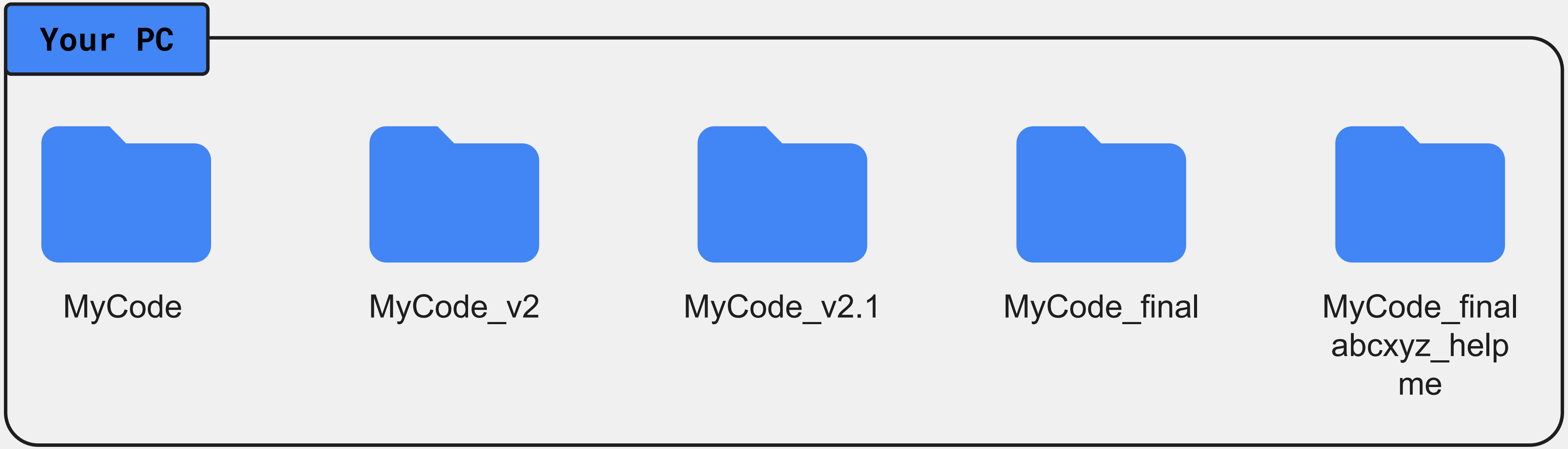
Google Developer Group
Hong Kong University of Science and Technology

Workshop 2: **Git**

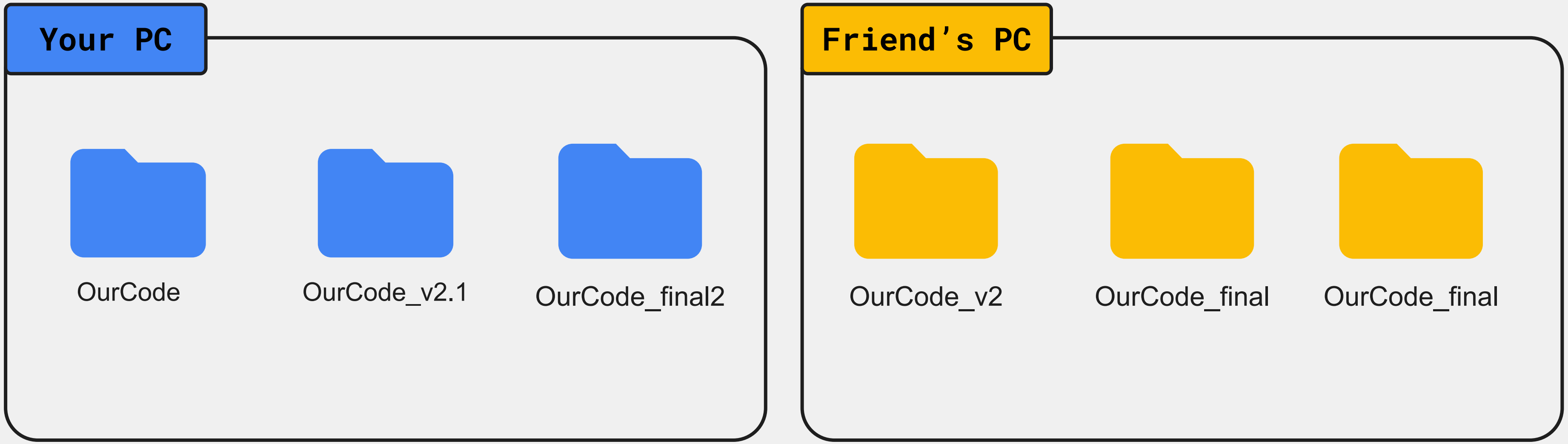
Definition: *an unpleasant or contemptible person (typically used of a man).*

Git

Let’s “git” started

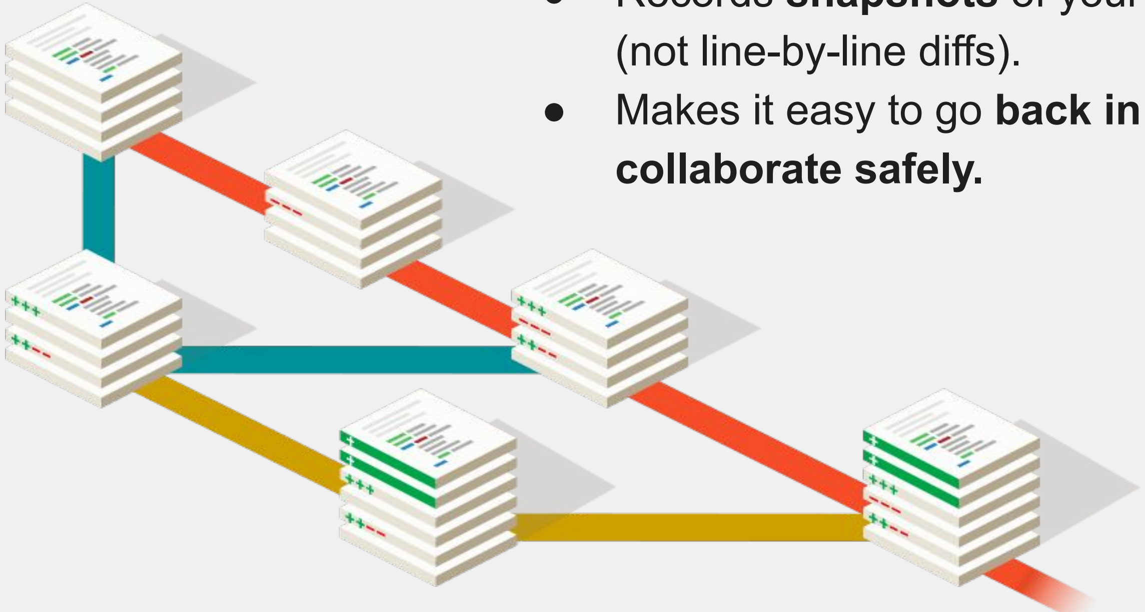


Let's "git" started



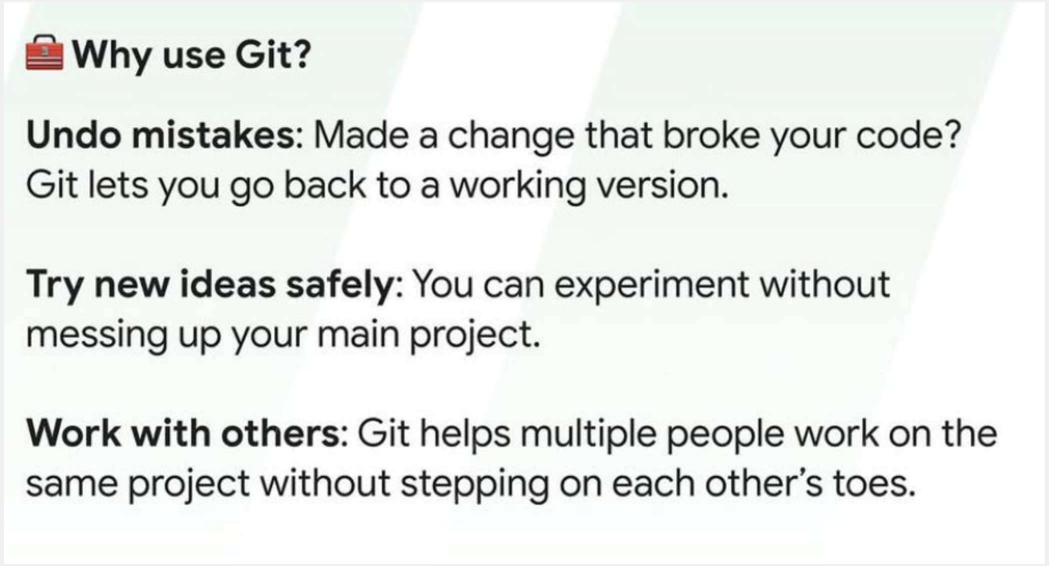
What is **Git**?

- A **local version control system** — everything is tracked on your machine first.
- Records **snapshots** of your project at each commit (not line-by-line diffs).
- Makes it easy to go **back in time, branch, and collaborate safely.**



Why use **git**?

According to the GDG@HKUST Instagram Post



🧰 **Why use Git?**

Undo mistakes: Made a change that broke your code? Git lets you go back to a working version.

Try new ideas safely: You can experiment without messing up your main project.

Work with others: Git helps multiple people work on the same project without stepping on each other's toes.

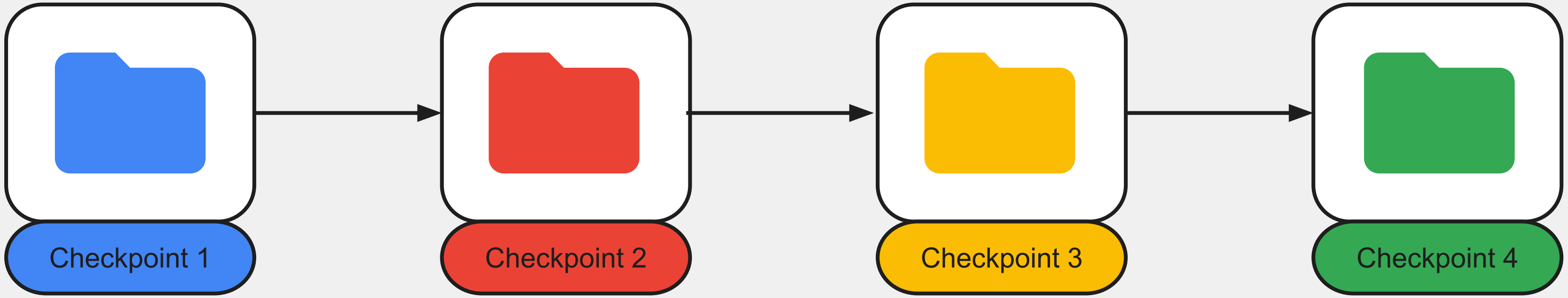
Please do follow if you have not @gdg_hkust

Makes you look like a competent developer! *

`*Works even if you're on the way there!`

Core concept

You build a history of checkpoints.
Each commit is a snapshot you can **revisit**, **merge**, or **share**.



Definitions

Repository (repo): A folder tracked by Git that stores your code and its version history.

Branch: A series of commit or a separate version of your project for new features or experiments.

Commit: A snapshot of your project at a specific point in time.

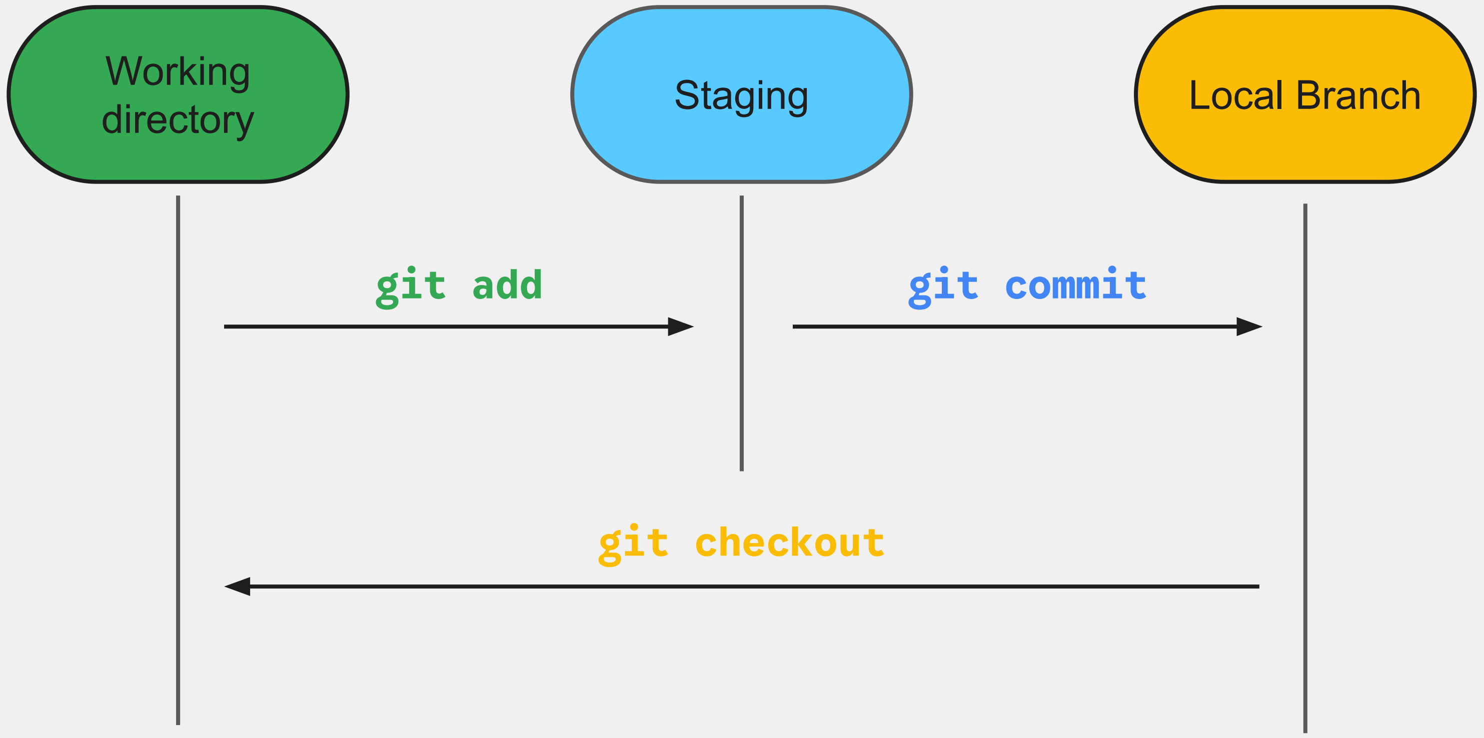
Working Directory: Your actual project files.

Staging Area (Index): A middle zone where you prepare changes before committing them.

Google Developer Group
Hong Kong University of Science and Technology

Git Basic Commands

Git workflow (local)



Committing in **git**

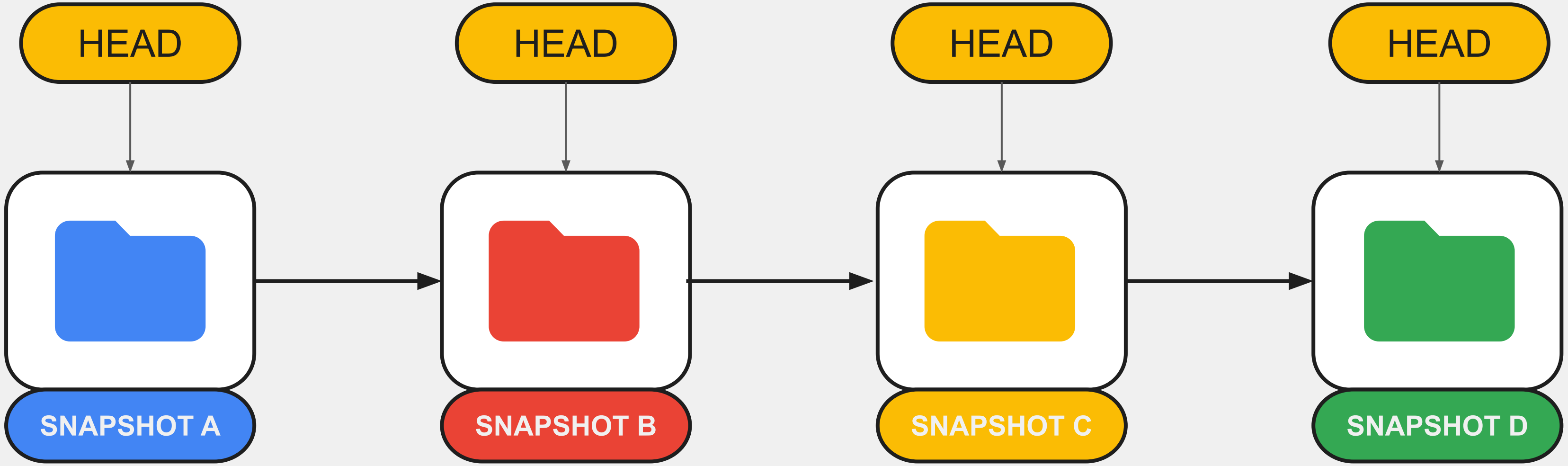
```
# Stage files
# Option 1: Add specific files
git add <file_1> <file_2> <file_3> …

# Option 2: Add all files
git add .

# Commit
git commit -m <a_very_nice_message>
```

Git HEAD

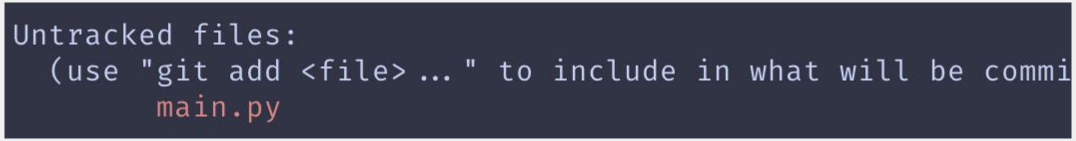
A head is simply a reference to a commit object. Each head has a name (branch name or tag name, etc).



By default, there is a head in every repository called master.

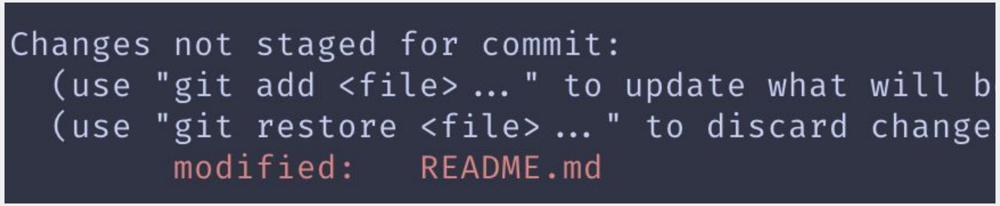
git status

Untracked
New files Git isn’t watching yet



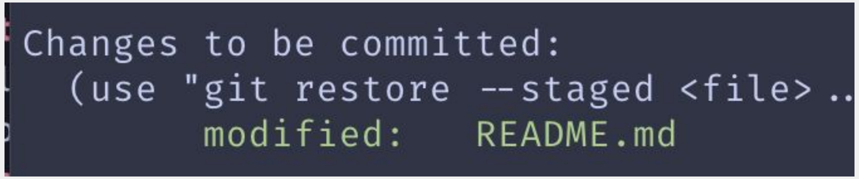
```
Untracked files:
  (use "git add <file>..." to include in what will be commi
        main.py
```

Modified
Tracked file has been changed



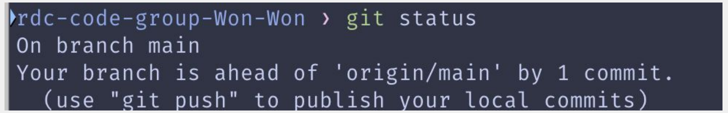
```
Changes not staged for commit:
  (use "git add <file>..." to update what will b
  (use "git restore <file>..." to discard change
        modified:   README.md
```

Staged
File ready to be committed



```
Changes to be committed:
  (use "git restore --staged <file>..
        modified:   README.md
```

Committed
Snapshot saved to history



```
rdc-code-group-Won-Won › git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

.gitignore

Tells Git which files or folders to ignore (not track or commit).

File names

```
secret.txt
config.json
.env
```

Directories (folder + content)

```
/build/
node_modules/
.venv/
```

File extensions / wildcards

```
# all .log files
*.log
# all .tmp files
*.tmp
```

Subdirectory patterns

```
secret.txt
config.json
.env
```

Leading slash (/)

```
/config.json   # only in root
config.json    # anywhere
```

Google Developer Group
Hong Kong University of Science and Technology

Undoing & Fixing Mistakes

Undo changes in the working directory (before staging)

```
git restore index.html

# Older git/method of doing it
git checkout -- index.html
```

⚠️ Overwrites your local changes (can’t undo)

Moves the HEAD and optionally modifies staging/working area

`git reset --<mode> <commit>` **(with caution!)**

Mode	Affects Staging?	Affects Working Dir?	Typical Use
`--soft`	❌	❌	Uncommit, keep changes staged
`--mixed`	✅	❌	Uncommit, keep changes unstaged
`--hard`	✅	✅	Reset everything (dangerous)

Example:

```
git reset --hard HEAD~1
```

Safer option: `git revert`

Use **revert** to undo something already pushed.

Command	Purpose	Safe for shared repos?
`git revert <commit>`	Creates a new commit that undoes changes	✅ Yes
`git reset <commit>`	Moves branch pointer (can rewrite history)	❌ No

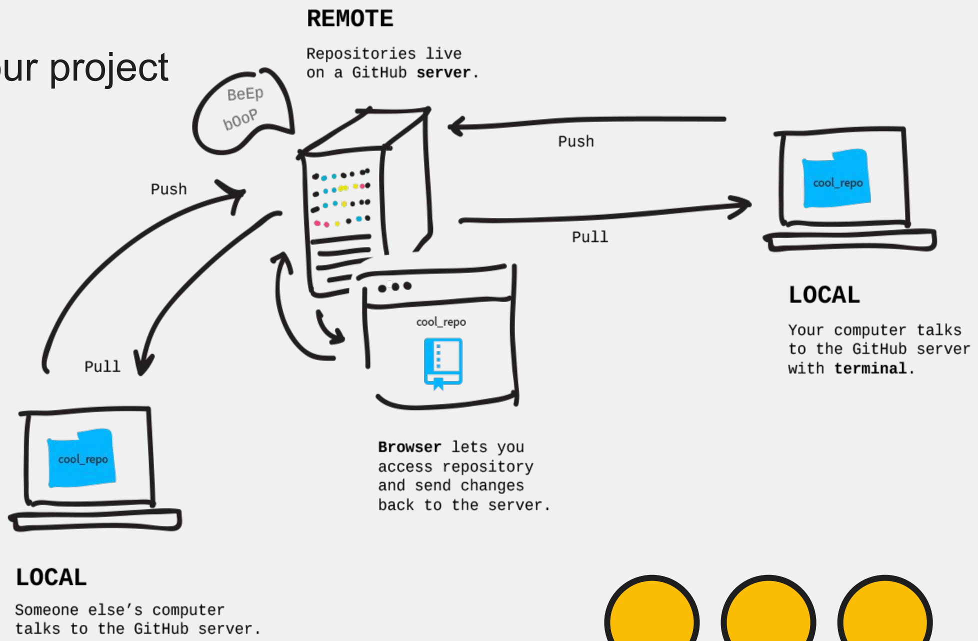
Google Developer Group
Hong Kong University of Science and Technology

Working with **Remotes**

Collaborating with Others

What’s a Remote?

- A remote repository is a version of your project hosted online (e.g., GitHub, GitLab).
- Lets you:
    - Back up your code
    - Collaborate with teammates
    - Review and merge changes


REMOTE
Repositories live on a GitHub **server**.
BeEp bOoP
Push
Pull
Push
Pull
cool_repo
cool_repo
cool_repo
LOCAL
Your computer talks to the GitHub server with **terminal**.
Browser lets you access repository and send changes back to the server.
LOCAL
Someone else’s computer talks to the GitHub server.

Setting up credentials

Set your identity

Git uses this info for every commit you make:

```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
git config --global credential.helper store
```

Check your settings

```
git config --list
```

* This works for public repo for private repos, you will need more authentication

Linking Local Git to GitHub

Option 1: Clone a Remote Repo

If the project already exists online:

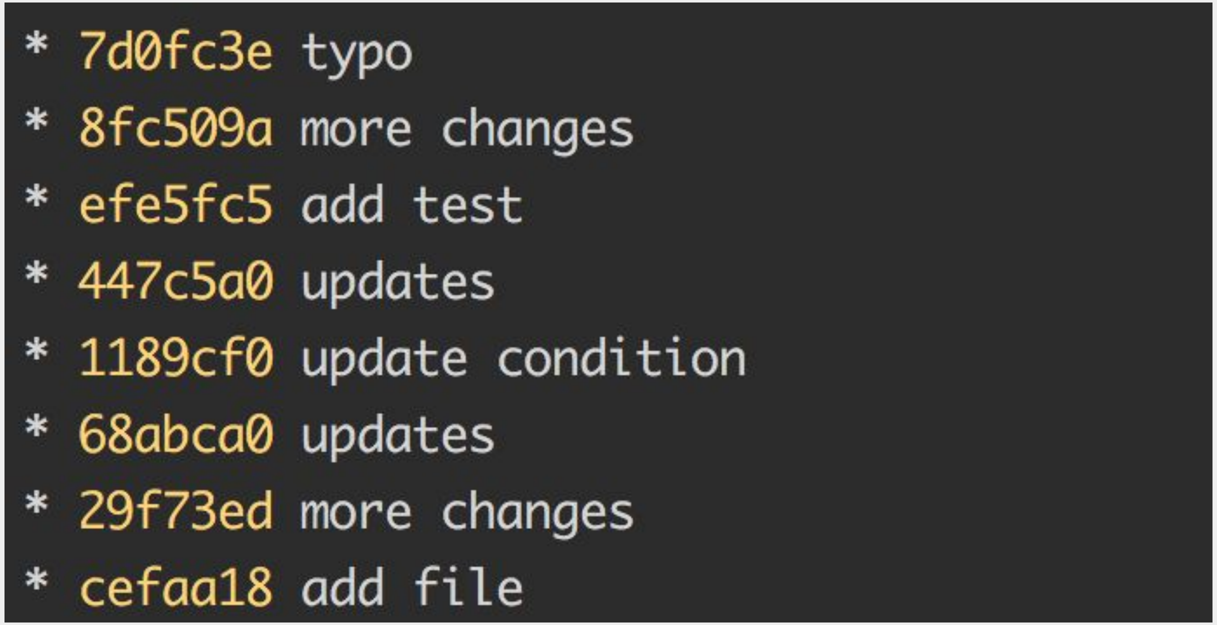
```
git clone https://github.com/quangngonz/git-workshop-demo.git

cd git-workshop-demo
```

Option 2: Add a Remote to an Existing Local Repo

If you started locally with git init:

```
git remote add origin
https://github.com/quangngonz/git-workshop-demo.git
git branch -M main
git push -u origin main
```



```
* 7d0fc3e typo
* 8fc509a more changes
* efe5fc5 add test
* 447c5a0 updates
* 1189cf0 update condition
* 68abca0 updates
* 29f73ed more changes
* cefaa18 add file
```

Good commit message

```
Commit Meesage:

<feat>: <description>

[optional body]
```

- `feat` – a new feature is introduced with the changes
- `Fix` – a bug fix has occurred
- `chore` – changes that do not relate to a fix or feature and don't modify src or test files (for example updating dependencies)
- `refactor` – refactored code that neither fixes a bug nor adds a feature

Google Developer Group
Hong Kong University of Science and Technology

Branch and **Pull Requests**

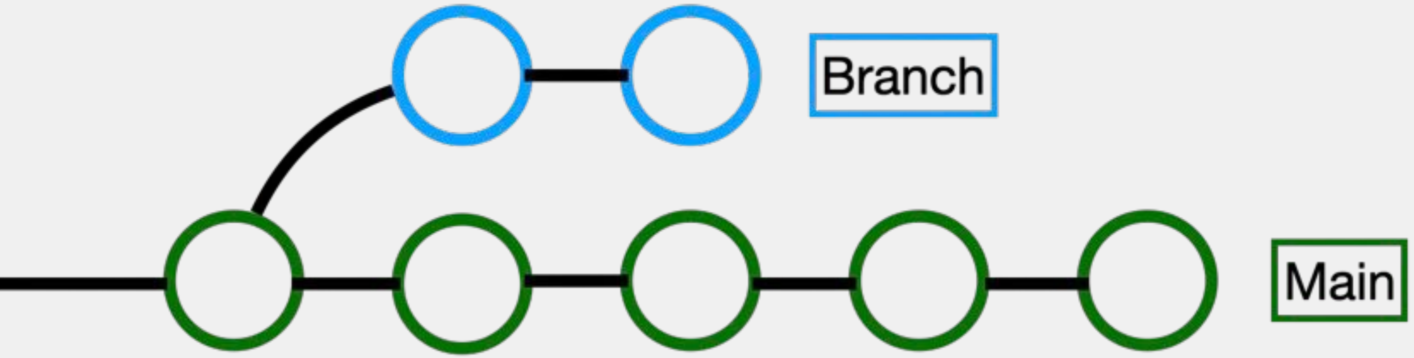
Branch

What is a Branch?

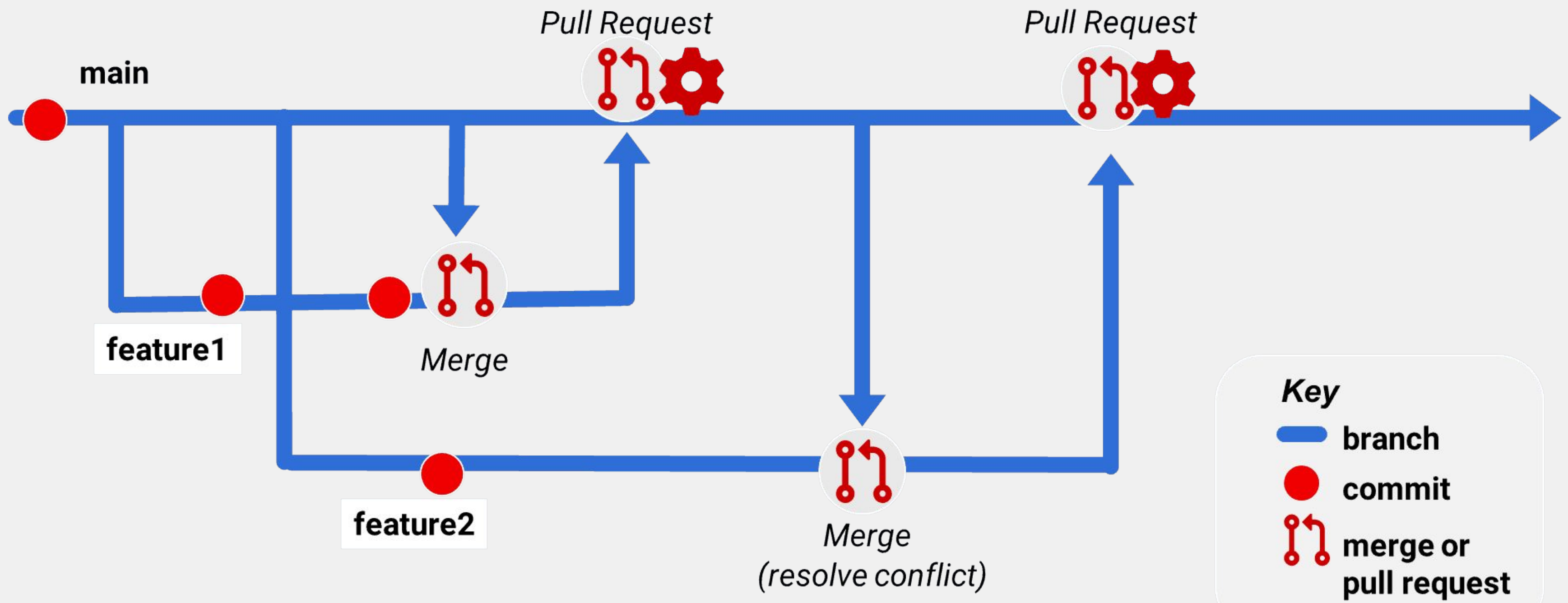
- A branch is like a separate workspace for your code.
- You can experiment without touching the main project.
- The default branch is usually called main.

Why Use Branches?

- Keep main clean and working
- Develop new features safely
- Test ideas or fix bugs separately
- Combine (merge) when ready



Branch and Pull Requests



Key

▬	branch
●	commit
⮍	merge or pull request

It is easy to shoot your foot off with **git**, but also easy to revert to a previous foot and merge it with your current leg.

- Jack William Bell

Google Developer Group
Hong Kong University of Science and Technology

Thank you for coming!

Definition: *an unpleasant or contemptible person (typically used of a man).*