh — 86x9". The terminal content shows a command to run git log with various formatting options.

```
git-tps / > git --no-pager log --since="2 weeks ago" \  
--pretty=format:"%C(yellow)%h %C(cyan)%ad %C(green)%an%C(reset)%n  %s%n%n" \  
--date=short
```


Tip #21 - Push the Limits

BONUS TIP



Putting all this together we can accomplish some cool stuff!

- I've made a personal repo
- I've been adding new tools and documenting my setup
- I created a personal site using github pages and added a /terminal
- Now I can access my tools from anywhere!

This will instantly give me access to all my custom tools from anywhere!

<https://github.com/jbeers/jtb-env/blob/main/shell-scripts/.bashrc>

<https://jbeers.github.io/terminal>

We've Barely Scratched the Surface



hooks	.git folder	signing	ref types
porcelain	gitflow	lost commits	lfs
-graph	monorepos	blobs	blame
merge strategies	diff	gc	config

And more!