Git and GitHub for Complete Beginners

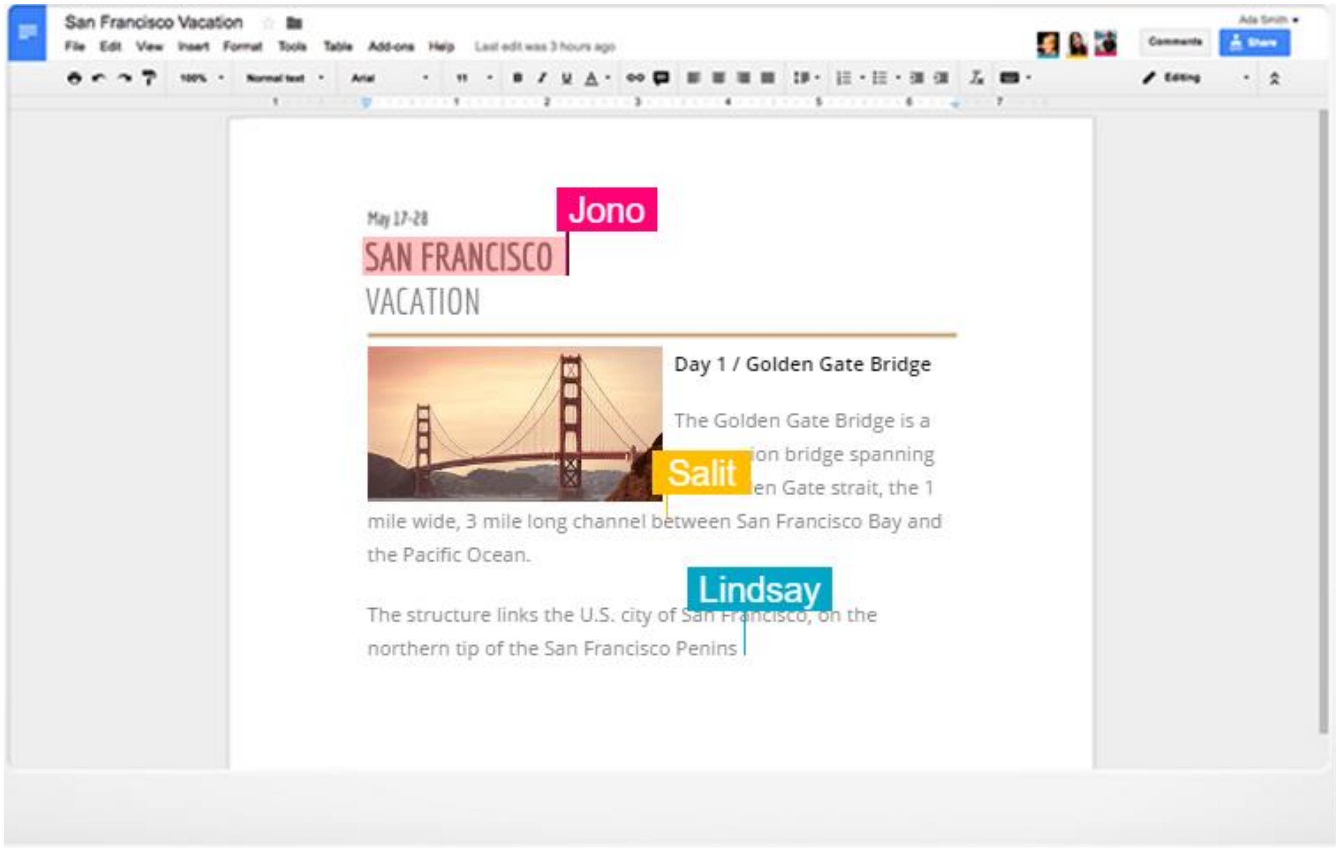
January 2017

Before we start:

- Install git: https://git-scm.com/download/
- Make a Github account: https://github.com/

Git = Version Control

Why version control?


San Francisco Vacation
File Edit View Insert Format Tools Table Add-ons Help Last edit was 3 hours ago
Comments
Share
Editing
Jono
Salit
Lindsay

May 17-28

SAN FRANCISCO
VACATION

Day 1 / Golden Gate Bridge

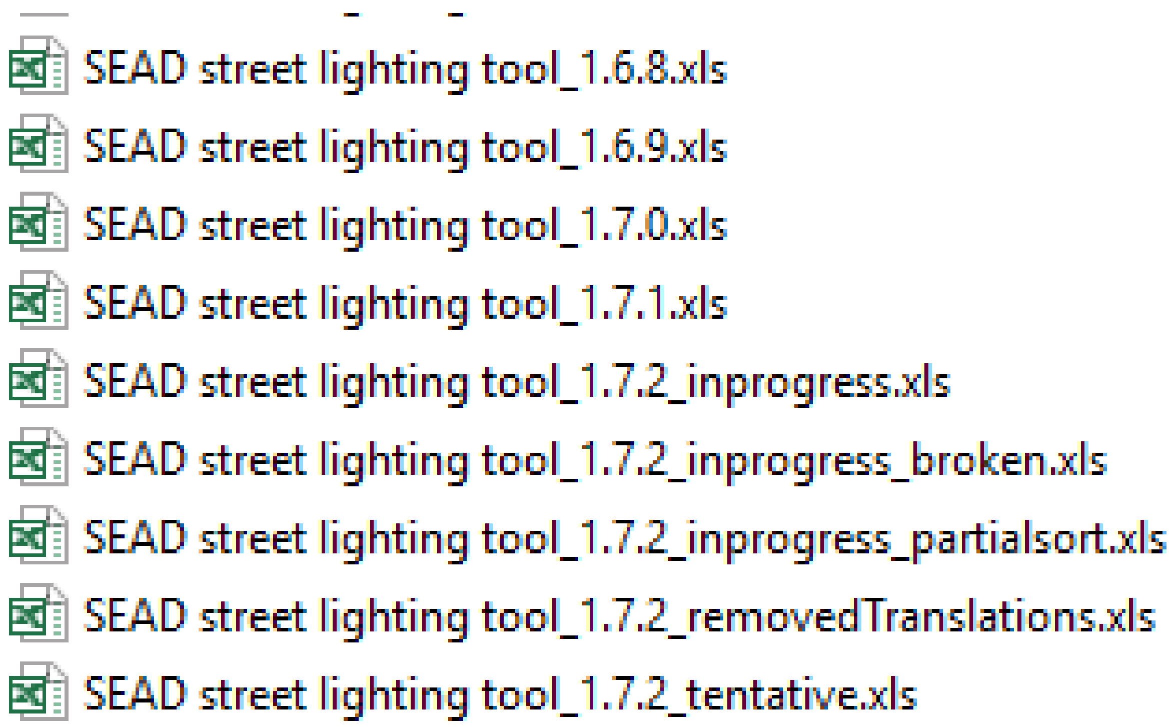
The Golden Gate Bridge is a ion bridge spanning en Gate strait, the 1 mile wide, 3 mile long channel between San Francisco Bay and the Pacific Ocean.

The structure links the U.S. city of San Francisco, on the northern tip of the San Francisco Penins

Google Docs - entirely synchronous, managed but a) poor history and b) online only

Git = Version Control

Why version control?


SEAD street lighting tool_1.6.8.xls
SEAD street lighting tool_1.6.9.xls
SEAD street lighting tool_1.7.0.xls
SEAD street lighting tool_1.7.1.xls
SEAD street lighting tool_1.7.2_inprogress.xls
SEAD street lighting tool_1.7.2_inprogress_broken.xls
SEAD street lighting tool_1.7.2_inprogress_partialsort.xls
SEAD street lighting tool_1.7.2_removedTranslations.xls
SEAD street lighting tool_1.7.2_tentative.xls

Copy the whole file (email it?) – asynchronous, and unmanaged

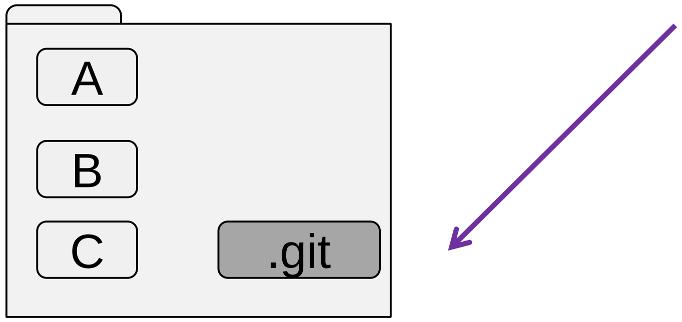
Git = Version Control

- You make changes independently
- You sync your changes periodically
- Git manages the relationship between your edits, other peoples edits, and old versions

What is Git?

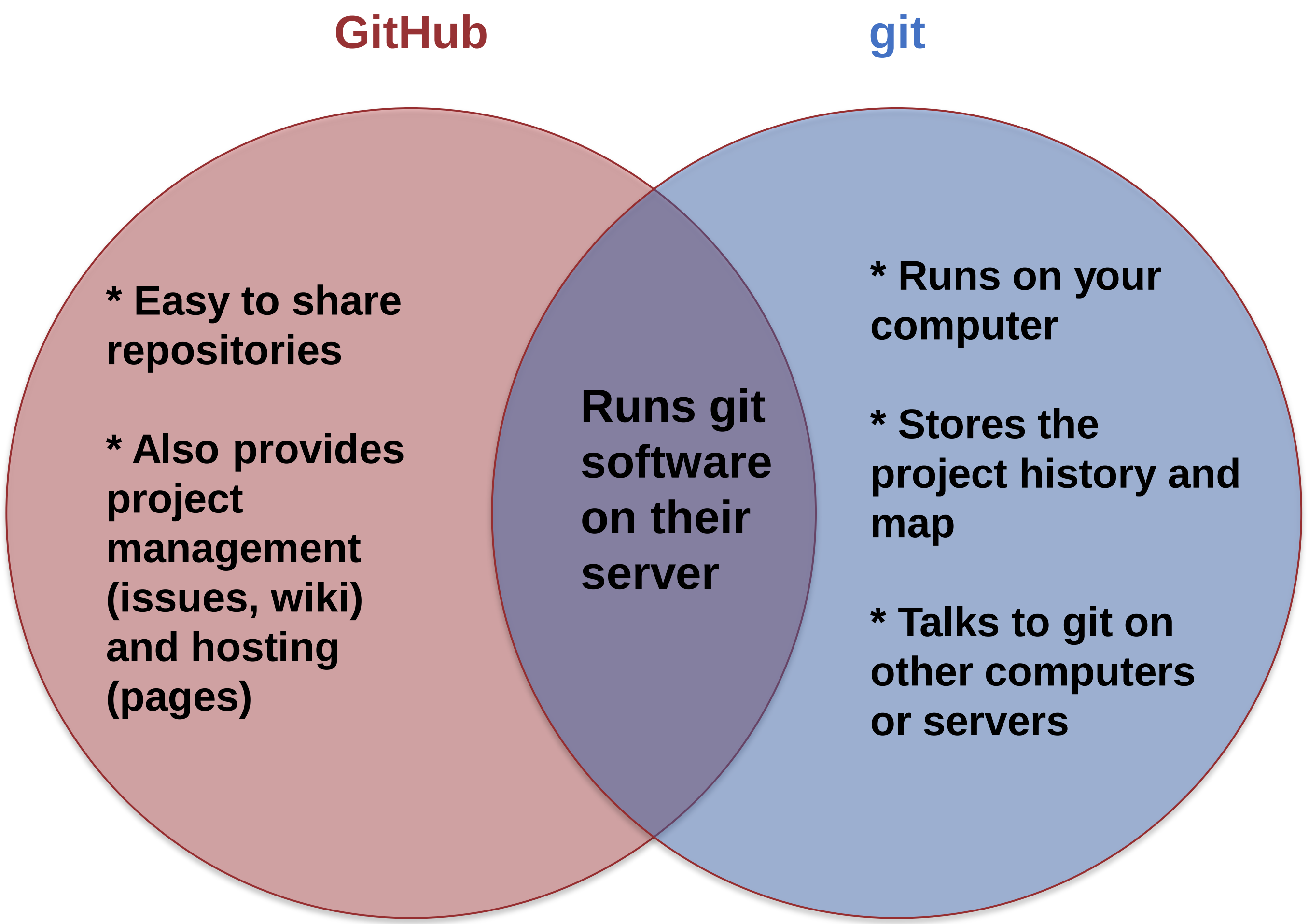
Version control system for a project ('repository'). Everyone has: all current files, all past files, and a map of how they are related to each other.

A repository (repo) on your computer:

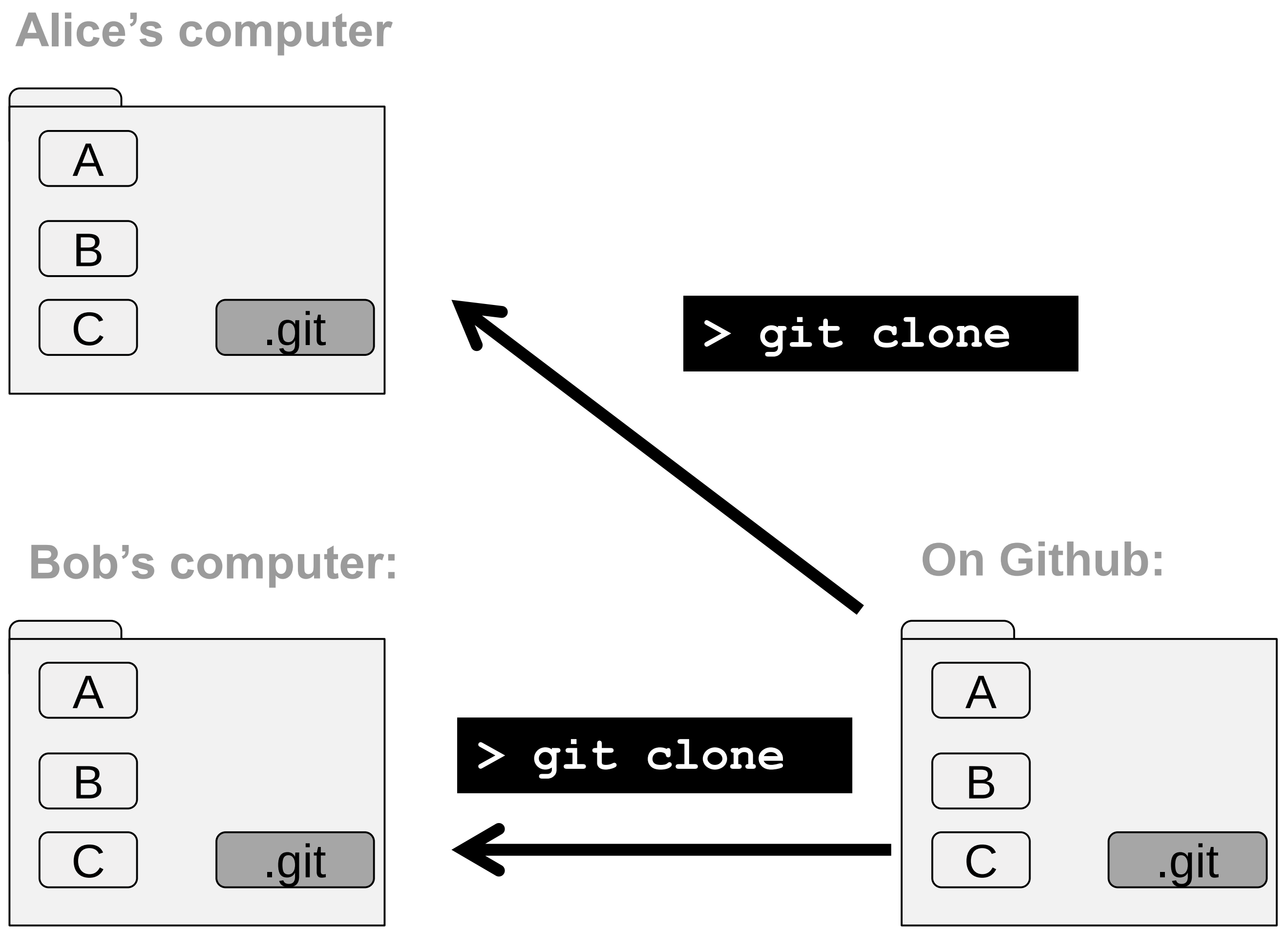


Hidden folder. This folder keeps track of past versions, and talks to the Git software.

The unscientific Venn Diagram of Github

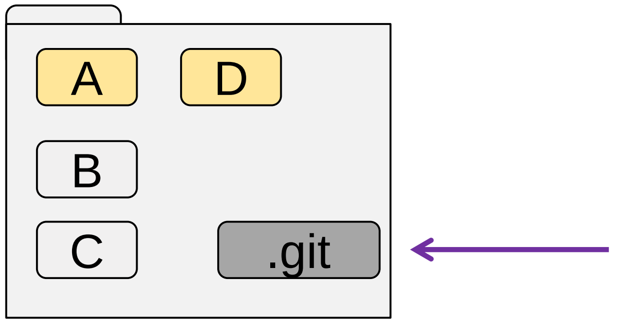

GitHub
git
* Easy to share repositories
* Also provides project management (issues, wiki) and hosting (pages)
Runs git software on their server
* Runs on your computer
* Stores the project history and map
* Talks to git on other computers or servers

Getting a repo



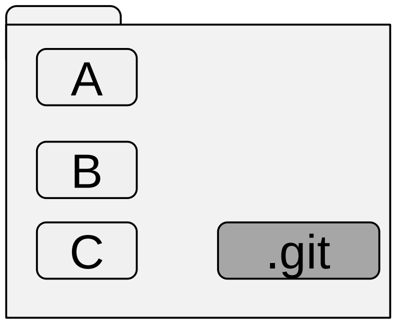
What happens when you edit files?

Alice edits, and clicks ‘File-save’:

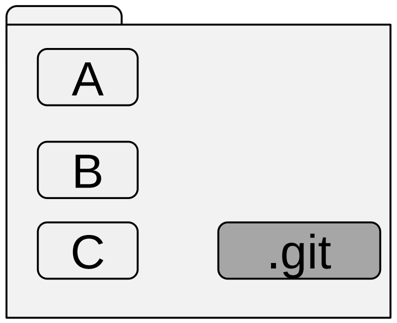


git on her computer knows they are different, but hasn’t recorded the difference

Bob’s computer:

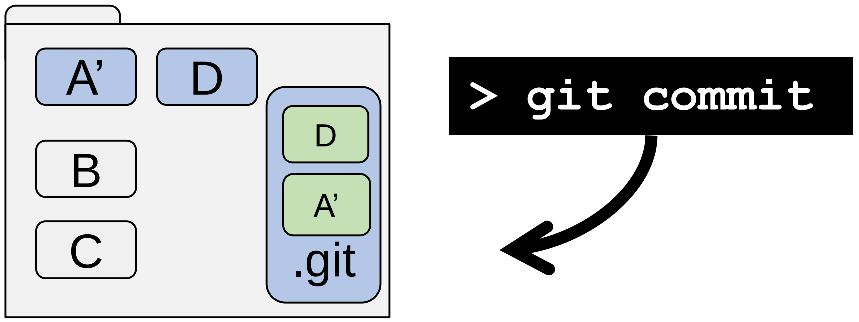


On Github:

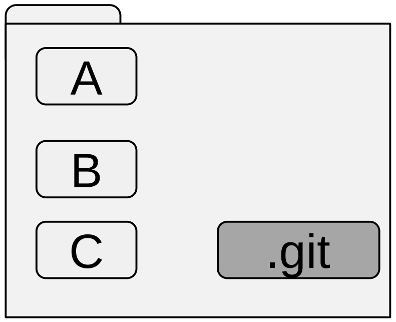


What happens when you edit files?

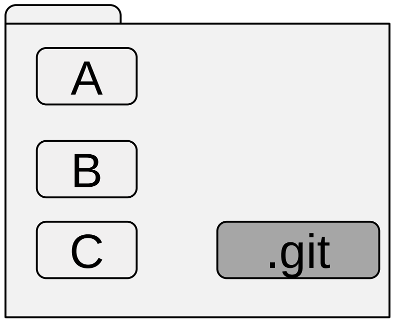
Alice tells her git to store a snapshot



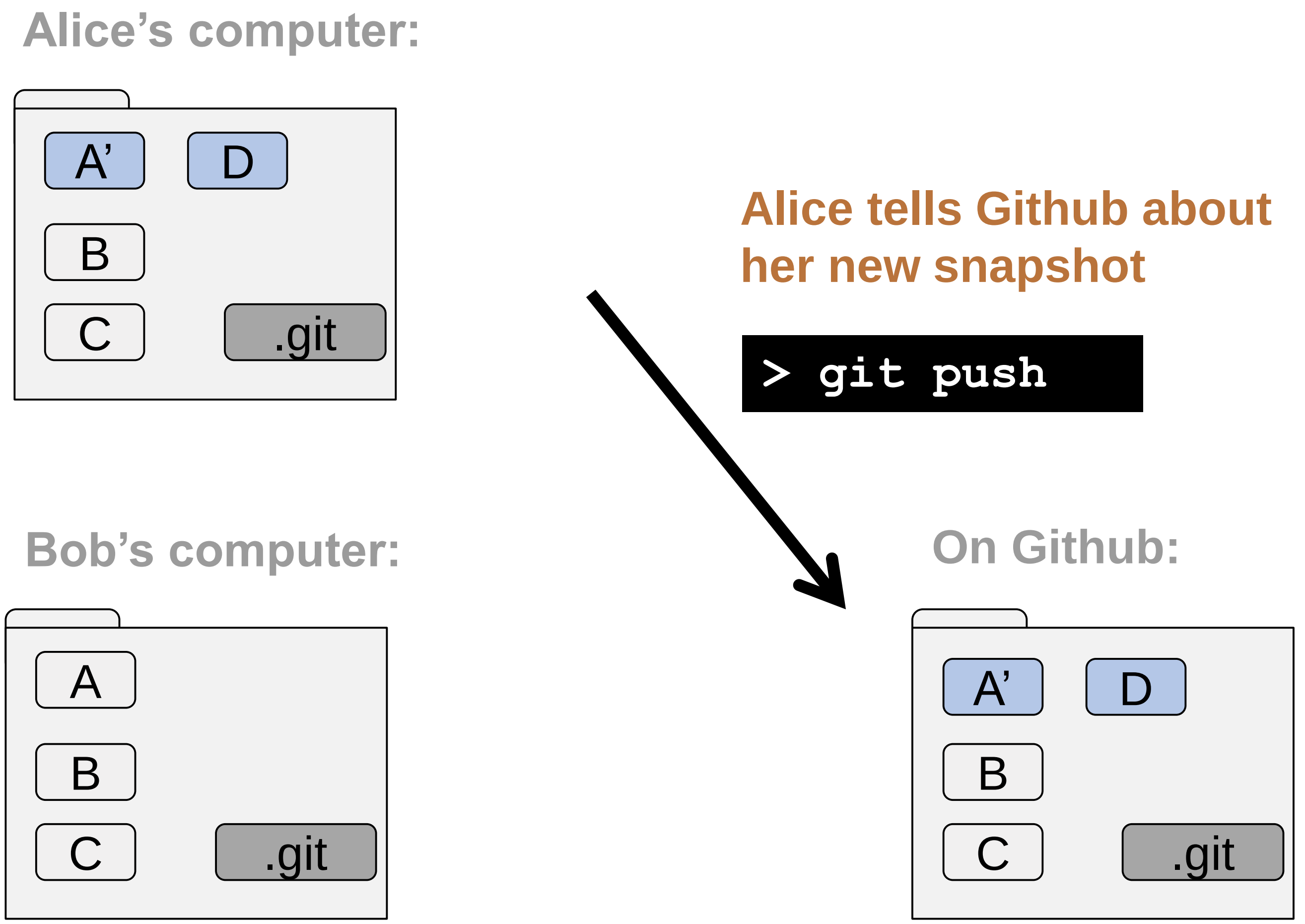
Bob’s computer:



On Github:



What happens when you edit files?

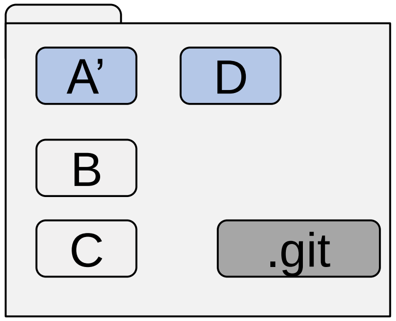


Alice tells Github about her new snapshot

```
> git push
```

What happens when you edit files?

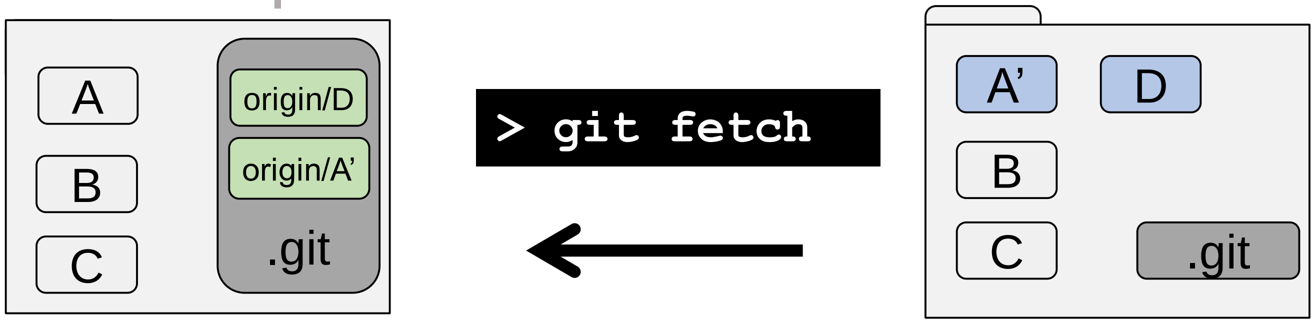
Alice's computer:



Bob asks Github for changes

Bob's computer:

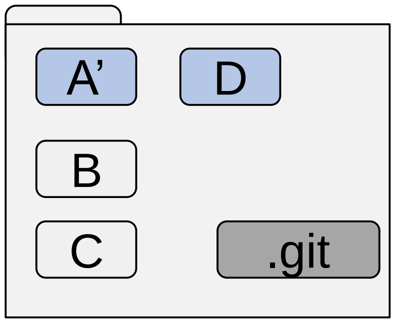
On Github:



```
> git fetch
```

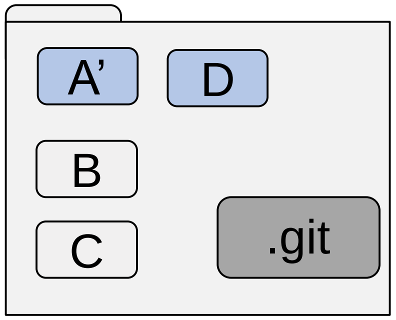
What happens when you edit files?

Alice's computer:

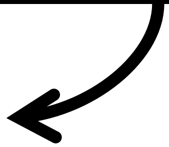


Bob merges the changes

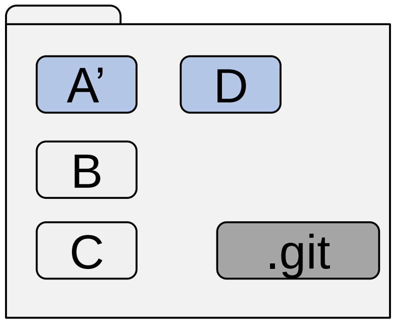
Bob's computer:



```
> git merge
origin/branch-name
```



On Github:

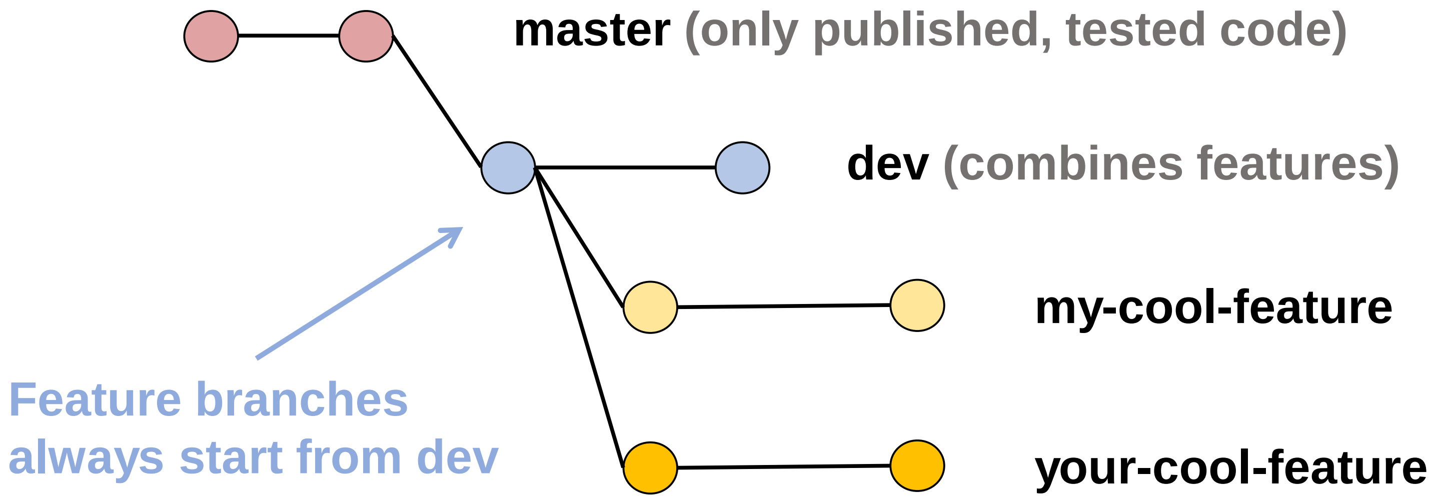


Branches

- Every commit knows which commits came before it
- Commits can only be created in the context of a branch, which points to the last commit
- Branches allow work to diverge – and then to get merged back together later.
- Merging creates a commit with 2 parents, which is added to whichever branch you have checked out when you do the merge.

Typical workflow – Feature Branches
Common variant is Git Flow

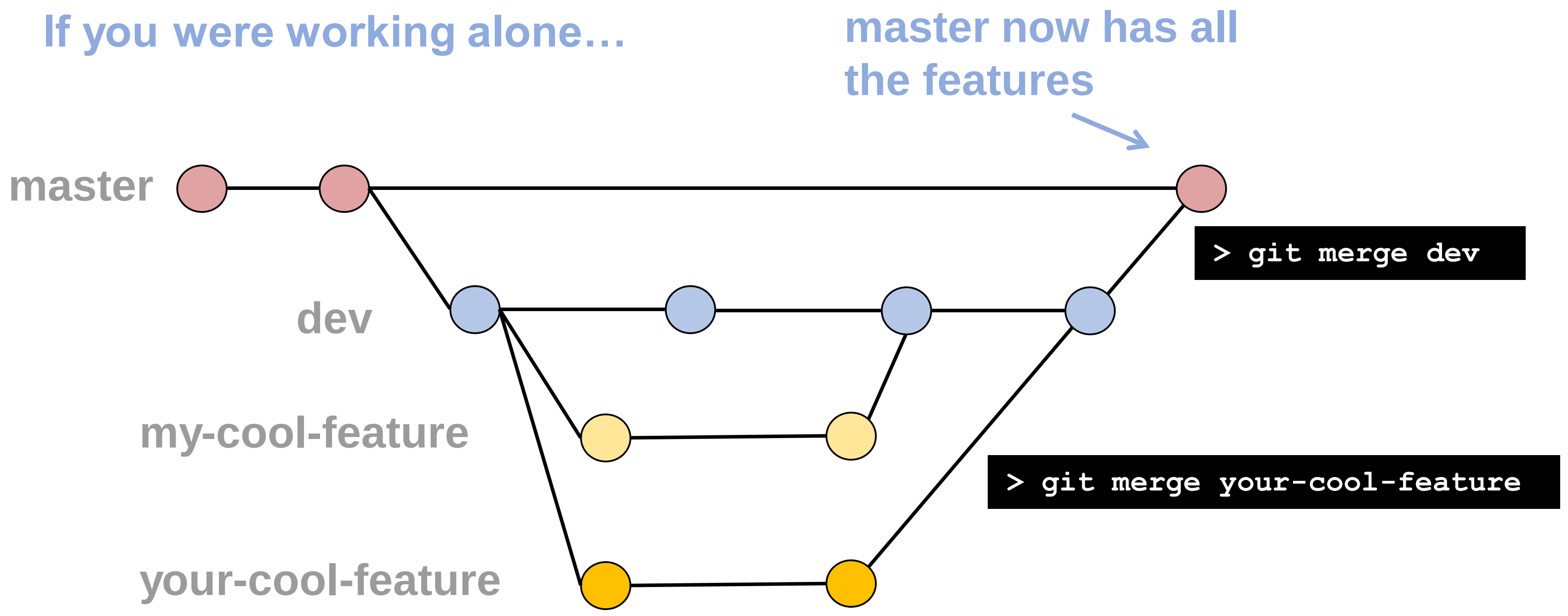
This is just one example (the one we use on #housing-insights) - how your team decides to use branches is up to you!


master (only published, tested code)
dev (combines features)
my-cool-feature
your-cool-feature
Feature branches always start from dev

Typical workflow – Feature Branches

Common variant is Git Flow

If you were working alone…



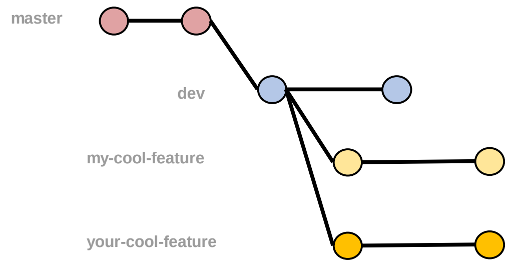
```
> git merge dev
```

```
> git merge your-cool-feature
```

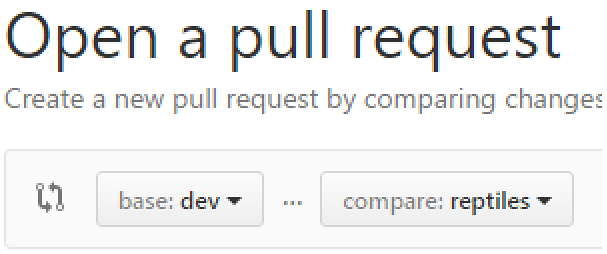
Typical workflow – Feature Branches

Common variant is Git Flow

Working in a team:

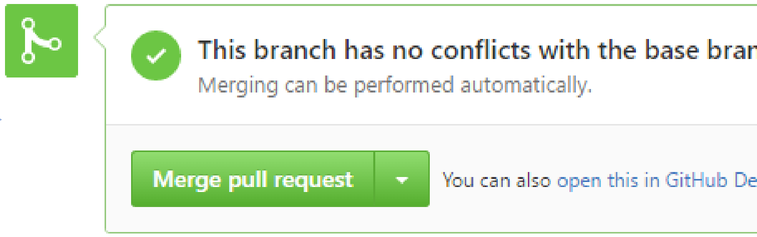


```
> git push origin my-cool-feature
```



Open a pull request

Create a new pull request by comparing changes

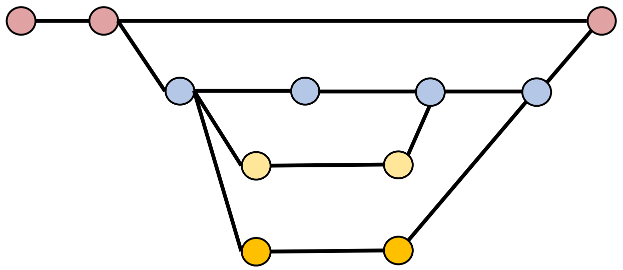


This branch has no conflicts with the base bran

Merging can be performed automatically.

You can also open this in GitHub De

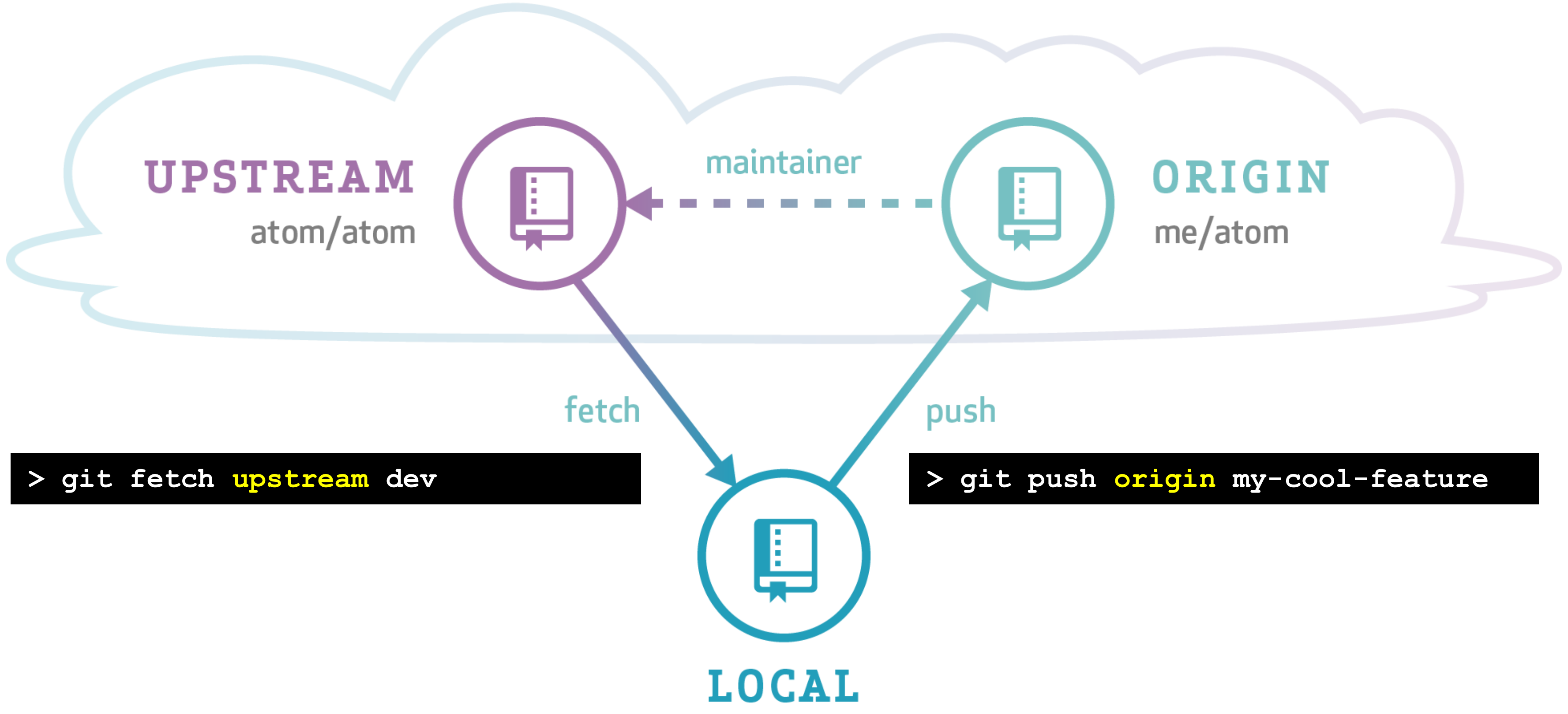
master now has all the features



```
> git fetch origin master
> git merge origin/master
```

Triangular Workflows

Every person has two copies of the repository – one on GitHub, and one on their computer



```
> git fetch upstream dev
```

```
> git push origin my-cool-feature
```

Navigating the git forest

If you were going on a hike, you’d need to know:

- Where you are
- Where you want to go
- How to get there

Where am I?

Am I in the right folder?

Am I on the right branch?

```
C:\GitRepos\git-playground> git status
On branch dev
Your branch is up-to-date with ‘origin\dev’.
nothing to commit, working tree clean
```

Are there any files that I haven’t committed yet?

Where do I want to go?

```
#What branches are available?
> git branch

#Where is each branch and commit?
> git log --oneline --graph --decorate --all
```

How do I get there?

```
# Lock the door before you leave - store your current changes
> git add --all
> git commit

# Or, bring them with you
> git stash

# Find the trailhead
> git checkout <starting branch e.g. dev>

# Start out on the trail
> git branch <my-cool-feature>

<<work on your project>>

# Save your changes before you go!
> git add --all
> git commit

# Post your selfie to Facebook (or GitHub…)
> git push origin my-cool-feature
```

Typical workflow

```
> git checkout dev
> git fetch upstream dev
> git merge upstream/dev
> git branch my-cool-feature

# write some code in my text editor

> git add --all
> git commit
#repeat above 2 as many times as needed

> git push origin my-cool-feature

# could also keep my-cool-feature checked out if doing work later
> git checkout dev
```

Demo!

- Everyone fork the repo
- I’ll demo a typical workflow (follow along if you want)
- Optional workshop after – make a pull request!

https://github.com/NealHumphrey/git-playground

Configuration for a Triangular Workflow

```
> git clone <url-of-your-fork>
> cd <repo-folder-name>
> git remote add upstream <url-of-source>
> git remote
  origin
  upstream
```

```
#Now you can use:
> git fetch upstream dev
> git push origin my-cool-feature

#Or default
```

Activity!

- Grab a handout (or in the git-playground README)
- Google ‘ASCII art’ or ‘ASCII animals’ to find things to add to the zoo.
- Don’t like animals? Make a new file and add to there instead (theme-park.txt? movie-theatre.txt? nature.txt?)