GitHub for SunSat

Document Overview

This document is basically just a quick intro to GitHub & why it's useful

I'll quickly introduce Git & GitHub, go through some of the basic actions, and then talk about how to actually use it.

This is only brief – some good quick resources for more detail:

Basic Git: https://guides.github.com/introduction/git-handbook/

Intro to GitHub: https://guides.github.com/activities/hello-world/

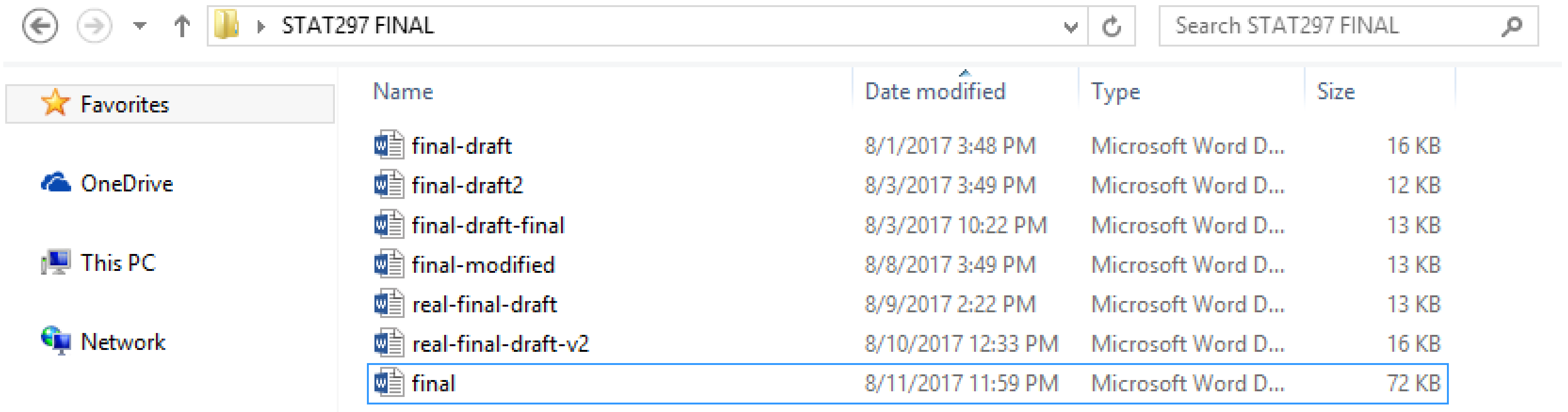
What is Git?

I'll cover some of the specifics a bit later, but in short, Git is a version control system.

It allows users to track the changes to their files over time, so it's easy to revert to a working copy if issues arise, and easy to work out where things started to go wrong!

Git is pretty much industry standard at this point, so knowing how to use it is a very useful skill!! ☺

Git wants to stop you from doing things like this ->



Name	Date modified	Type	Size
final-draft	8/1/2017 3:48 PM	Microsoft Word D...	16 KB
final-draft2	8/3/2017 3:49 PM	Microsoft Word D...	12 KB
final-draft-final	8/3/2017 10:22 PM	Microsoft Word D...	13 KB
final-modified	8/8/2017 3:49 PM	Microsoft Word D...	13 KB
real-final-draft	8/9/2017 2:22 PM	Microsoft Word D...	13 KB
real-final-draft-v2	8/10/2017 12:33 PM	Microsoft Word D...	16 KB
final	8/11/2017 11:59 PM	Microsoft Word D...	72 KB

What is GitHub?

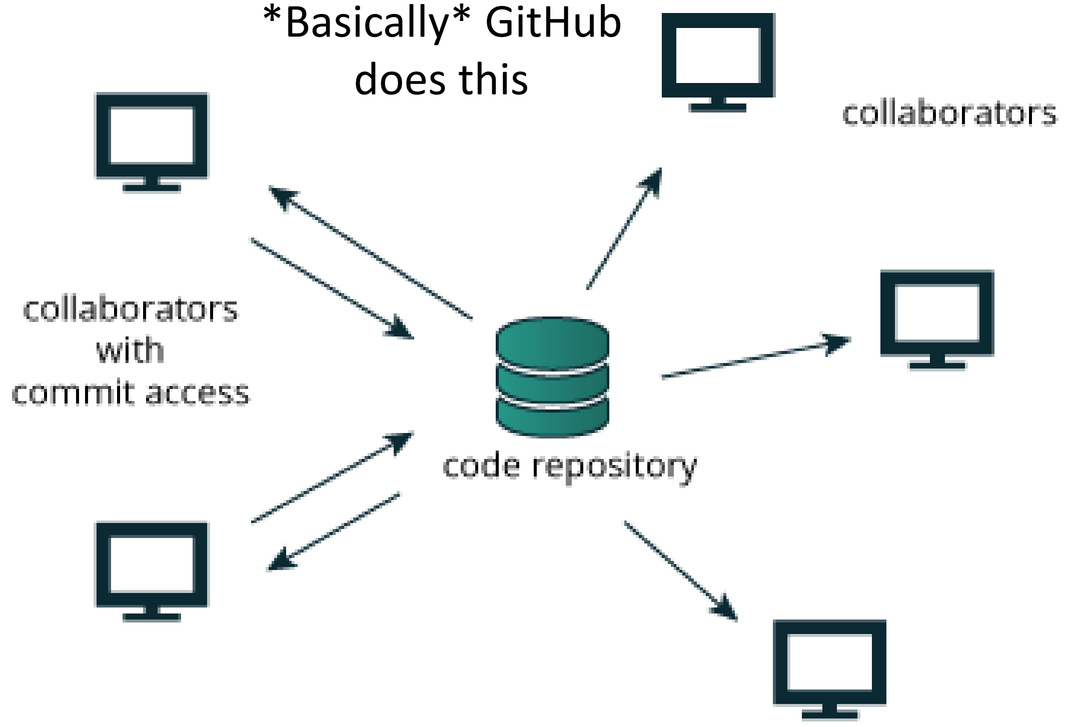
Git in its purest form is run on your computer. You don't need internet access and all of the to-and-fro is going on inside your machine

GitHub is a Git repository (I'll explain what this is shortly) hosting service. It hosts your work in the cloud.

Git is totally free, and so is GitHub (as far as we're concerned)

GitHub is powerful because it enables collaboration – so in our SunSat team, we can look over work that others have done and easily suggest changes or make a 'branch' (I'll explain this too!) to test cool new things.

Basically GitHub does this



Some Terminology

Repository (Repo) | The place where the current state of your work is stored. It also contains all of the history of the project in the form of a ‘snapshot’.

‘Remote’ Repo | The ‘shared’ repository. In GitHub, the remote repo is where all of the work lives, and is located in the cloud. Our remote repo is here: https://github.com/SunSat-Avionics/Avionics-Code-for-Arduino-

‘Local’ Repo | This one is yours and yours alone! You get a local repo by ‘cloning’ (copying) the remote repo onto your computer. Initially, you will have an exact copy of the remote repo which you can now make changes to ‘locally’ without affecting the code that anyone else is using. This means you can develop new features or test bugs without worrying about breaking anything for any of the other users.

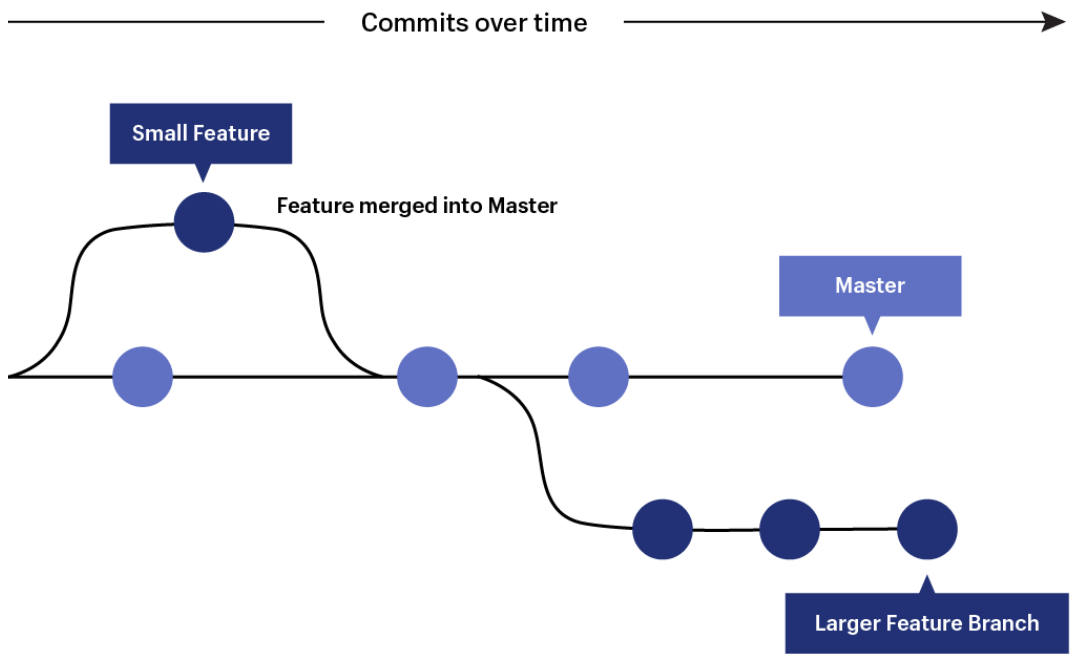
Making Changes

Now if you decide that you want to make some changes to the code, you can create a ‘branch’ in your local repo. You’re creating a copy of all of the work at the point in time that you created the branch.

You now have a bit of a playground – you can make changes to the work in the branch and test new features until you’re happy you’ve got something that works. You can switch between active branches and work on multiple features at a time – this way you can cleanly separate the work that you’re doing to tackle different tasks, whilst still maintaining the ‘master’ branch at all times.

Each branch should be created with a purpose. You might want to add a new function, or fix a bug that’s causing some trouble. Once you’re satisfied that you’ve done what you’ve set out to in this branch, you can ‘merge’ the branch back into the master – you’re basically adding this feature into the main code on your remote repo.

Whenever you make a bit of a change (e.g. adding a little function or some new lines), you should ‘commit’ the changes to your branch. In effect this is ‘saving your changes’

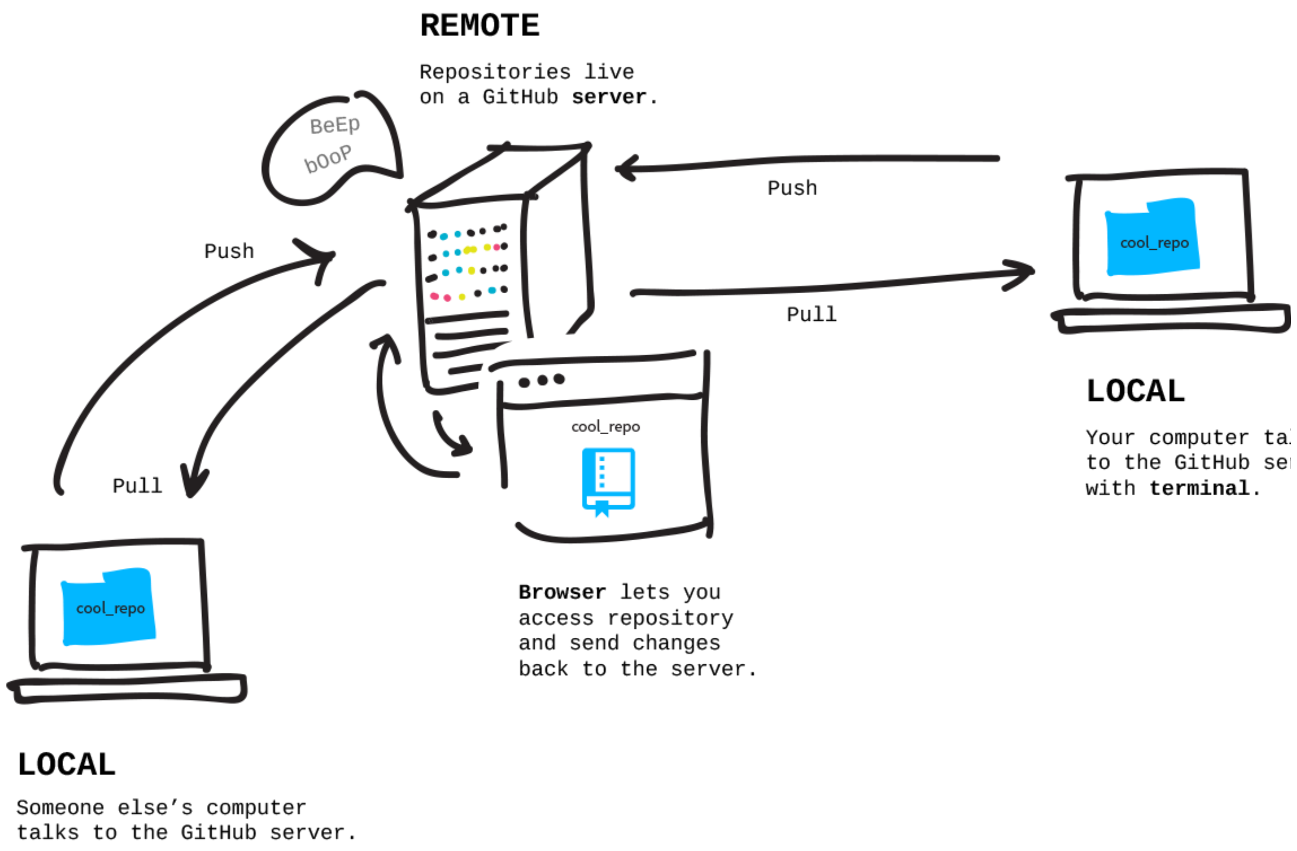


Making Changes

Okay, let's say you've made some changes on your remote repo. For example, maybe you wanted to add a feature that allowed the satellite to send some extra telemetry data back to the ground station. These changes are now on your master branch and you've hopefully verified that they've done the job, but no-one else can see them yet!

You now need to 'push' your changes to the remote repository. This just means that you are sending your changes to the cloud, and updating the repo that everyone has access to.

Once you've done this, the remote master has updated. Now other users can 'pull' the changes into their local repo. Their local master branch will be updated to reflect that changes that you have just pushed.


REMOTE
Repositories live on a GitHub **server**.
BeEp
bOoP
Push
Pull
Push
Pull
cool_repo
cool_repo
cool_repo
LOCAL
Your computer ta
to the GitHub se
with **terminal**.
Browser lets you access repository and send changes back to the server.
LOCAL
Someone else's computer talks to the GitHub server.

Pull Requests

If the change is especially crucial and you'd really like it to be reviewed before it is merged with the remote master, you can create something called a 'pull request'.

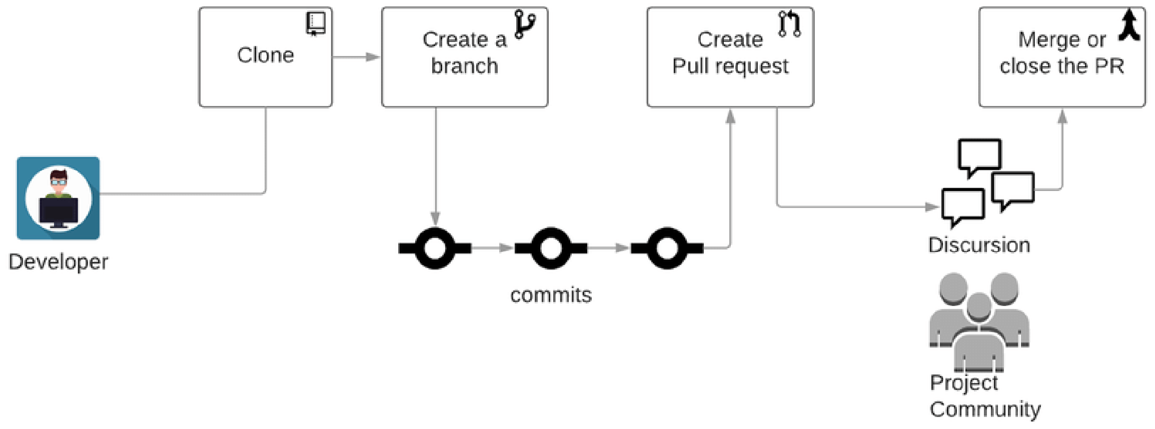
A pull request is basically you telling everyone that you've made some changes, and letting them take a look. Then, people can look over your changes and see if everything looks okay – if not, they can comment on your code or suggest changes.

It's basically an extra layer of verification, just to make sure everyone is happy before the changes are fully committed.

Some changes might be really small and simple, and so you might not need to create a pull request (I think generally speaking it's good practice to create a pull request for every significant change you make, but don't create one if you've just corrected a typo!) Especially in this application, you might not feel like it's necessary, but just know it's there any it's useful if you want to use it ☺

You can assign people to pull requests. If there are two people working on one bit of code, you can assign the other person, rather than asking *everyone* to have a look (unless you want everyone to have a look…)

1. Clone the remote repo onto your machine. This is your local repo now
2. Create a branch to add a feature or fix a bug
3. Make incremental changes to the remote branch – committing changes as you go
4. Create a pull request – you're asking the other people to review your changes before they're officially committed
5. When everyone's happy, merge the branch into the master.



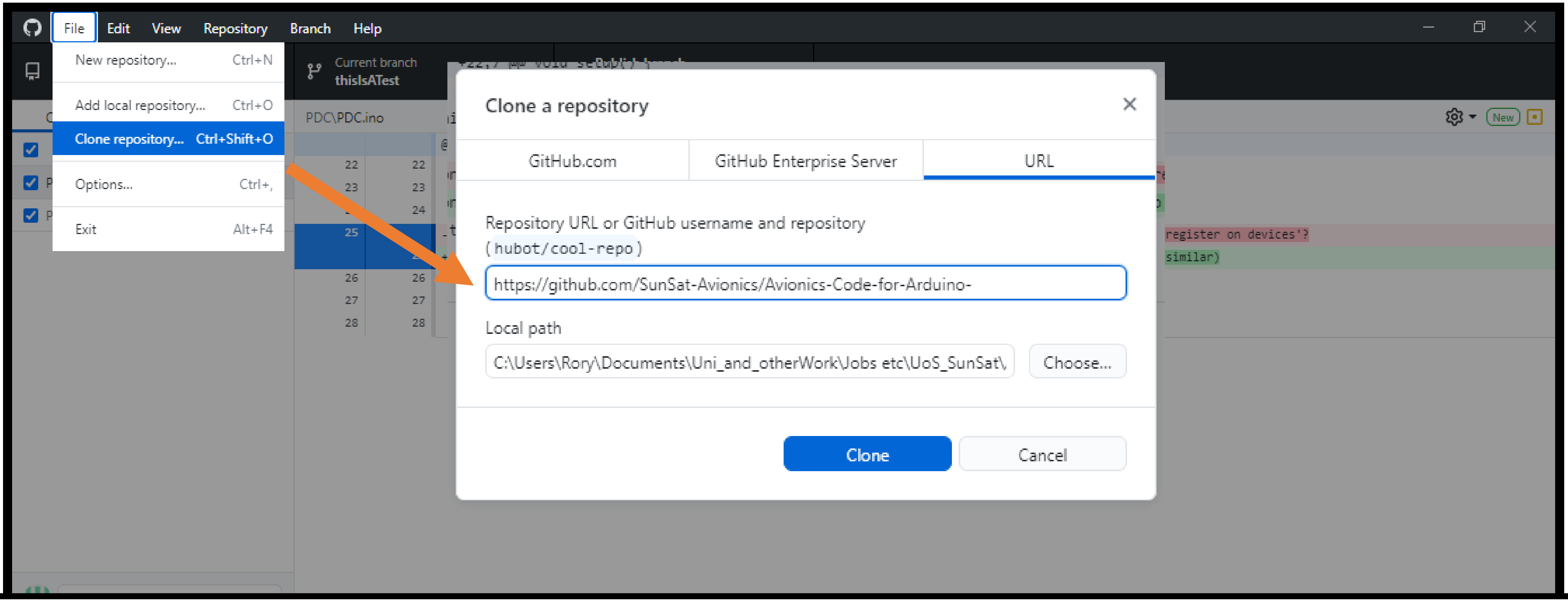
Using GitHub

Okay so now you (hopefully) have a pretty decent grasp of some of the basic logistics – how does it all actually work?

First things first – I'd recommend downloading GitHub for desktop (windows/mac). GitHub is totally accessible via command line if you'd prefer (*it's relatively intuitive as there aren't many commands that you'll *need* - here's a cheatsheet if you're interested.*)

The rest of this tutorial will be based on the GitHub website and the desktop version, though, as a heads up!

Cloning the Remote Repo



Repository URL or GitHub username and repository
(hubot/cool-repo)

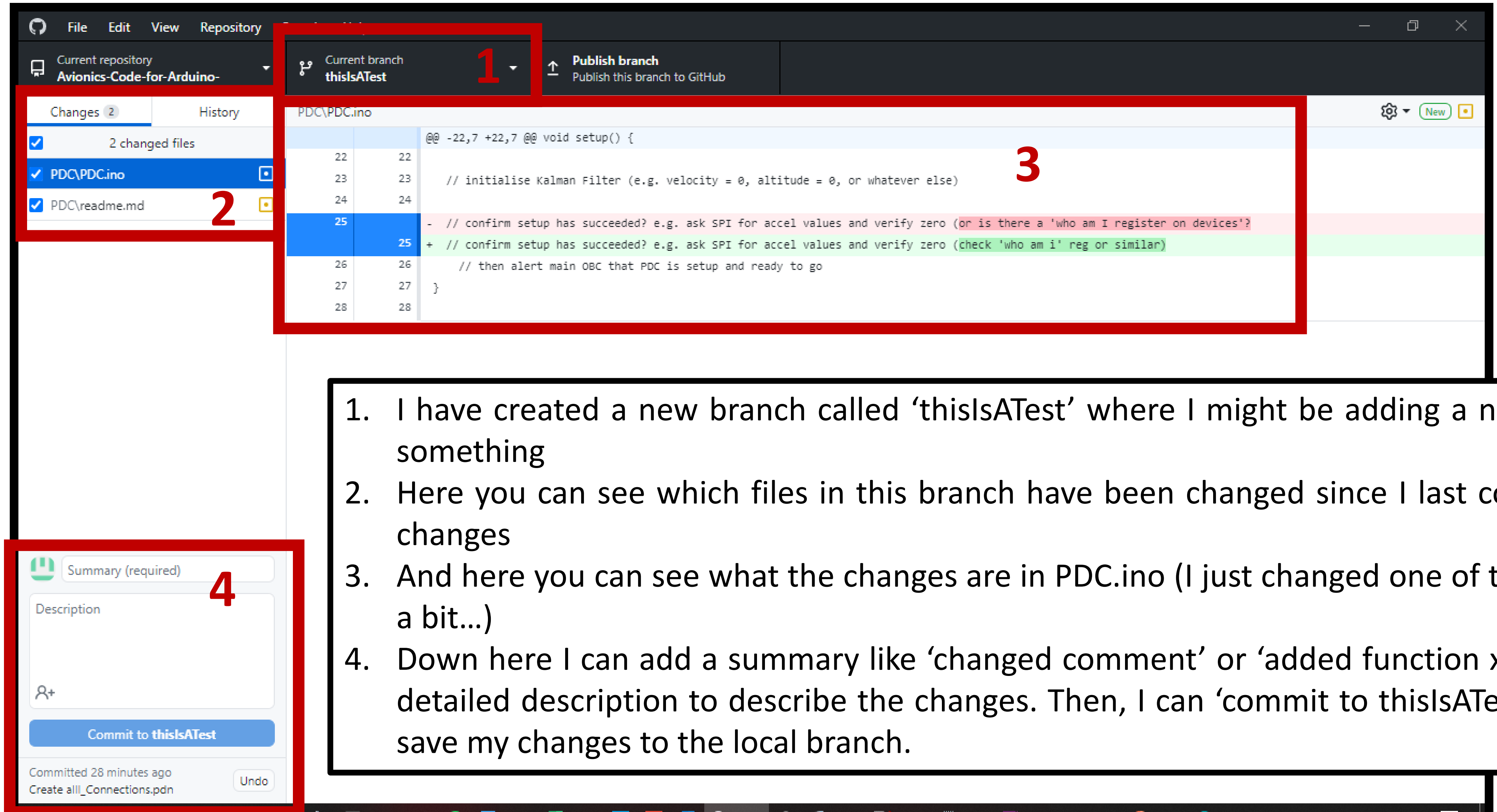
https://github.com/SunSat-Avionics/Avionics-Code-for-Arduino-

Local path

C:\Users\Rory\Documents\Uni_and_otherWork\Jobs etc\UoS_SunSat\

1. From GitHub desktop, go to *File > Clone Repository*,
2. In the window that opens, go to the 'URL' tab, and copy the URL of the remote repo (https://github.com/SunSat-Avionics/Avionics-Code-for-Arduino-)
3. Select the location on your computer where you'd like to store your remote repo, then hit clone!

Making Changes in a Branch

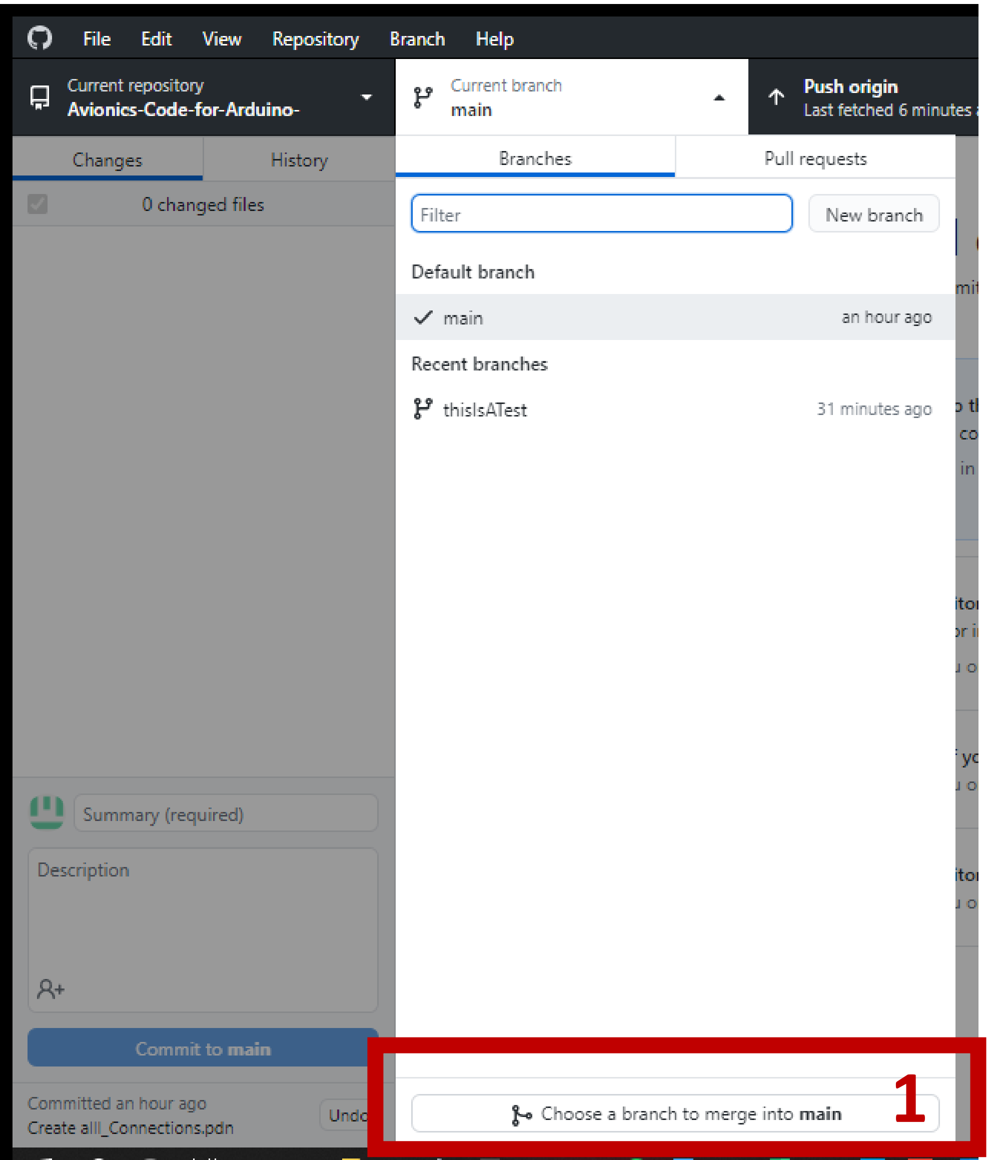


PDC\PDC.ino

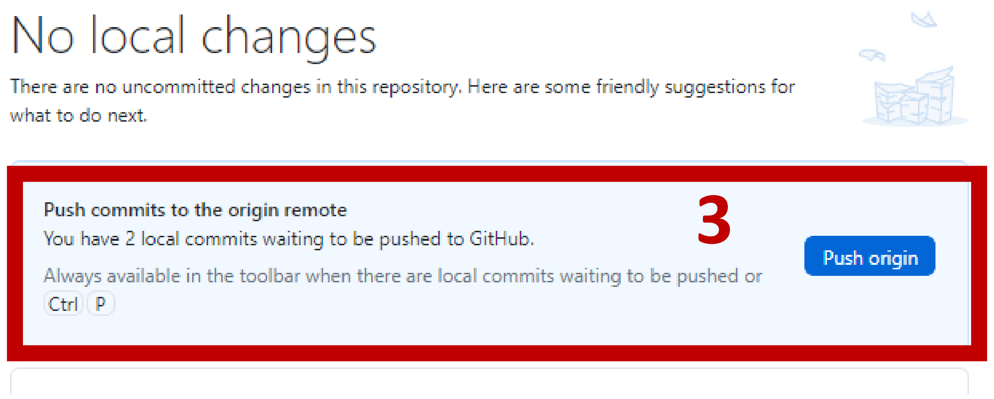
```
@@ -22,7 +22,7 @@ void setup() {

  // initialise Kalman Filter (e.g. velocity = 0, altitude = 0, or whatever else)

- // confirm setup has succeeded? e.g. ask SPI for accel values and verify zero (or is there a 'who am I register on devices'?
+ // confirm setup has succeeded? e.g. ask SPI for accel values and verify zero (check 'who am i' reg or similar)
    // then alert main OBC that PDC is setup and ready to go
}

```

1. I have created a new branch called ‘thisIsATest’ where I might be adding a new feature or something
2. Here you can see which files in this branch have been changed since I last committed any changes
3. And here you can see what the changes are in PDC.ino (I just changed one of the comments a bit…)
4. Down here I can add a summary like ‘changed comment’ or ‘added function x’, and a more detailed description to describe the changes. Then, I can ‘commit to thisIsATest’, which will save my changes to the local branch.

Pushing Changes Without Pull Request



1. If you don't think you need to create a pull request and have other people review your code (it can be a bit long-winded and tedious and sometimes you just want to push a little thing through), then you can merge the new branch into your local master using this button at the bottom.
2. After you've merged the branch, you're just left with the master branch
3. Now you can just press this 'push origin' blue button to push the changes to the remote master, and there won't be any pull request process to worry about! If you go to the remote repo now, you should see that the code over there has changed to reflect the changes you just pushed



No local changes

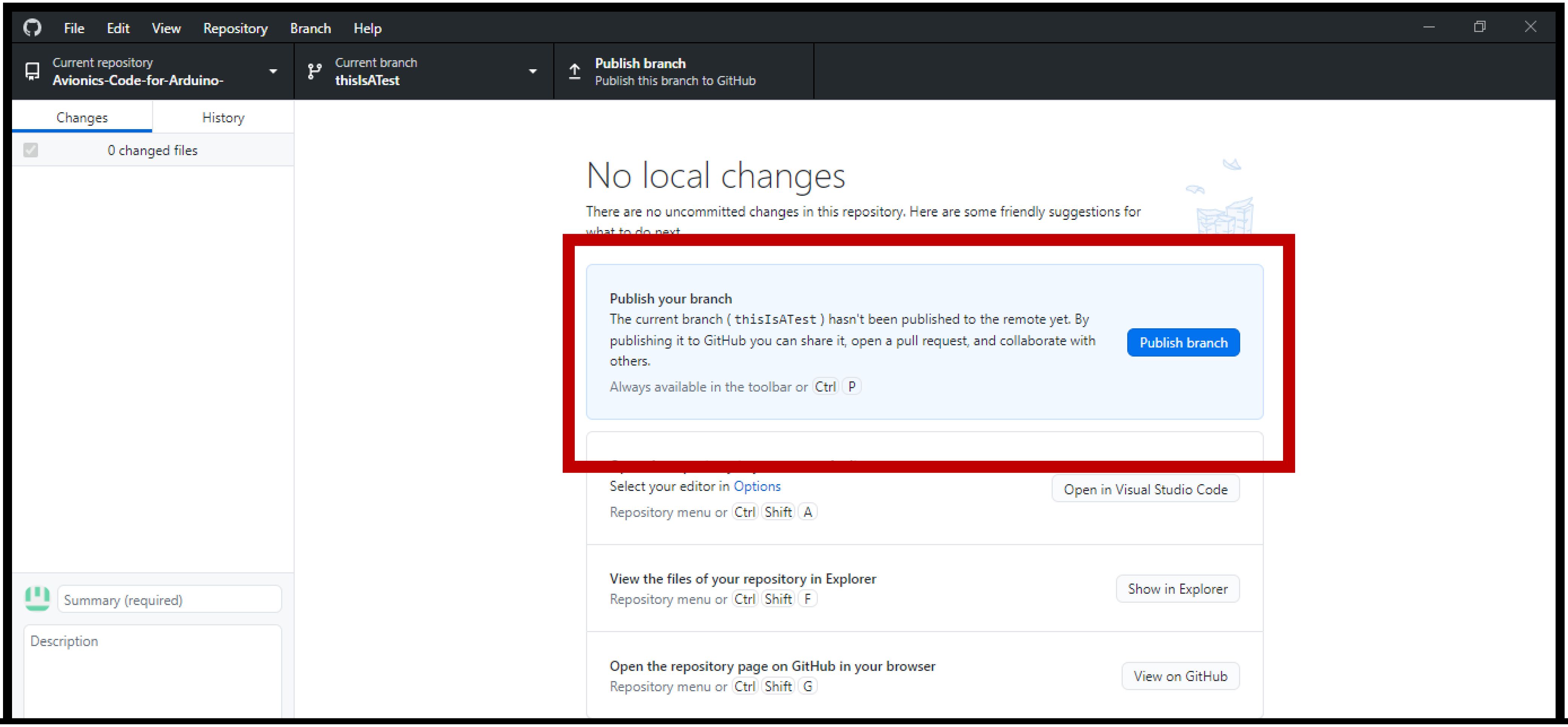
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Push commits to the origin remote

You have 2 local commits waiting to be pushed to GitHub.

Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Publishing and Pull Requesting



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Publish your branch

The current branch (`thisIsATest`) hasn't been published to the remote yet. By publishing it to GitHub you can share it, open a pull request, and collaborate with others.

Always available in the toolbar or Ctrl P

Select your editor in Options

Repository menu or Ctrl Shift A

View the files of your repository in Explorer

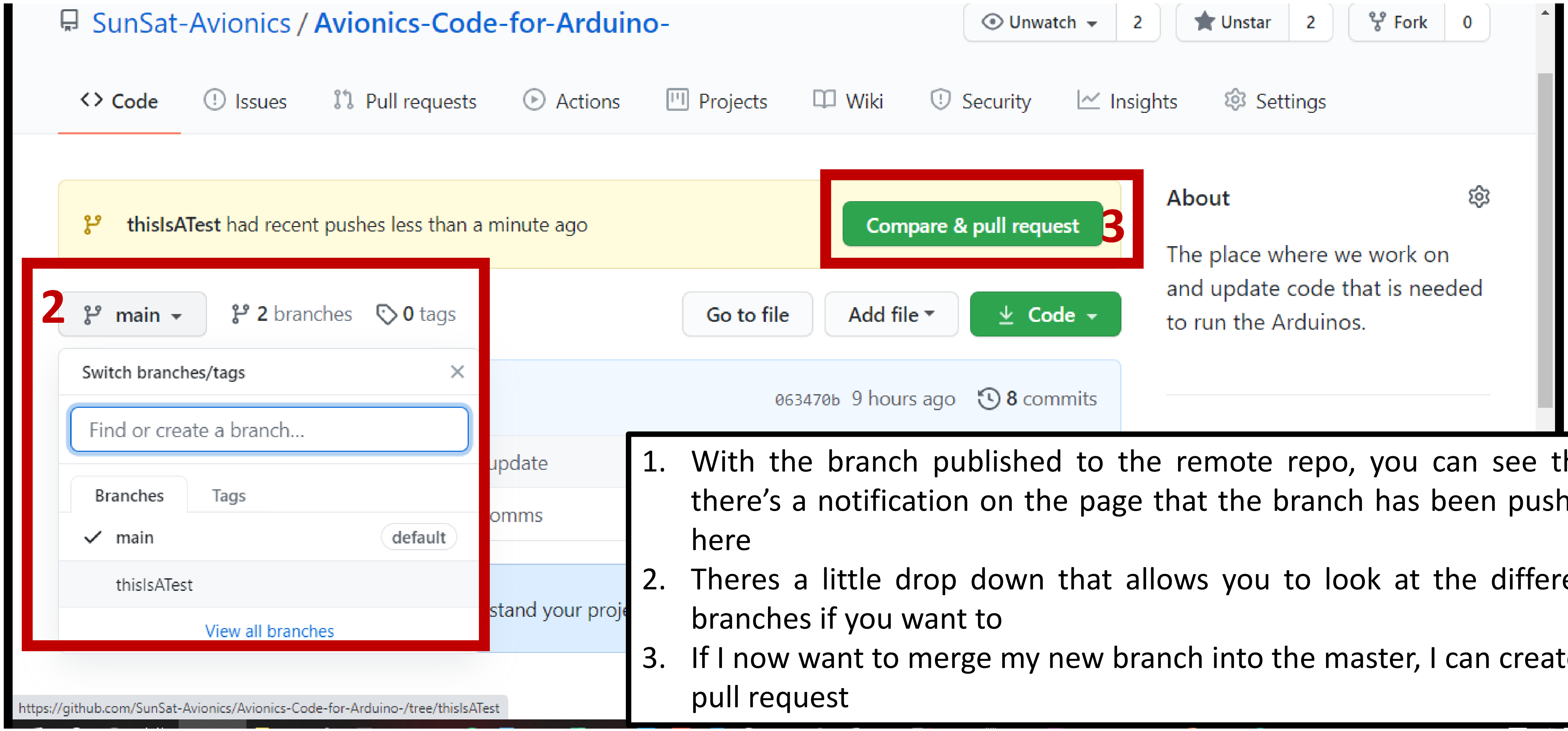
Repository menu or Ctrl Shift F

Open the repository page on GitHub in your browser

Repository menu or Ctrl Shift G

1. If I want to go through the pull request process, I *don't* merge my new branch into master, but I instead publish the branch to the remote repo.
2. Once I publish, the branch, it will appear on the remote repo

The Remote Repo


SunSat-Avionics / Avionics-Code-for-Arduino-

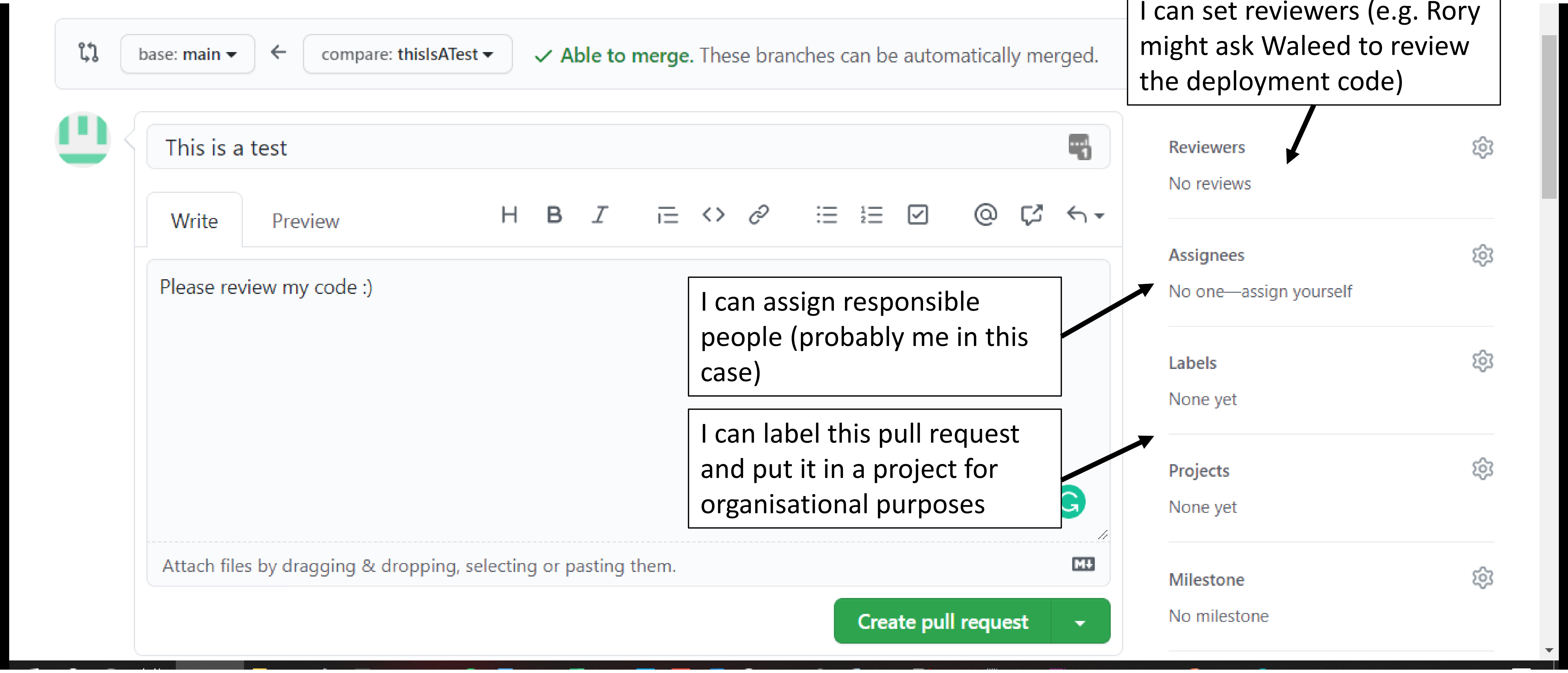
thisIsATest had recent pushes less than a minute ago

The place where we work on and update code that is needed to run the Arduinos.

https://github.com/SunSat-Avionics/Avionics-Code-for-Arduino-/tree/thisIsATest

1. With the branch published to the remote repo, you can see that there's a notification on the page that the branch has been pushed here
2. Theres a little drop down that allows you to look at the different branches if you want to
3. If I now want to merge my new branch into the master, I can create a pull request

Pull Requests



base: main ← compare: thisIsATest ✓ Able to merge. These branches can be automatically merged.

This is a test

Write Preview

Please review my code :)

Attach files by dragging & dropping, selecting or pasting them.

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

I can set reviewers (e.g. Rory might ask Waleed to review the deployment code)

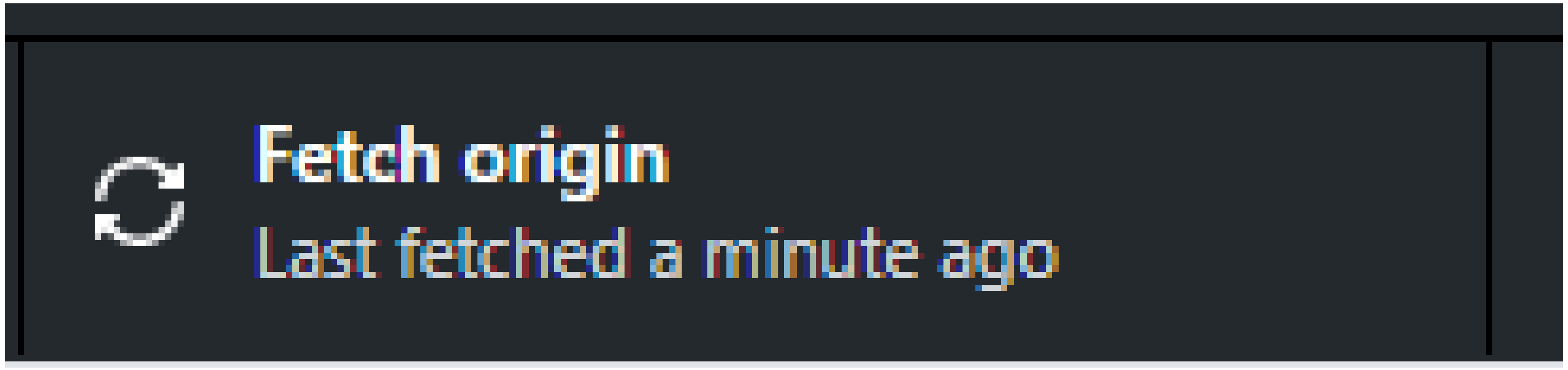
I can assign responsible people (probably me in this case)

I can label this pull request and put it in a project for organisational purposes

Pulling changes

It’s worth getting into the habit of fetching changes from the remote repo. People may have committed changes recently, and it’s useful for you to have those in your remote repo before you continue making any more changes.

You can simply do this by pressing this button in the desktop GitHub



The End! (kind of)

Basically this is all there is to know to get started. It should be enough to help you make changes, submit them for review, and track changes over time.

From personal experience, it can be a bit of a long process to start with, but once you get into the rhythm of branching, changing, committing, and then either publishing & pull requesting, or merging then publishing, it's quite a straightforward process!

If you've got any questions at all, please ask! I'm always happy to help so don't hesitate ☺