Version Control: Git

Agenda

- Introduction to version control
- Git Basics
- Github Basics
- Python Project Setup
- Github Actions (CI/CD)

Have you ever worked on a Python project that stopped working after you made a change and you weren’t quite sure how to get it back?

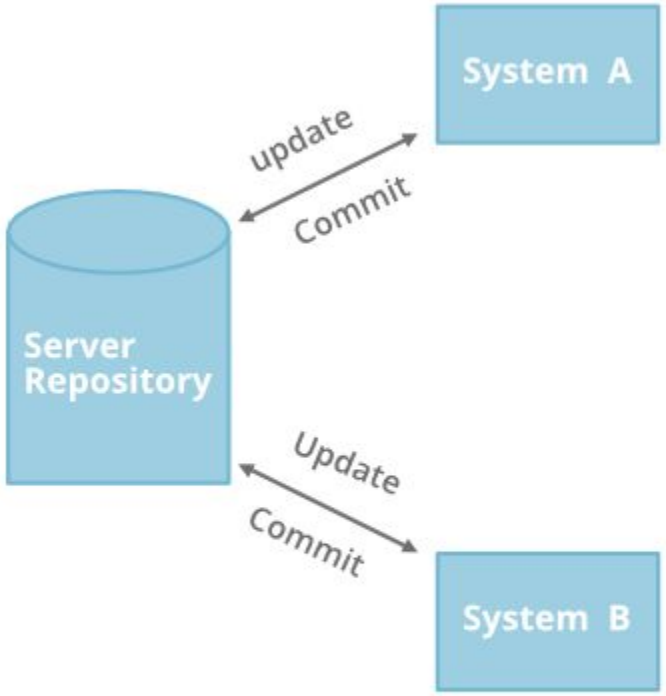
Version Control

- A *version control system* (VCS) is a set of tools that track the history of a set of files. This means that you can tell your VCS (Git, in our case) to save the state of your files at any point. Then, you may continue to edit the files and store that state as well. Saving the state is similar to creating a backup copy of your working directory. When using Git, we refer to this saving of state as *making a commit*.

- When you make a commit in Git, you add a commit message that explains at a high level what changes you made in this commit. Git can show you the history of all of the commits and their commit messages. This provides a useful history of what work you have done and can really help pinpoint when a bug crept into the system.

- In addition to showing you the log of changes you've made, Git also allows you to compare files between different commits.

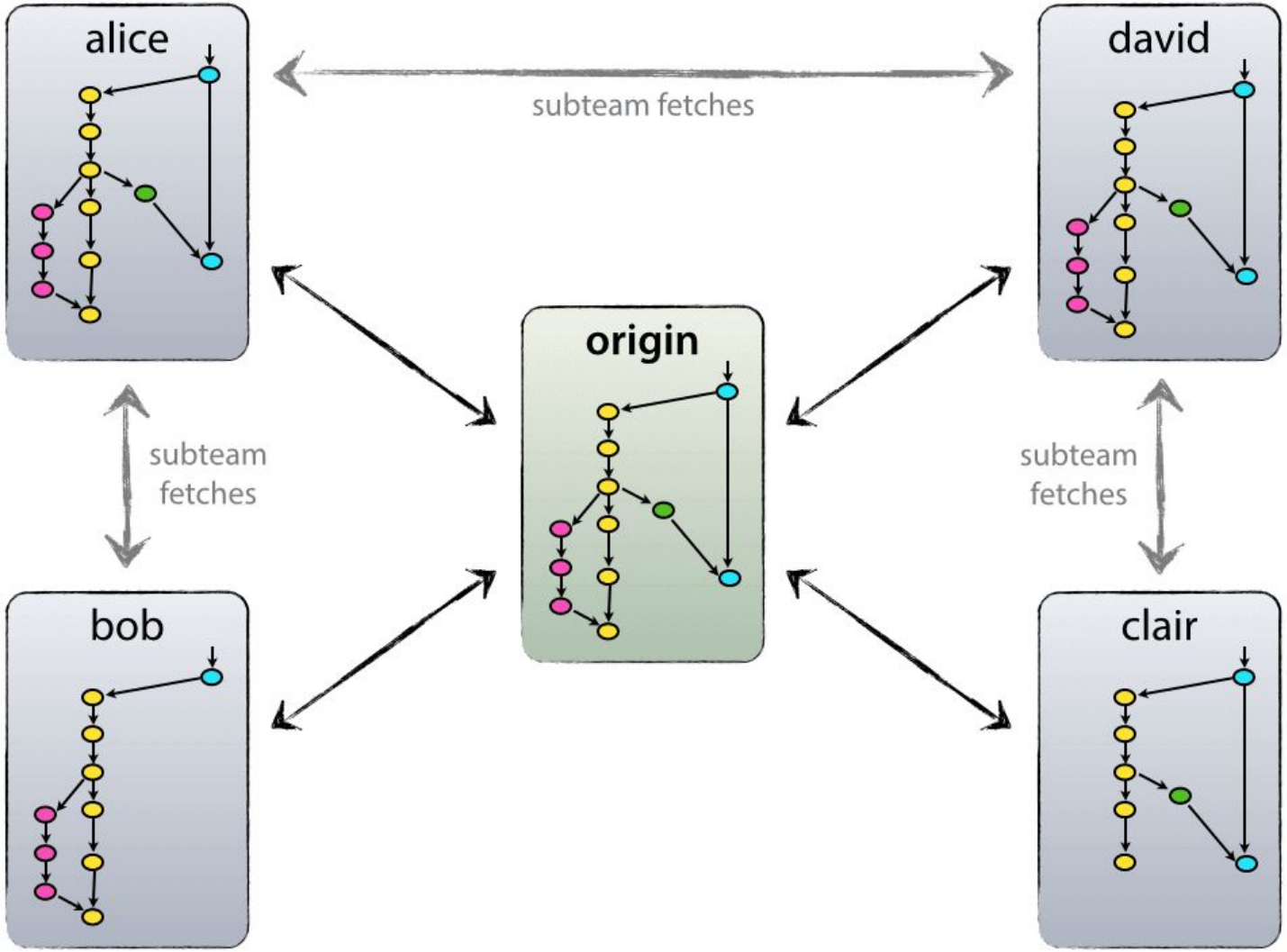
Centralized Vs Distributed Version Control System



Centralized

SVN

Distributed Version Control System



Each developer pulls and pushes to origin.

But besides the centralized push-pull relationships, each developer may also pull changes from other peers to form sub teams.

First Time Git Users

git config --global user.name <name>

git config --global user.id <email>

Create Repo

Local Git Repo:

```
$ mkdir example
$ cd example
$ git init
```

Github Repo:

1. Create new repo on github.com
2. Clone the repo locally.

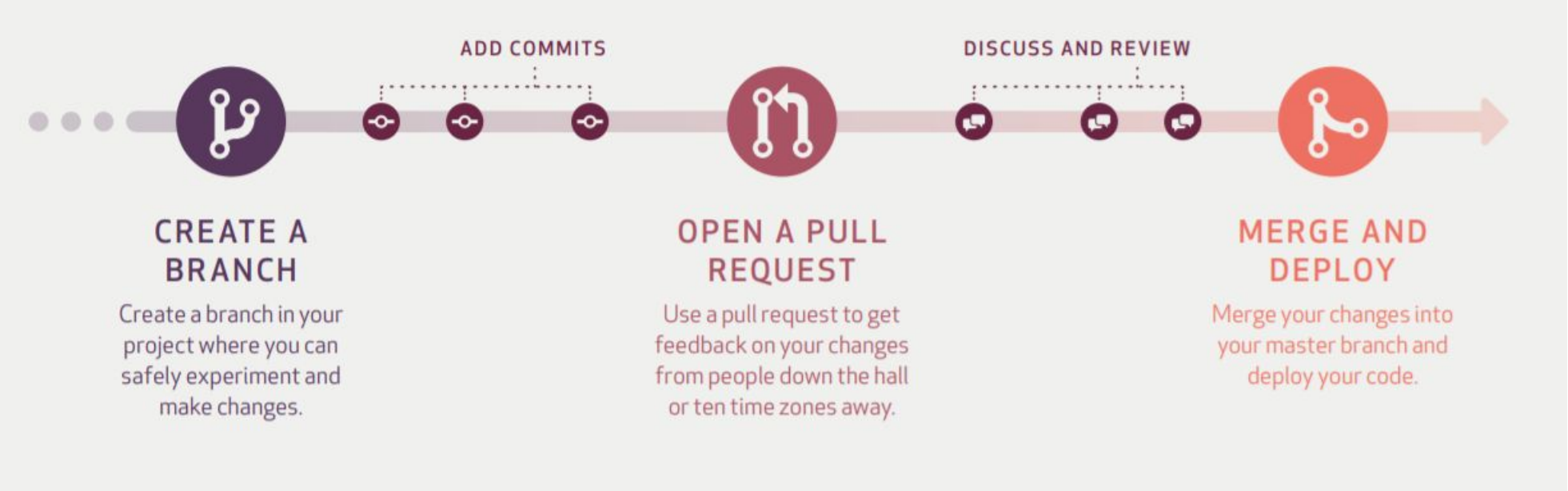
Git Workflow - Demo

Git Workflow

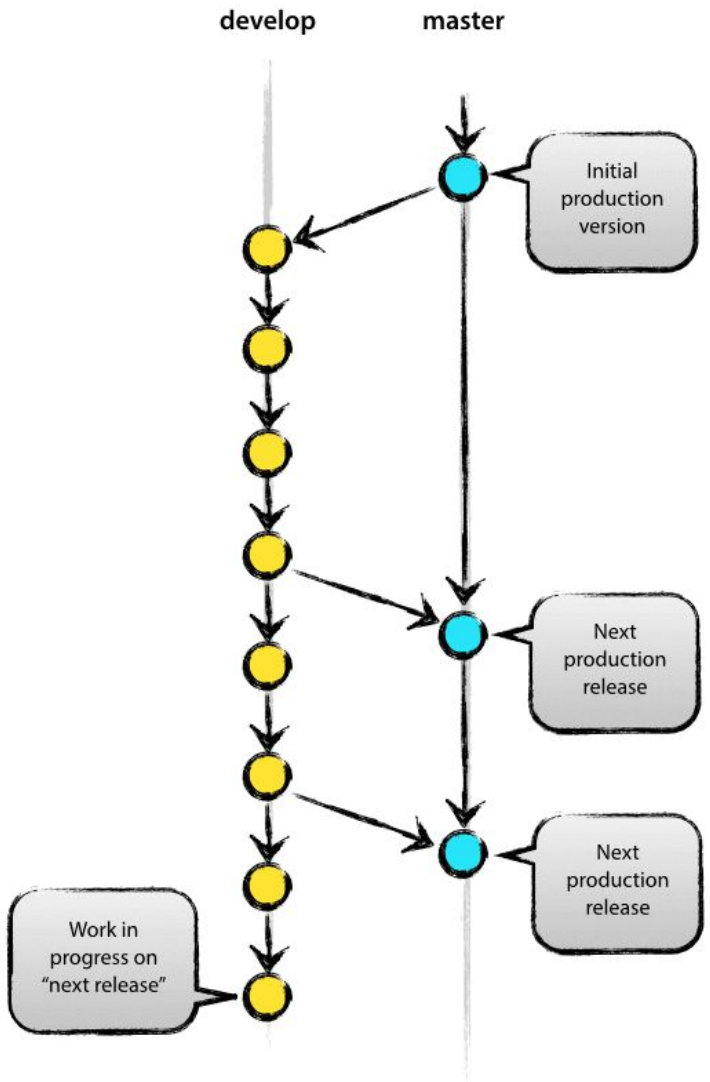
1. `git status` – Make sure your current area is clean.
2. `git pull` – Get the latest version from the remote. This saves merging issues later.
3. Edit your files and make your changes.
4. `git status` – Find all files that are changed. Make sure to watch untracked files too!
5. `git add [files]` – Add the changed files to the staging area.
6. `git commit -m "message"` – Make your new commit.
7. `git push origin [branch-name]` – Push your changes up to the remote.

Git Branches

https://guides.github.com/introduction/flow/



Git Branching Strategy



At the core, the development model is greatly inspired by existing models out there. The central repo holds two main branches with an infinite lifetime:

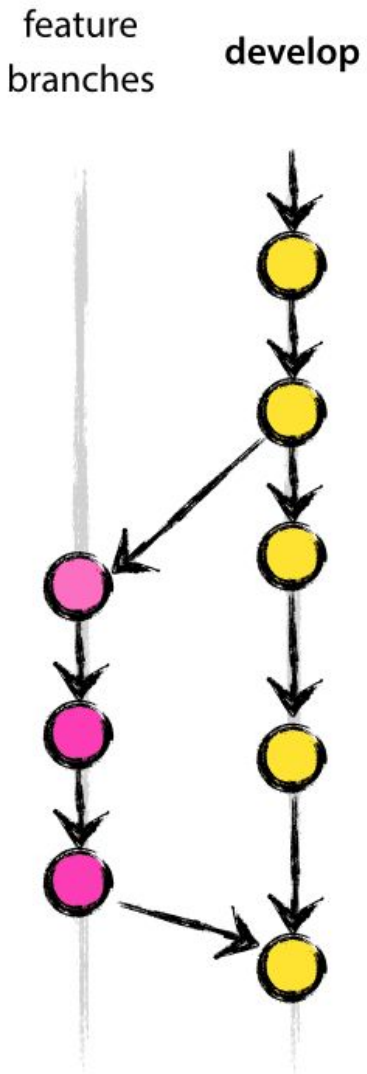
1. master
2. develop

We consider origin/master to be the main branch where the source code of HEAD always reflects a **production-ready state**.

We consider origin/develop to be the main branch where the source code of HEAD always reflects a state with the **latest delivered development changes for the next release**. Some would call this the "integration branch".

https://nvie.com/posts/a-successful-git-branching-model/

Feature Branch



Keeping separate branch for every feature.

Examples of features in data science project are related to:

1. New Modelling Iteration / Retraining with updated dataset.
2. Data Engineering Pipeline Development / Bugfix
3. API Development / Bugfix

Note:

1. We don't want our feature branches to live for very long. They should eventually merge in develop and finally in master at frequent intervals. (Otherwise, difficult to merge later).

2. Ideally, we don't want non-related things to be developed in same feature branch. (May cause conflicts / overlapping changes)

3. Ideally, we don't want multiple people to be working on same feature branch .

What is a merge conflict?

- If you changed the same part of the same file differently in the two branches you're merging, Git won't be able to merge them cleanly. It will result into a merge conflict.

- Let's try it out.

Github Actions

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

https://docs.github.com/en/actions

Example: CI Pipeline to run test cases in example_pkg
https://github.com/AlmaBetter-School/packaging_tutorial/blob/main/.github/workflows/main.yml

Appendix

References

- https://realpython.com/python-git-github-intro/#gitignore
- https://towardsdatascience.com/a-quick-primer-to-version-control-using-git-3fbdbb123262
- https://nvie.com/posts/a-successful-git-branching-model/
- https://git-scm.com/book/en/v2\

Git Interactive:
https://learngitbranching.js.org/
https://try.github.io/