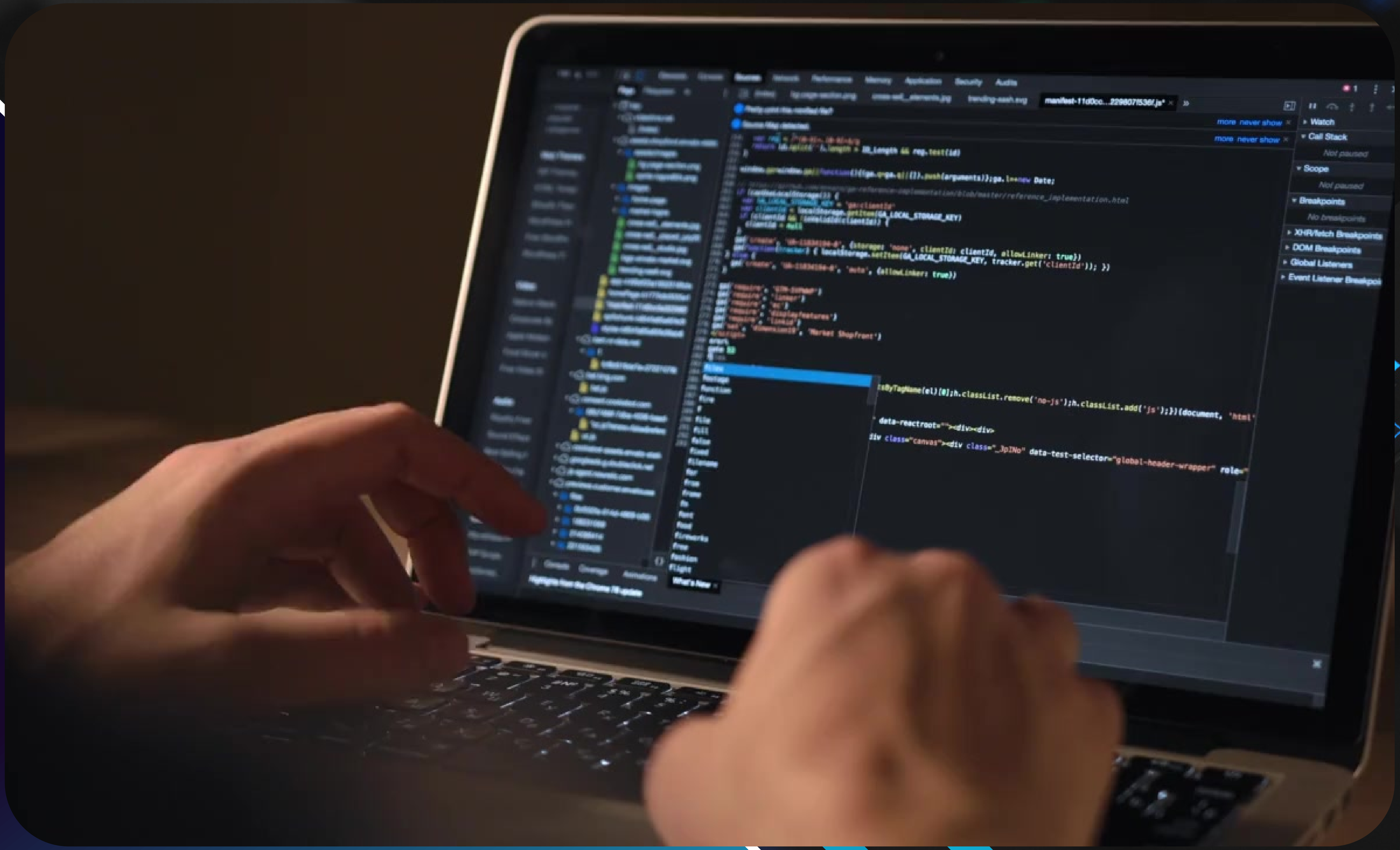
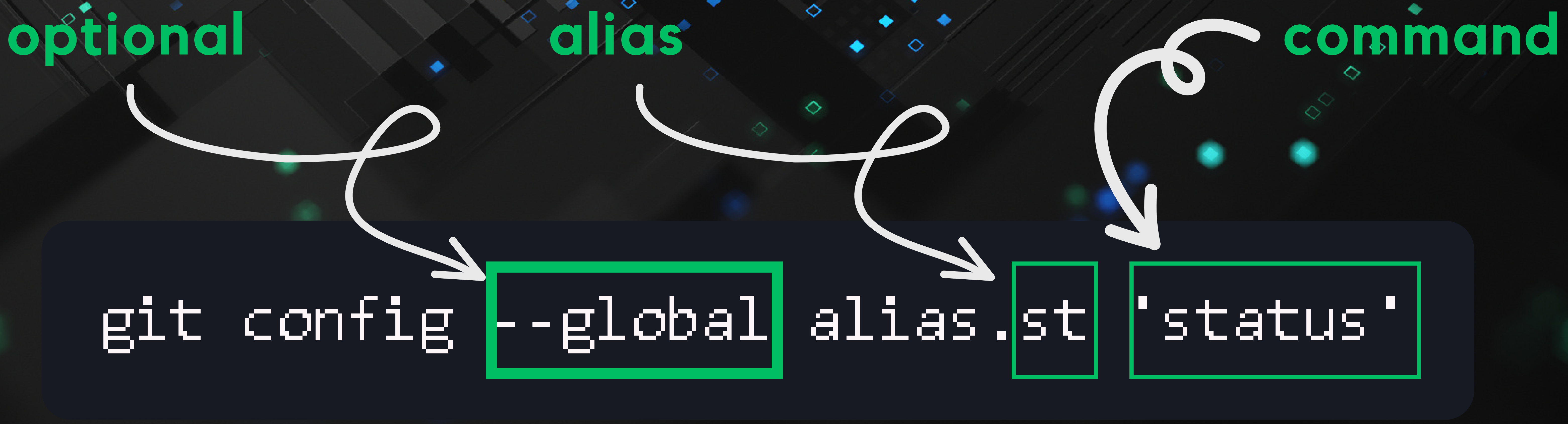
Git Shortcuts



Productivity

The main reason we should use shortcuts more is to increase our productivity and the speed at which we work.

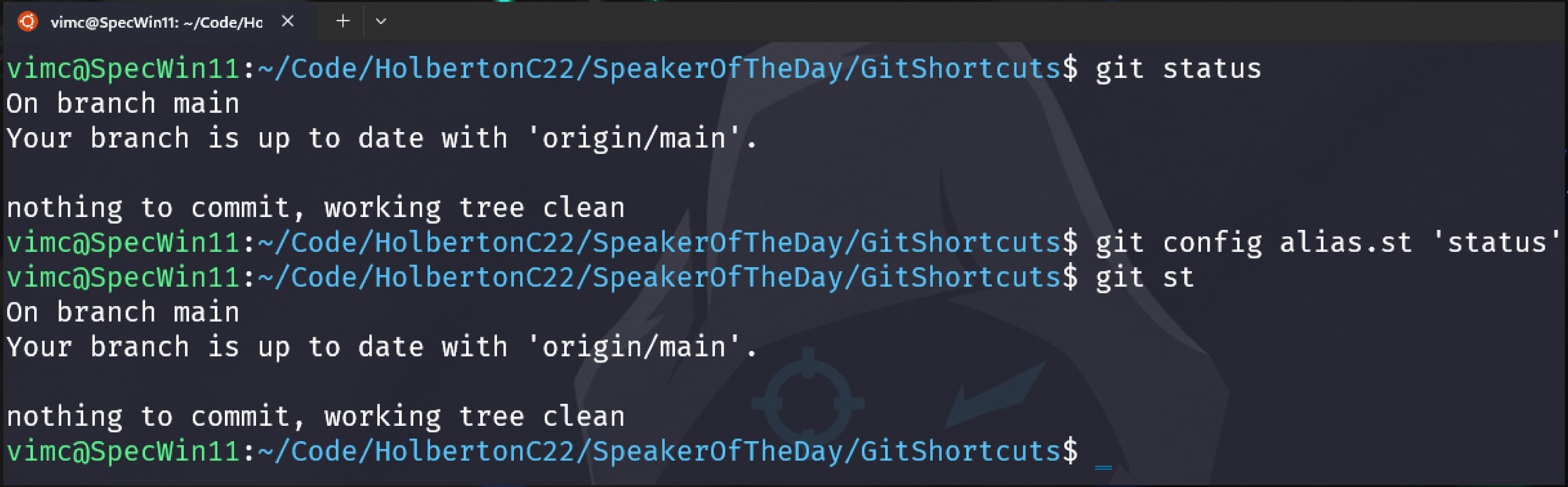
how to create git command aliases



```
git config --global alias.st 'status'
```

GITSHORTCUTS

example

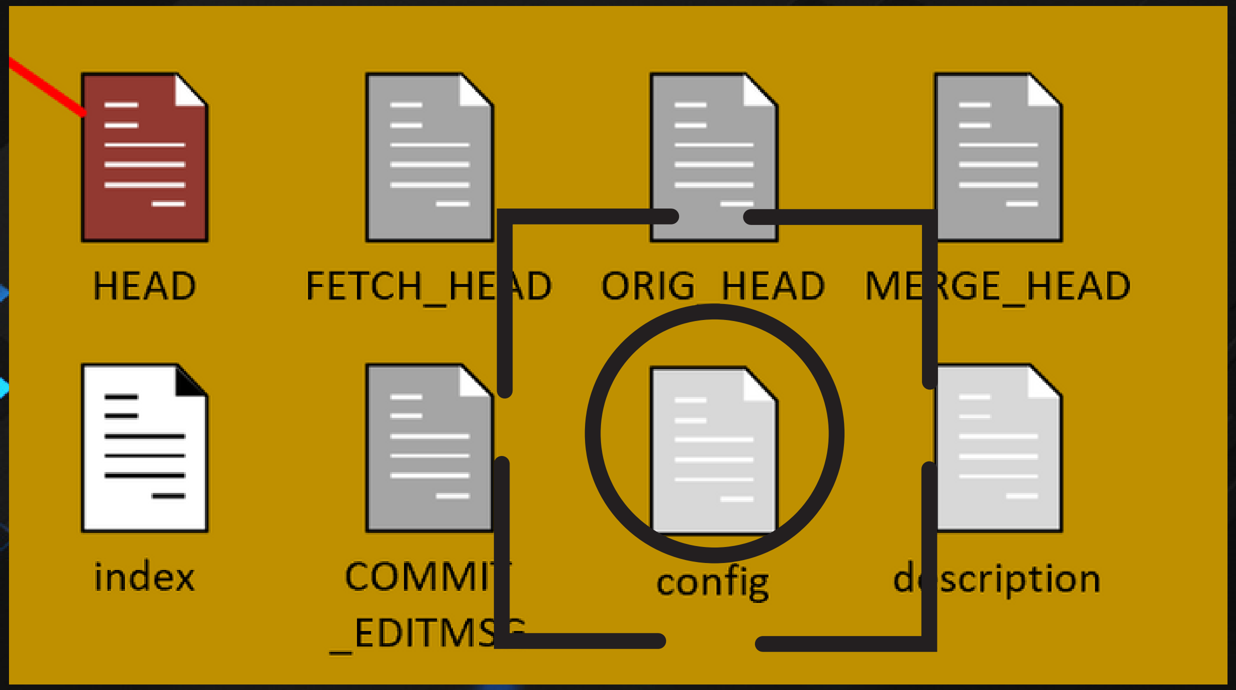


```
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts$ git config alias.st 'status'
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts$ git st
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts$ _
```

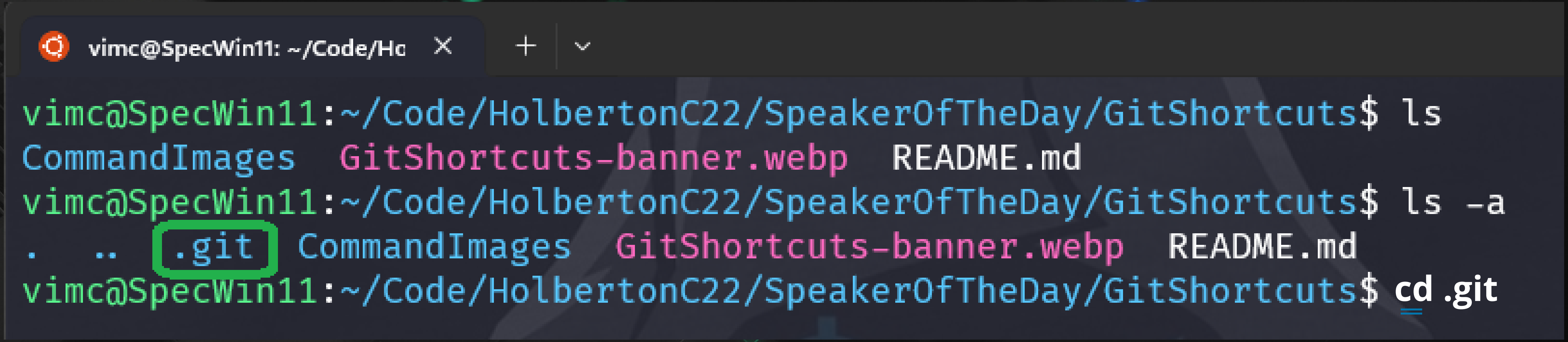
where are our aliases?



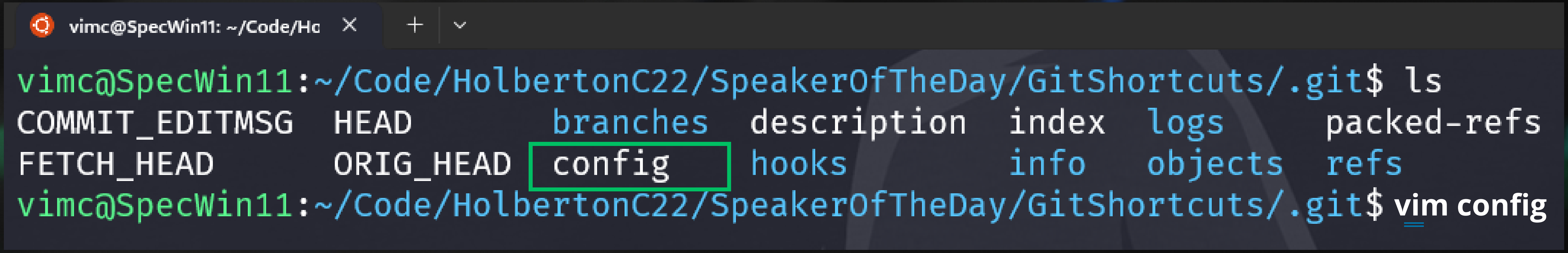
HEAD FETCH_HEAD ORIG_HEAD MERGE_HEAD

index COMMIT_EDITMSG config description

the .git folder

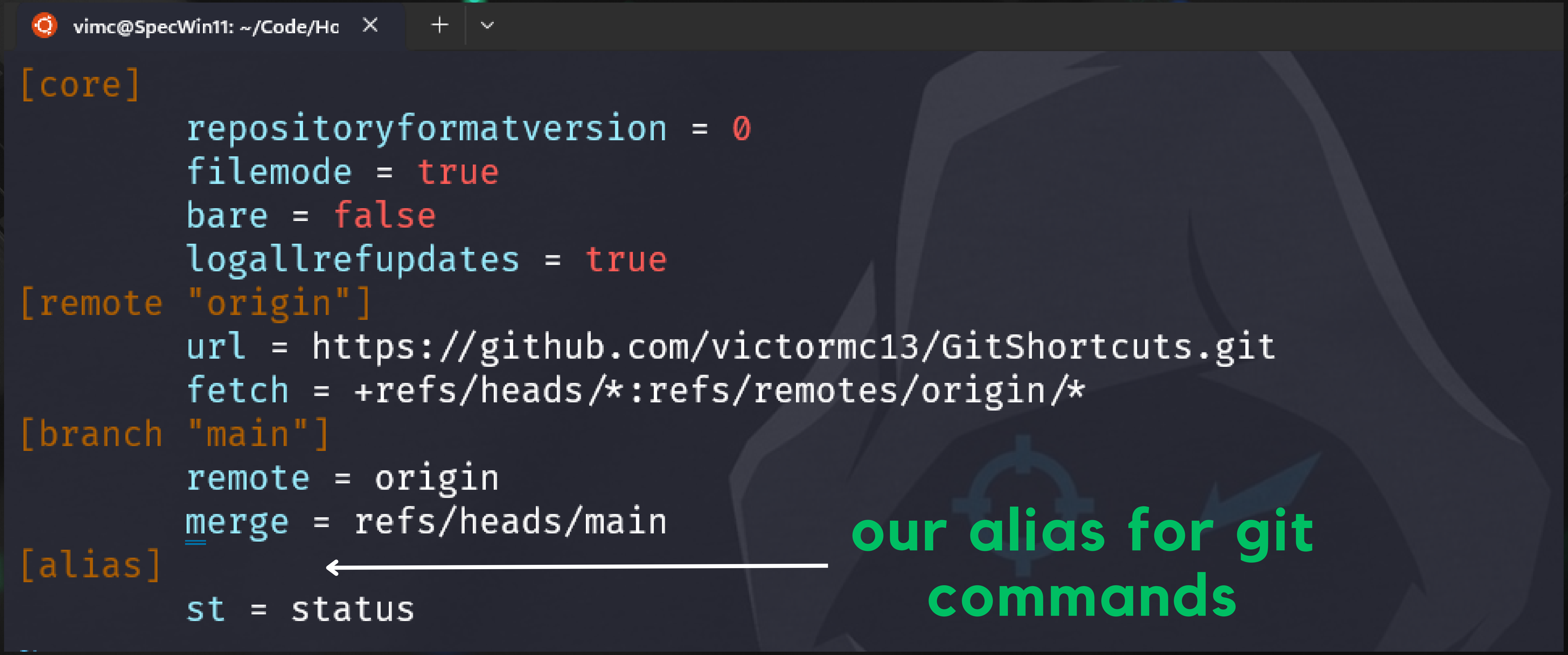


```
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts$ ls
CommandImages  GitShortcuts-banner.webp  README.md
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts$ ls -a
.  ..  .git  CommandImages  GitShortcuts-banner.webp  README.md
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts$ cd .git
```



```
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts/.git$ ls
COMMIT_EDITMSG  HEAD       branches  description  index  logs     packed-refs
FETCH_HEAD      ORIG_HEAD  config    hooks        info   objects  refs
vimc@SpecWin11:~/Code/HolbertonC22/SpeakerOfTheDay/GitShortcuts/.git$ vim config
```

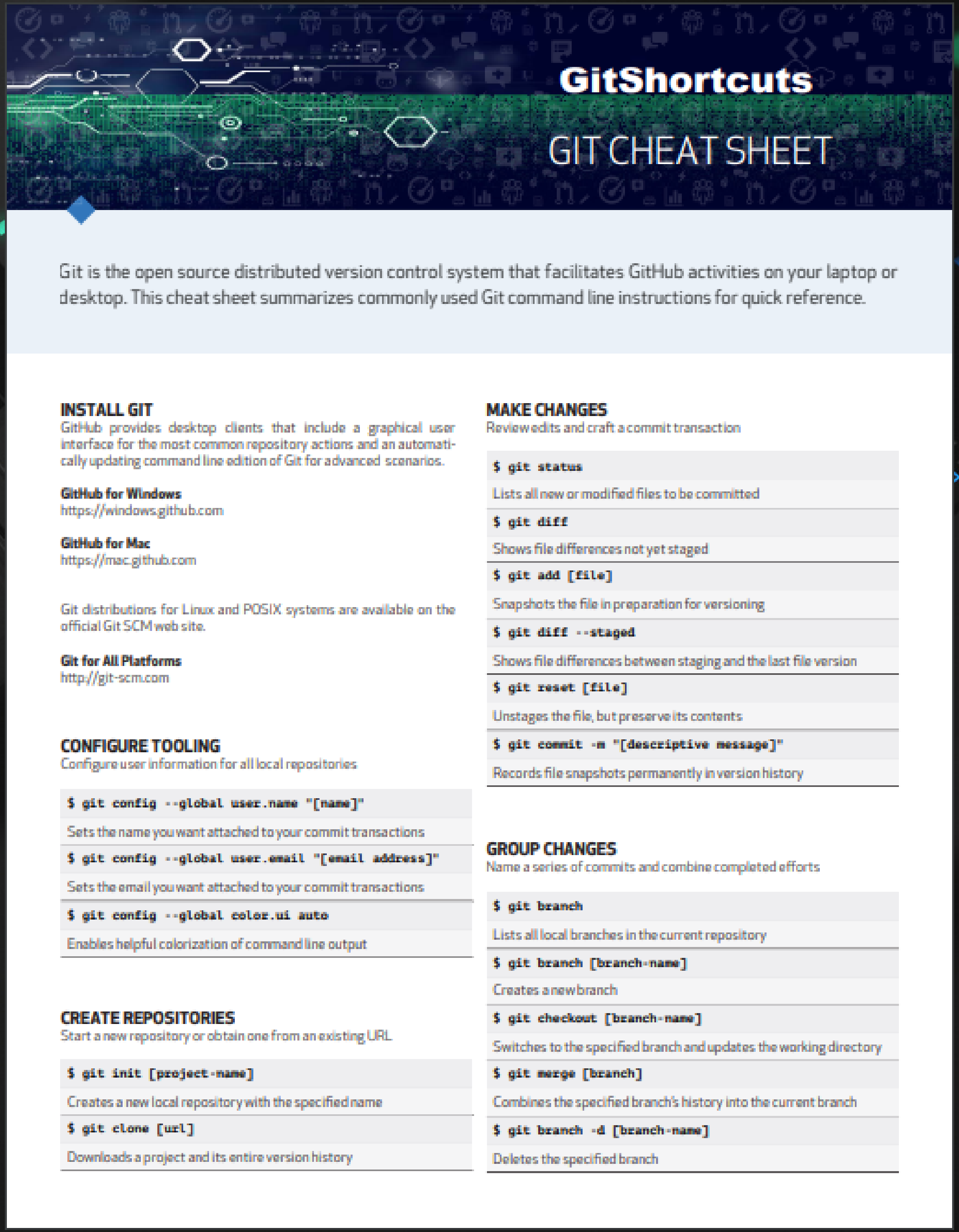
config file



```
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = https://github.com/victormc13/GitShortcuts.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main
[alias]
        st = status
```

our alias for git commands

https://github.com/victormc13/GitShortcuts



GitShortcuts

GIT CHEAT SHEET

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

INSTALL GIT

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Git for All Platforms
http://git-scm.com

CONFIGURE TOOLING

Configure user information for all local repositories

`$ git config --global user.name "[name]"`	Sets the name you want attached to your commit transactions
`$ git config --global user.email "[email address]"`	Sets the email you want attached to your commit transactions
`$ git config --global color.ui auto`	Enables helpful colorization of command line output

CREATE REPOSITORIES

Start a new repository or obtain one from an existing URL

`$ git init [project-name]`	Creates a new local repository with the specified name
`$ git clone [url]`	Downloads a project and its entire version history

MAKE CHANGES

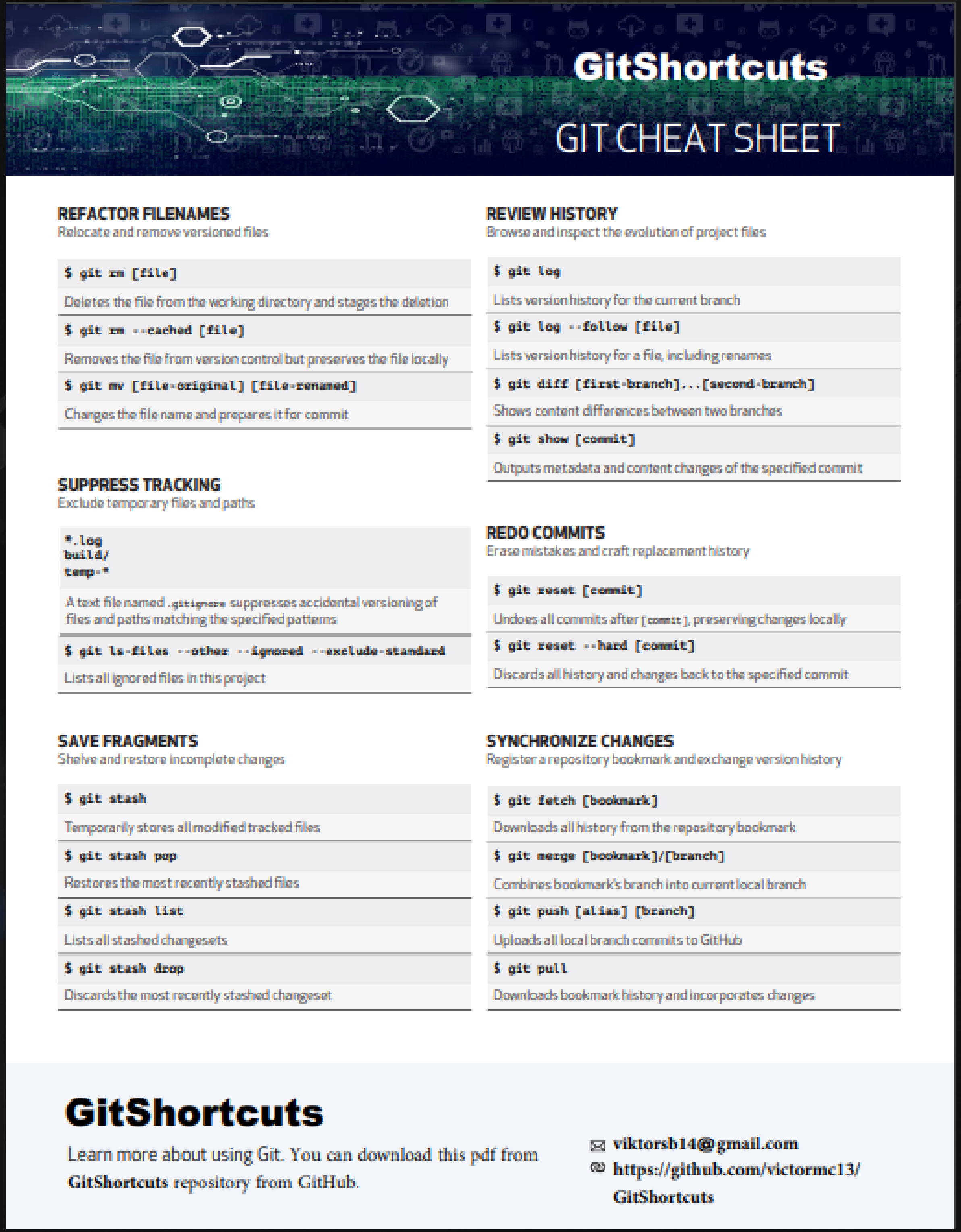
Review edits and craft a commit transaction

`$ git status`	Lists all new or modified files to be committed
`$ git diff`	Shows file differences not yet staged
`$ git add [file]`	Snapshots the file in preparation for versioning
`$ git diff --staged`	Shows file differences between staging and the last file version
`$ git reset [file]`	Unstages the file, but preserve its contents
`$ git commit -m "[descriptive message]"`	Records file snapshots permanently in version history

GROUP CHANGES

Name a series of commits and combine completed efforts

`$ git branch`	Lists all local branches in the current repository
`$ git branch [branch-name]`	Creates a new branch
`$ git checkout [branch-name]`	Switches to the specified branch and updates the working directory
`$ git merge [branch]`	Combines the specified branch's history into the current branch
`$ git branch -d [branch-name]`	Deletes the specified branch



GitShortcuts

GIT CHEAT SHEET

REFACTOR FILENAMES

Relocate and remove versioned files

`$ git rm [file]`	Deletes the file from the working directory and stages the deletion
`$ git rm --cached [file]`	Removes the file from version control but preserves the file locally
`$ git mv [file-original] [file-renamed]`	Changes the file name and prepares it for commit

SUPPRESS TRACKING

Exclude temporary files and paths

```
*.log
build/
temp-*
```

A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

`$ git ls-files --other --ignored --exclude-standard`	Lists all ignored files in this project

SAVE FRAGMENTS

Shelve and restore incomplete changes

`$ git stash`	Temporarily stores all modified tracked files
`$ git stash pop`	Restores the most recently stashed files
`$ git stash list`	Lists all stashed changesets
`$ git stash drop`	Discards the most recently stashed changeset

REVIEW HISTORY

Browse and inspect the evolution of project files

`$ git log`	Lists version history for the current branch
`$ git log --follow [file]`	Lists version history for a file, including renames
`$ git diff [first-branch]...[second-branch]`	Shows content differences between two branches
`$ git show [commit]`	Outputs metadata and content changes of the specified commit

REDO COMMITS

Erase mistakes and craft replacement history

`$ git reset [commit]`	Undoes all commits after `[commit]`, preserving changes locally
`$ git reset --hard [commit]`	Discards all history and changes back to the specified commit

SYNCHRONIZE CHANGES

Register a repository bookmark and exchange version history

`$ git fetch [bookmark]`	Downloads all history from the repository bookmark
`$ git merge [bookmark]/[branch]`	Combines bookmark's branch into current local branch
`$ git push [alias] [branch]`	Uploads all local branch commits to GitHub
`$ git pull`	Downloads bookmark history and incorporates changes

GitShortcuts

Learn more about using Git. You can download this pdf from **GitShortcuts** repository from GitHub.

✉ viktorsb14@gmail.com
🔗 https://github.com/victormc13/GitShortcuts

victormc13

THANK YOU