Workshop outline

DAY 1: git

1) Concept
2) Main tools

→ Practical part 1

3) Terminal vs GUI
4) Handling merge conflicts
5) Example workflows
6) Tips ‘n tricks

→ Practical part 2

DAY 2: GitHub

1) Github tour
2) Git-Github syncing

→ Practical part 1

3) Github collaboration
4) Git: down the rabbit hole

→ Practical part 2

GitHub tour

Important Github topics

- Navigation
- READMEs and .gitignore
- File organisation conventions
- Cloning and forking
- Comparing commits
- Setting up a Github token
- Secret tip

Tips

- VS Code on Github
  - Modify URL: add “1s”

    “https://github.com/idf-io/Git-Github_Workshop”

    “https://github1s.com/idf-io/Git-Github_Workshop”

- Always initiate your repo on Github and clone to local!
  - Other way around is much more complicated

Syncing Git and Github

Sync setup

- Setup

```
> git config --global user.name [name]
> git config --global user.email [email]
```

```
> git remote add origin [URL]
```

- “origin” is a convention name for your remote repository

```
> git remote
> git remote -v
```

View local and remote branches

- Remote branches

```
> git branch -r
```

- All branches (local and remote)

```
> git branch --all
> git branch -vv
```

Get remote data

- Download changes from the remote repo:

```
> git fetch origin
> git fetch origin [branch name]
> git fetch origin --all
```

- `git fetch` doesn’t apply the changes
- Now you can access them as well (don’t work on them in this state!)

```
> git branch --all
> git branch -vv
> git checkout origin/myBranch
```

Tracking remote branches

```
> git push origin [branch to upload]
```

Incorporating remote data

- If you know that no changes have happened in remote or you have configured git to automatically merge or rebase:

```
> git checkout [target branch]
> git pull
```

- Usual case: manual merging

```
> git checkout [target branch]
> git fetch origin [target branch]
> git status
> git merge origin/[target branch]
```

Download new branches

- Download (pull) changes from the remote repo:

```
> git checkout (-t) [origin/branch-name]
```

Upload your changes

```
> git push origin [branch to upload]
```

- If you want to overwrite remote changes (careful!)

```
> git push origin --force  [branch to upload]
```

- Delete remote branches

```
> git push origin --delete [remote branch name]
```

Tag sync

```
> git push origin [tagName]
```

- Push one tag
- View remote tags: directly in remote repository platform

```
> git push origin --tags
```

- Delete branch

```
> git push origin --delete [tagName]
```

- Delete remote tag

Github collaboration

Collaborate

- Same remote repo but different branches
- Formal collaboration via “pull requests”

Some more Git

More terminologies

- Remote (repository)
- Local (repository)
- Push: “Send” local changes to remote
- Pull: “Receive” changes from remote

Extended tool-set

- Diff
- Stash
- Reset

Explore changes

```
> git diff [branch1..branch2 | ref1..ref2 | commit1..commit2]
```

- View changes/differences between commits
- HEAD: where you are
- HEAD~2: two commits before HEAD
- HEAD~1 == HEAD^: parent commit
- .. : difference
- …: difference between the common ancestor

Save temporary changes

```
> git stash
```

- View stashes

```
> git stash list
```

- Retrieve a stash

```
> git stash pop (stash)
> git stash apply (stash)
```

Save temporary changes

- Also stash untracked files

```
> git stash -u
```

- Name the stash

```
> git stash save “Message”
```

How to undo changes

- If you want to change the last commit

```
> git reset [commit]
```

- Restore data to a specific point in the past and keeps the changes since as modified and untracked files

- If you want to discard all changes since a commit

```
> git reset --hard [commit]
```

- Careful!!!

How to correct your mistakes

- Rename the last commit message (! creates a new commit!)

```
> git commit --amend “Message”
```

- Add a file to the past commit

```
> git add file
> git commit --amend [file] --no-edit
```

- Add many files to the past commit

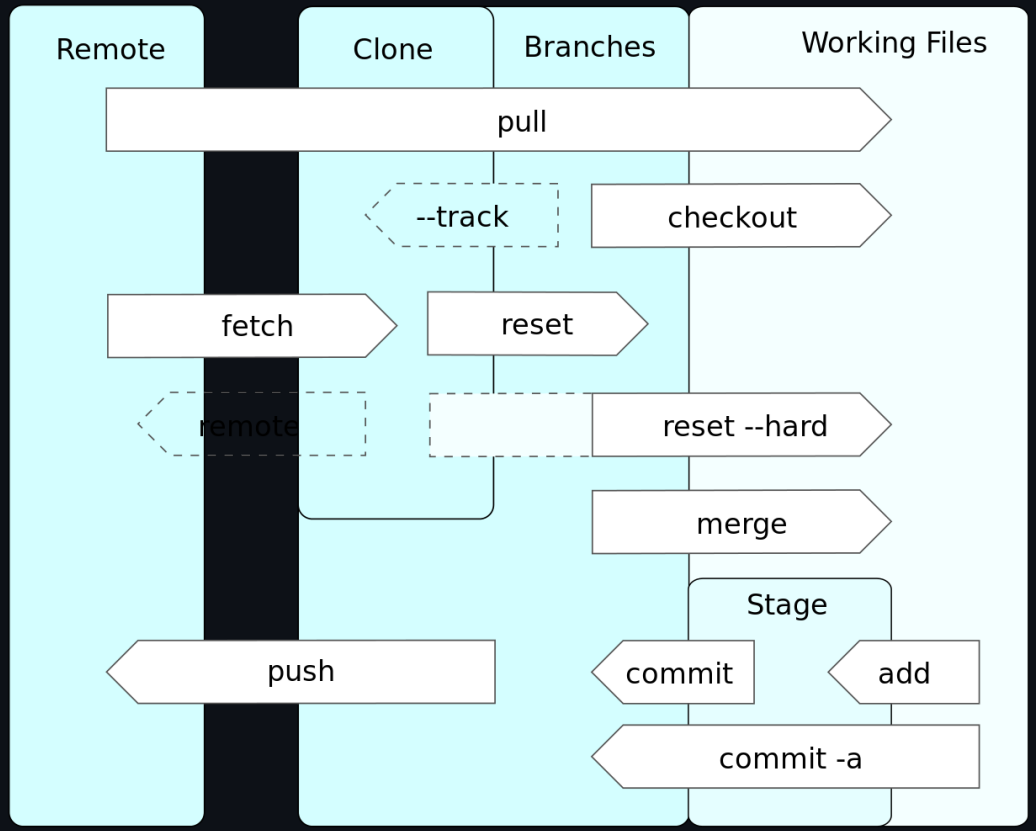
```
> git add [file1] [fileN]
> git commit --amend
```

THM:

- git is a super versatile tool to manage versions of a project
- Integration with GitHub allows you to back up, collaborate and make your code public
- All that I have presented in this workshop can be learned in detail in this resource by Atlassian:

  https://www.atlassian.com/git/tutorials/setting-up-a-repository

Git/Github overview



https://en.wikipedia.org/wiki/Git#/media/File:Git_operations.svg

Git/Github cheatsheet by Atlassian

Down the rabbit hole

Rebase a branch

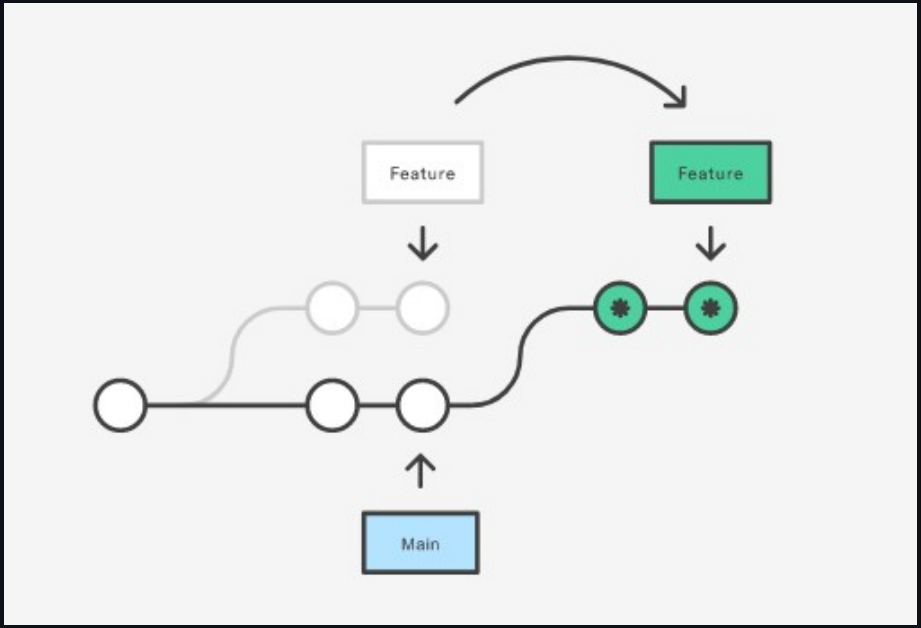
- 2 schools of Git thought:
  1) Git is the detailed history of what happened in a project

     → git merge

  2) Git is a clean history of changes

     → git rebase (+ git merge)

==> Details



Rewrite history (careful!)

- Homework for the curious

```
> git rebase -i [Nr of commits back in time]
```

- Try it out in the terminal!

Cherry-pick commits

```
> git cherry-pick [commit]
```

- Applies a single commit to HEAD
- VS merge VS rebase
- Usecases:
  1) Hotfixes
  2) Get specific code

```
a - b - c - d   Main
      \
      e - f - g   Feature
```

```
a - b - c – d - f   Main
      \
      e - f - g   Feature
```

Tips ‘n tricks

- Find common commit ancestor

```
> git merge-base [branch1] [branch2]
```