Week 4: Coding Workflows

SESSION 1: GIT'ER DONE!

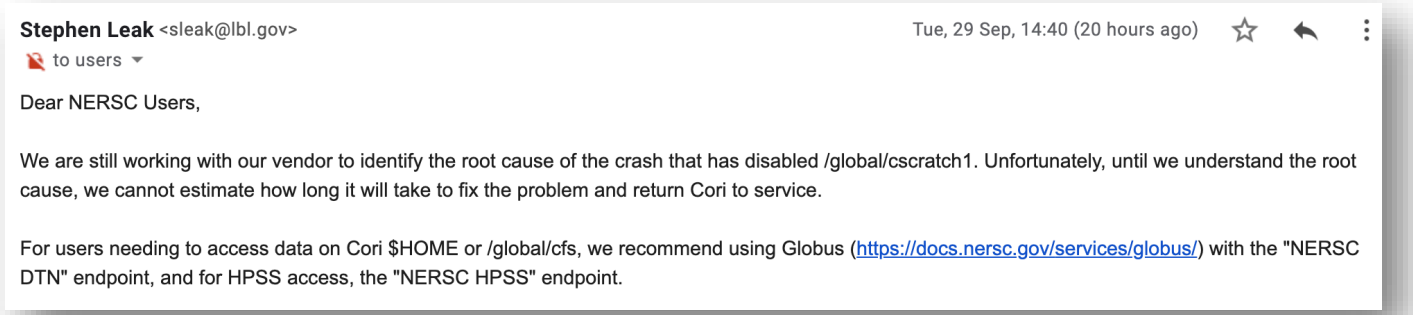
STARFISH SCHOOL 2021

Version control (but why??)

Why would you want to use Version Control?

- Multiple versions without screwing working code up
- Keep track of changes
- Use in a paper – and can cite to a specific version
- Open access and changes made by other users
- Rollback – when you screw up, you can fix!
- Collaboration

Version control



Stephen Leak <sleak@lbl.gov> Tue, 29 Sep, 14:40 (20 hours ago)

to users

Dear NERSC Users,

We are still working with our vendor to identify the root cause of the crash that has disabled /global/cscratch1. Unfortunately, until we understand the root cause, we cannot estimate how long it will take to fix the problem and return Cori to service.

For users needing to access data on Cori $HOME or /global/cfs, we recommend using Globus (https://docs.nersc.gov/services/globus/) with the "NERSC DTN" endpoint, and for HPSS access, the "NERSC HPSS" endpoint.

COOL CATCH

Losing access to your code when a bit super computer is down isn't cool at all...

Git

The basics of git

- Initializing/Cloning
- Committing
- Making a branch/Checking Out
- Pulling/Pushing
- Merging
- Conflicts

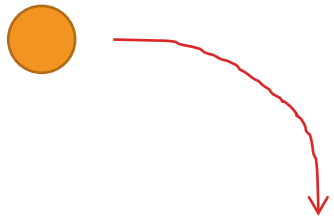
The basics of git

- Initializing/Cloning
- Committing
- Making a branch/Checking Out
- Pulling/Pushing
- Merging
- Conflicts

HOT TIP

When you start an RStudio project you can make a new git repo by default

Your Git Tree

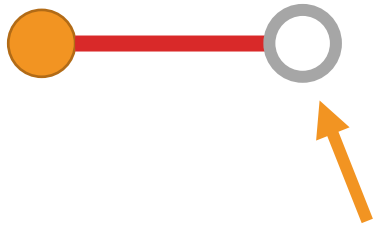


The initial commit

main.py

```
import numpy as np

# This is my program

.....



print("This is my program")
```

Your Git Tree

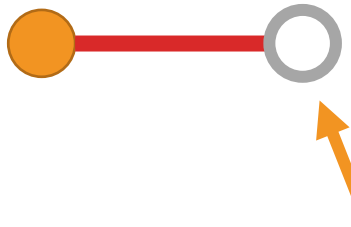


main.py

```
import numpy as np

# This is my program

.....
print("This is my new line")


print("This is my program")
```

Making a change
(not staged)

Your Git Tree

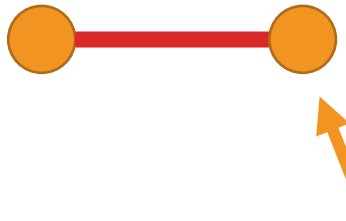


main.py

```
import numpy as np

# This is my program

.....
print("This is my new line")


print("This is my program")
```

Making a change

Your Git Tree

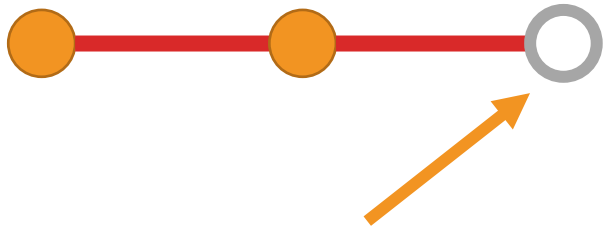


main.py

```
import numpy as np

# This is my program

.....
print("This is my new line")


print("This is my program")
```

Now committed

Your Git Tree

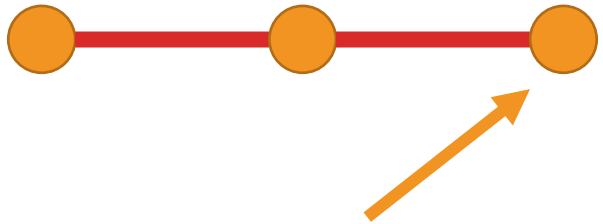


main.py

```
import numpy as np

# This is my program


.....
print("This is my new line")
print("Another new line")

print("This is my program")
```

Additional change

Your Git Tree



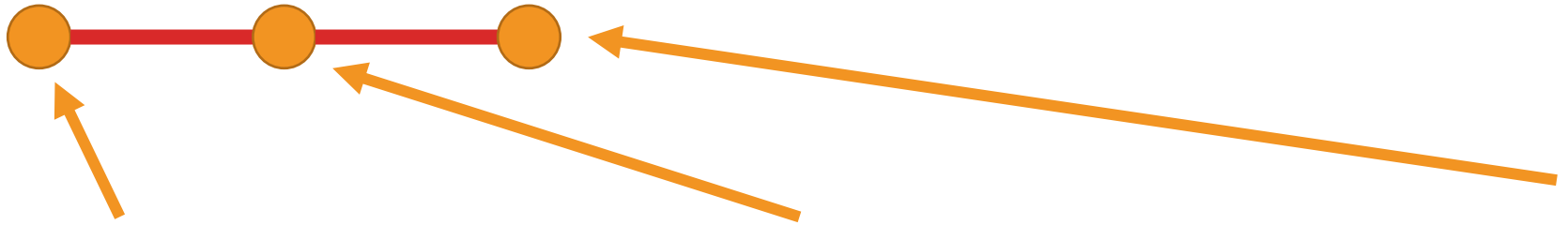
main.py

```
import numpy as np

# This is my program


.....
print("This is my new line")
print("Another new line")

print("This is my program")
```

Now committed

Your Git Tree

All of the individually committed versions are stored



main.py

```
import numpy as np

# This is my program

.....



print("This is my program")
```

main.py

```
import numpy as np

# This is my program

.....
print("This is my new line")


print("This is my program")
```

main.py

```
import numpy as np

# This is my program

.....
print("This is my new line")
print("Another new line")

print("This is my program")
```

Creating a repository

From an already existing directory:

```
git init .
```

Creates a local repository with all your local code.

Creating a repository

From an already existing directory:

```
git init .
```

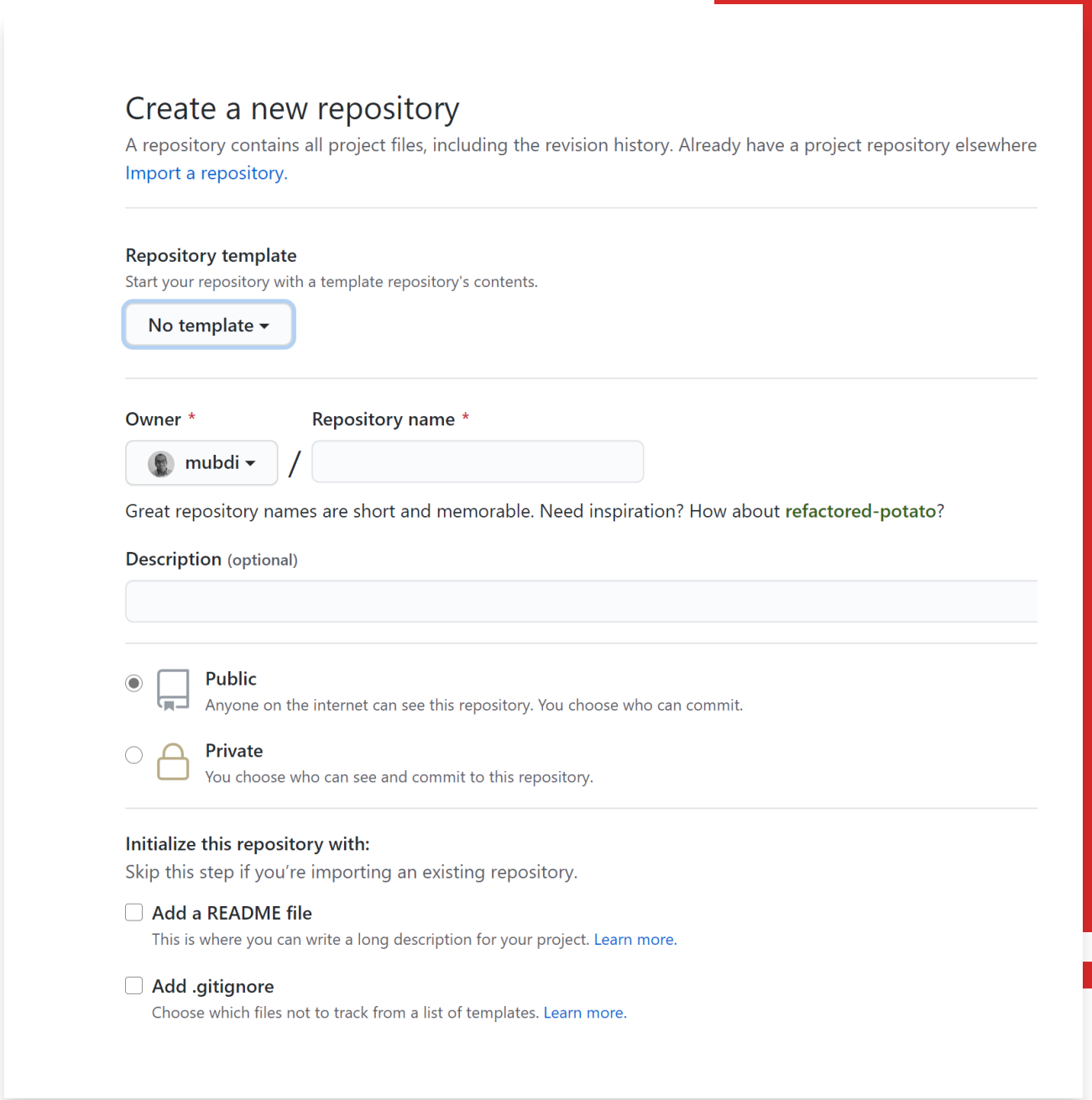
Creates a local repository with all your local code.

HOT TIP

If you have a bunch of code you've been using for years and you want to start tracking it, initialize the git repo this way

Creating a repository

Can create a new repo directly on github:

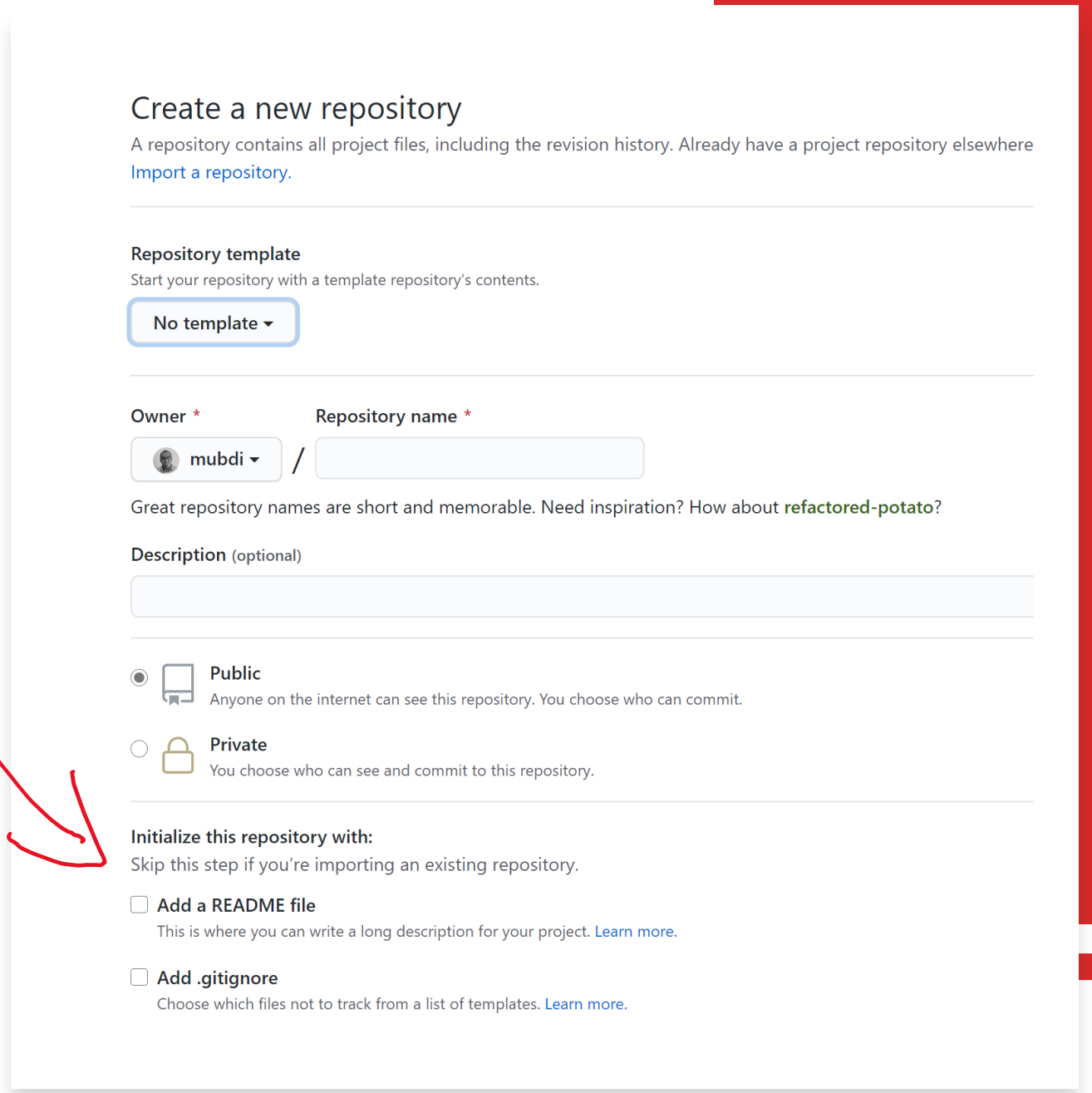


HOT TIP

If you want to create a repo on github for an existing repository (i.e. one you have already made on your computer), make sure you don't initialize it.

Creating a repository

Can create a new repo directly on github:



Cloning an existing repository

Make a “local” copy of a repository:

```
git clone <URL>
```

This makes a local copy of your “remote” repository.

Cloning an existing repository

Make a “local” copy of a repository:

```
git clone <URL>
```

This makes a local copy of your “remote” repository.

HOT TIP

You can also clone a local path or network path this way as well (i.e., if you’re storing your repo on a Department/CITA computer)

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

COOL CATCH

Git is really meant to track "small" files (think text files and code, not really large datasets.) In general, don't add large datasets to your git repo.

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

COOL CATCH

Git is really meant to track "small" files (think text files and code, not really large datasets.) In general, don't repo.

HOT TIP

If you accidentally added large files to your git repo, check out conflict resolution that we'll talk about in a bit.

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

HOT TIP

To see what state your repository is in, use the command `git status`

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

HOT TIP

Sometimes you have files in the directory. You can tell git that you never want to add them by adding them to a **.gitignore** file

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

HOTTER TIP

You don't even need to make your own .gitignore file! You can get pre-made templates for most languages here: https://github.com/github/gitignore

Sometimes you have files in the directory. You can tell git that you never want to add them by adding them to a **.gitignore** file

Committing Changes

Once you've added files, you can commit that change using

```
git commit <filename>
```

or for all files that have been changed/added:

```
git commit -a
```

which will open your default editor to add a message to describe your change.

Committing Changes

Once you've added files, you can commit that change using

```
git commit <filename>
```

or for all files that have been changed/added:

```
git commit -a
```

which will open your default editor to add a message to describe your change.

HOT TIP

You can specify your message on the command line using

```
git commit -a -m "Your Message"
```

Committing Changes

Once you've added files, you can commit that change using

```
git commit <filename>
```

or for all files that have been changed/added:

```
git commit -a
```

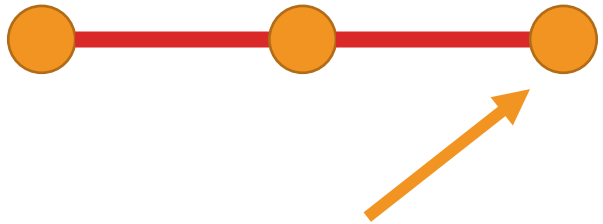
which will open your default editor to add a message to describe your change.

HOT TIP

Commit often! Just do it! Don't worry if things aren't perfect!

Git Branches

Main Branch

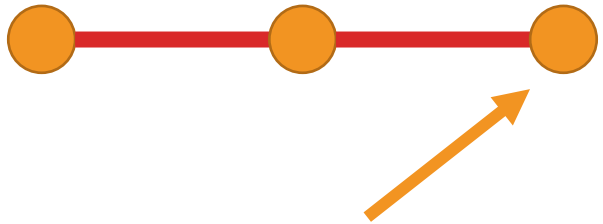


main.py

```
import numpy as np

# This is my program


.....
print("This is my new line")
print("Another new line")

print("This is my program")
```

Git Branches

Main Branch



main.py

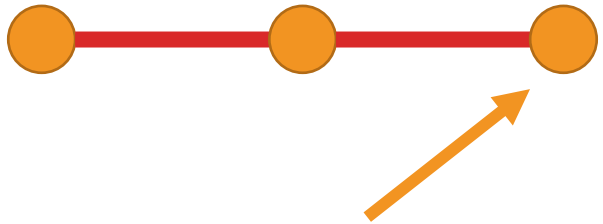
```
import numpy as np

# This is my program


.....
print("This is my new line")
print("Another new line")

print("This is my program")
```

COOL CATCH

Previously, the primary branch in git used to be referred to by the problematic term "master". You may see this terminology still every once in a while.

Git Branches

Main Branch



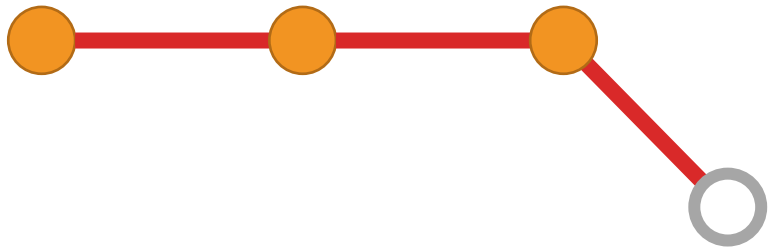
main.py

```
import numpy as np

# This is my program


.....
print("This is my new line")
print("Another new line")

print("This is my program")
```

Sometimes, you'll want to work on something separate from your working code. You can create a "branch"

Git Branches

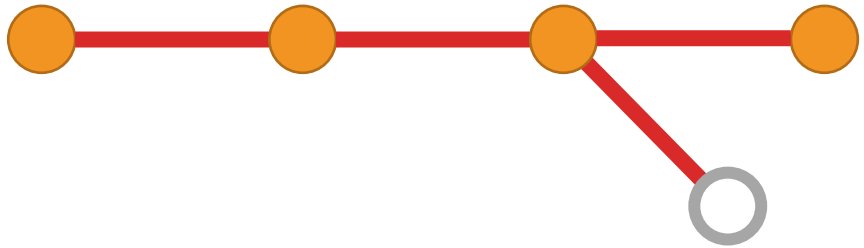
Main Branch



New Branch

Git Branches

Main Branch

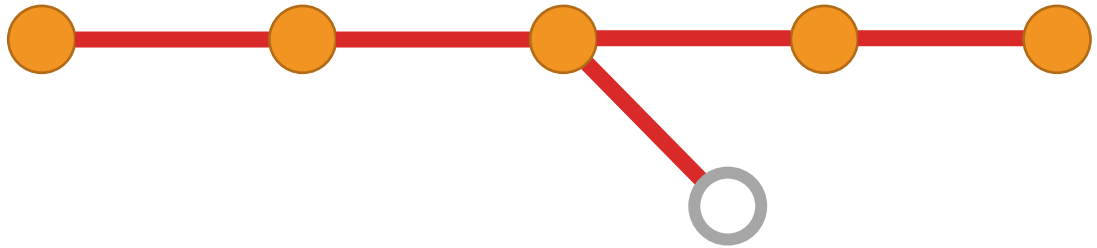


New Branch

Life continues on the main branch

Git Branches

Main Branch

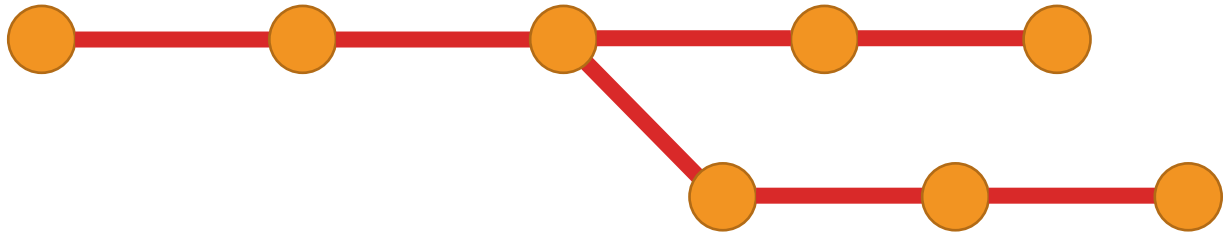


New Branch

Life continues on the main branch

Git Branches

Main Branch

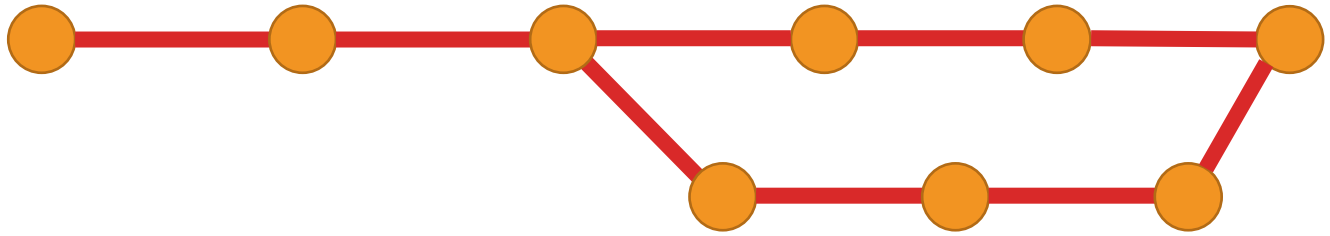


New Branch

But you can continue developing on the new branch by “checking it out” and committing as usual

Git Branches

Main Branch

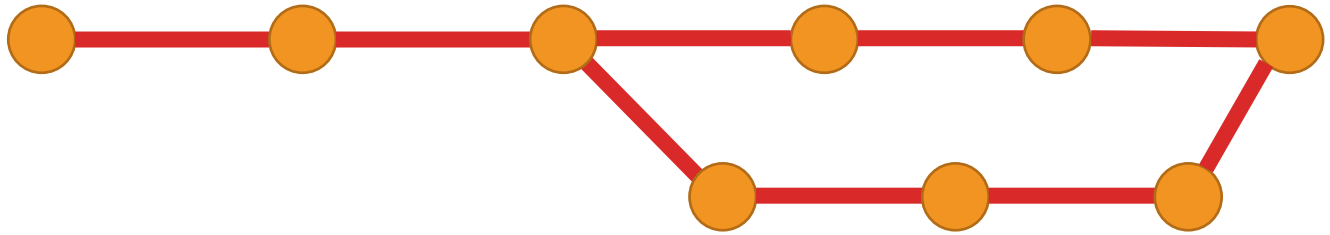


New Branch

Once you're done with your new branch, you can **merge** it back to the main branch

Git Branches

Main Branch



New Branch

Once you're done with your new branch, you can **merge** it back to the main branch

If there are changes that can't be merged together, there's a conflict!

Creating a new branch and checking it out

You can create a new branch using the command line:

```
git branch <branch_name>
```

once you've created the branch, you can move to it by checking it out:

```
git checkout <branch_name>
```

Creating a new branch and checking it out

You can create a new branch using the command line:

```
git branch <branch_name>
```

once you've created the branch, you can move to it by checking it out:

```
git checkout <branch_name>
```

HOT TIP

You can see which branch you're on with `git status`

Creating a new branch and checking it out

You can create a new branch using the command line:

```
git branch <branch_name>
```

once you've created the branch, you can move to it by checking it out:

```
git checkout <branch_name>
```

HOT TIP

You can see all of your available branches by

```
git branch -a
```

Merging back to the main

You can change back to the main branch:

```
git checkout main
```

and you can merge your old branch:

```
git merge <branch_name>
```

Break!

Mergi th

You can change

and

CONFLICT

Dealing with conflicts

When you merge, on occasion the branches will have a conflict. Git will tell you about it. What do you do?

1. Git will tell you about it, and save both versions in the same file
2. Fix the file and save it
3. Commit the new version.
4. Be proud that you defeated the conflict!

COOL CATCH

If you are using a terminal rather than an IDE you *might* get the conflict message as a vi file.

Dealing with conflicts

When you merge, on occasion the branches will have a conflict. Git will tell you about it. What do you do?

1. Git will tell you about it, and save both versions in the same file
2. Fix the file and save it
3. Commit the new version.
4. Be proud that you defeated the conflict!

Demo Time!
Real Life Git Conflict

COOL CATCH

If you are using a terminal rather than an IDE you *might* get the conflict message as a vi file.

Github and Remote Git Repositories

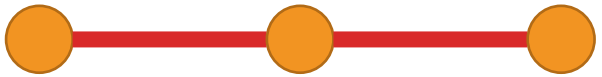
Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository

Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository



Remote Repository

Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository

Remote Repository

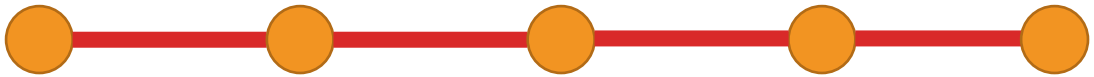
COOL CATCH

GitHub, while popular, isn't the only remote/cloud git service. You may also see people using BitBucket or GitLab, amongst others

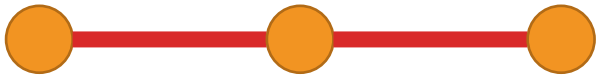
Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository



Can add new commits

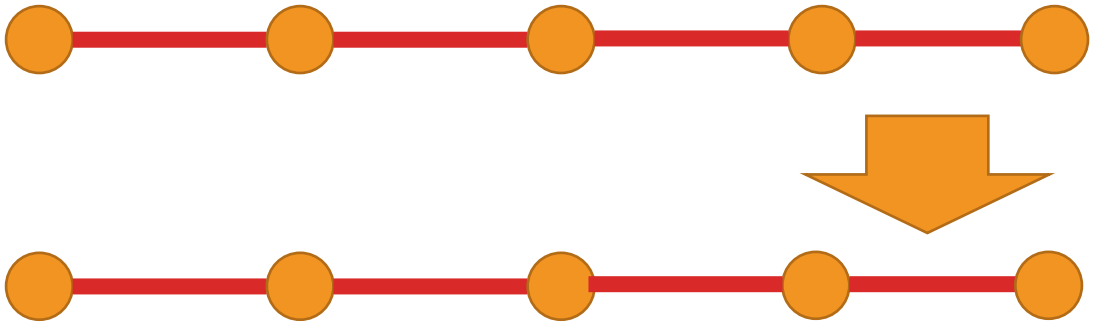


Remote Repository

Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository



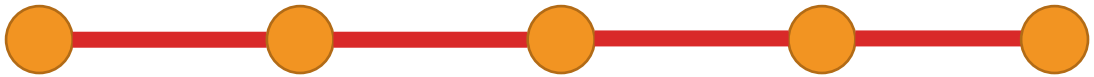
Remote Repository

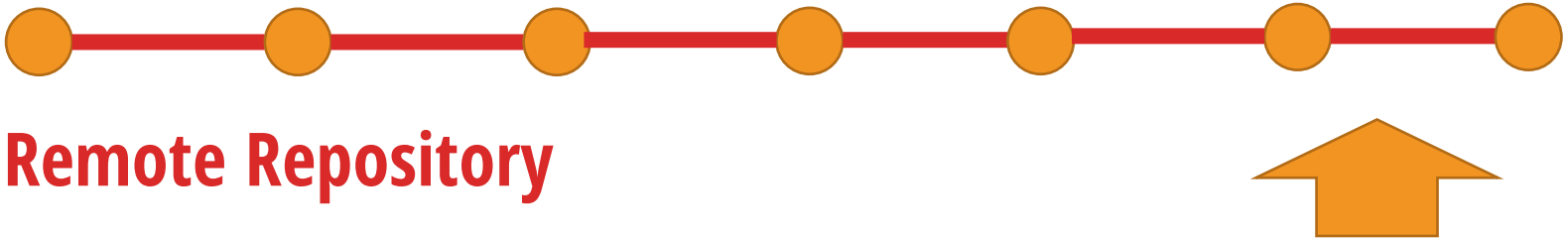
Can then `git push` them back to the remote

Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository





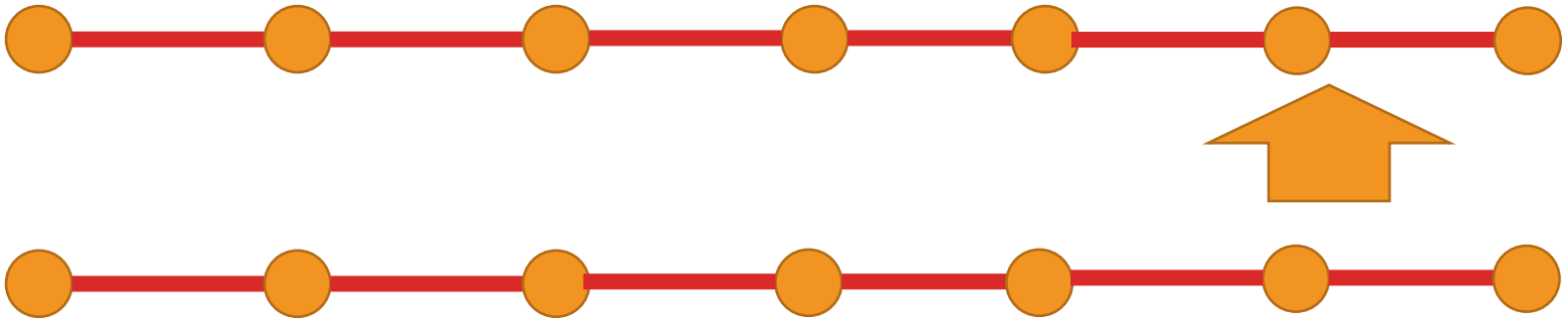
Remote Repository

Other people can push to the remote repo

Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository



Remote Repository

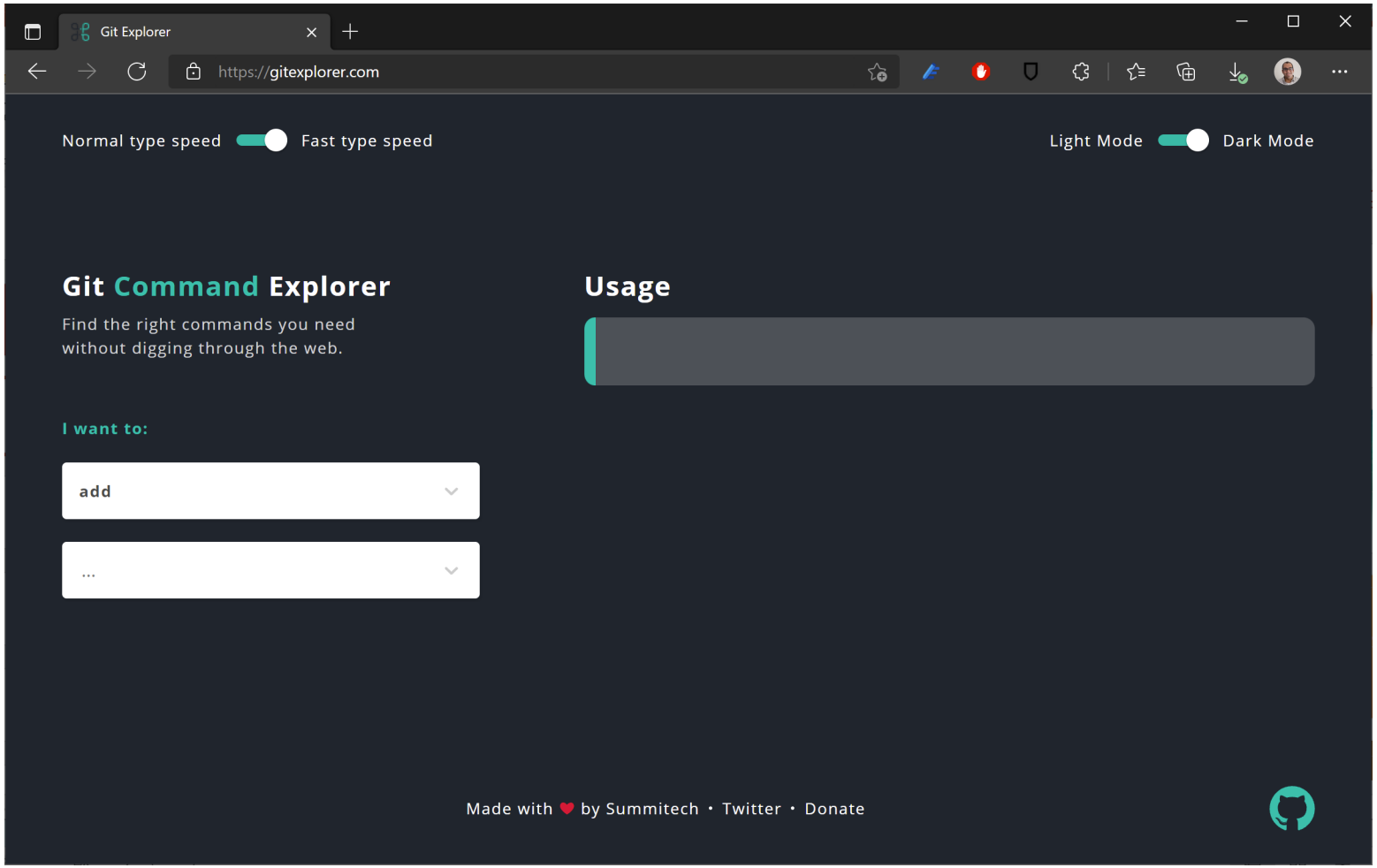
Can then **`git pull`** them back to your local repo

Useful References for Git

- Software Carpentry (intro to version control with git)
  - https://swcarpentry.github.io/git-novice/
- Atlassian Tutorials
  - https://www.atlassian.com/git/tutorials/what-is-version-control
- Git Cheat Sheet from Atlassian
  - https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- The Simple Guide
  - https://rogerdudler.github.io/git-guide/

I don’t usually remember all git commands. It’s a bit of a waste of time.

Git Command Explorer



http://gitexplorer.com

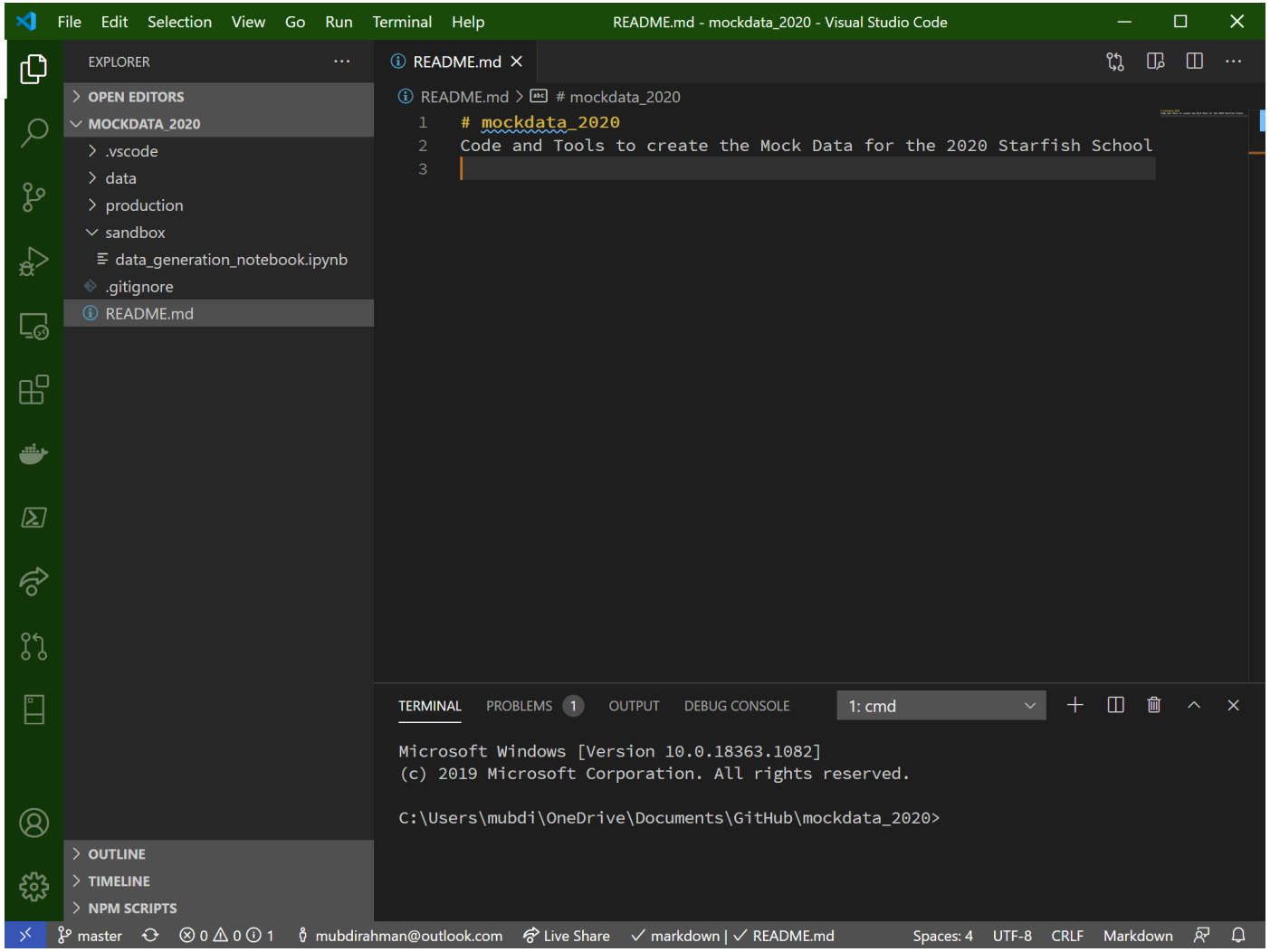
Using a Code Editor: VS Code

Made for efficient coding practices

Takes care of git right through the environment

Linting, autocomplete, error checking

Helps with debugging (come back to this place next week)



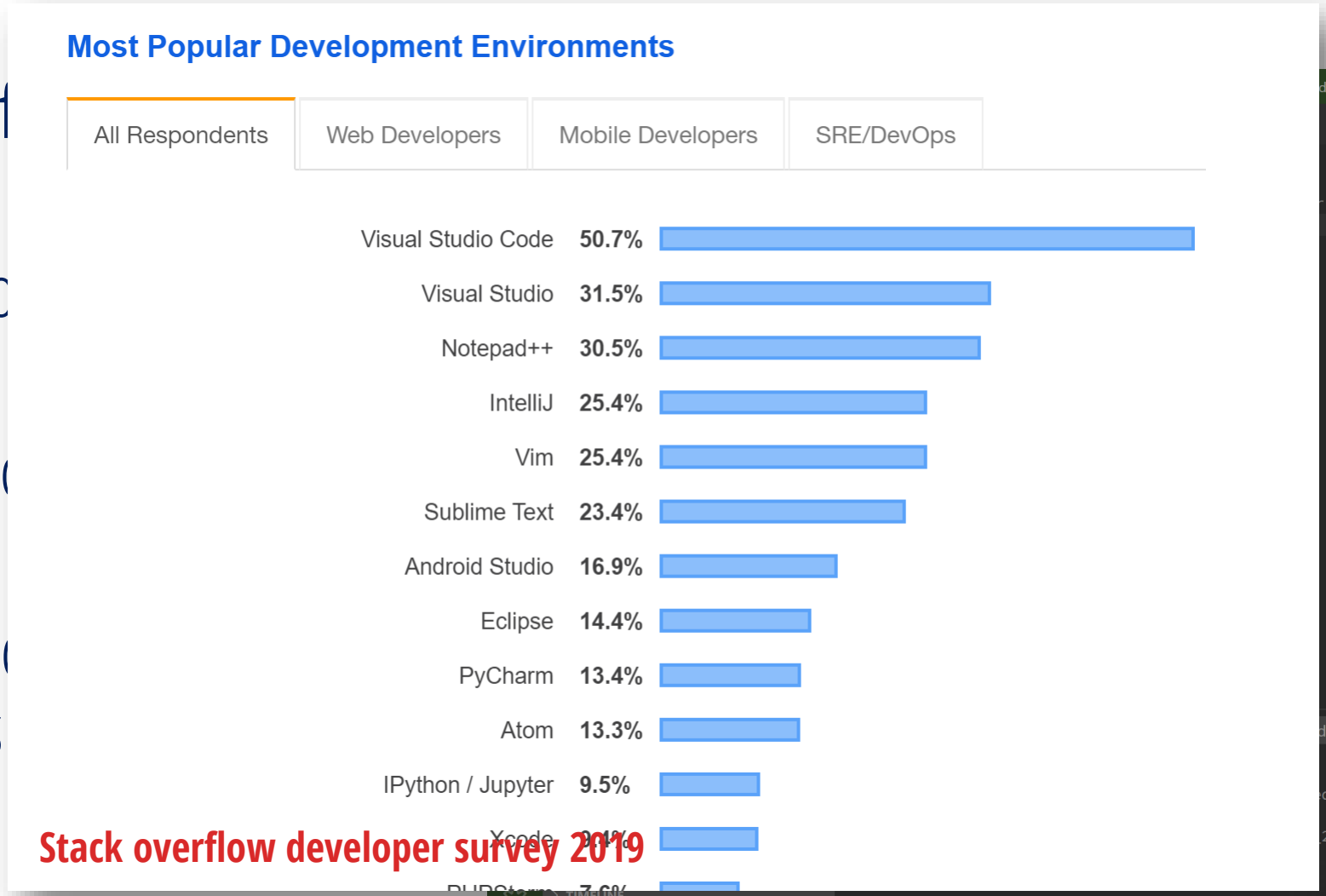
Using a Code Editor: VS Code

Made for ef
practices

Takes care c
the environ

Linting, aut
checking

Helps with
back to this



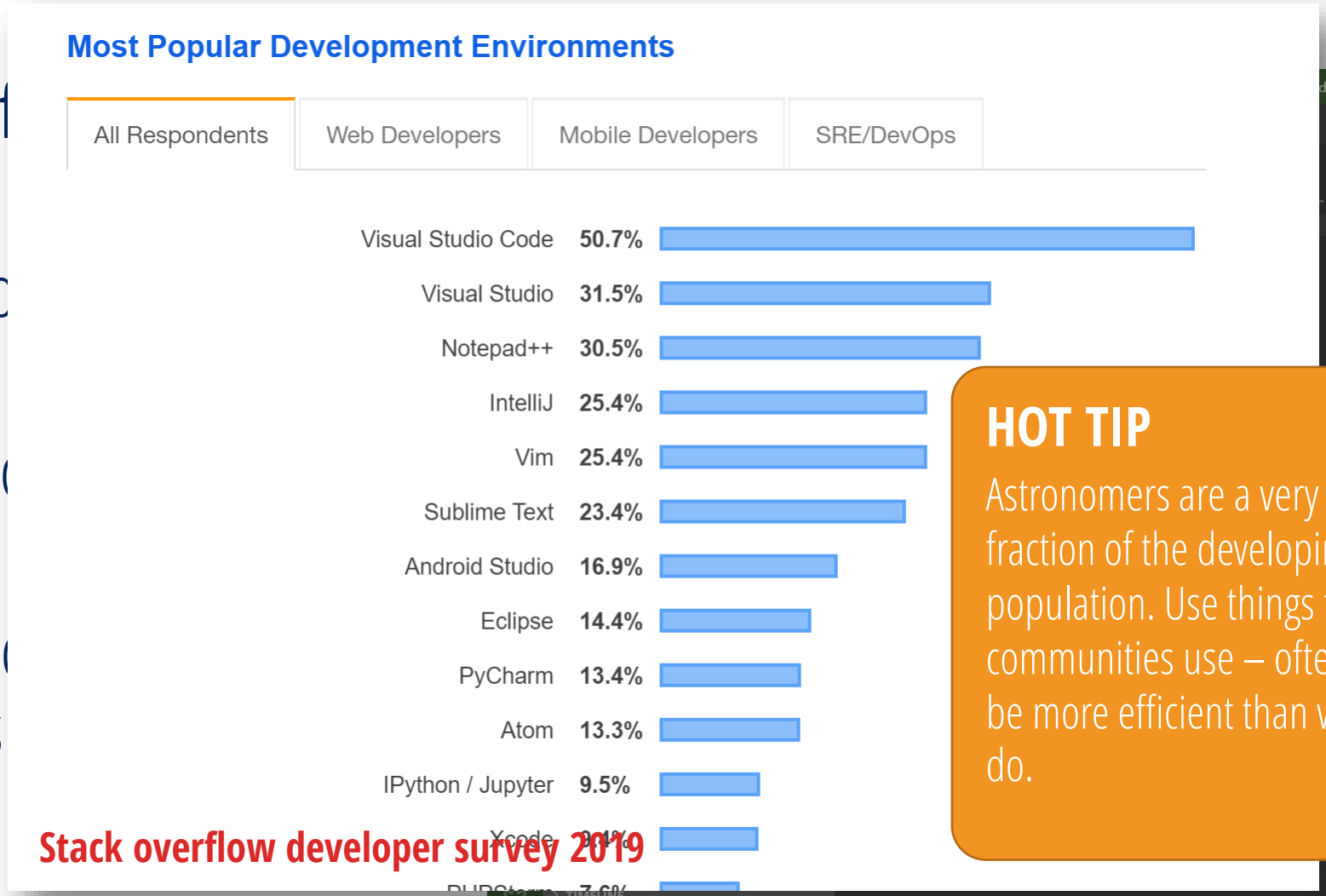
Using a Code Editor: VS Code

Made for ef practices

Takes care c the environ

Linting, aut checking

Helps with back to this



HOT TIP

Astronomers are a very small fraction of the developing population. Use things that other communities use – often, they'll be more efficient than what we do.

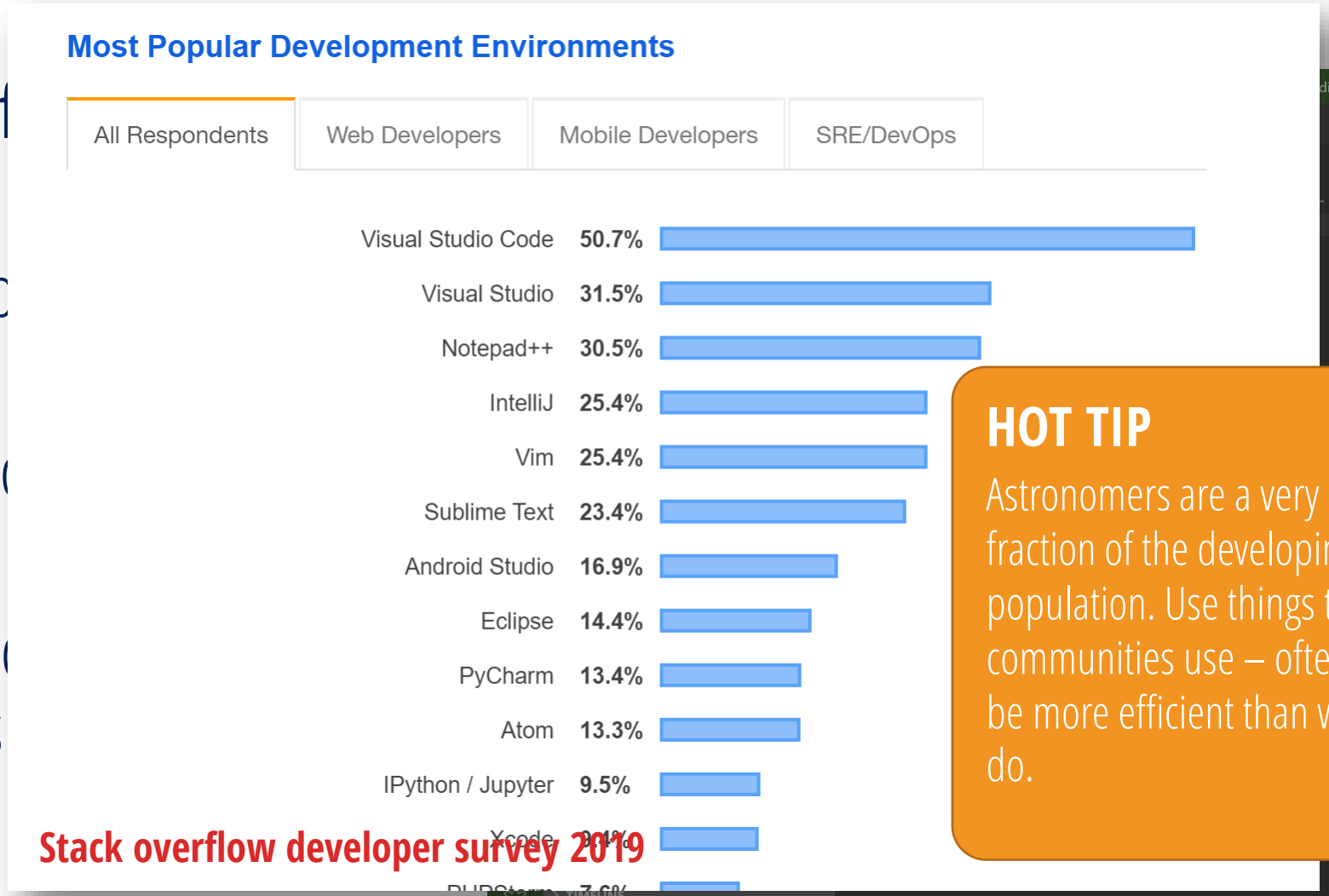
Using a Code Editor: VS Code

Made for ef practices

Takes care c the environ

Linting, aut checking

Helps with back to this



HOT TIP

Astronomers are a very small fraction of the developing population. Use things that other communities use – often, they'll be more efficient than what we do.

Demo Time!
VS Code

Exercise

You've tried these before, but let's do it again:

- Create a directory with a couple of new python files.
- Initialize a git repository within the directory
- Make changes to the files and commit them to your repository
- Make a new branch and commit a new change to your python files
- Merge the new branch down to the main branch
- Check out the git log to see all of your commits

Exercise

- Create a new uninitialized git repository on github
- Push your repository to the new repository
- Clone the remote repo to a new folder
- Create a conflict: make changes to both the old and new repository
- Fix the conflict and push everything back to the remote