Fundamentals of Git

Deakin Software Engineering Club (DSEC)

26th of March (T1) 2021

Welcome 

The first DSEC event of 2021!

Hiya  I'm Ravindu
Studying Bachelor of Software Engineering (Honours) @ Deakin
Secretary, DSEC

RavinduL

Agenda

1. Overview
2. Installation & configuration
3. Repositories & commits
4. Branching, merging & conflict resolution
5. Collaboration

Git

Version Control System (VCS)—keeps track of changes made to files over time.

Widely used, free & open-source!

https://git-scm.com

Installing Git

Windows https://git-scm.com/download/win

Download & run the installer

macOS https://git-scm.com/download/mac

```
$ brew install git
```

Linux https://git-scm.com/download/linux

```
$ apt install git # Debian (e.g., Ubuntu)
$ dnf install git # Fedora
```

Configuring Git

Specify your name & email address

```
$ git config --global user.name "Your Name"
$ git config --global user.email "your.email@deakin.edu.au"
```

Specify a default branch name

```
$ git config --global init.defaultBranch main
```

(Optionally) Specify which text editor you'd like to use

```
$ git config --global core.editor "code -n -w" # Visual Studio Code
```

Terminology

Repository = folder in which Git tracks changes.

Committing = saving changes to the repo, making a “checkpoint” in its timeline.

Creating a Repo

Create an empty folder for your project

```
$ mkdir my-project
$ cd my-project
```

“Initialize” the folder as a repository

```
my-project$ git init
Initialized empty Git repository in my-project/.git/
```

Commits

Stage changes

Marks changes to be included
in the next commit.

```
$ git add <files>
```

Commit staged changes

```
$ git commit -m <commit message>
```

Commit Logs

View the commit log

```
$ git log
```

View changes of a commit

```
$ git show <hash> [<files>]
```

Commit Messages

Keep commits focused, and their changes cohesive.

Commit messages describe the effect of the commit. Prefer imperative mood.

```
$ git commit -m <para1> -m <para2>
```

```
$ git commit -e
```

View single-line logs

```
$ git log --oneline
```

Staging & Diffs

Stage specific changes of files

```
$ git add -e [<files>]
```

View staged changes

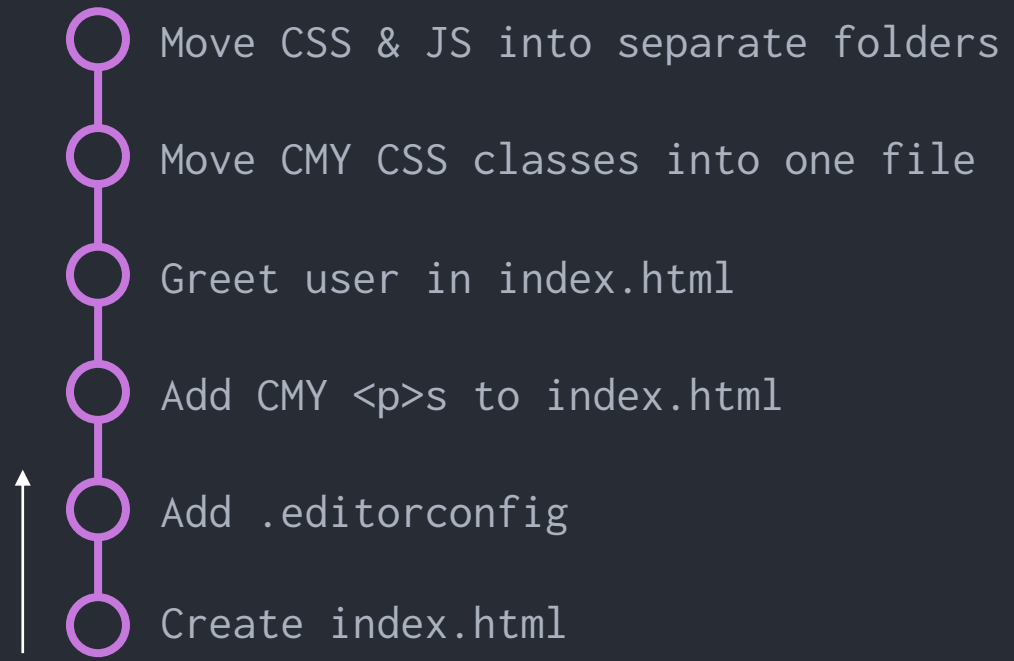
```
$ git diff --cached [<files>]
```

View unstaged changes of tracked files

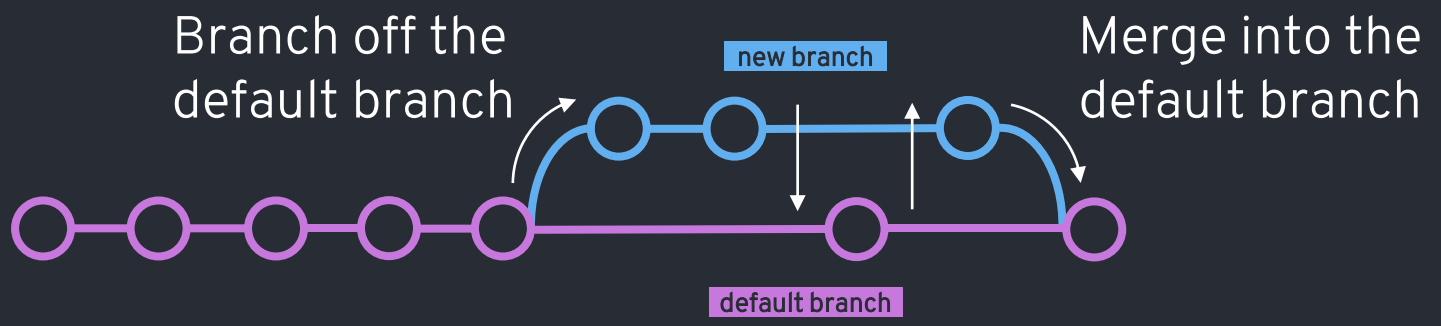
```
$ git diff [<files>]
```

Commit Graph

Chronological representation of commits in a repo.



Branches


Branch off the
default branch
new branch
Merge into the
default branch
default branch

Branches

Create a branch

```
$ git branch <name>
```

List branches

```
$ git branch
```

Switch to (checkout) a branch

```
$ git checkout <name>
```

Create & checkout a branch

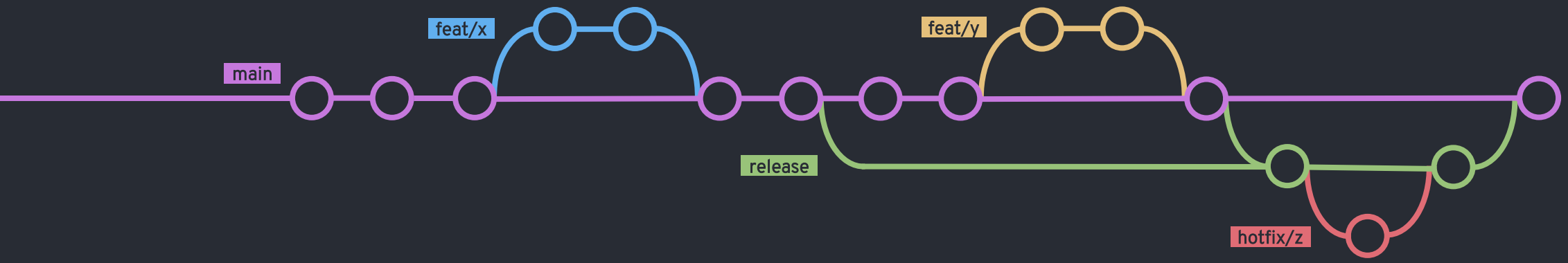
```
$ git checkout -b <name>
```

Merging

Merge a branch into the current branch

```
$ git merge <name>
```

Branching Convention



main	Development branch.
release	Stable versions (“releases”) of the project.
feat/xyz	Features added to the project. Replace “xyz” with a feature ID. Branch off main.
hotfix/xyz	Fixes for bugs found in releases. Replace “xyz” with a fix ID. Branch off release. Merge into main.

Branches

Rename the current branch

```
$ git branch -m <name>
```

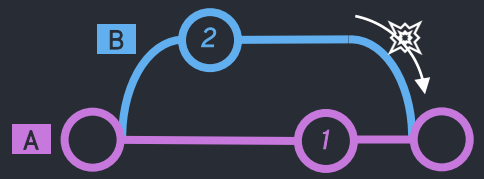
Delete a branch

```
$ git branch -D <name>
```

Conflict Resolution

💥 Merge conflicts occur when,

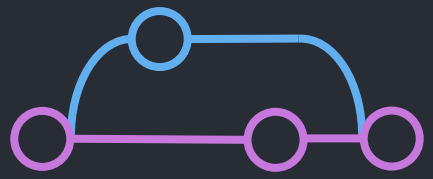
- Different changes are made to the same line(s) on both branches.
- A file deleted in one branch has changed in the other.



🤔 Git doesn't understand what changes to retain/discard,

- Keep changes from *1*?
- Keep changes from *2*?
- Some combination of these?

Conflict Resolution

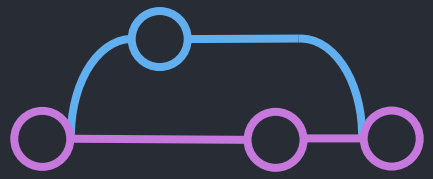


Running `git merge`,

- Merges as much as possible.
- Stages files that are successfully merged.
- Markers conflicting lines of code with merge conflict markers.

```
<<<<<<< HEAD
changes on current branch
=======
incoming branches
>>>>>>> branch/being/merged
```

Conflict Resolution



Merge commit

Fix the conflicts, stage changes & commit.

```
$ git commit
```

Abort a merge

```
$ git merge --abort
```

Terminal Interface

Scrollable commit logs

`j` & `k` to scroll down & up

`Ctrl` `d` & `Ctrl` `u` to page down & up

`q` to exit

Check the status often!

```
$ git status
```

Use git help for reference docs

```
$ git help <command>
```

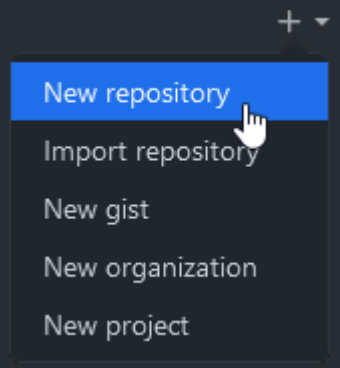
Collaboration

Host your Git repositories on a hosting service such as,

- GitHub
- GitLab
- Bitbucket

Collaboration

Create a GitHub repo



Specify the repo name (unique within your account) & privacy settings.

Collaboration

Add a remote to your local repo

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

Push a new branch to the remote

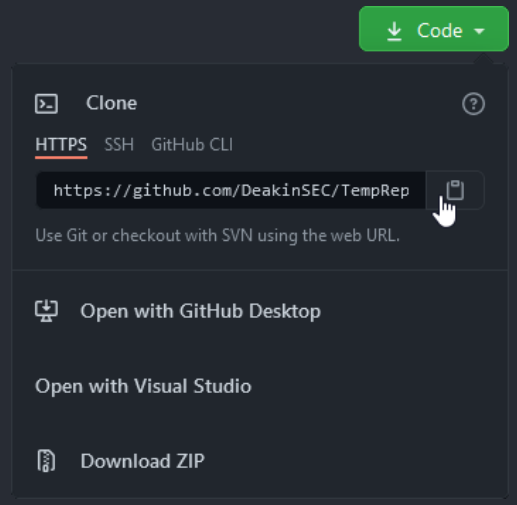
```
$ git push -u <remote name> <branch name>
```

Your project should now be hosted online 

Collaboration

Clone a repo

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



Collaboration

Push the current branch to the remote

```
$ git push
```

⚠ committing != pushing

Collaboration

Pull changes the current branch from the remote

```
$ git pull
```

✔ Branch often to avoid conflicts

Git Projects

Specify files that must be ignored by Git (e.g. autogenerated files/build artefacts) via a `.gitignore` file.

- https://github.com/github/gitignore contains common `.gitignore`s.

Recommended documentation at the root of your repo,

1. `README.md` describes the project.
2. `CONTRIBUTING.md` describes how contributors would get started.
3. `LICENSE` contains the project license. https://choosealicense.com.
4. `CODE_OF_CONDUCT.md` https://docs.github.com/github/building-a-strong-community/adding-a-code-of-conduct-to-your-project

GitHub-flavoured Markdown: https://github.github.com/gfm

Graphical Clients

Terminals

+ Fast, lightweight, complete
+ Allows scripting, automation

Graphical Clients

+ Better visualisation of diffs & repo
- Not universal, possibly functionally limited

✔ Know the underlying Git concepts & terminal commands even if you're using a GUI!

Graphical Git Client

 SourceTree *Free [WM]*

 GitHub Desktop *Free [WM]*

 GitKraken *Free for non-commercial [WML]*

 Fork *[WM]*

 Tower *[WM]*

 Sublime Merge *[WML]*

https://git-scm.com/downloads/guis

Review

✓Installed & configured Git

✓Created a repo
✓Created commits
✓Created & merged branches
✓Resolved merge conflicts

✓Published the repo to GitHub
✓Collaborated with contributors

Thank you!

We'd appreciate your feedback!

https://bit.ly/DSECGitGoodFeedback

Deakin Software Engineering Club

Skill workshops, projects, quizzes, hackathons & more!

Join https://bit.ly/DSECJoin


https://bit.ly/DSECYouTube


DeakinSEC


_deakinsec

Credits

All logos including those of Git, GitHub, GitLab, Bitbucket, SourceTree, GitHub Desktop, GitKraken, Fork, Tower, Sublime Merge, YouTube, Facebook and Instagram, are property of their respective owners.

This presentation uses the following fonts licensed according to the Open Fonts License (OFL),

- Overpass (headings, body)
- Inconsolata (code)