



# Git and GitHub Workshop

Ali Bagheri | March 2022

# Contents

## 01 Intro to git

- Why do we use git
- Alternatives
- How to install git
- GUI Git

## 03 git branch and merge

- What is a branch?
- Add a branch
- Move between branches
- Why do we merge
- Easy way
- Best way

## 05 Extra

## 02 git basics

- Setting up a repo
- git add
- git commit
- git log

## 04 GitHub

- Why?
- How
- Setting it up
- Pull Push

# Intro to git





# git

## git

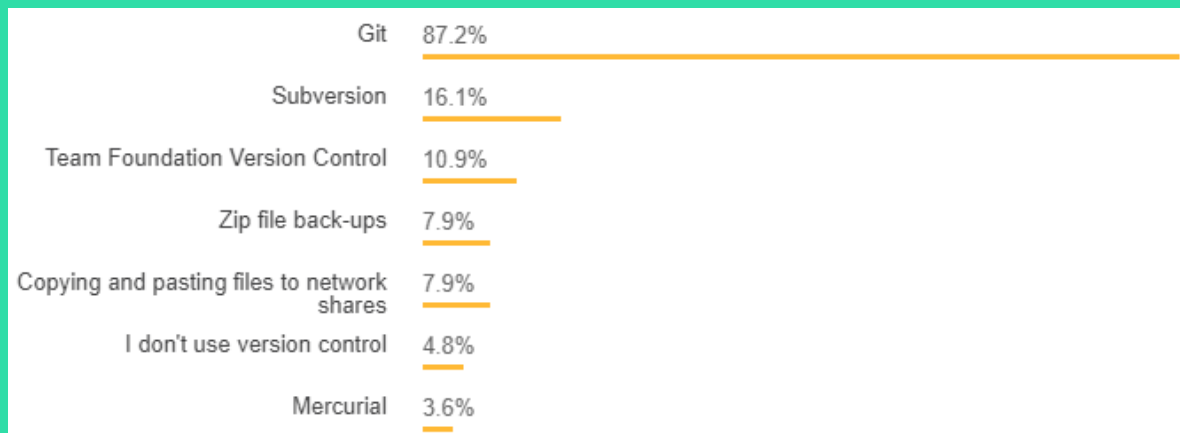
The world's most popular  
version control system  
( VSC )

## Version Control

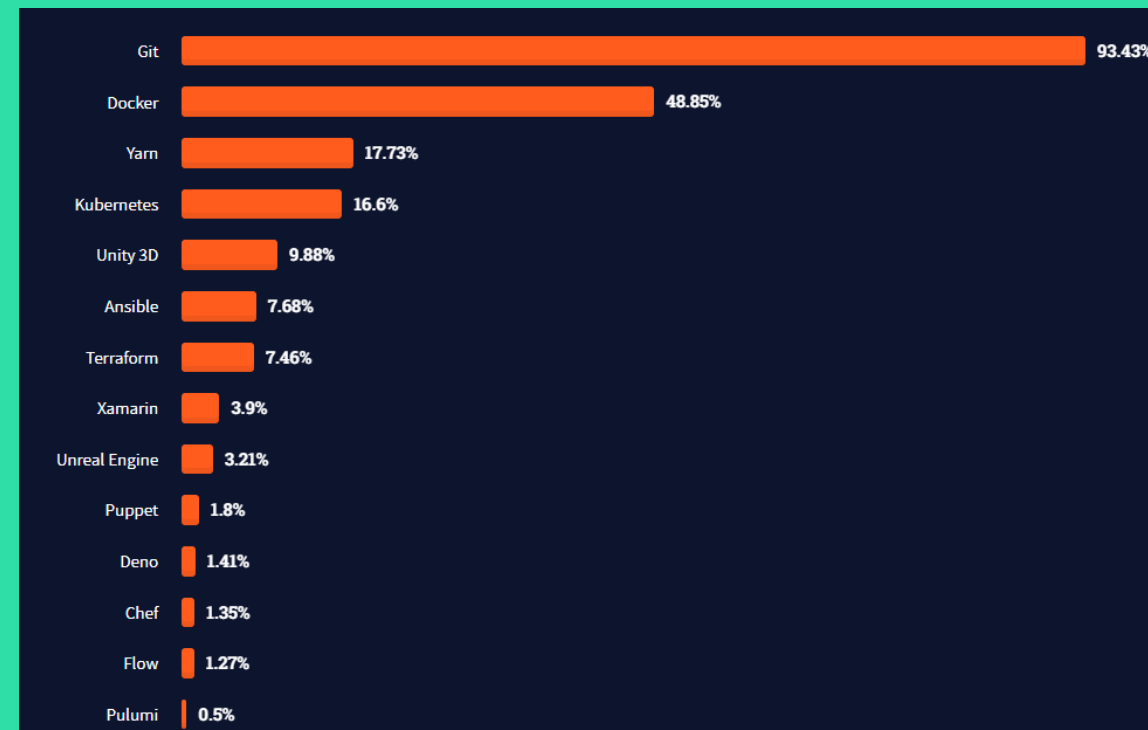
Software that tracks and  
manages changes

## Other Options

Subversion  
CVS  
Mercurial



Stack overflow survey 2018  
\* Last year including version controls



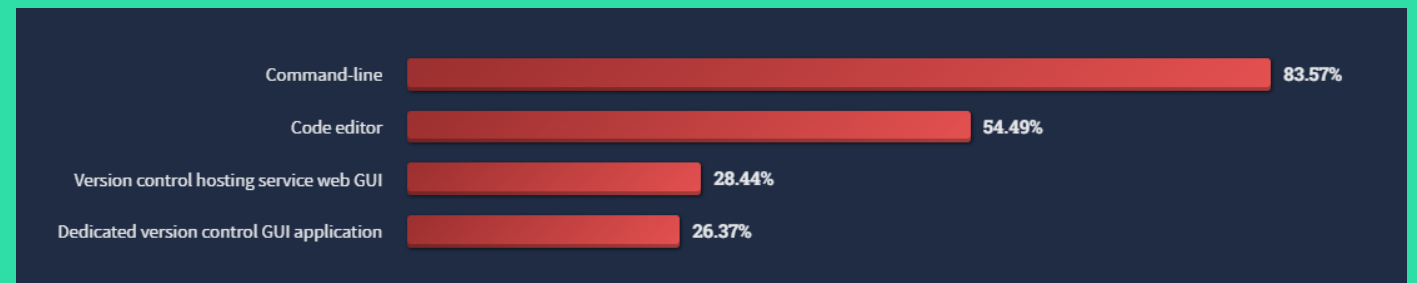
Stack overflow survey 2021 :  
Over 90% of respondents use Git, suggesting that it is a  
fundamental tool to being a developer.

# Stack overflow survey 2022

## Version control systems

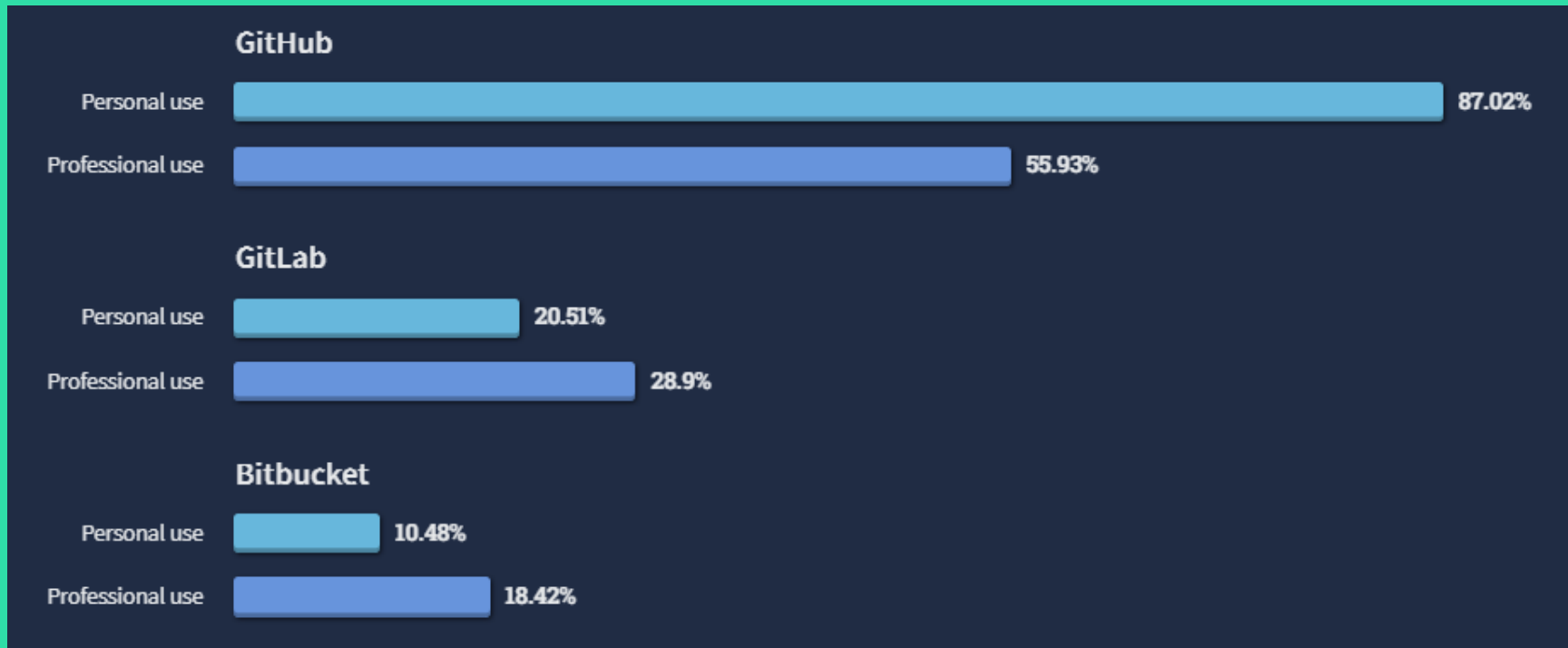


## Interacting with version control systems



# Stack overflow survey 2022

## Version control platforms





# git Usage

- Track changes across multiple files
- Compare versions of a project
- "Time travel" back to old versions
- Revert to a previous version
- Collaborate and share changes
- Combine changes





# Installation


## Windows

- Download from <https://git-scm.com> and continue installation

## Linux

- In Debian-based distros use “sudo apt install git”
- In Fedora (or any RPM-based distros) use “sudo dnf install git-all”

## MacOS

- Check for existence with “git --version”
  - If was not installed download it from <https://git-scm.com> and continu
- 

# Configuring Identity

In Terminal or Command Prompt ( CMD ) type these commands :

```
> git config --global user.name "Ali Bagehri"
```

```
> git config --global user.email "AliBagehri@mail.com"
```

# Git GUI



GitKraken

Syntax Highlighting,  
Drag and drop  
functionality



Sourcetree

Free



GitHub Desktop

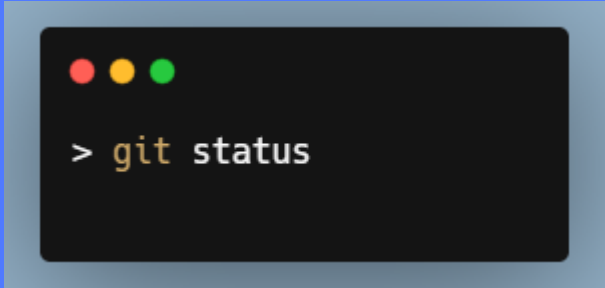
Developed and  
maintained by github

# git basics

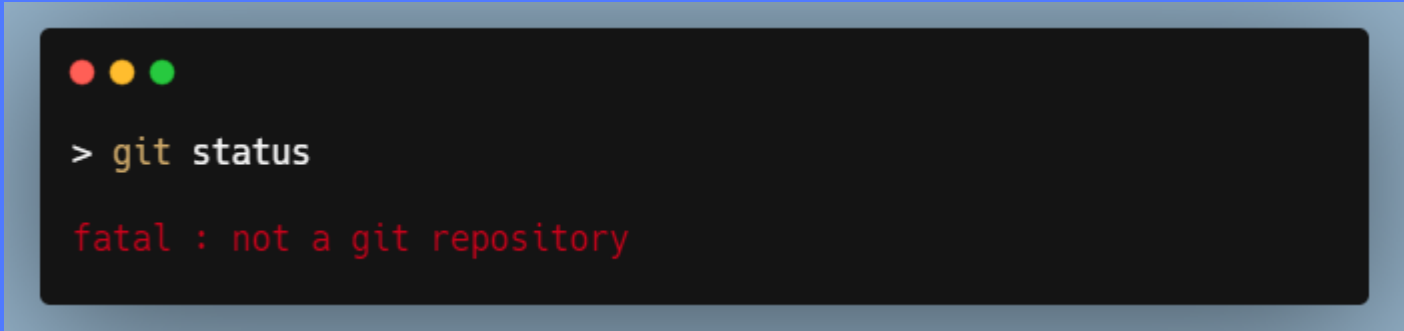


# First command

“git status” gives information on the current status of a git repository and its contents.



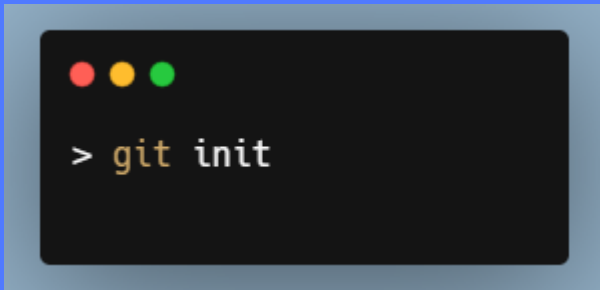
```
> git status
```



```
> git status  
fatal : not a git repository
```

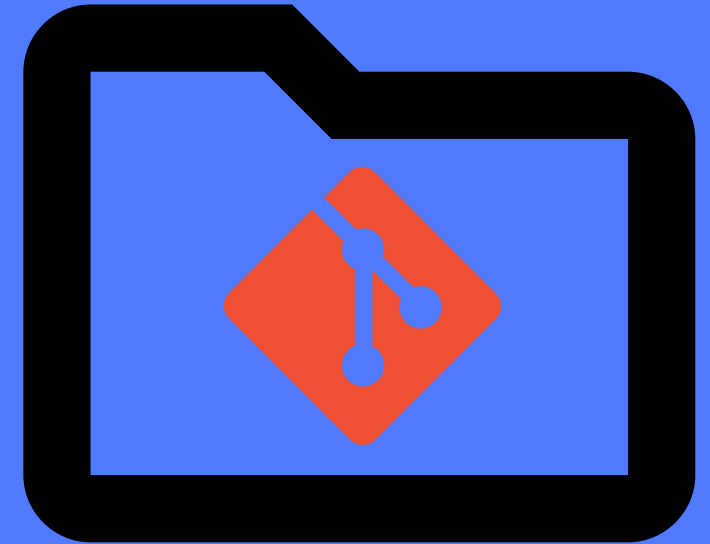
# Repository

A git "Repo" is a workspace which tracks and manages files within a folder.



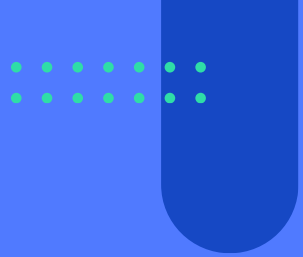
**Warning:**  
Do not init a repo inside of a repo!

Always check existence of a repo before initializing one with  
"git status"





# git workflow



## Working directory

modified index.html

created style.css

deleted html/about.html

modified html/team.html

modified script.js

## Staging area

## Repository





# git add



Adds specified files to staging area

```
> git add <file1> <file2> <folder1>
```

```
> git add .
```

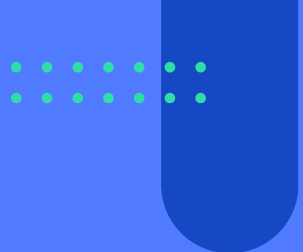
Adds all changes to staging area







```
> git add index.html
```



## Working directory

- created style.css
- deleted html/about.html
- modified html/team.html
- modified script.js

## Staging area

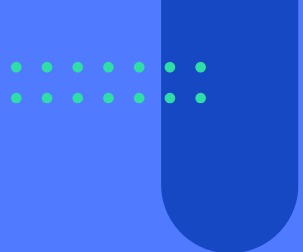
- modified index.html

## Repository





```
> git add style.css script.js
```



## Working directory

- deleted html/about.html
- modified html/team.html

## Staging area

- modified index.html
- created style.css
- modified script.js

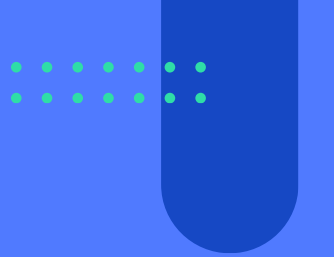
## Repository

- 





# git commit



Commits all staged changes.

```
• • •  
> git commit
```

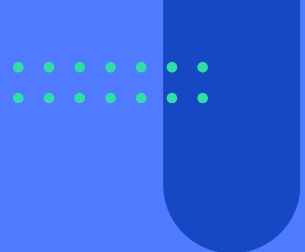
```
• • •  
> git commit -m "commit message"
```

-m specifies commit message





```
> git commit -m "updating main files"
```



## Working directory

deleted html/about.html

modified html/team.html

## Staging area

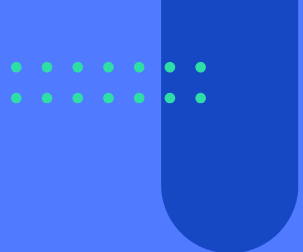
## Repository

adding main files





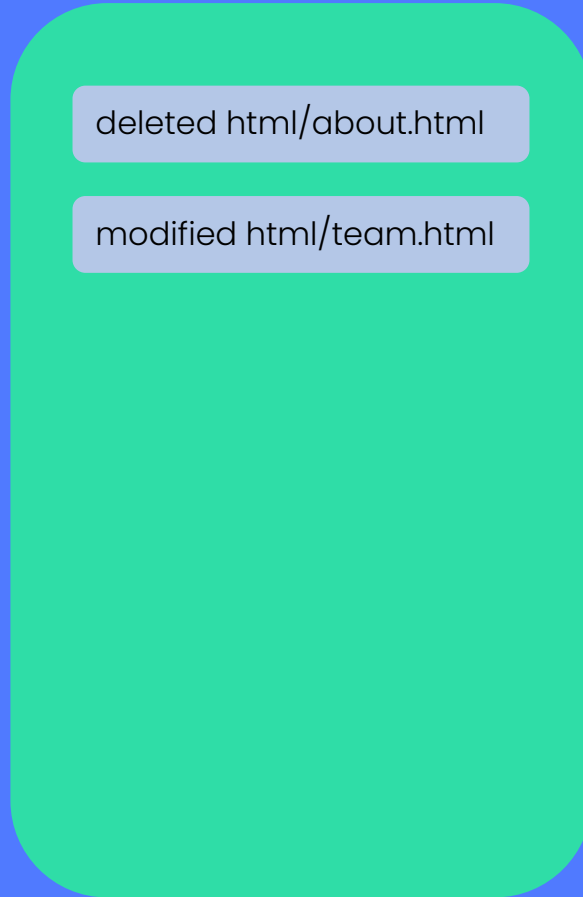
```
> git add html
```



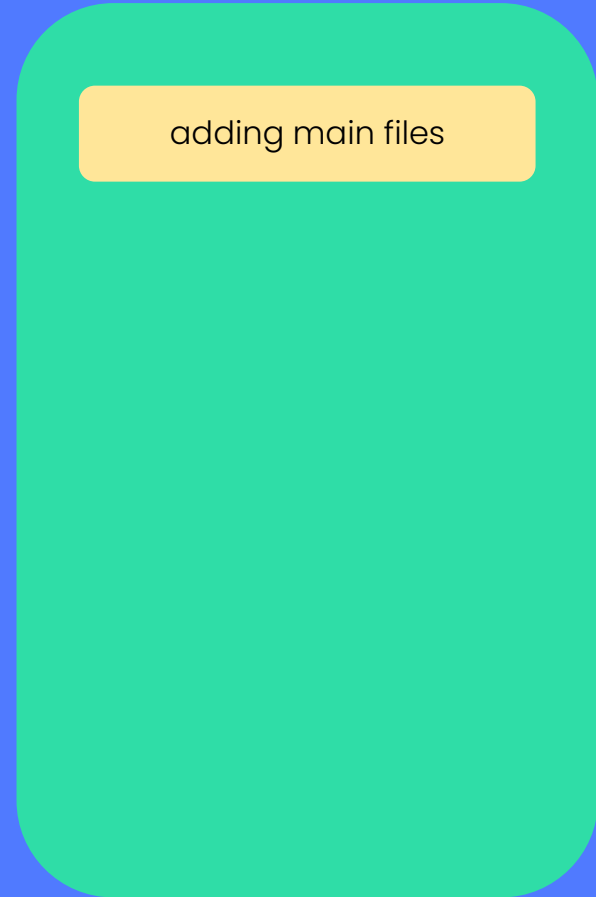
## Working directory



## Staging area

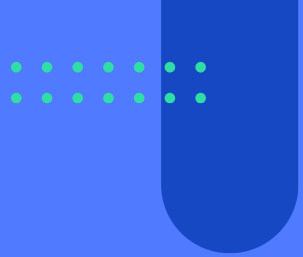


## Repository





```
> git commit -m "updating html folder  
> about.html and team.html"
```



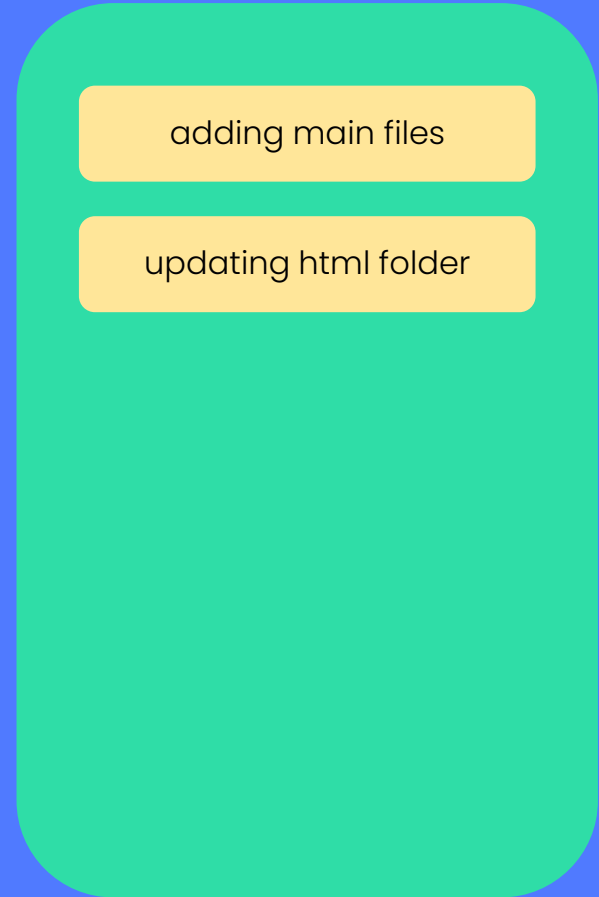
## Working directory



## Staging area



## Repository





# git log

Logs all commit

```
> git log
commit e66256e3556f8ac38765b85a116304f98f580cfd (HEAD -> master)
Author: Ali Bagheri <alibagheri@gmail.com>
Date:   Mon Mar 14 16:23:01 2022 +0330

    updating html folder
    about.html and team.html

commit 120ed1b549f2bbb3900cd9a41551cbb8c47f70
Author: Ali Bagheri <alibagheri@gmail.com>
Date:   Mon Mar 14 16:21:55 2022 +0330

    updating main files
```



# git log



```
> git log --oneline
```

```
e66256e (HEAD -> master) updating html folder about.html and team.html
```

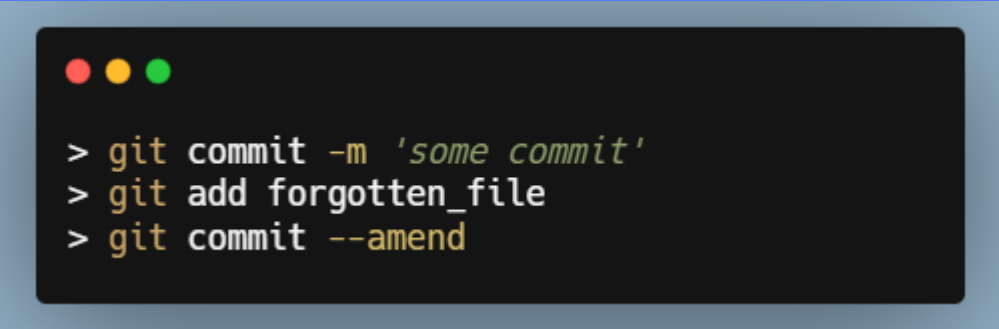
```
120ed1b updating main files
```







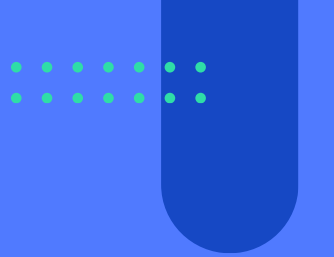
# Forgotten changes



```
> git commit -m 'some commit'
> git add forgotten_file
> git commit --amend
```



# Some Basic Guidelines



- Commit early and often
- Make commits atomic (group similar changes together, don't commit a million things at once)
- Write meaningful but short commit messages





# Commits text

Git Docs:

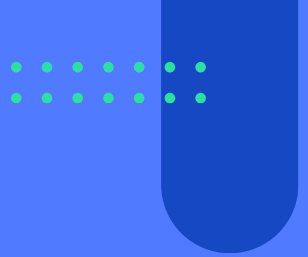
"Describe your changes in imperative mood, e.g. "make xyzzy do frotz" instead of "[This patch] makes xyzzy do frotz" or "I changed xyzzy to do frotz", as if you are giving orders to the codebase to change its behavior."

Note: It's not mandatory but only a suggestion





# Atomic Commits



When possible, a commit should encompass a single feature, change, or fix. In other words, try to keep each commit focused on a single thing.

This makes it much easier to undo or rollback changes later on. It also makes your code or project easier to review.





# Ignoring Files

We can tell Git which files and directories to ignore in a given repository, using a .gitignore file. This is useful for files you know you NEVER want to commit, including:

- Secrets, API keys, credentials, etc.
- Operating System files (.DS\_Store on Mac)
- Log files
- Dependencies & packages

Create a file called .gitignore in the root of a repository. Inside the file, we can write patterns to tell Git which files & folders to ignore.

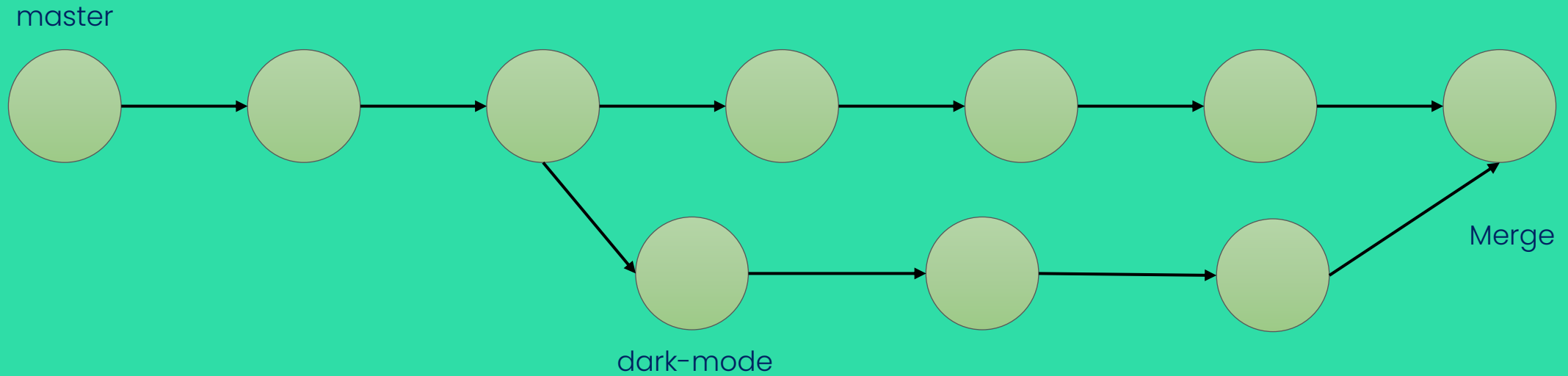
```
node_modules
configs
.idea
logs
package-lock.json
.env*
```



# git branch and merge



# Branch



Some people say master is “source of truth”.

# Branch

```
> git log
commit e66256e3556f8ac38765b85a116304f98f580cfd (HEAD -> master)
Author: Ali Bagheri <alibagheri@gmail.com>
Date:   Mon Mar 14 16:23:01 2022 +0330

    updating html folder
    about.html and team.html

commit 120ed1b549f2bbb3900cd9a41551cbb8c47f70
Author: Ali Bagheri <alibagheri@gmail.com>
Date:   Mon Mar 14 16:21:55 2022 +0330

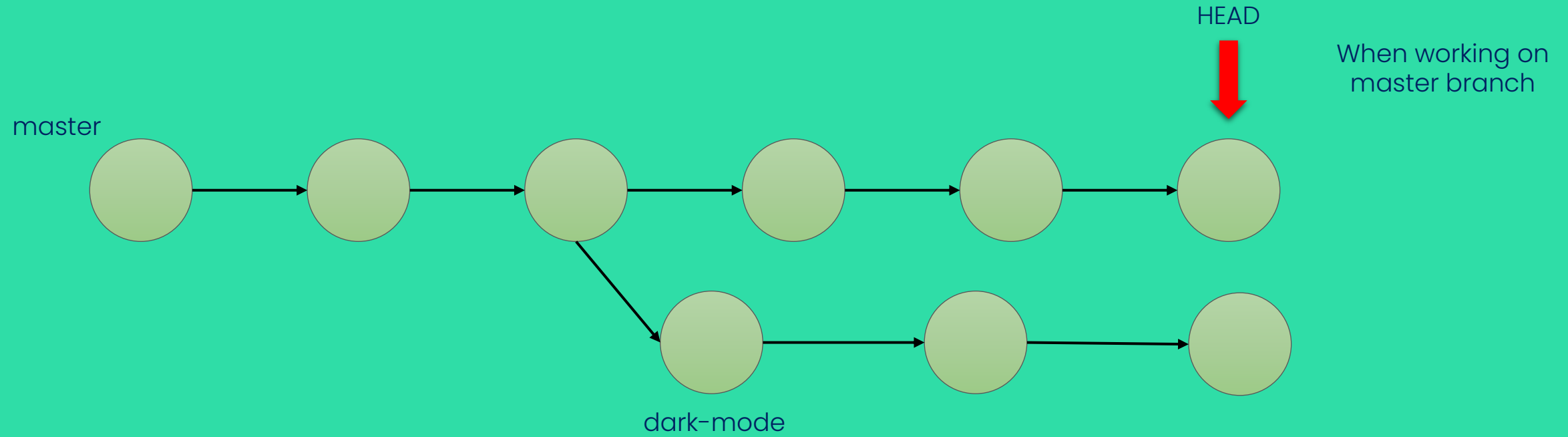
    updating main files
```

HEAD is our current location

master is a branch

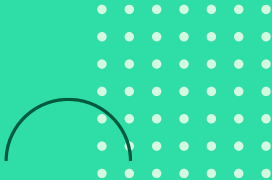


# Branch

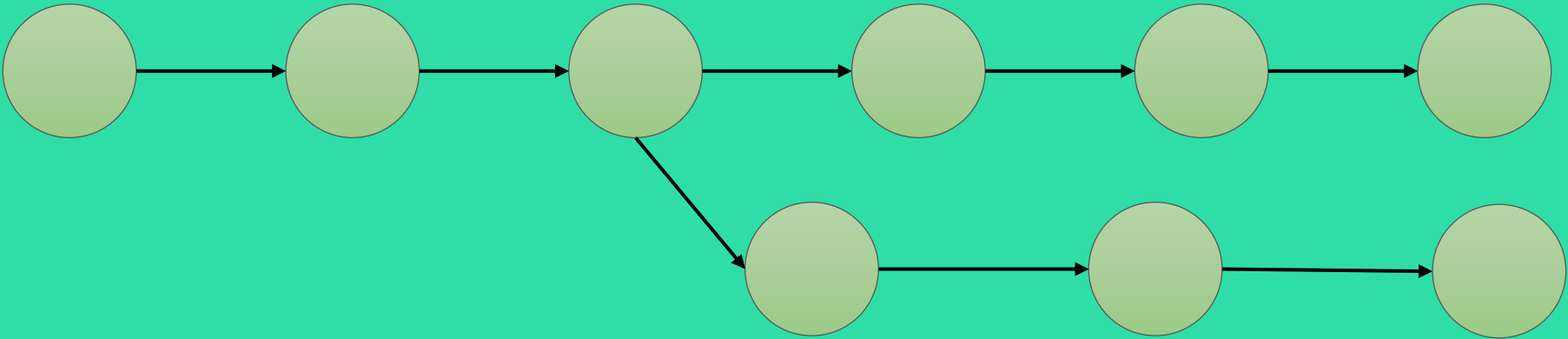




# Branch



master



dark-mode



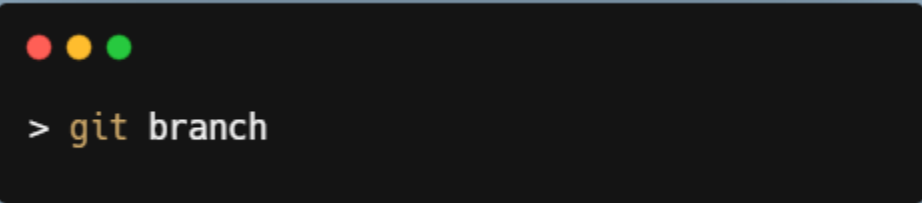
HEAD

When working on  
dark-mode branch



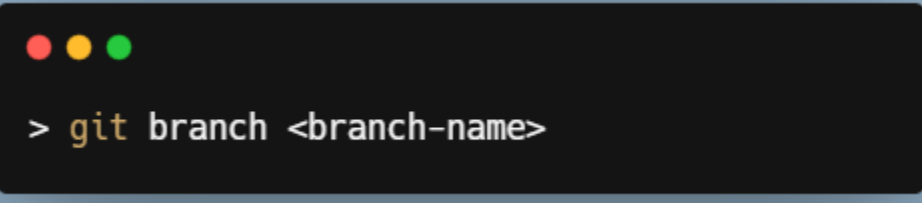


# Branch



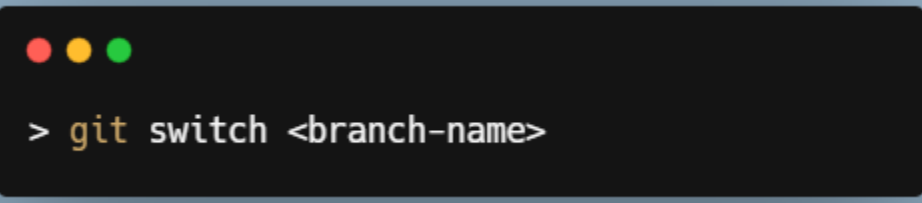
```
> git branch
```

To see all and current branch



```
> git branch <branch-name>
```

Creates a new branch



```
> git switch <branch-name>
```

Switches to another branch



# Branch



```
> git checkout <branch-name>
```

Another way of switching branch



```
> git switch -c <branch-name>
```

Creates a branch and switch to it

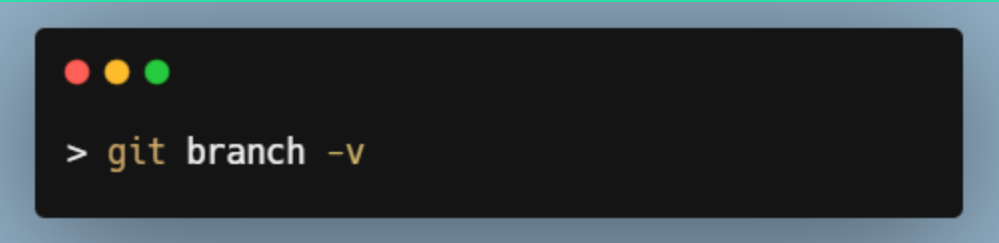


```
> git checkout -b <branch-name>
```

Another way of creating and switching

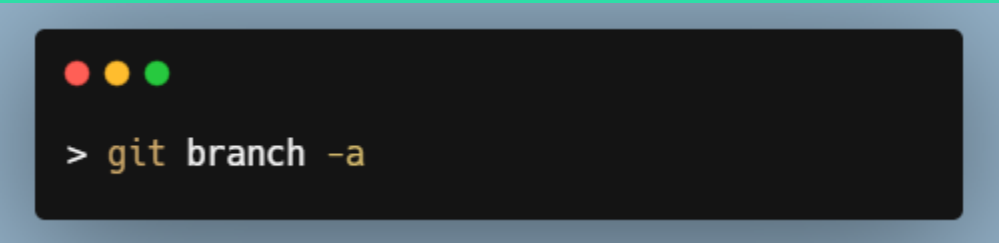


# Branch



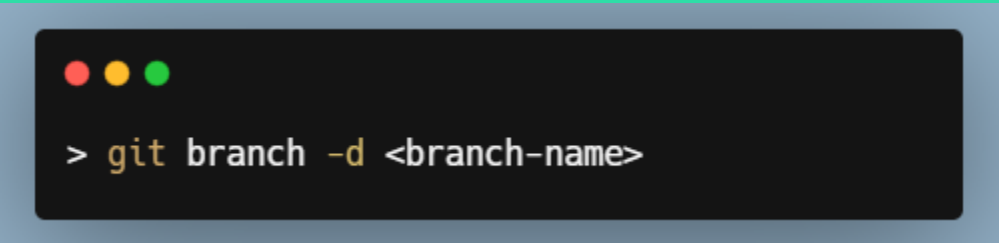
```
> git branch -v
```

More info about branches



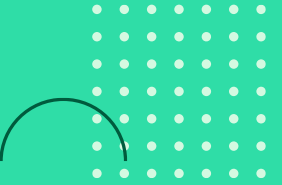
```
> git branch -a
```

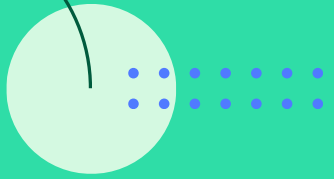
Listing all branches



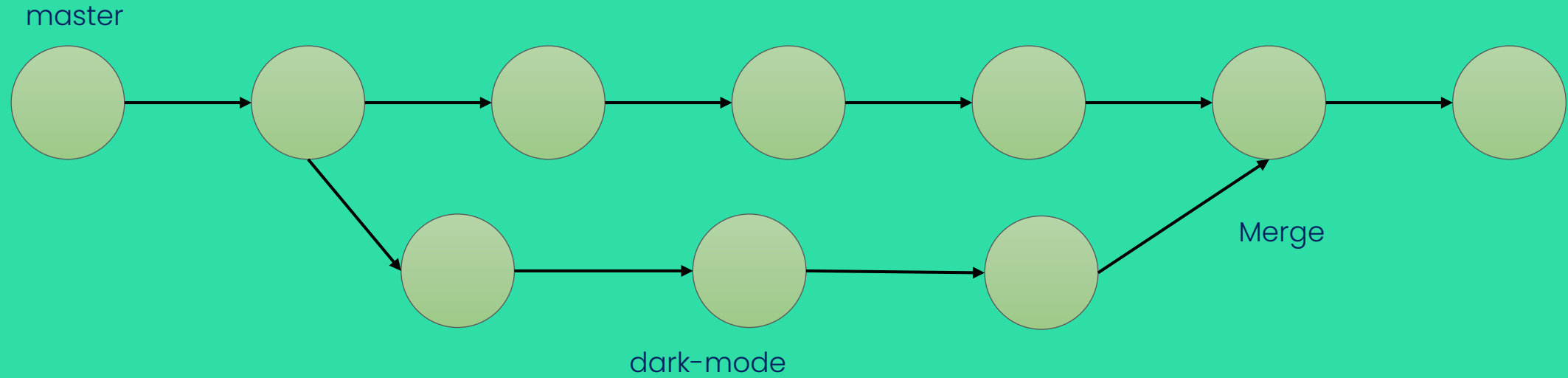
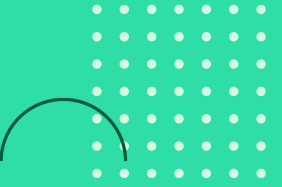
```
> git branch -d <branch-name>
```

Delete a branch





# Merge

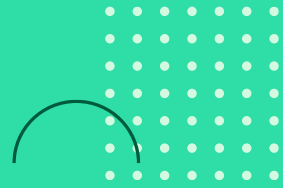


All changes will be added to destination branch.





# Merge



```
> git switch master  
> git merge dark-mode
```

Merge dark-mode to master





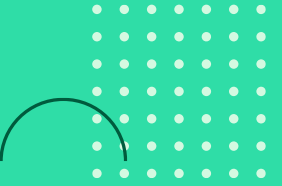
# Merge



```
> CONFLICT (content): Merge conflict in blah.txt
```

```
Automatic merge failed; fix conflicts and then commit the result
```

Some times new commits in destination branch, will prevent auto merge !





# Merge

```
<<<<<< HEAD
```

```
color: #FFFFFF;  
border: 1px solid red;
```

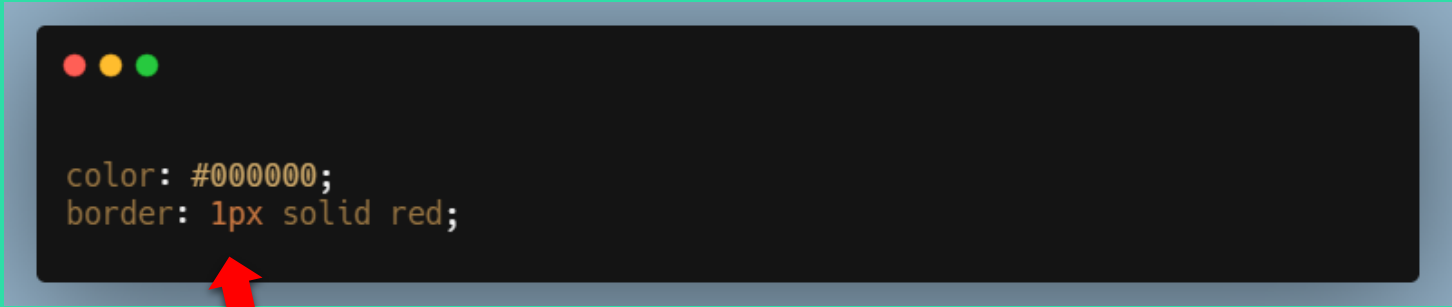
```
=====
```

```
color: #000000;
```

```
>>>>>> dark-mode
```

Section of code that was  
changed in both branch

# Merge



```
color: #000000;  
border: 1px solid red;
```

To fix this just delete markers, and resolve conflicts.  
Then commit it.



# GitHub

# What is GitHub?



Github is a hosting platform for git repositories. You can put your own Git repos on Github and access them from anywhere and share them with people around the world.

Beyond hosting repos, Github also provides additional collaboration features that are not native to Git (but are super useful). Basically, Github helps people share and collaborate on repos.



Git is the version control software that runs locally on your machine. You don't need to register for an account. You don't need the internet to use it. You can use Git without ever touching Github.



Github is a service that hosts Git repositories in the cloud and makes it easier to collaborate with other people. You do need to sign up for an account to use Github. It's an online place to share work that is done using Git.

# Other Options



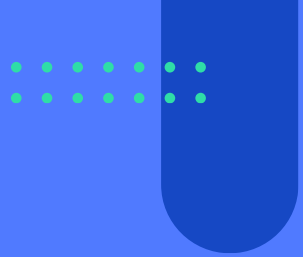
GitLab



BitBucket



Your own Git  
Server



Founded in 2008, Github is now the world's largest host of source code. In early 2020, Github reported having over 40 million users and over 190 million repositories on the platform.

Other benefits of Github:

- It's Free!
- Collaboration
- Open Source Projects
- Exposure
- Stay Up To Date



# Getting a repo from Github

```
> git clone <url>
```

It will clone a repo from a git server to your current directory.

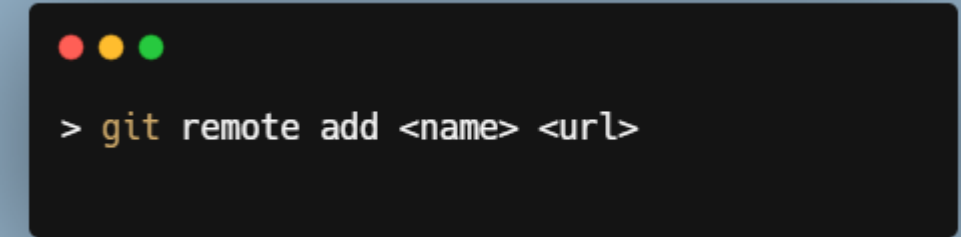
Alert:  
You should setup your remote account first.





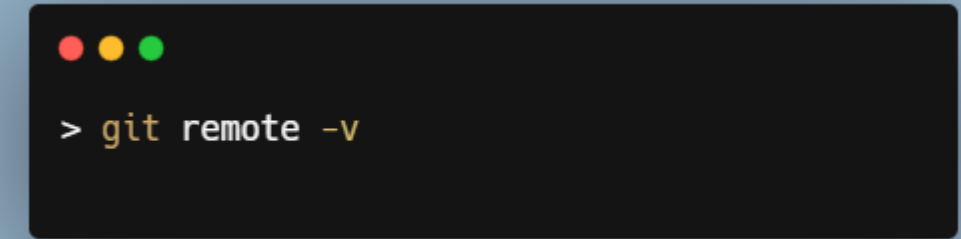
# Connecting a repo to remote server

The default name of remote is usually “origin”.



```
> git remote add <name> <url>
```

You can view remotes using this command.



```
> git remote -v
```



# Other remote commands

Renaming a remote

```
> git remote rename <old> <new>
```

Removing a remote

```
> git remote remove <name>
```

# Pushing commits to remote

Pushes all of your commits of named branch to named remote repository

```
> git push <remote> <branch>
```

The -u option allows us to set the upstream of the branch we're pushing. You can think of this as a link connecting our local branch to a branch on Github.

```
> git push -u <remote> <branch>
```

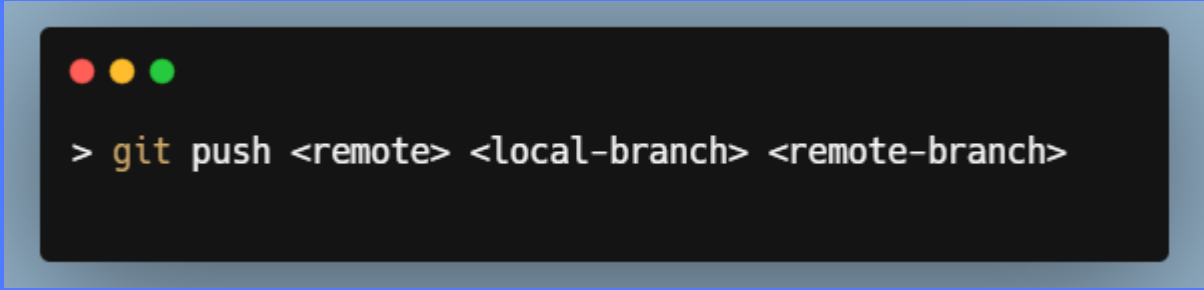
After that on this branch you can use this short form

```
> git push
```



# Push in details

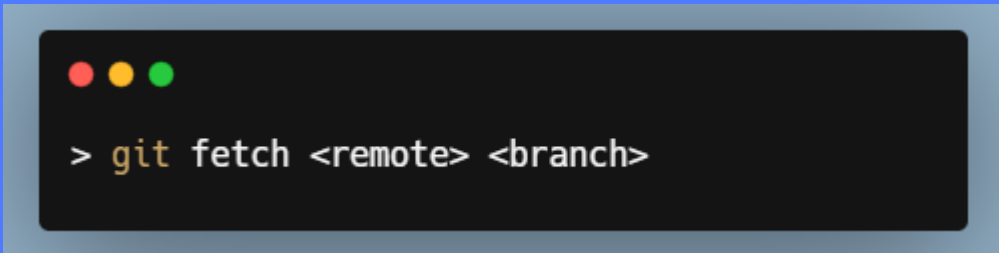
When you push to a branch with different name.



```
> git push <remote> <local-branch> <remote-branch>
```




# Fetch



```
> git fetch <remote> <branch>
```

Fetching allows us to download changes from a remote repository, BUT those changes will not be automatically integrated into our working files.



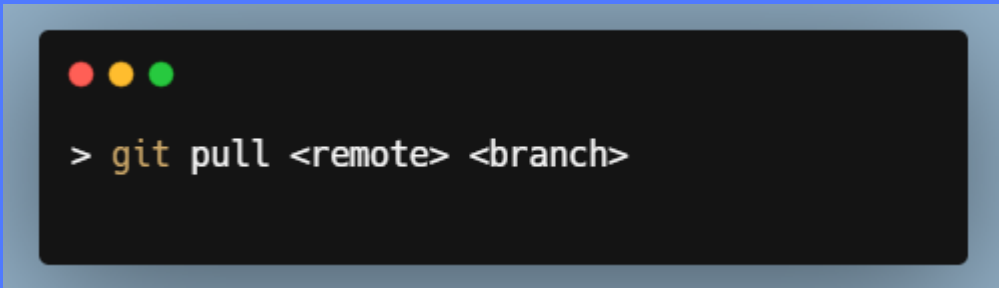


# Pull


git pull is another command we can use to retrieve changes from a remote repository. Unlike fetch, pull actually updates our HEAD branch with whatever changes are retrieved from the remote.


git pull = git fetch + git merge

Alert: pulls can result in conflicts!!

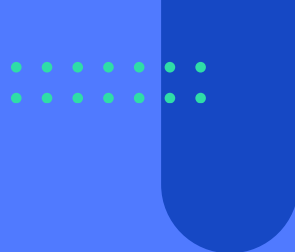


```
> git pull <remote> <branch>
```






In this short form default remote is origin and default branch is your current branch.



```
> git pull
```




## git fetch

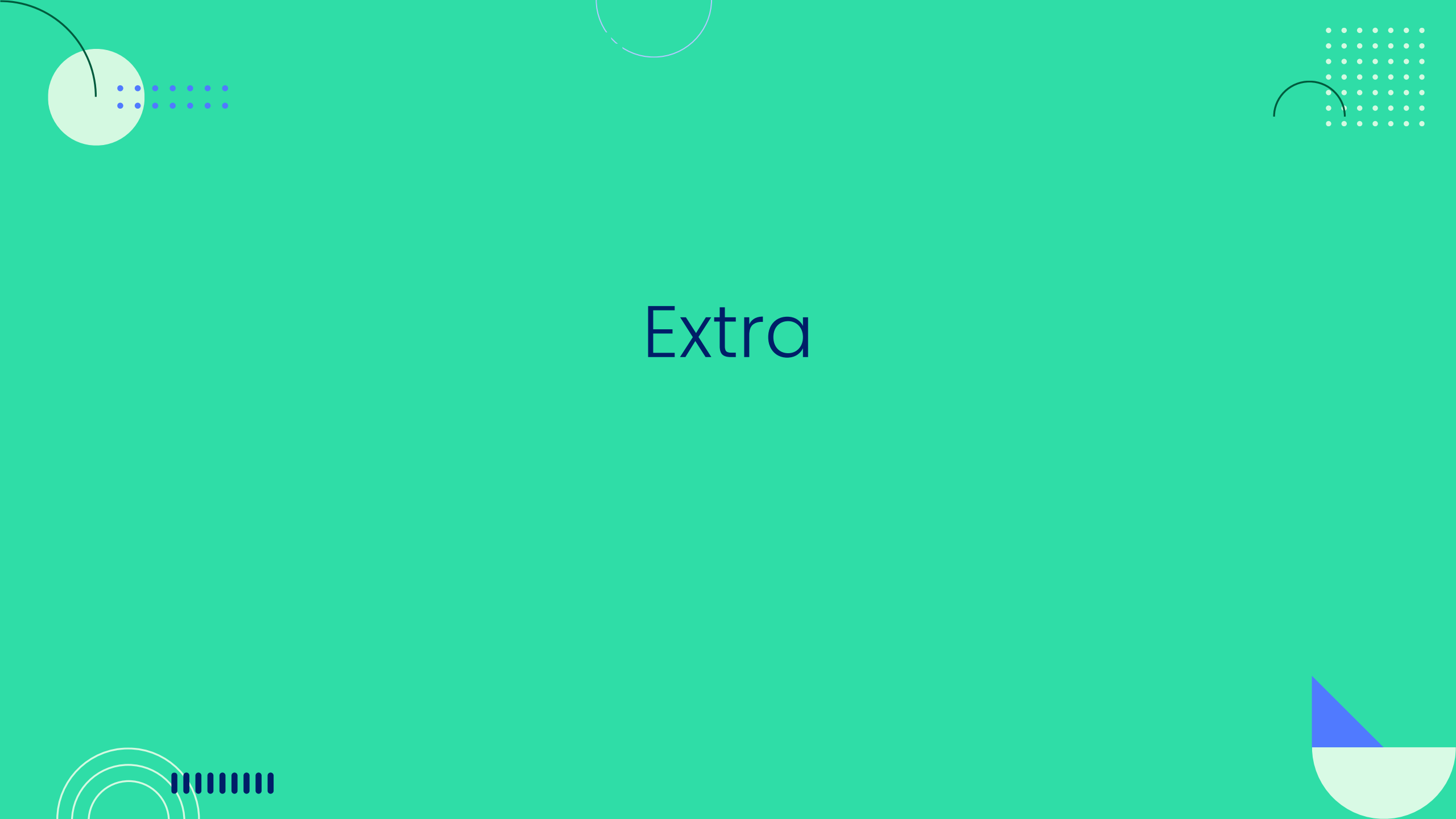
- Gets changes from remote branch(es)
  - Updates the remote-tracking branches with the new changes
  - Does not merge changes onto your current HEAD branch
  - Safe to do at anytime
- 



## git pull

- Gets changes from remote branch(es)
  - Updates the current branch with the new changes, merging them in
  - Can result in merge conflicts
  - Not recommended if you have uncommitted changes!
- 





Extra

- 
- 
- 
- Comparisons via tools
  - Stashing



# Git workflows

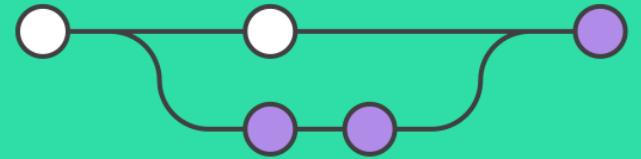
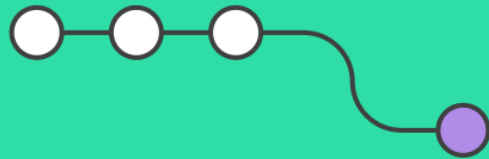
## Workflows with 1 protected branch

- master/main

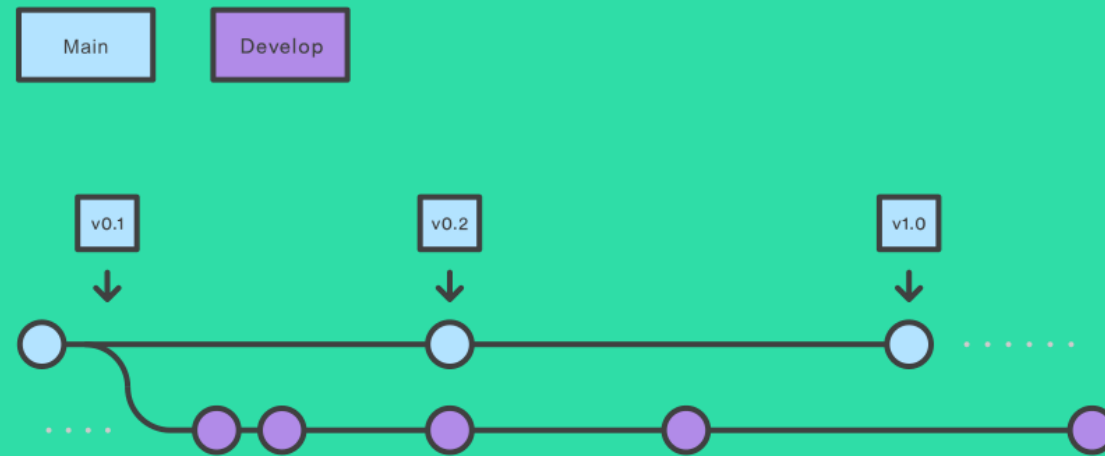
## Workflows with 2 protected branch

- master/main
- develop

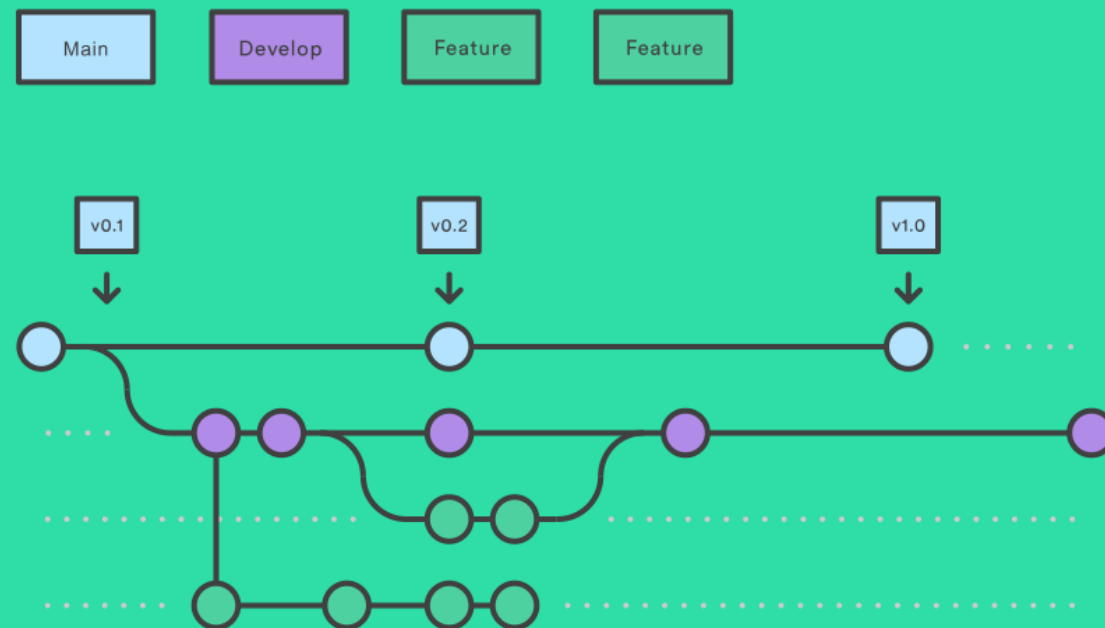
# Feature Branch Workflow



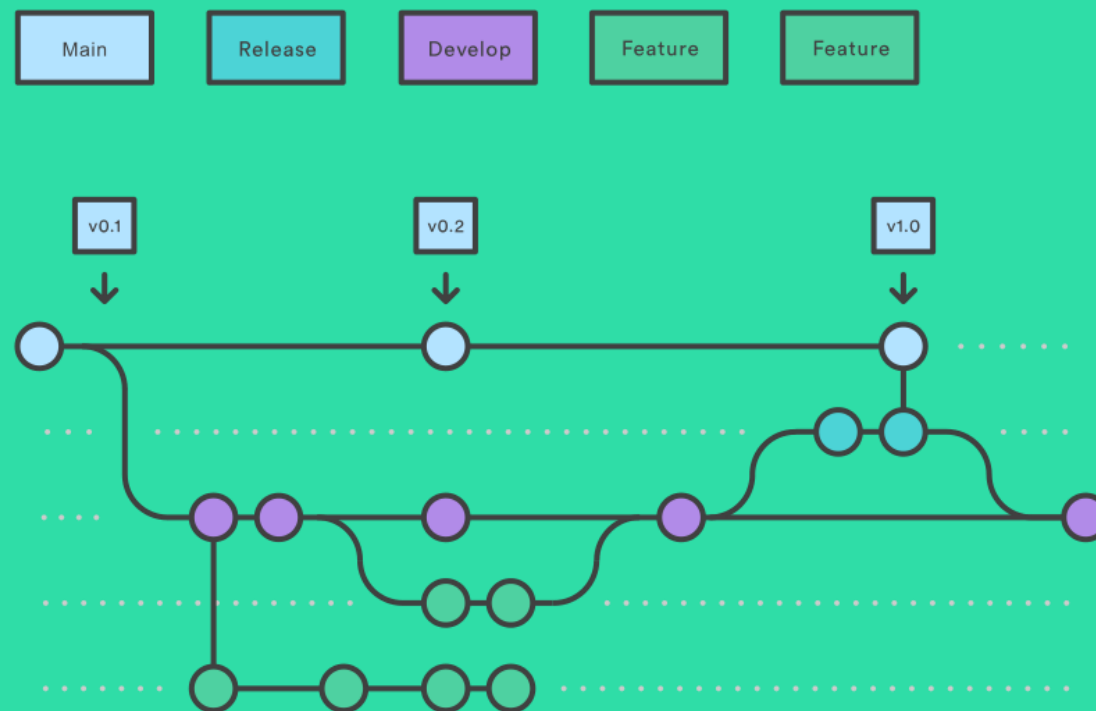
# Gitflow Workflow



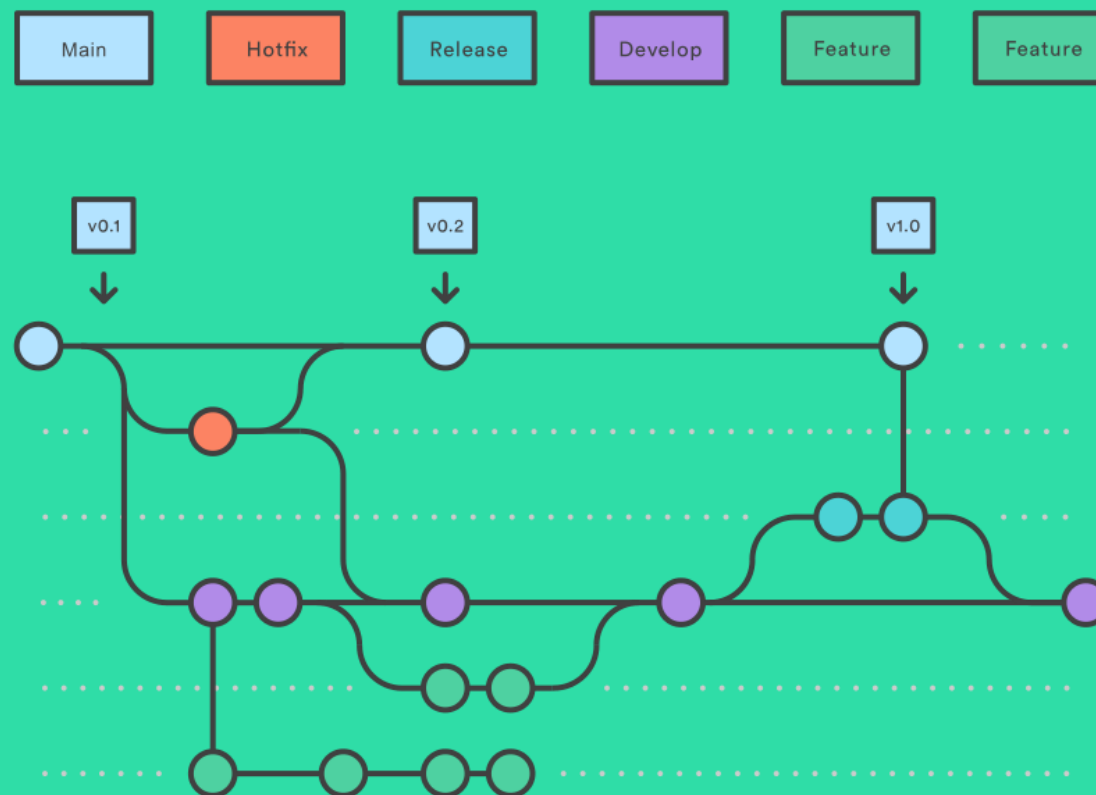
# Gitflow Workflow



# Gitflow Workflow

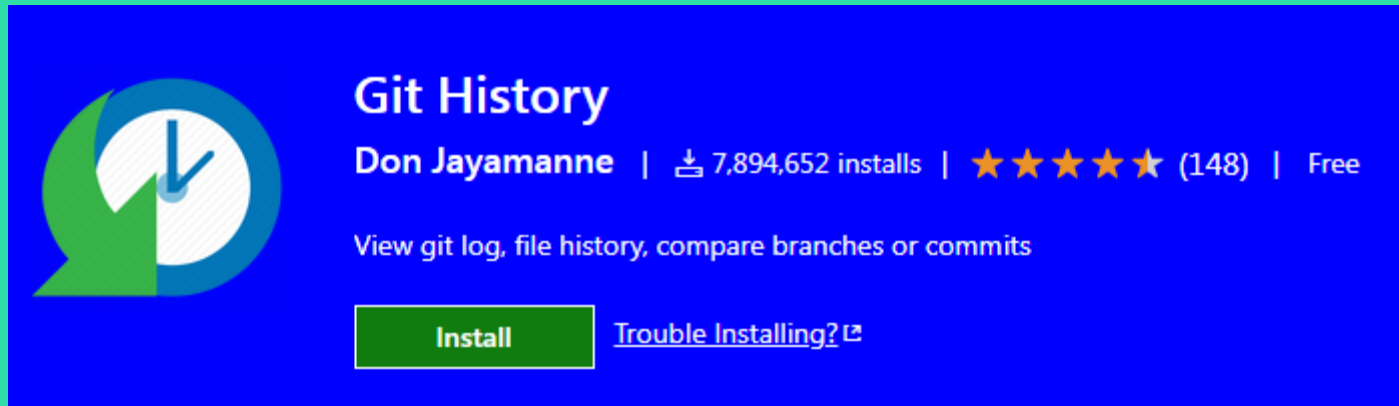


# Gitflow Workflow





# Some VSCode Extensions



The image shows a screenshot of the 'Git History' extension page in the Visual Studio Code marketplace. The card has a blue background. On the left is the extension's icon, which features a green speech bubble and a blue clock face. To the right of the icon, the text 'Git History' is displayed in white. Below the title, the author 'Don Jayamanne' is listed, followed by a download icon and the text '7,894,652 installs'. This is followed by five yellow star icons and the text '(148)'. To the right of the stars, the word 'Free' is shown. Below this information, a description reads 'View git log, file history, compare branches or commits'. At the bottom of the card, there is a green 'Install' button and a blue link that says 'Trouble Installing?' with an external link icon.

**Git History**  
Don Jayamanne | 📄 7,894,652 installs | ★★★★★ (148) | Free

View git log, file history, compare branches or commits

[Install](#) [Trouble Installing?](#)

# Some VSCode Extensions



## GitLens — Git supercharged

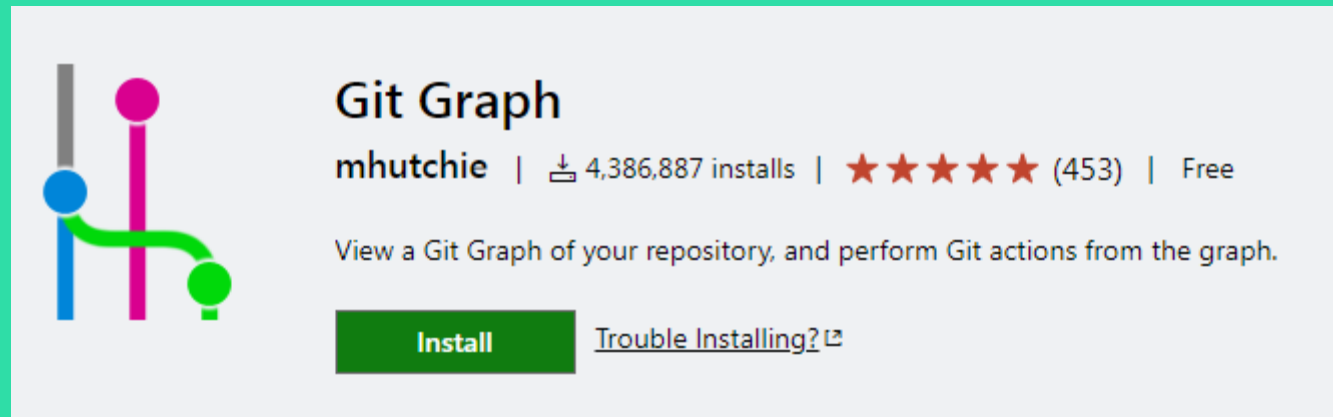
GitKraken  |  21,092,479 installs |      (672) | Free

Supercharge Git within VS Code — Visualize code authorship at a glance via Git blame annotations and CodeLens, seamlessly navigate and explore Git repositories, gain valuable insights via rich visualizations and powerful comparison commands, and so much more

[Install](#)

[Trouble Installing?](#) 

# Some VSCode Extensions



The image shows a screenshot of the VS Code extension marketplace for the 'Git Graph' extension. The card has a light gray background. On the left is a logo with a blue vertical bar, a pink vertical bar, and a green line connecting them with circles. To the right of the logo, the text 'Git Graph' is in a large, bold, black font. Below that, 'mhutchie' is followed by a download icon, '4,386,887 installs', five red stars, '(453)', and 'Free'. A description below reads 'View a Git Graph of your repository, and perform Git actions from the graph.' At the bottom, there is a green 'Install' button and a link 'Trouble Installing?' with an external link icon.

**Git Graph**  
mhutchie | 📄 4,386,887 installs | ★★★★★ (453) | Free


View a Git Graph of your repository, and perform Git actions from the graph.

[Install](#) [Trouble Installing?](#)

# Some VSCode Extensions




## Git Blame

Wade Anderson |  1,633,677 installs | ★★★★★ (40) | Free

See git blame information in the status bar.

[Install](#)

[Trouble Installing?](#) 

You can download this slides in link below:



<https://bagheriali.dev/files/workshops/git/slides.pdf>