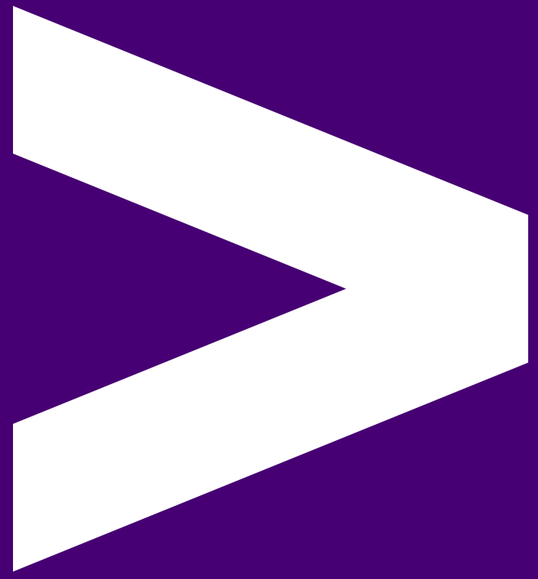
School of Tech

part of accenture

Source Control with Git



School of Tech
part of accenture

Overview

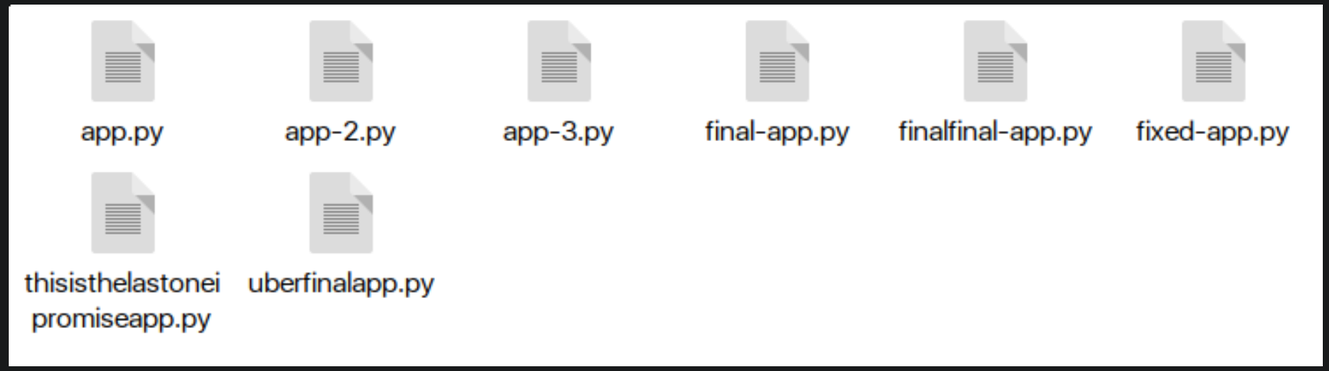
- Source Control
- Git
- Sharing Code with git
- Branching
- Pull Requests

Learning Objectives

- Describe what source control is, and why we use it
- Add and commit features / folders / files, and push to remote
- Create a Git repository
- Create branches
- Create, review, and merge pull requests

School of Tech
part of accenture

A World Without Version Control



- If we wanted to keep historical versions of a file, we would need to manually create a file for each version with a new name.
- Keeping track of this would become cumbersome.

What is Source Control?

- Source control allows you to keep track of your files and every change made to them, ever.
- It also allows you to share these files and their change history with others, so they can make changes too.

Git

- A distributed source control management system (SCM)
- It has become the most widely used tool to manage source code nowadays
- It has top level "repositories", which are like a big folder full of all your work
    - Inside these we can have many files and folders

First, some setup

> Before we get into some theory and practicals, we will check your Git software and account are set up correctly.

Exercise - check local installation

Check you already have it installed:

```
$ git --version
git version 2.24.3 (Apple Git-128)
```

Install Git using your package manager if you haven't:

```
$ sudo apt-get install git   # Unix
$ brew install git           # macOS - you need Homebrew first
```

Windows: browse to https://gitforwindows.org/

Exercise - Check local configs

Configure Git on your machine. Run each of these one line at a time. Use your own name and email:

```
git config --global user.name "Alice Bloggs"
git config --global user.email "alice.bloggs@someserver.com"
git config --global init.defaultBranch main
git config --global core.pager cat
git config --global core.editor "nano"
git config --global pull.rebase false
git config --global fetch.prune true
```

(Windows users should do this in Powershell and again in GitBash.)

School of Tech
part of accenture

Exercise - Set up a GitHub account

- Create a GitHub Account
  - See also the laptop-setup-sot session
  - Accenture staff should use their work email as the primary registered email, and verify it
- Then setup Multi-Factor Authentication (MFA), preferable with an Authenticator

Exercise - Set up an SSH key

Set up an SSH key for GitHub:

- Follow Generating a new SSH key
- Then follow Adding a new SSH key to your GitHub account
- Then follow Testing the SSH connection

Exercise - Join a GitHub organisation

GitHub organisations are shared accounts where groups or businesses can collaborate across many projects at once.

- Your instructor will invite you to an organisation for your cohort using your email address.
- Once you are invited, you will receive an email invitation, with a link in it
- You can also accept the invitation at `https://github.com/orgs/{organisation-name}/invitation` (you will need to update the link to use the organisation for your cohort)

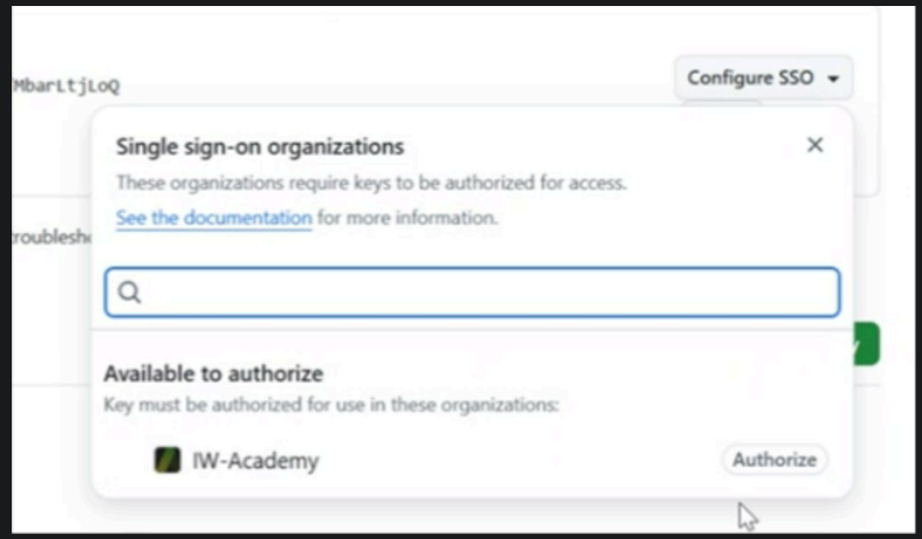
School of Tech
part of accenture

Exercise - Configure your SSH Key for SSO in the Organisation

This is required as an extra security layer - without it, your SSH key will not work in the terminal.

- Follow the steps at Authorizing SSH keys for SSO

It will look a bit like this:

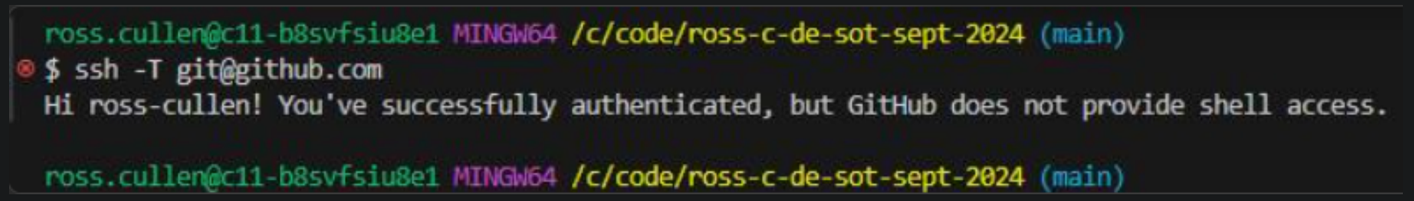


Checking we can reach GitHub on port 22

In a Bash or GitBash window run this:

```
ssh -T git@github.com
```

You should see something like this, which indicates success:



If you don't, you may ned to check your home router or the office WiFi.

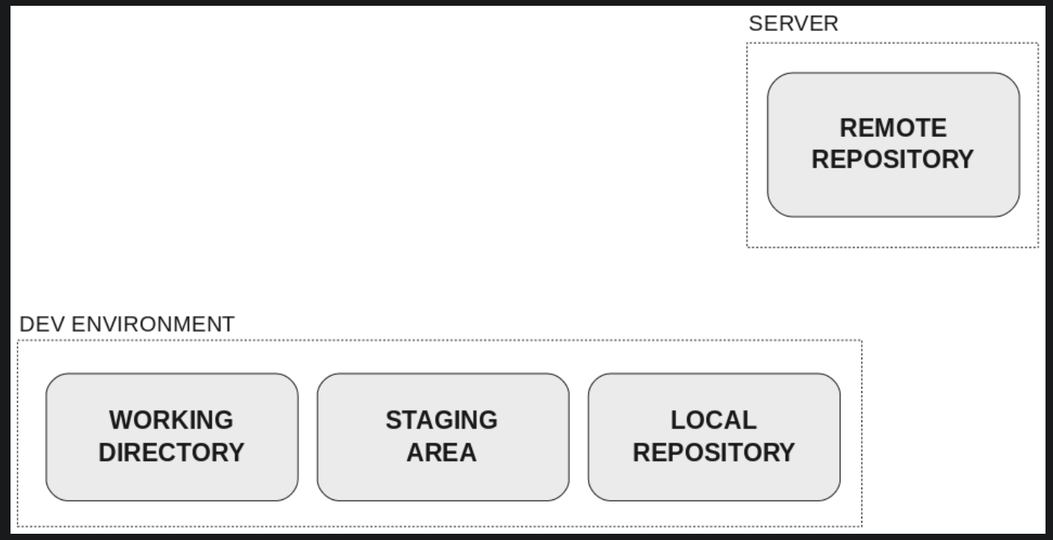
School of Tech
part of accenture

Emoji Check:

Are your local Git and your GitHub Account set up correctly now? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

Key Git components



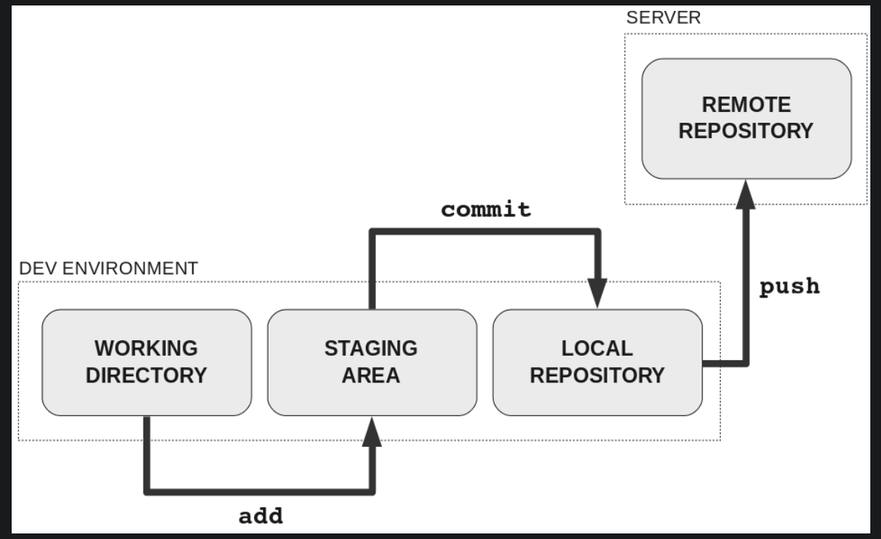
Fear not, this diagram will be explained on the next slide.

Key Git components

- Working directory: your private workspace, where changes are made. It may contain tracked and untracked files.
- Staging area: all changes you want to bundle up in the next commit (transaction) to your local repository.
- Local repository: your local version of the Git repository.

School of Tech
part of accenture

Making changes in Git

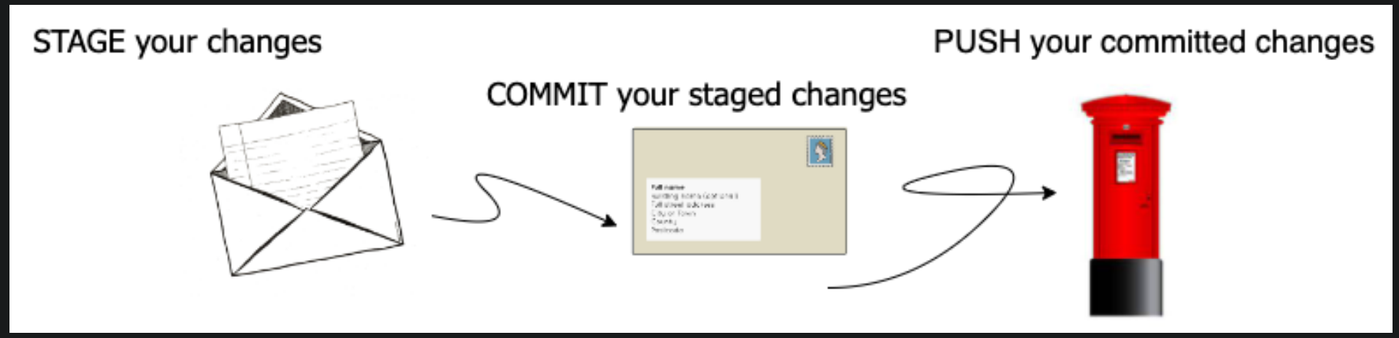


Making changes in Git

Sending your local changes to remote repo ~ Sending a letter:

- `staging` your local changes ~ Putting your letter (i.e. your local changes) in an envelope.
- `committing` your staged changes to your local repo ~ Double checking that the letter is ready to be sent, you seal the envelope, add the address and a stamp.
- `pushing` your committed changes from your local repository to the remote ~ You're ready to do the deed, taking it to the postbox and sending it.

Making changes in Git



Git in practice

> Over the next few slides we will get everyone to make a "repository", then update it with some edits and a new file, then add, commit and push your changes.

GitHub organisation

We have a shared GitHub org.

This is where you will add your own mini project repositories, and later, ones for your group projects.

> Let's take a look at it now...

- `https://github.com/{organistion-name}`

Your own GitHub repository

> A "repository" is a collection of files and folders where we can save all our work. We can also share this with others in our cohort or project team for collaboration.

On the following slides, your tasks will be to

- Create a remote repository
- Add and update a few files
- Add the files for your mini-project

School of Tech
part of accenture

Demo - New repository

- Create a new internal GitHub repository for your mini-project and session exercises
- Make sure to select our organisation as the `Owner`
- Set the `Repository name`, including your own name
  - e.g. `sue-smith-sot-de-month-year`
- Add a description
- Set it to internal
- Add a default README file
- Add a default Python .gitignore file
- Add an MIT License
- Clone this repository to your local machine

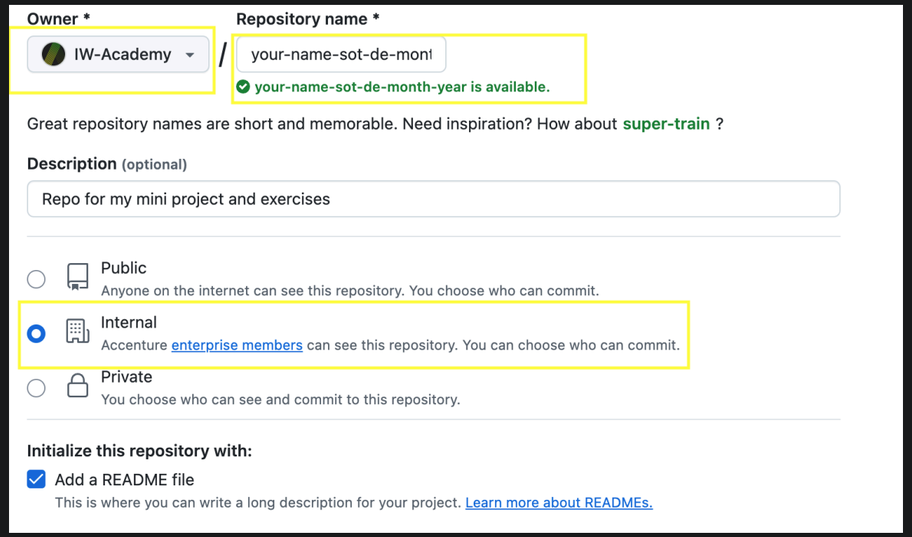
Images on next slides!

School of Tech
part of accenture

Exercise - Create a new repo

School of Tech
part of accenture

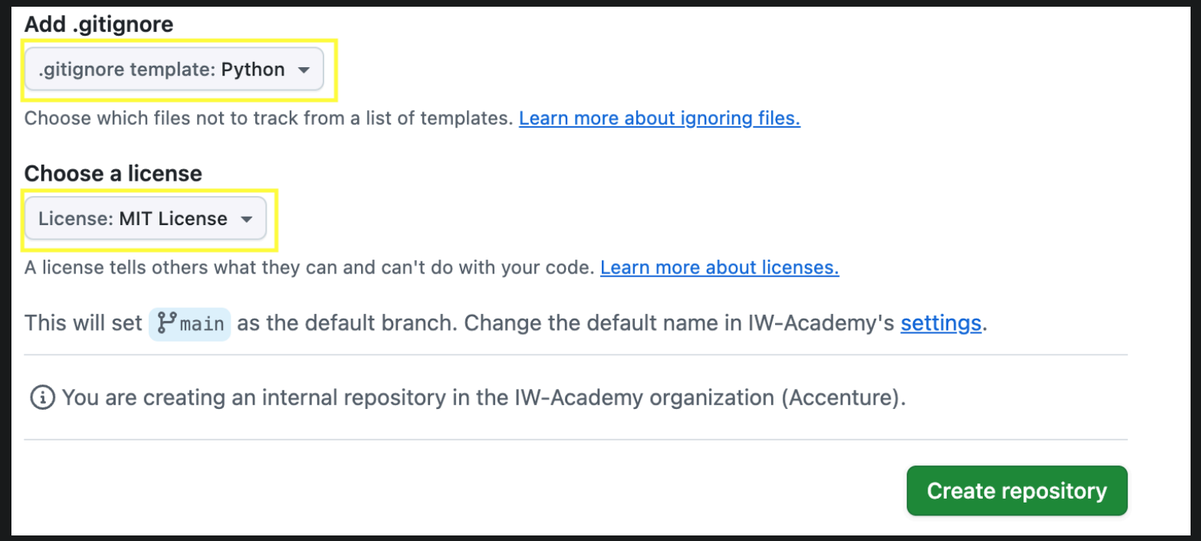
Exercise - Select owner and add a repo name



- Make sure to select our organisation as the `Owner`
- Set the `Repository name`, including your own name
  - e.g. `sue-smith-sot-de-month-year`
- Add a description
- Set it to internal
- Add a default README file

School of Tech
part of accenture

Exercise - Update repo settings

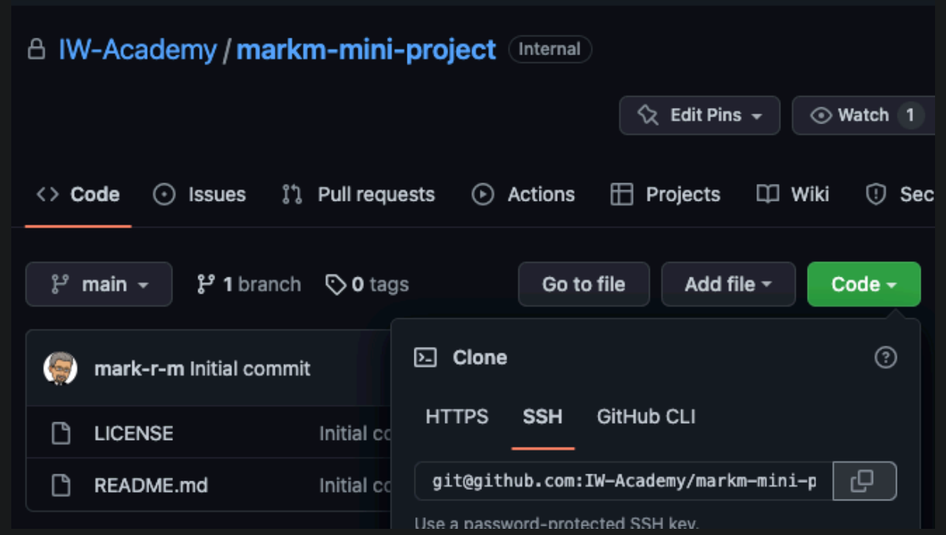


- Add a default Python `.gitignore` file
- Add an MIT License

School of Tech
part of accenture

Exercise - Clone the repo using SSH

When on the `Code` tab on your repo, select the green `Code` button and pick the SSH option to get the URL to clone

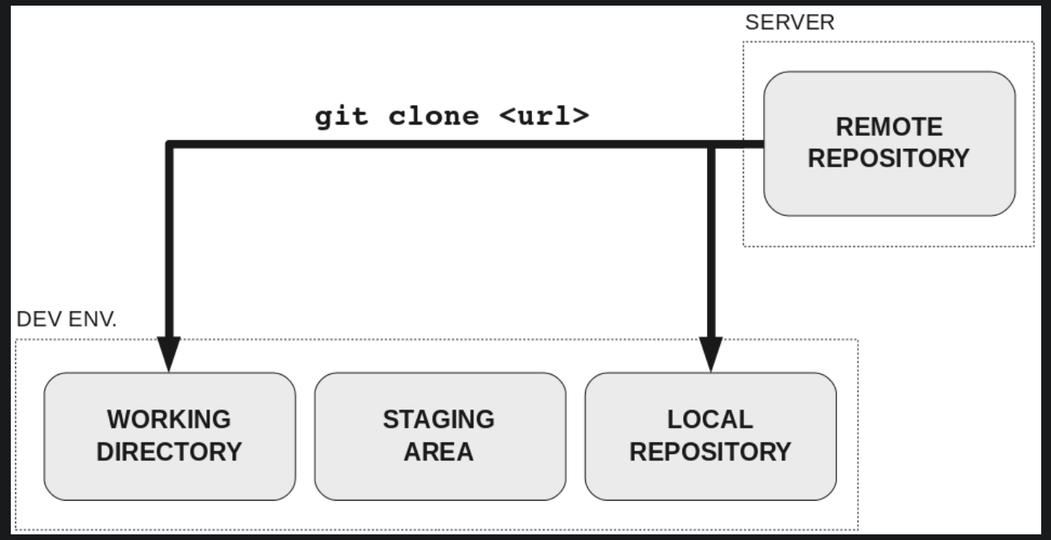



```
cd C:\sot-de-<month>-<year>
git clone <ssh-link-from-green-code-button>
```

Make sure you are in the correct folder before yo udo the "clone"!

Git clone - what did we do?

This "copies" the remote information into your machine:



School of Tech
part of accenture

Emoji Check:

Do you feel you understand making a repository and cloning it? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

Your first updates

> Your new repository is a bit like an empty bag, waiting to be filled up.

On the next few slides we will get to work on it.

git add

Tells Git to record all changes to a file. You can also use it on folders.

Behind the scenes, Git puts a copy in the staging area - think "snapshot of my changes".

```
# Stage one file
git add myfile.py
# Stage everything in a folder
git add foldername

# Stage everything in the current directory (and subdirectorie
# Not recommended!
git add .
```

Exercise - create a file and stage it

In VSCode:

- Add a new text file to your repo called `my-hobbies.txt`
- Open the file
- Add a line with a description e.g. "My hobby is reading books"

In the terminal run these:

- `git status`
- `git add my-hobbies.txt`
- `git status`

> What do the commands show you?

git commit

Bundles up all staged (`add`-ed) changes in a new commit to apply to the local repository.

```
# Write commit message in text editor
git commit
# Or inline commit message
# This is preferred
git commit -m "Update function definition"
```

Exercise - commit a file to the repo

In the terminal run these:

- `git status`
- `git commit -m "added hobbies file"`
- `git status`

> What do the commands show you?

git push

Upload any changes made in your local repository to the remote repository:

```
git push
```

Exercise - push to the remote repo

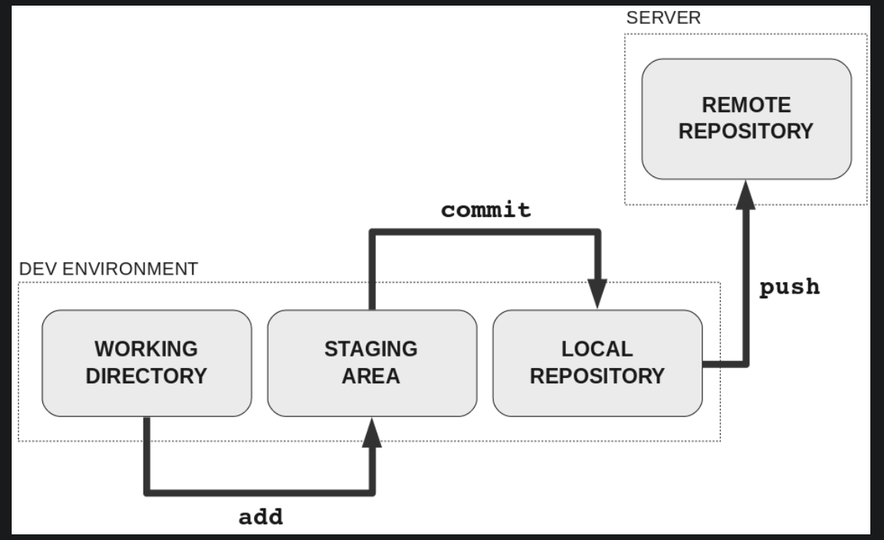
In the terminal run these:

- `git status`
- `git push`
- `git status`

> What do the commands show you?

git add, commit and push

Let's revisit that sequence:



Emoji Check:

Do you feel you understand the git sequence of `add`, `commit`, and `push`?
Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively

When should I commit?

Commits should:

- Pass all tests, and not have any known bugs to be fixed in a future commit
- Small self-contained changes that work
- Have short and meaningful commit messages
- Act as a save point for your code changes

Example commit message: change function to return integer value rather than string value

One important message about commits

NEVER commit sensitive information! This includes:

- Passwords
- Keys
- Private information about a person or entity

As professional software engineers, we can use handy tools like git-secrets to catch when we might accidentally commit sensitive information.

Check out the academy-git-secrets-example repo for a demonstration of how this tool can be used in your projects.

When should I push?

> "Frequently"

- Code that isn't pushed exists only on your machine. So while it is a "save point" per commit, it's not backed up, and your team can't see it or use it. And if your machine crashes, it's still lost!
- Some teams like to to push with every commit.
- At least - do it when you leave your machine, like lunchtime and at the end of the day.

.gitignore

- We can use a special file called a `.gitignore` to tell git not to track certain files
- This is a good way to protect sensitive information
- It can also come in handy for lots of other things, e.g. local config files, or very large package-related directories
- We can use file names like `bad-file.txt`
- Or folder names like `passwords-here/`
- And use wildcards like `personal-data-*.*`

Demo: .gitignore

> The instructor can show you how this works:

- Demonstrate adding a file you don't want like "personal-data.txt"
- then ignore it
- then add/commit/push the ignore file
- then show the "personal-data.txt" is not in the remote repository
- then show further edits to the bad file are also ignored

School of Tech
part of accenture

Quiz Time! 🤓

Which git command would you use to apply your locally staged changes to your local repository?

1. `git status`
2. `git add`
3. `git commit`
4. `git push`

Answer: `3`

Which git command would you use to add new files and folders to the staging area?

1. `git status`
2. `git add`
3. `git commit`
4. `git push`

Answer: `2`

Emoji Check:

Do you feel you understand git a bit better now? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

Sharing Code

Not only for emergencies - we can use Git to share code between a Team of people

Sharing Git repositories

- Git is distributed, which means most of the changes happen in people's individual local repositories (repos) rather than in a central location
- Everyone's local repos have all information about the repo
- There is no 'one and only' source of repo information

Sharing Git repositories

- Because Git is distributed, all information about a repository is copied every time it is cloned.
- You upload your local changes to remote repositories hosted on private or public servers, and also download changes made by other people into yours.

Sharing Git repositories

- Git repositories can be hosted in a private Git server, although this is something few people do.
- Most people prefer to utilise code hosting services like BitBucket, SourceForge, GitLab or GitHub.

School of Tech
part of accenture

GitHub

- GitHub is the most popular code hosting website
- GitHub is free and offers loads of tools to aid coders:
  - Code hosting
  - Code discovery and search
  - Bug reports
  - Project task boards
  - Project wikis

Git as a shared resource

Git repositories (repos) are a great way to share data, files and projects between a team.

On the next few slides we'll uses a shared repo and all put edits into it!

> Instructor to share the SSH clone link for the shared repo they have set up

School of Tech
part of accenture

Exercise - Clone a shared repository

- Follow the instructor!
- Navigate to your programme directory (wherever you are storing your files for this programme) and clone the repository (the instructor will give you the correct repository link):

```
cd C:\sot-de-<month>-<year>
git clone <repo-ssh-link>
```

- This should create a new directory that you can `cd` into

Lets make a change

- In this repo make a new text file and call it after your name (e.g. `linzi-c.txt`) and write your name in there
- Once you are happy let's get this added, committed and pushed
- Who can remember the commands we need to do these three things?

Push those changes!

- `git add <my-name.txt>`
- `git commit -m "Added a new file called <my name>"`
- `git push`

What's changed?

- Once you have pushed your changes, have a look at the remote repo and see if you can find the file you just pushed
- `https://github.com/{organisation-name}/{repo-name}`
- Hopefully you should see your file up there (along with some others...?)

School of Tech
part of accenture

Emoji Check:

Do you feel you understand how to push changes now? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

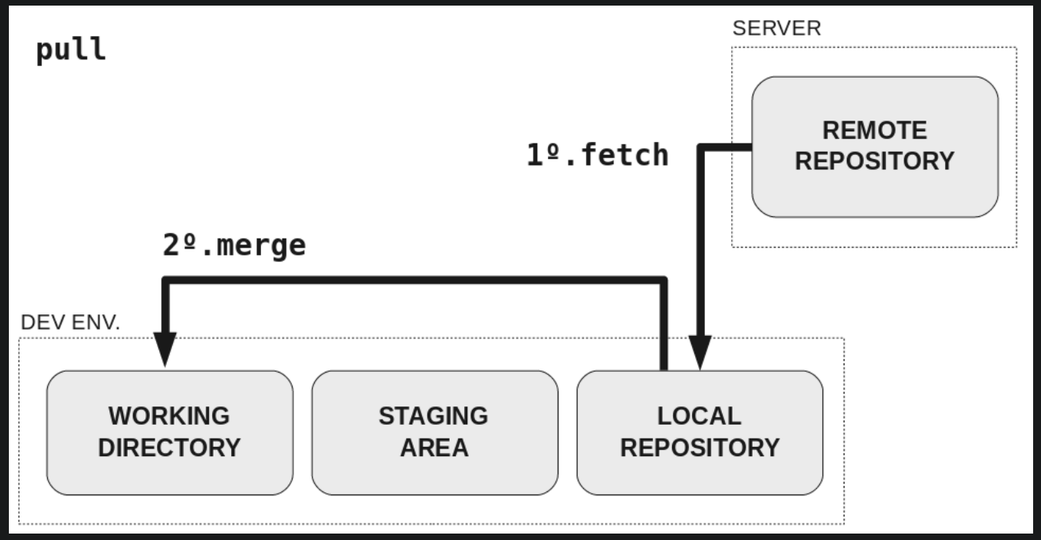
Local updates

What if other folks have also sent work to the remote sever, and you don't have those yet?

- I can't see those new files on my machine
- This is because the remote has been updated, but your local repo is still behind

Local Updates - Pull

- To update the files locally you can use the `git pull` command to apply any changes to your local files that other people have made
  - This only updates your current branch - more on this later



Conflicts (1/3)

- Lets try and edit a file that everyone has access to
- In the repository you should see a file called `your-hobbies.md`
- Now write something about yourself in the designated section and then `git add, git commit and git push` that code

Conflicts (2/3)

- Once you have done this, try and pull down the changes. There's a chance you see the following message:

```
→  NJA-source-control-2021 git:(main) git pull
Auto-merging about-me.txt
CONFLICT (content): Merge conflict in about-me.txt
Automatic merge failed; fix conflicts and then commit the result.
→  NJA-source-control-2021 git:(main) ✗
```

- This is known as a conflict, where git was unable to merge changes because two separate sources have changed the same line

School of Tech
part of accenture

Conflicts (3/3)

- If you open up vscode again, if you navigate to the offending file then you will see something funky like this:

```
Add your name and a fun fact about yourself between the lines:


_____________________________________________
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes | Start Live Share Sessi·
<<<<<<< HEAD (Current Change)
I sure hope this doesnt conflict
=======
Tom C - I like Leeds United. They don't like me, they made me sad.
>>>>>>> 49152ffdbf2e2e20b0f6d275da77498032ede6f5 (Incoming Change)
_____________________________________________
```

- This will let you know what your change is (current change) and what the conflicting change in the repo is (incoming change)

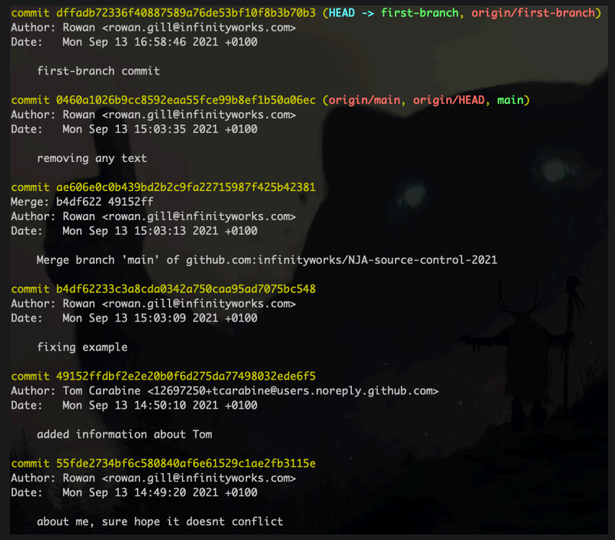
Fixing mistakes

- The simplest way to fix a mistake is to change them back in your local directory, and then make a new commit
- There are other ways to achieve this, but they involve potentially re-writing git history and need to be treated with caution
- For now, let's just delete our changes and make new commits

School of Tech
part of accenture

Viewing our previous changes

You can use `git log` to take a look at the commit history for a repository.



School of Tech
part of accenture

Emoji Check:

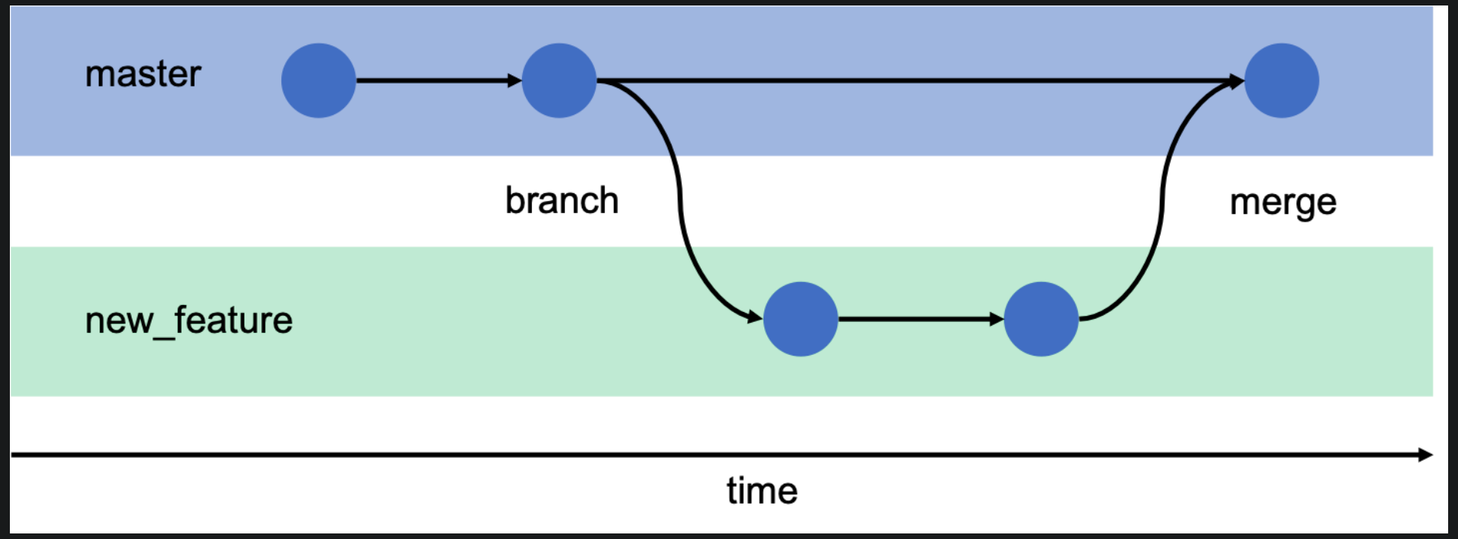
Do you feel you understand conflicts? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

Branches

- A branch is an alternative development path in your repository
- Branches are usually created from `main`, but they can branch from anywhere (like a tree)
- Branches allow us to commit our work-in-progress, without affecting the original branch

What is a branch?



Creating and checking out branches

```
git branch <new_branch>
git checkout <new_branch>
```

```
# You can create and switch to a branch in one go
git checkout -b <new_branch>
```

Pushing branches

`--set-upstream` creates an upstream branch in the remote that matches our local branch of the same name.

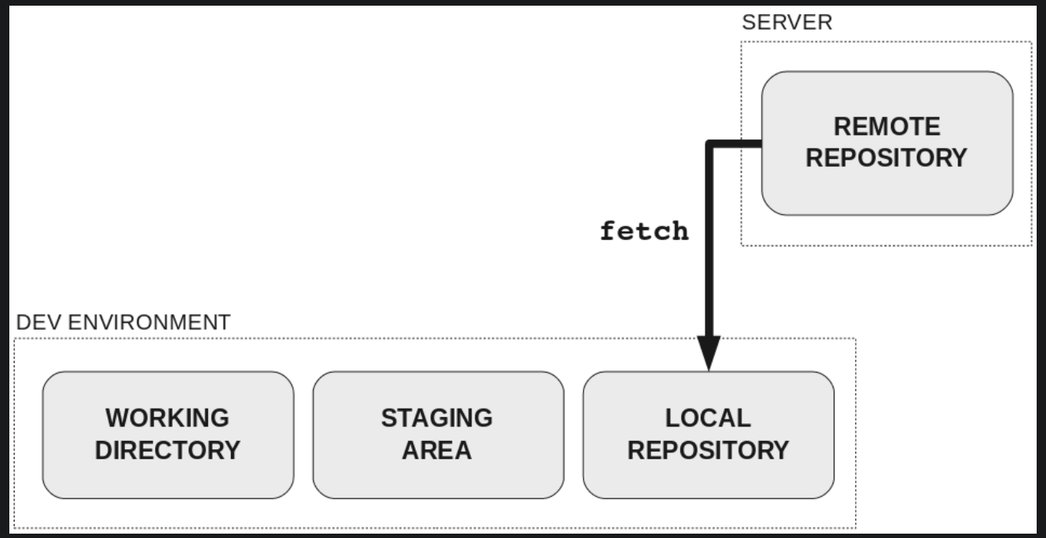
```
git push --set-upstream origin <new_branch>
```

(This is not necessary if pushing to `main`, as that's the default)

School of Tech
part of accenture

Local Updates - Fetch

- To get your local machine up to date with all the remote branch names, we use `git fetch`
- There is an extra `--prune` option to make sure our list of branches has old ones removed
- Because of the global setting we used previously, the `--prune` is now your default



Scenario

Alice has written some code on a feature branch and pushed her branch up to the remote. You want to have a look at her code and try it out for yourself. How can you do this?

1. `git branch alices-branch`
2. `git fetch && git branch alices-branch`
3. `git checkout alices-branch`
4. `git fetch && git checkout alices-branch`

Answer: `4`

School of Tech
part of accenture

Exercise - Our first branch

> Do this in the demo shared repo, not your personal one

- So we've seen how to make and checkout a branch
- Create a new unique branch with your name (eg `linzi-first-branch`)
    - and change a line in the file you made earlier `yourname.txt`
- When you are happy commit and push that branch
- Open GitHub in the browser and find it

School of Tech
part of accenture

Emoji Check:

Do you feel you understand branches? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

What now?

So we have a branch pushed up to the remote repository. We want to merge those changes to our `main` branch. How do we do this?

- There are a few ways to achieve this
- One of the most common is to raise a pull request

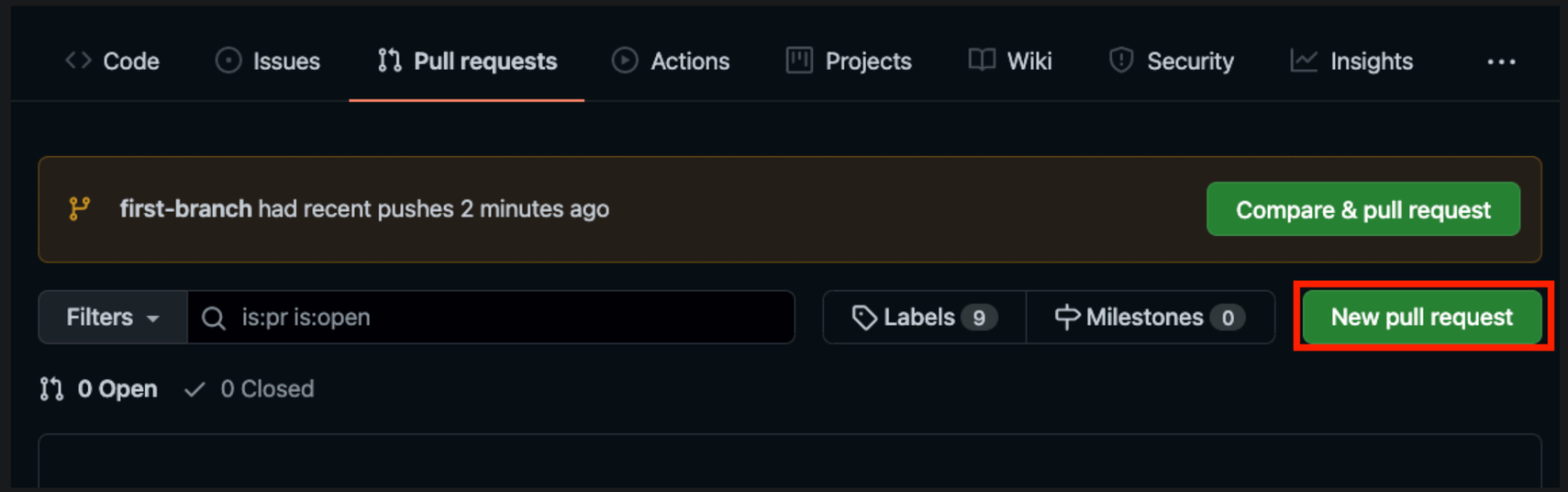
Pull Requests

- Pull requests allow us to control changes going into our `main` branch
- They can be made to need passing test suites before they can be merged, creating a safety net for your code
- Provides an opportunity for review pairing and knowledge share

School of Tech
part of accenture

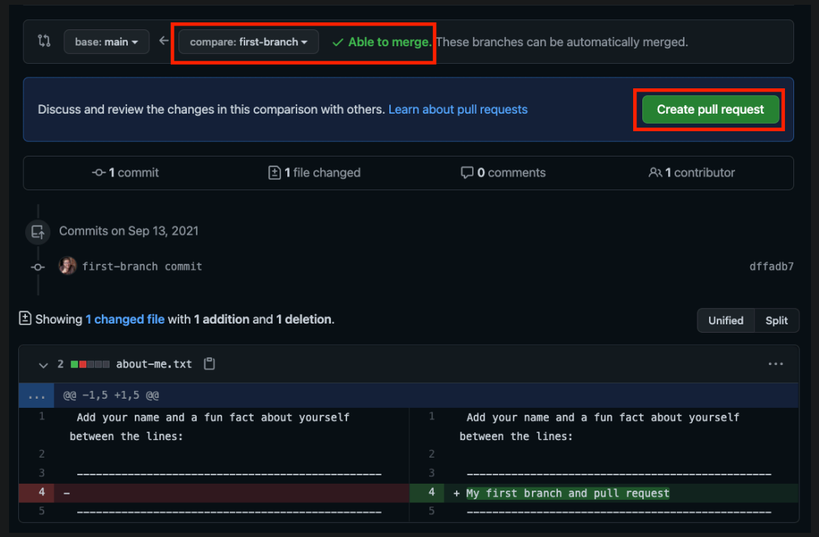
Raising a Pull Request

- To raise a pull request head to the github page for the repository and select the pull requests tab



Setting up a PR

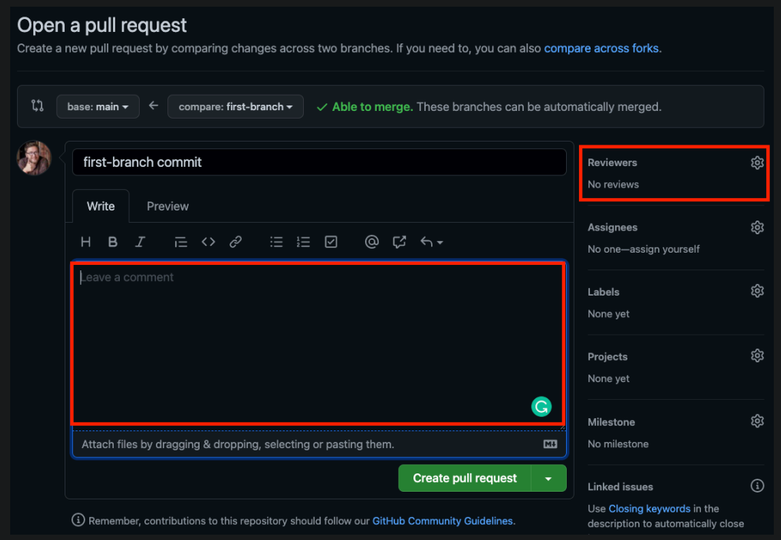
- There's some important information you need to provide to create a pull request, such as specifying the branch you want to merge



School of Tech
part of accenture

Setting up a PR

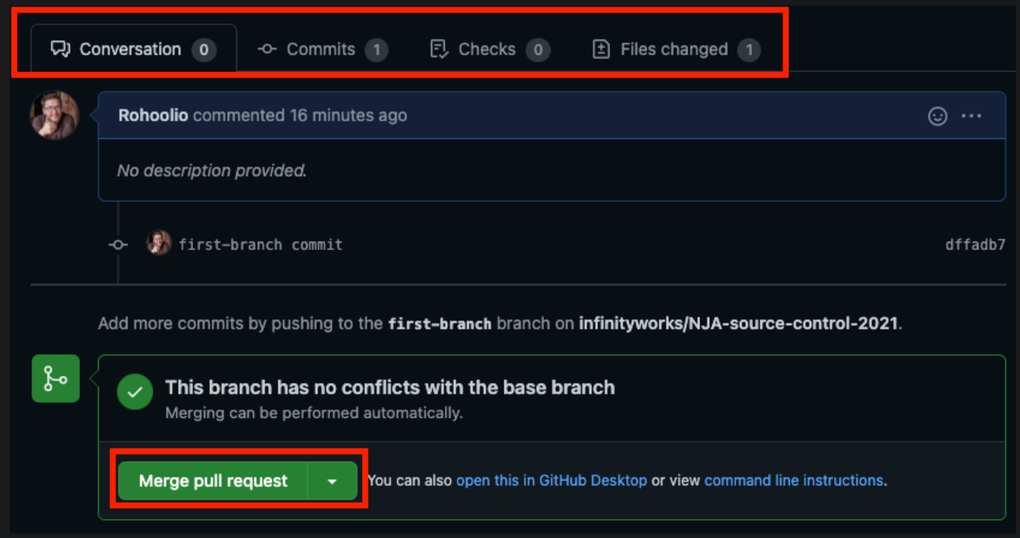
- As well as some useful additional information like the description and adding reviewers (these can still be changed after the pull request is made)



School of Tech
part of accenture

Complete a PR

- When all the previous steps are done others can then view, comment and suggest changes to the code



Whomever made the PR can't also Approve it - why not?

School of Tech
part of accenture

Emoji Check:

Do you feel you understand pull requests? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

Mini project under source control?

Yes please!

> We will now move your mini-project files into your personal GitHub repos so they are under version control and backed up!
>
> Also, add all the folders for these session like python-1/2, data-persistence, and so on.

School of Tech
part of accenture

Exercise - Mini Project

- Move your mini project files into the your new repository

We suggest a folder structure like this:

```
.gitignore
README.md
sessions/
mini-project/
  src/*
  tests/*
  requirements.txt
```

This way you can put all of your app code into `src/` and unit tests (more on that in a future session) into `tests/`.

You'll have a `data/` directory too in your local repo containing any persisted data for your app. Add `data/*` to your `.gitignore` file so that personal data doesn't get pushed to the remote repo.

Exercise - Push your mini-project files

- In a terminal:
  - Add all your files to staging wit `git add ...`
  - Commit all your files with a message
  - Push your files
- Then in your web browser:
  - Check you can see your mini-project files in your remote repository in GitHub

Introducing the daily kata exercises

Because git is really really important, you will be practicing the commands we have covered today (as well as a few new ones) in a daily kata, starting tomorrow.

> The instructor(s) will share a task each day, and then you will practice it in project time.

Daily pushes to GitHub

After the this session, you will be expected to push your Mini Project work into your repo on a regular basis (preferably every day, if not more). It does not need to be "working" or "complete" - back it up anyway.

This repo will also contain all your exercises from all the other taught sessions, (e.g. python-1, python-2, data-persistence and so on).

Between working on exercises and the Mini Project, there should be something to push to your repo every day!

Terms and Definitions - recap

- Repository: Stores all your code, and all history and tracking metadata
- Commit: A number of changes bundled up as a single transaction
- Branch: Alternative code path originating from a specific commit in another branch
- Staging: Telling Git what changes we want to include in the next commit

Terms and Definitions - recap

- Clone: Copy a remote Git repository and all of its metadata into our local environment
- Fork (GitHub): Make a copy of a repo hosted online into another hosted repo under our account so we can freely change it
- Pull Request / Merge Request: Request to merge a branch/fork into the main branch/fork it originated from (upstream)

Overview - recap

- Source Control
- Git
- Sharing Code with git
- Branching
- Pull Requests

Objectives - recap

- Describe what source control is, and why we use it
- Add and commit features / folders / files, and push to remote
- Create a Git repository
- Create branches
- Create, review, and merge pull requests

Further Reading

- Roadmap.sh - Learn Git and GitHub
- Pro Git Book (free)
- git tutorials
- Pull requests tutorial
- Credits to Rachel Carmena
- Great introductory article to git
- `man git`

School of Tech
part of accenture

Emoji Check:

On a high level, do you think you understand the main concepts of this session? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively