

Introducing PROS 3 GIT / GItHub

by
Willem Scholten
Learning Access Institute

Revision: 07/01/2019

- **Copyright © 2019 by the Learning Access Institute / Willem Scholten**
- **All rights reserved.** No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.
- All material contained within this publication, including referred to software samples, are for the soul use by institutions participating in the **CWU GEARUP robotics program**, and as such have been granted permission to share, reproduce this material for the benefit of their students.

PROS 3 Intro

- The **Introduction to PROS 3** is a collection of separate guides which cover the various fundamental aspects for students to be successful in using the PROS 3 development environment to program either a Cortex or V5 based robot.
- These guides cover the programming language and commands, as well as the tool sets to create and maintain code in a professional manner.

PROS 3 Intro

- List of Guides:
 - C/C++ Language Guide
 - Cortex Programming Guide
 - V5 Programming Guide
 - PROS 3 Interface Guide
 - GIT/GIThub guide

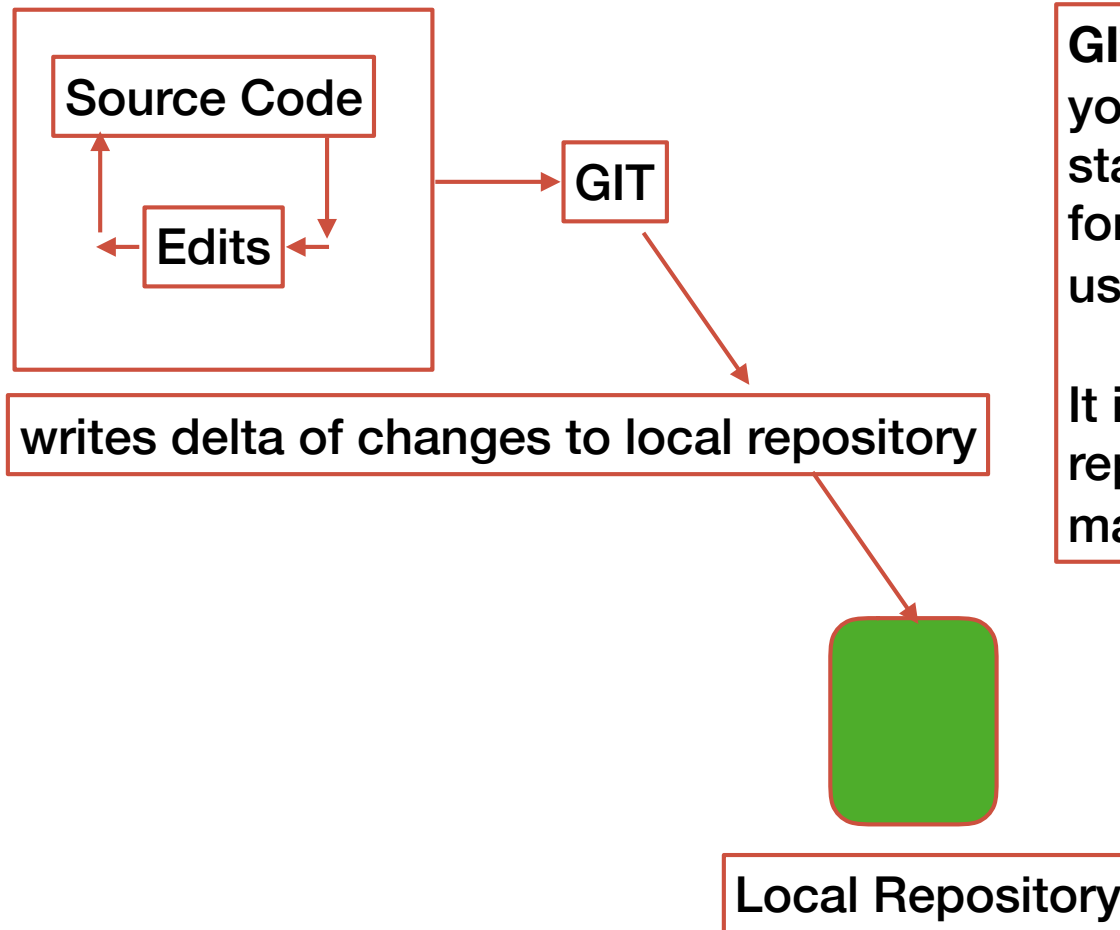
gitHUB Access

- Sample code repositories for learning the Cortex (and the V5) can be cloned from the following URL:
 - <https://github.com/sprobotics>

gitHUB Access

- **What is Git and gitHUB:**
 - **Git** is the **client** that manages version / source control, and is installed on **your development machine**.
 - **Git** has versions for most operating systems, including **Windows, MAC OS and Linux**
 - **Git** interacts with the central **hosted Git repository**.
 - A **Git** repository most typically lives on **gitHUB** a cloud based repository service, in larger companies a private in house repository server may exists.

gitHUB Access



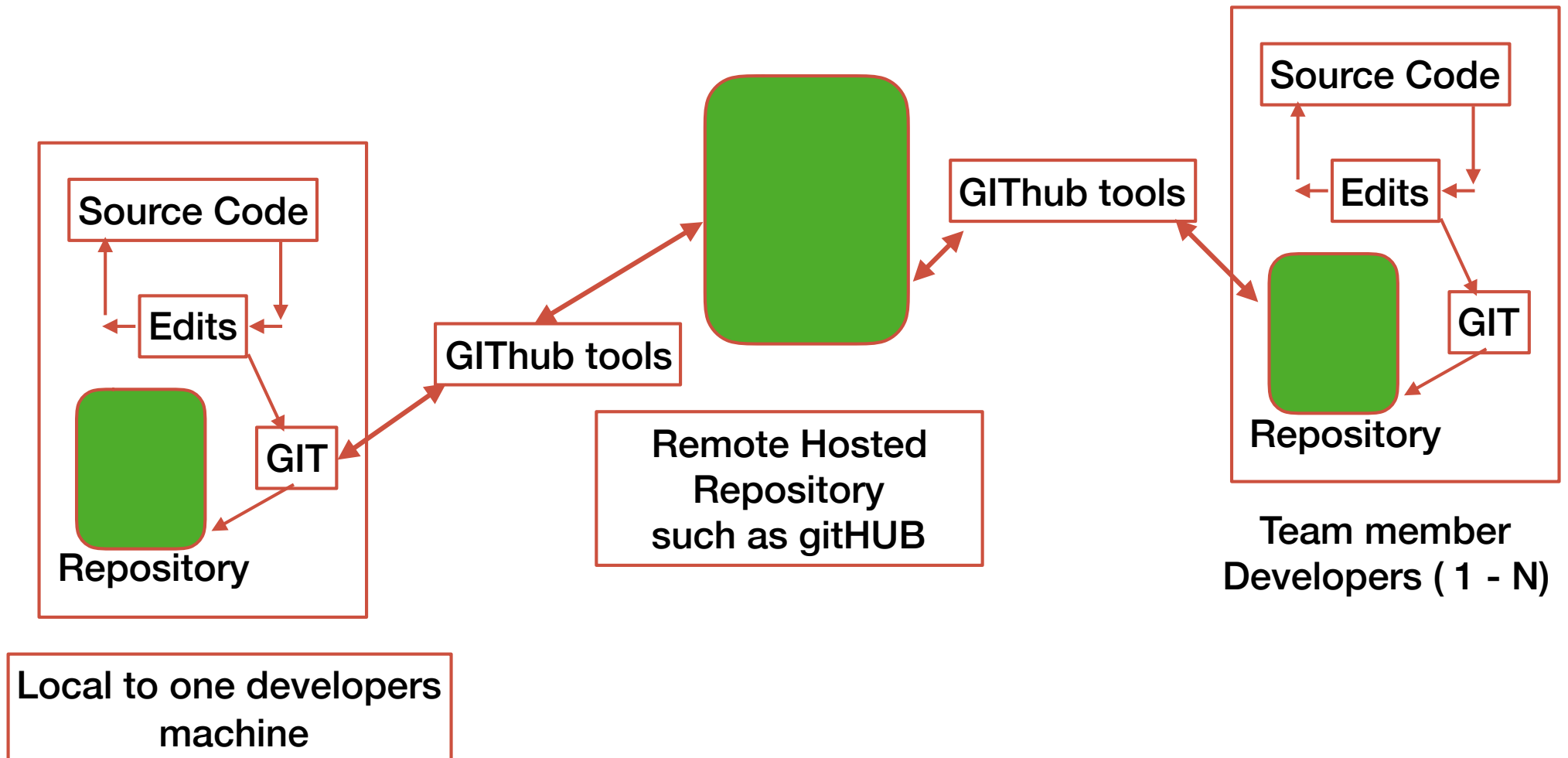
GIT - builds a repository of changes of your local files, these delta's are stamped (uniquely identified) and allow for the roll-back to a previous state, using a variety of GIT commands.

It is important to note that this repository of delta's lives on your local machine only

gitHUB Access

- **Interaction between Git and gitHUB:**
 - **Git** — provides source code control using repositories
 - **gitHUB** — host the repositories, including documentation for your team and beyond

gitHUB Access



gitHUB Access

- The following slides will **cover increasingly** in-depth how GIT and gitHUB as well as PROS integrate and facilitate development in teams.
- At a **minimum** you should think of GIT / gitHUB as a tool set, integrated with PROS to help you safeguard your code base against accidental loss, 'bad code decisions' you like to roll-back (remove)
- Yes the **best practice** is to learn to use the tool to facilitate seamless multi developer code development and sharing among sub teams.

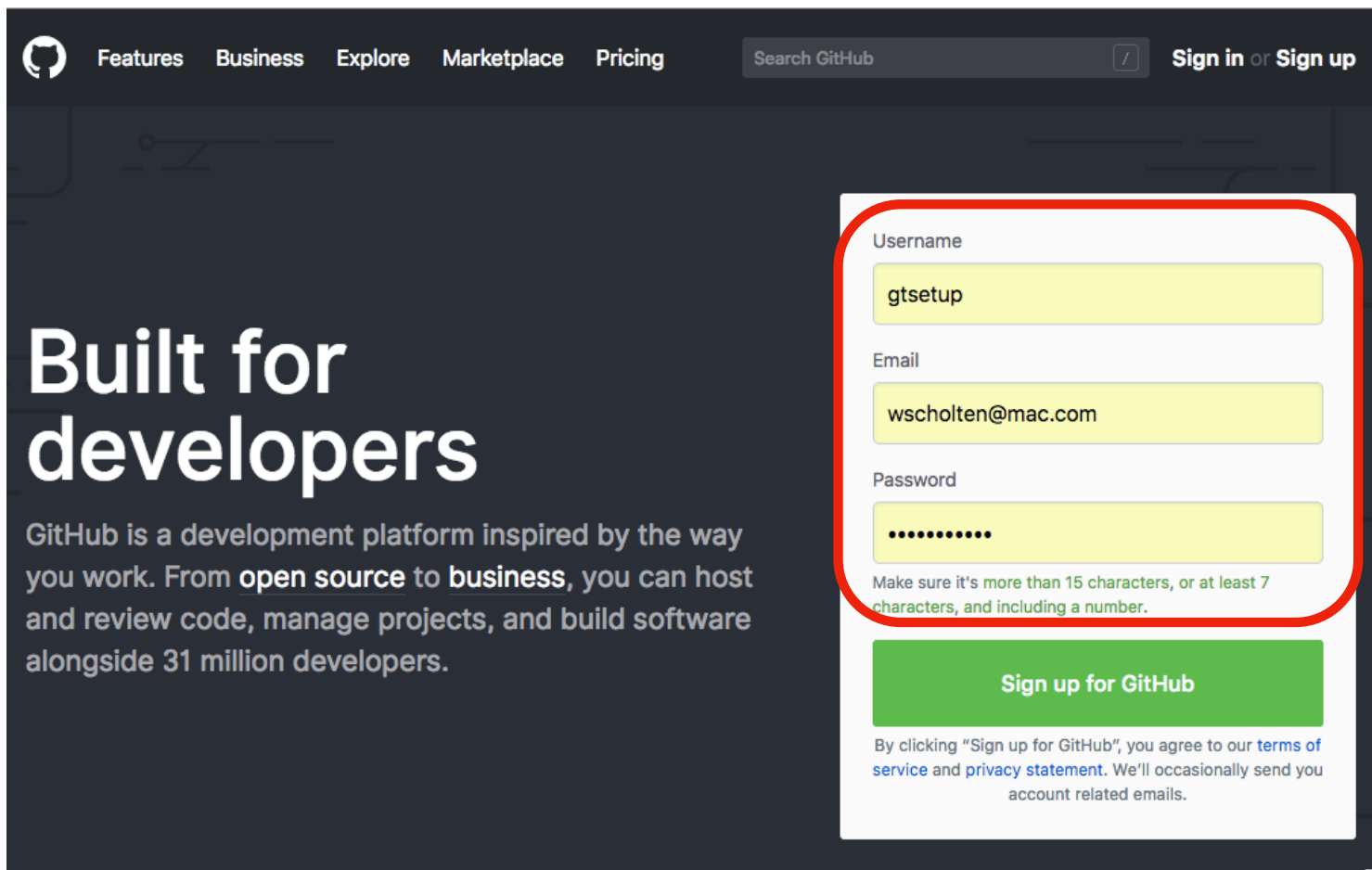
gitHUB Access

- Tools required for safe and team based development:
 - GIT tools (GIT command line tools)
 - gitHUB account and tool set (gitHUB client for your machine)
 - PROS 3 GIT/gitHUB integration plugin

gitHUB Access

Create gitHUB repository server account:

Goto: <https://github.com>



The screenshot shows the GitHub homepage with a dark blue background. On the left, the text "Built for developers" is prominently displayed. To the right, a white sign-up form is highlighted with a red rounded rectangle. The form contains three input fields: "Username" with the value "gtsetup", "Email" with the value "wscholten@mac.com", and "Password" with masked characters. Below the password field is a green button labeled "Sign up for GitHub". At the bottom of the form, there is a line of small text: "By clicking 'Sign up for GitHub', you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails."

Username

gtsetup

Email

wscholten@mac.com

Password

.....

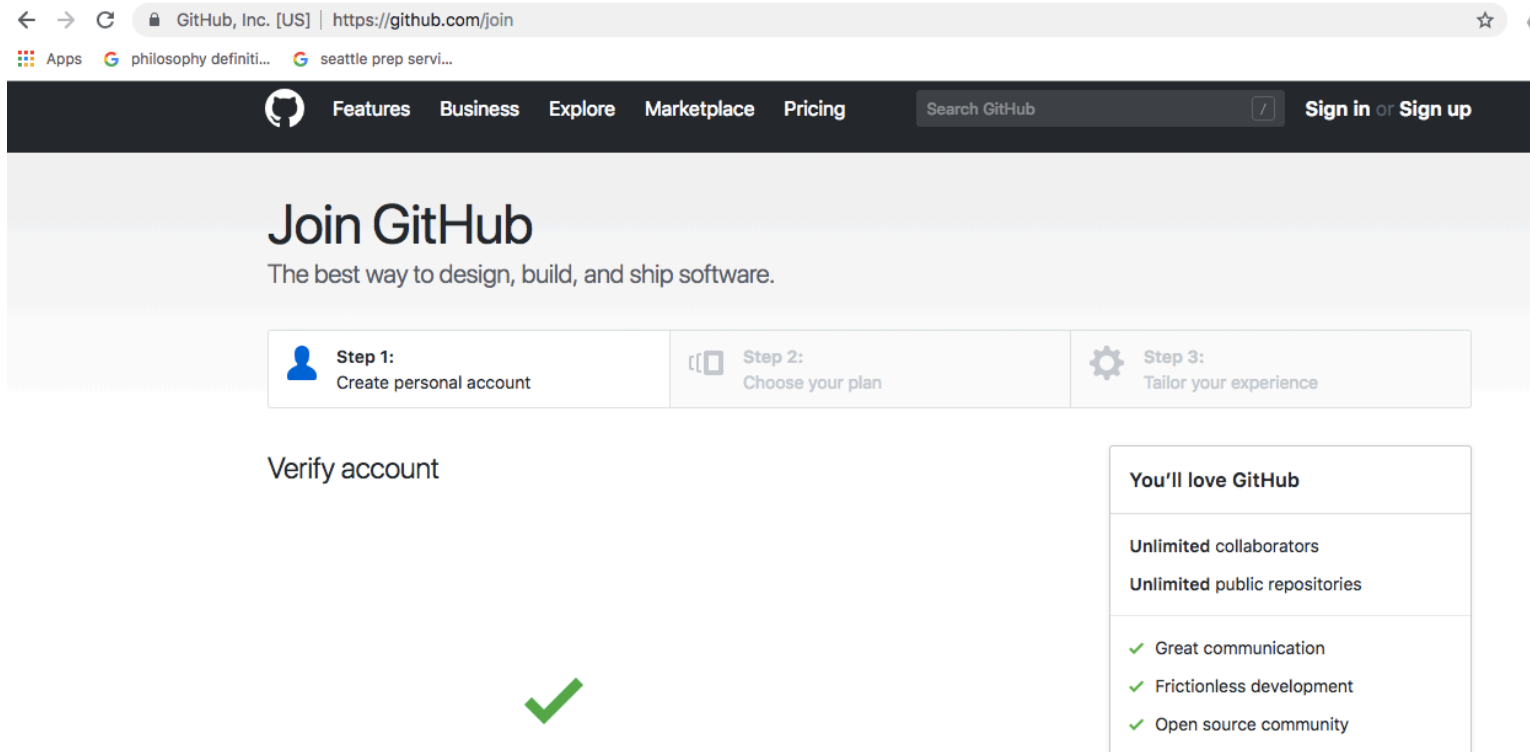
Make sure it's more than 15 characters, or at least 7 characters, and including a number.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

Pick username,
add your email
and pick a
password

gitHUB Access



gitHUB Access


You've taken your first step into a larger world, @gtsetup.

 **Completed**
Set up a personal account

 **Step 2:**
Choose your plan

Choose your personal plan


Every plan comes with GitHub's most-loved features: Collaborative code review, issue tracking, the open source community, and the ability to join organizations.


Free

\$0
per month

Includes:
Personal account
Unlimited public repositories
Unlimited collaborators

There are millions of public projects on GitHub. Join one or start your own for free.


Developer

\$7
per month

Includes:
Personal account
Unlimited public repositories
Unlimited private repositories
Unlimited collaborators

Free for students as part of the [Student Developer Pack](#).

Pick the free plan

gitHUB Access

You'll find endless opportunities to learn, code, and create, @gtsetup.



Completed
Set up a personal account



Step 2:
Choose your plan



Step 3:
Tailor your experience

How would you describe your level of programming experience?

- ☐ Very experienced ☒ Somewhat experienced ☐ Totally new to programming

What do you plan to use GitHub for? (check all that apply)

- ☐ Design ☒ Development ☐ Research
☐ School projects ☐ Project Management ☐ Other (please specify)

Which is closest to how you would describe yourself?

- ☐ I'm a hobbyist ☒ I'm a student ☐ I'm a professional
☐ Other (please specify)

What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

Submit

[skip this step](#)

**Just answer
what makes
sense**

gitHUB Access

Create Repositories as needed or via Window gitHUB client

The screenshot shows the GitHub homepage for a user who has not yet created any repositories. At the top, a large banner encourages learning Git and GitHub without code, featuring a green 'Read the guide' button and a grey 'Start a project' button. Below the banner, on the left, are two notification banners: a dark grey 'Launch report' and a white 'Our new Terms of Service and Privacy Statement' banner. Below these is a 'Repositories' section with a 'New repository' button and a message stating 'You don't have any repositories yet!'. On the right, the 'Browse activity' section features a card titled 'Discover interesting projects and people to populate your personal news feed.' with an 'Explore GitHub' button. A 'Discover repositories' link is also visible in the top right of the activity section.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#) [Start a project](#)

Launch report
Everything we released at GitHub Universe

Our new Terms of Service and Privacy Statement are in effect.

Repositories [New repository](#)

You don't have any repositories yet!

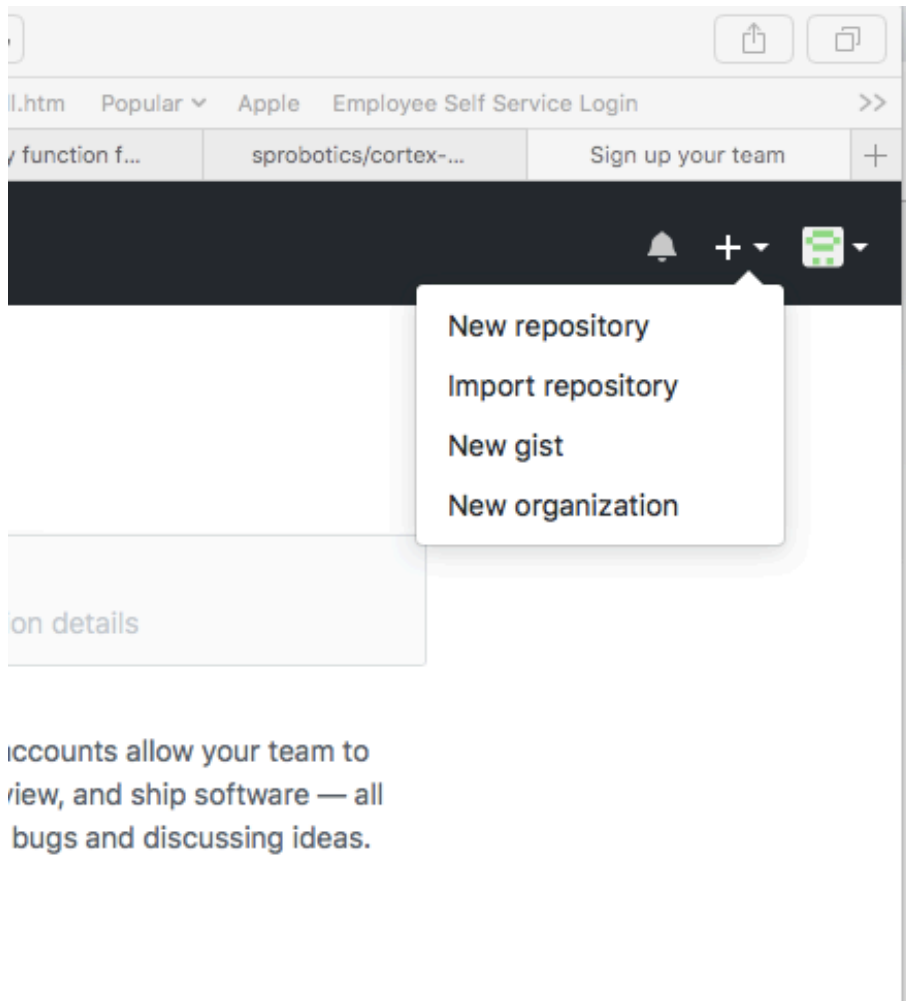
Browse activity [Discover repositories](#)

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you [watch](#) and people you [follow](#).

[Explore GitHub](#)

gitHUB Access



You can create an organization - being your team, and then add others to the repository to submit code changes.

gitHUB Access

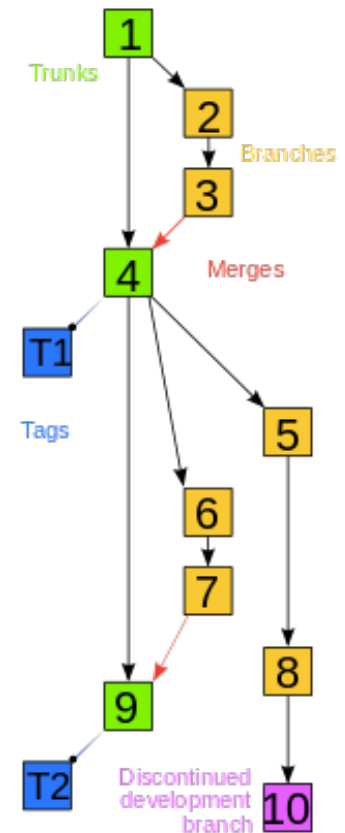
- **Installing the local Git and gitHUB clients:**
 - For **Windows** and **MAC OS** go to: <https://desktop.github.com/>
 - For **Linux** - depending on your distribution you may want to install **gitKraken** - available here: <https://www.gitkraken.com/download>
 - There are also gitHUB clients for IOS and android, allowing you to manage your repository.

gitHUB Access

- **Using GIT to manage your project:**
 - GIT uses a process of storing '**delta's**' of code changes and allows you to **roll back** to previous stable versions.
 - GIT uses **master, branches, forks and clones** to give multiple team members access to the shared code repository and avoid code development conflicts
 - GIT publishes **releases** - a stable code base - ready for distribution/production

gitHUB Access

- **GIT** - source code control is a version **control** system designed to track changes in **source code** and other text files during the development of a piece of software. This allows the user to retrieve any of the previous versions of the original **source code** and the changes which are stored.

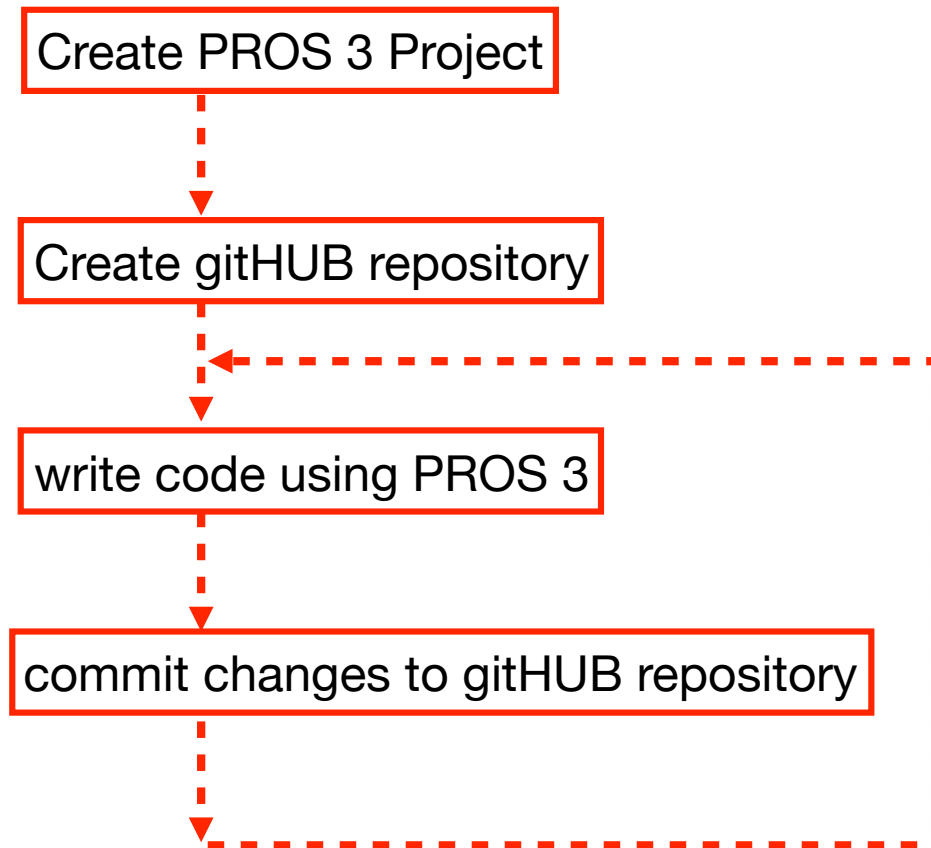


gitHUB Access

- A **repository**, or Git project, encompasses the entire collection of files and folders associated with a project, along with each file's revision history.
- The **file history** appears as snapshots in time called **commits**, and the commits exist as a linked-list relationship, and can be organized into **multiple lines of development** called **branches**.

gitHUB Access

Simple workflow of code development using gitHUB

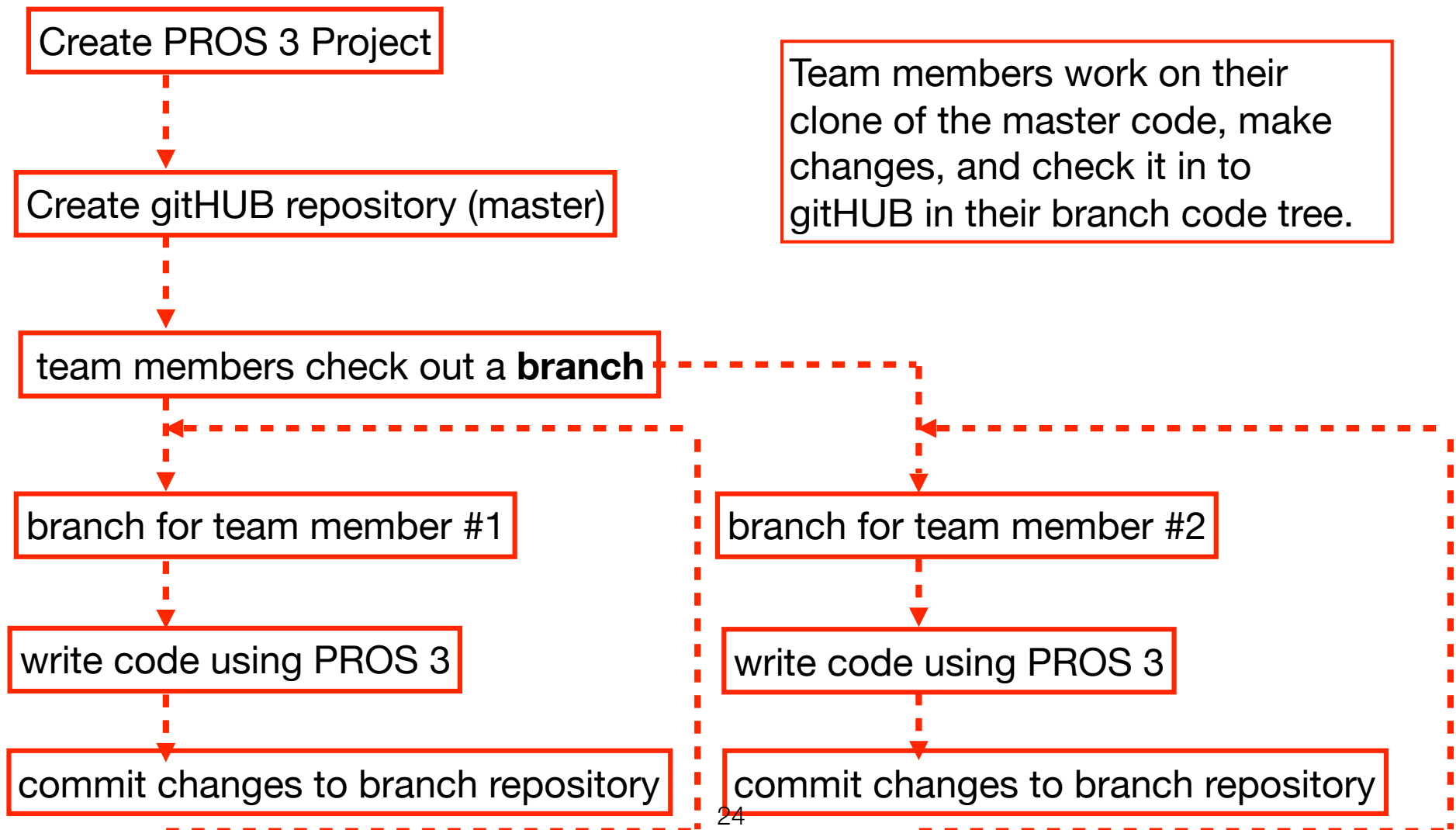


This method works well when it is a single person working on the code, it allows you to track your changes, publish simple releases (v1.0, 1.01 etc) to track your code progress.

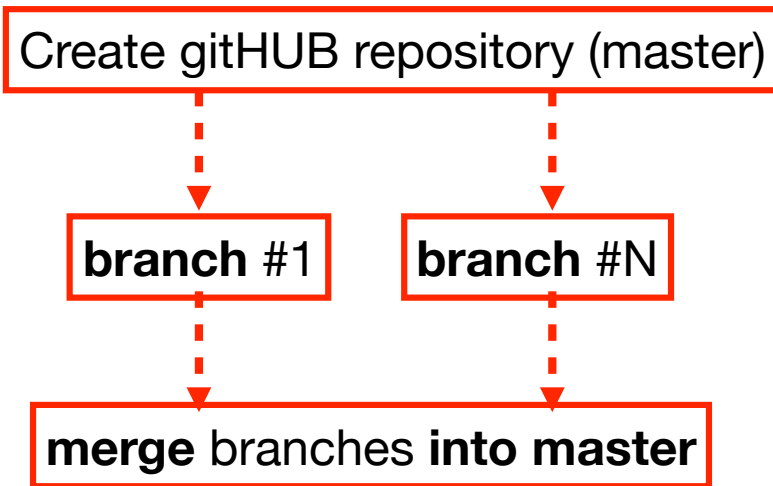
gitHUB Access

- Using **gitHUB** with **multiple developers** working on the same code base - this is where gitHUB / GIT's strength comes in, allowing each developer to work on the code base independently - **branches** - and then **merging** all the code together to a new agreed upon **master** version to then be **released**.
- gitHUB helps with code conflict resolutions - two or more developers submitting conflicting changes which need to be resolved.

gitHUB Access



gitHUB Access



During the merge into the master, if there are conflicts between branches, they must be resolved first prior to the merge being able to succeed.

The new master after merge will represent all agreed upon code merges.

gitHUB Access

- Once branches are **merged** into the master, one of two things can happen:
 - team members check out a **new branch** based on the newly created master
 - a **release** is created

gitHUB Access

- When to create a **release**:
 - When there is **solid** agreed upon **code base** which can be handed over **to testing**
 - Code should always be **released** for deployment to a **competition day robot**, so that any observations and new code designs can be implemented on a well defined **check point** during the development cycle.
 - Release are **solid checkpoints** you can **roll-back** to

gitHUB Access

- A **release**:
 - A **release** has a **Major** number and **Minor** Number, for example V1.0 - indicating first full release based on the **specification**.
 - Code fixed or enhanced based still on the **same specifications**, become **minor release** increments, for example V1.1, V1.2 or V1.0.1, V1.0.2
 - Code which is written as a **subsequent release** based on **new specification** should increase the Major number, for example: V2.0

gitHUB Access

- To **Fork** or **Clone** a project from gitHUB?
 - **Clone** - if you are part of a small team and you are all working on a shared agreed upon code project - typical for a VEX robotics team.
 - **Fork** - when you want to use a codebase in a repository as a starting point for your own subsequent code base - typical in a classroom - teachers code base is forked to cached students account, and then students use that to create their own code using standard clone/branches/merge etc

gitHUB Access

- **Forked projects** can be offered back to the original repository owner and changes pulled into the master repository if so decided. This is a **common** practice in **large scale Open Source code** development scenarios.

gitHUB Access

According to gitHUB:

Creating a “fork” is producing a personal copy of someone else’s project. Forks act as a sort of bridge between the original repository and your personal copy. You can submit *Pull Requests* to help make other people’s projects better by offering your changes up to the original project. **Forking is at the core of social coding at GitHub.**

gitHUB Access

Navigate to the repository you want to fork: <https://github.com/sprobotics/>

The screenshot shows a web browser with three tabs: 'Vision Sensor C++ API — PROS f', 'Using Color Codes - CMUcam5 P', and 'Seattle Preparatory School Robot'. The address bar shows 'GitHub, Inc. [US] | https://github.com/sprobotics'. The GitHub navigation bar includes a search bar, 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main header for the 'Seattle Preparatory School Robotics Program' repository is displayed, featuring the school's logo, the repository name, and a description: 'Software Development Repository for the Seattle Prep Robotics Program'. Below the header, there are statistics for 'Repositories 2', 'People 0', and 'Projects 0'. A search bar and filters for 'Type: All' and 'Language: All' are present. The repository list shows two items: 'cortex-demo01' (Cortex Demo project for teaching V1.0, updated an hour ago) and 'cortex-lcd' (Cortex LCD Chooser Example, updated 6 days ago). On the right, a sidebar shows 'Top languages' with 'C' as the only language, and 'People' with a message stating 'This organization has no public members. You must be a member to see who's a part of this organization.'

Seattle Preparatory School Robotics Program

Software Development Repository for the Seattle Prep Robotics Program

Seattle, WA 98119 | wscholten@seaprep.org

Repositories 2 | People 0 | Projects 0

Find a repository... | Type: All | Language: All

cortex-demo01
Cortex Demo project for teaching V1.0
C 1 Updated an hour ago

cortex-lcd
Cortex LCD Chooser Example
C Updated 6 days ago

Top languages
C

People 0 >
This organization has no public members.
You must be a member to see who's a part of this organization.

gitHUB Access

Select a repository you want to fork,
Then press the **'Fork'** button to fork the project to your personal GitHub account/repository

The screenshot shows the GitHub web interface for the repository `sprobotics / cortex-demo01`. The browser's address bar shows the URL `https://github.com/sprobotics/cortex-demo01`. The repository page includes a header with the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the repository name, there are buttons for Watch (1), Star (0), and Fork (0). The Fork button is highlighted with a red rectangle. Below the repository name, there are tabs for Code, Pull requests (0), Projects (0), Wiki, and Insights. The main content area shows the repository description "Cortex Demo project for teaching V1.0" and a summary of repository statistics: 8 commits, 3 branches, 2 releases, and 2 contributors. At the bottom, there are buttons for "Branch: master", "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". A commit history table is visible at the bottom, showing a commit by `seaprep` titled "Update README.md" and a file `.vscode/cquery_cached_index` with the message "Added tank to arcade switch".

gitHUB Access

Once the repository has been forked to your account, you will see that the project is now in your account (repository) and is forked from what source.

The screenshot shows a web browser window with the GitHub website. The browser's address bar shows the URL `https://github.com/gtsetup`. The GitHub navigation bar is visible at the top, with links for Pull requests, Issues, Marketplace, and Explore. The user profile for 'gtsetup' is displayed, featuring a green and white pixelated avatar. A 'ProTip!' banner at the top right encourages updating the profile. Below the profile picture, there are buttons for 'Add a bio' and 'Edit profile'. The 'Overview' tab is selected, showing statistics for Repositories (1), Stars (0), Followers (0), and Following (0). A red rounded rectangle highlights the 'Popular repositories' section, which lists the repository 'cortex-demo01'. This repository is noted as being forked from 'sprobotics/cortex-demo01' and is described as a 'Cortex Demo project for teaching V1.0'. The repository language is listed as C.

gtsetup

Add a bio

Edit profile

ProTip! Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you. [Edit profile](#)

Overview Repositories 1 Stars 0 Followers 0 Following 0

Popular repositories

[cortex-demo01](#)
Forked from [sprobotics/cortex-demo01](#)

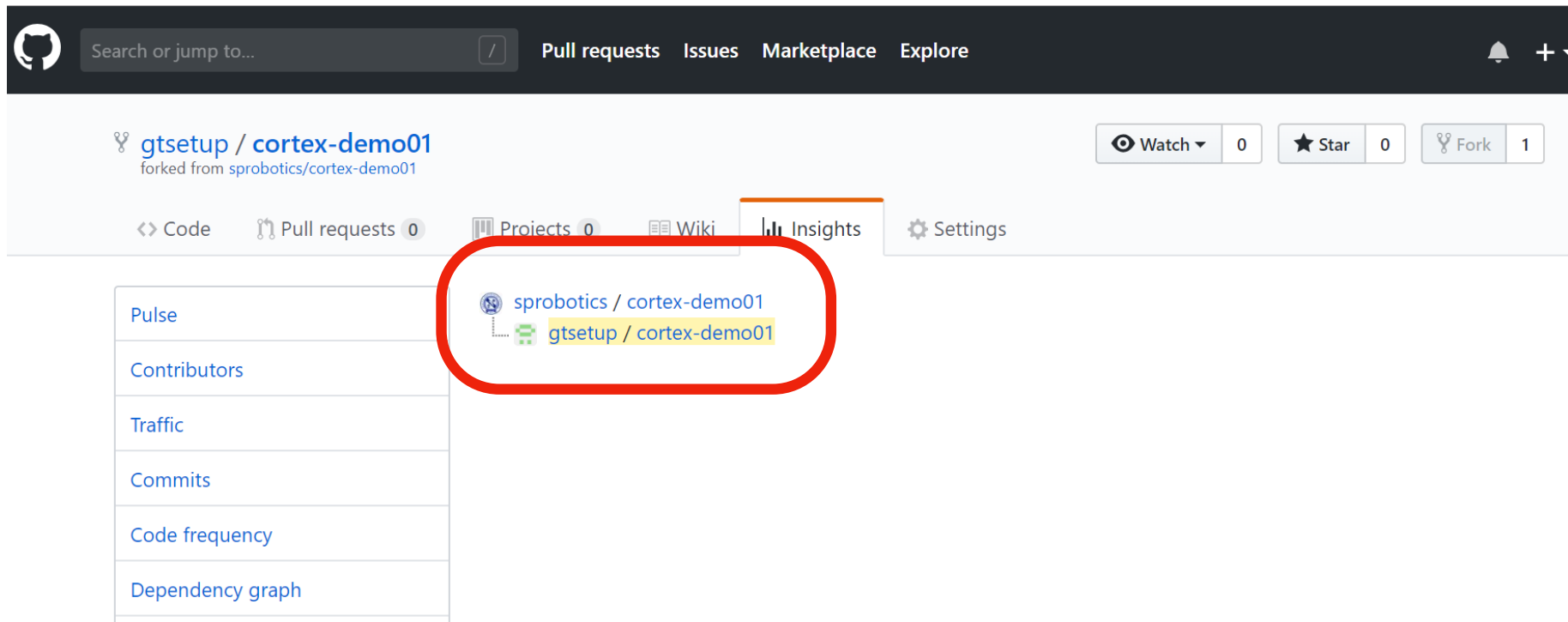
Cortex Demo project for teaching V1.0

C

Customize your pinned repositories

gitHUB Access

Once the repository has been forked to your account, you can click on the **‘Fork’ button** and see that the project is now in your account (repository) and is forked from what source.



gitHUB Access

- You may **FORK** sample code repositories from the following URL at any time: <https://github.com/sprobotics>
- The repositories available there are all set to **read-only**, so your code can not be contributed back to it. They are however intended for students use and starting points of their code.
- Once a project is forked, you use the same technique, checkout a branch, make changes commit etc.
- See for additional help: <https://guides.github.com/activities/forking/>

gitHUB Access

- **Learning more:**
 - <https://lab.github.com/courses>
 - <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>
 - <https://help.github.com>

gitHUB access in PROS 3

- GIT and gitHUB directly integrate into the PROS 3 development environment through Atom and is installed by default
- When integrated code can be directly checked into the repository, and branches of the code managed
- You must ensure that both the following are set:


```
git config --global user.email "email address"  
git config --global user.name "user name"
```

Forking / Cloning Template PROS Project

FORK A Template Project

News ▾ www-html english-intro...tatus=OnSale Yahoo! Google Maps getmain?req...OY3S905XG9 YouTube Wikipedia jetpack_J4t_install.htm

melodic/Installation - ROS W... VEX Robotics Polycarbonate... UP board kernel support : u... librealsense/distribution_linu... Tran

 **Seattle Preparatory School Robotics Program**
Software Development Repository for the Seattle Prep Robotics Program
Seattle, WA 98119 ✉ wscholten@seaprep.org

Repositories 27 People 11 Teams 3 Projects 0 Settings

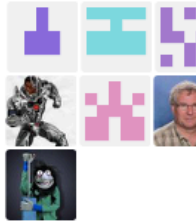
Find a repository... Type: All Language: All Customize pinned re

98333C Private
Contains all of the source code for the cortex robot for the 98333C team.
Updated 3 days ago

prosv5-98333B Private
Team 98333B competition Code
C++ Updated 3 days ago

cortex-template
Cortex Robot Software Template
C 1 Updated 4 days ago

Top languages
C C++

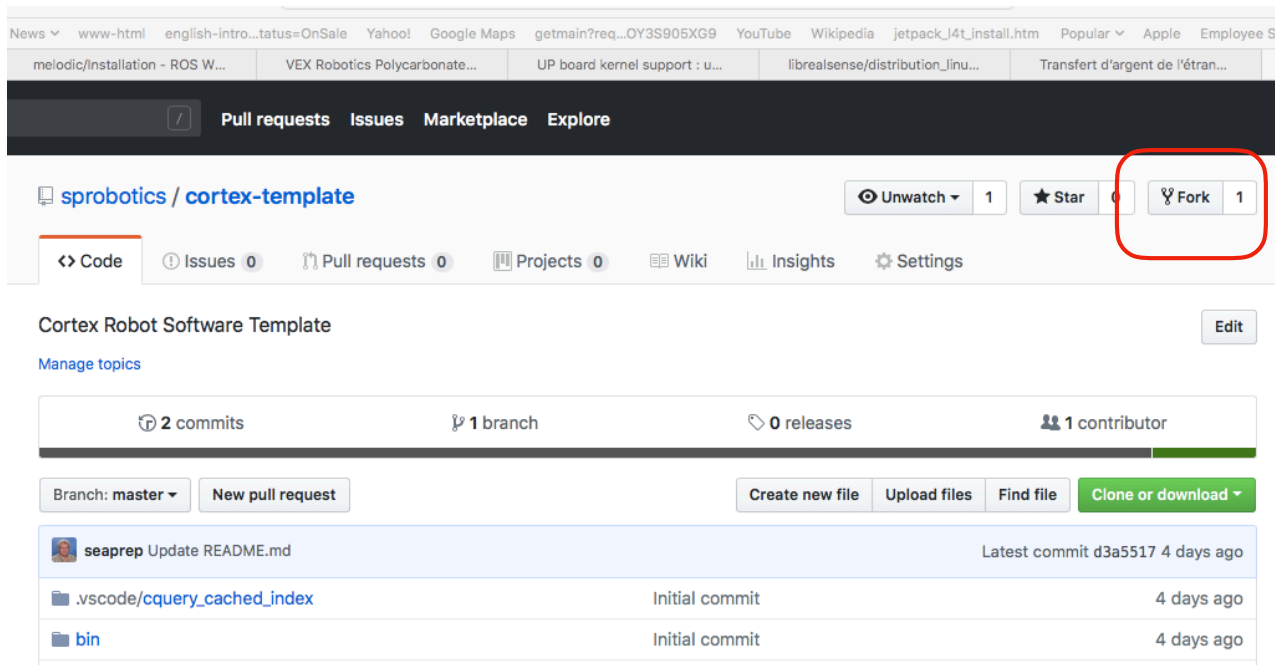
People

Invite someone

Step 1: login in to your gitHUB account

Step 2: Goto [GitHub.com/sprobotics](https://github.com/sprobotics)

Step 3: Find the repo: cortex-template

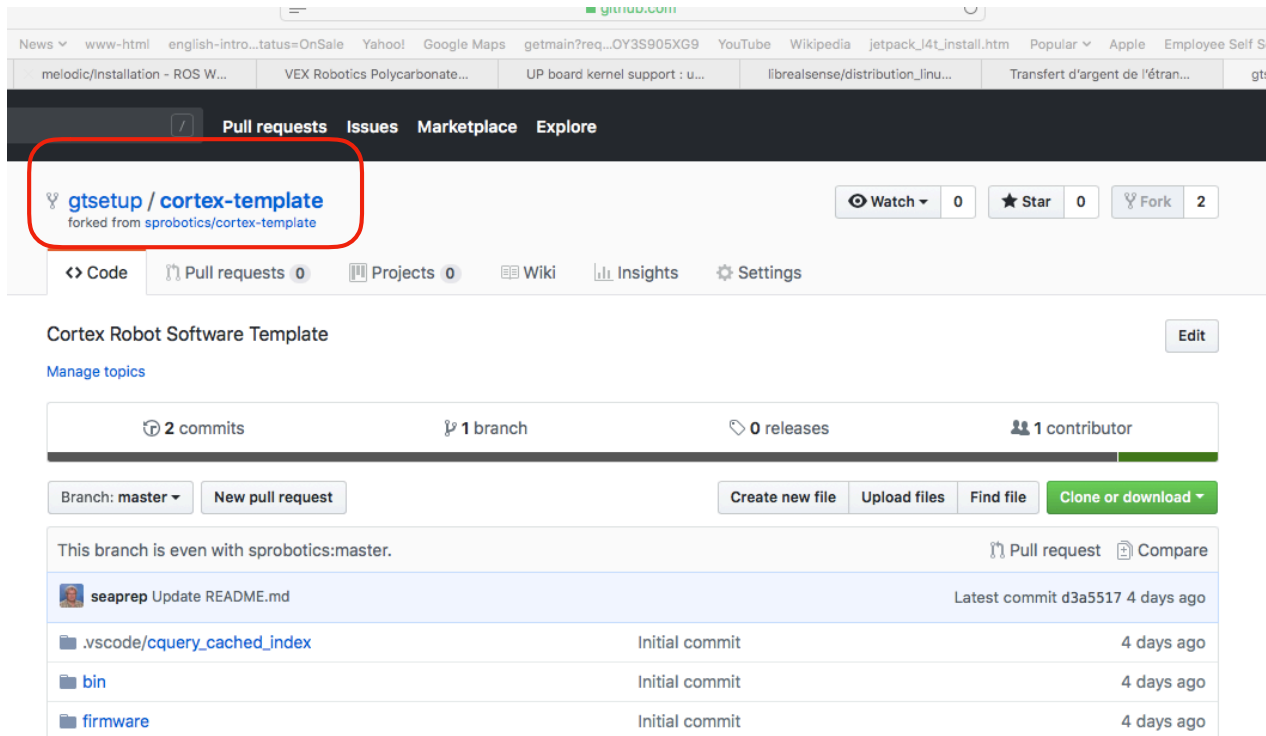
FORK A Template Project



Step 1: Click **Fork**

Step 2: Watch and follow prompts, project will be forked into your repo's

FORK A Template Project



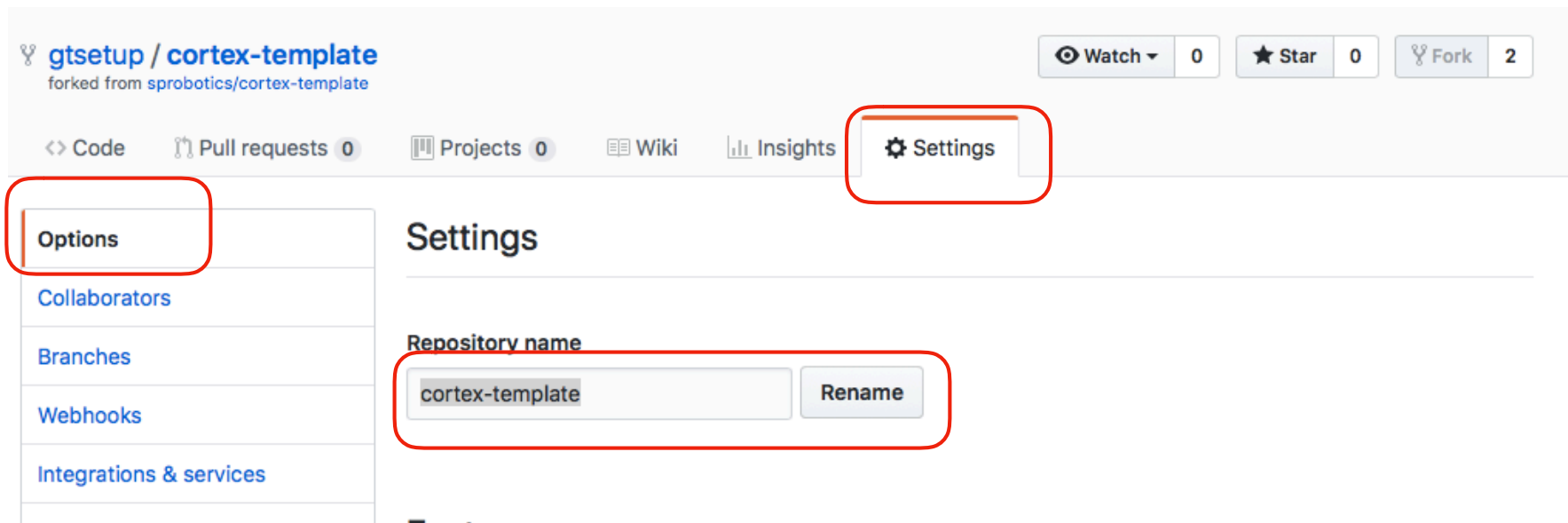
When successful you will see your Repos and the Forked project open in your GitHub repo.

FORK A Template Project

Step 1: Now click “Settings” for the cortex-template repo

Step 2: Goto “Options” and “Settings”

Step 3: Goto the Repo Renaming box



FORK A Template Project

gtsetup / cortex-template

forked from sprobotics/cortex-template

<> Code

Pull requests 0

Projects 0

Wiki

Insights

Settings

Options

Collaborators

Branches

Webhooks

Settings

Repository name

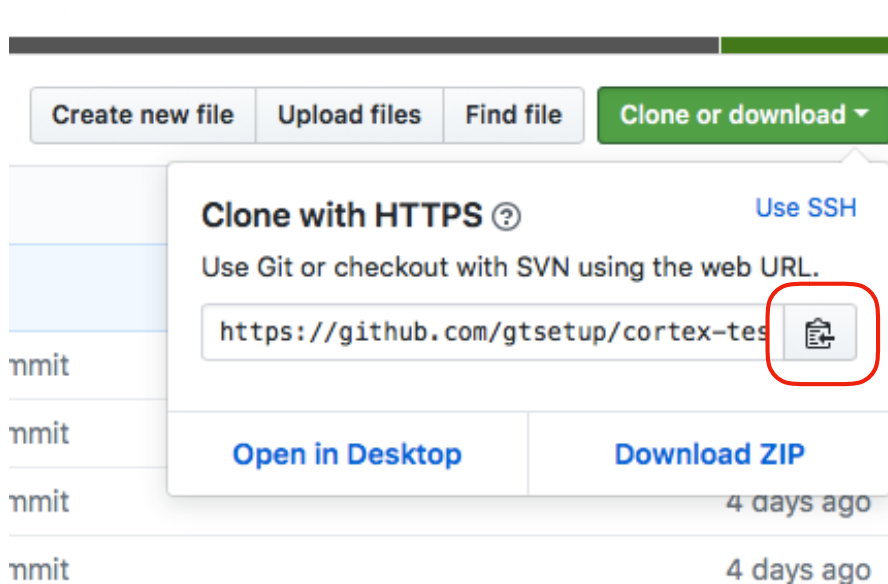
cortex-testCWU

Rename

Step 1: Change name and press “Rename”

Step 2: You will be taken back to main repo page, but will notice repo name has changed to your new name

Clone A Forked Template



Step 1: Now get ready to clone repository onto your workstation to work with it.

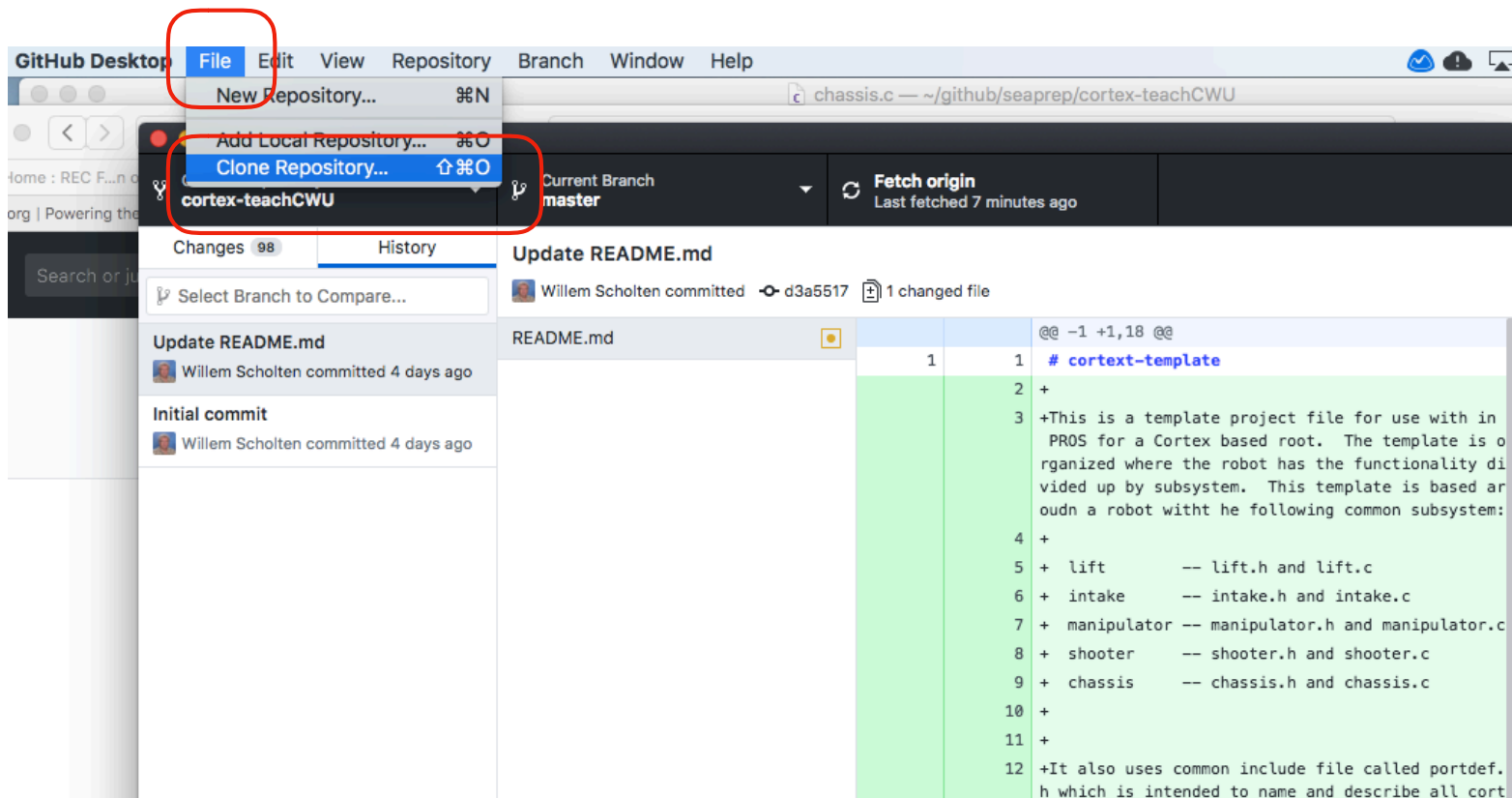
Step 2: Click “clone or download”

Step 3: Click on clipboard to grab the URL

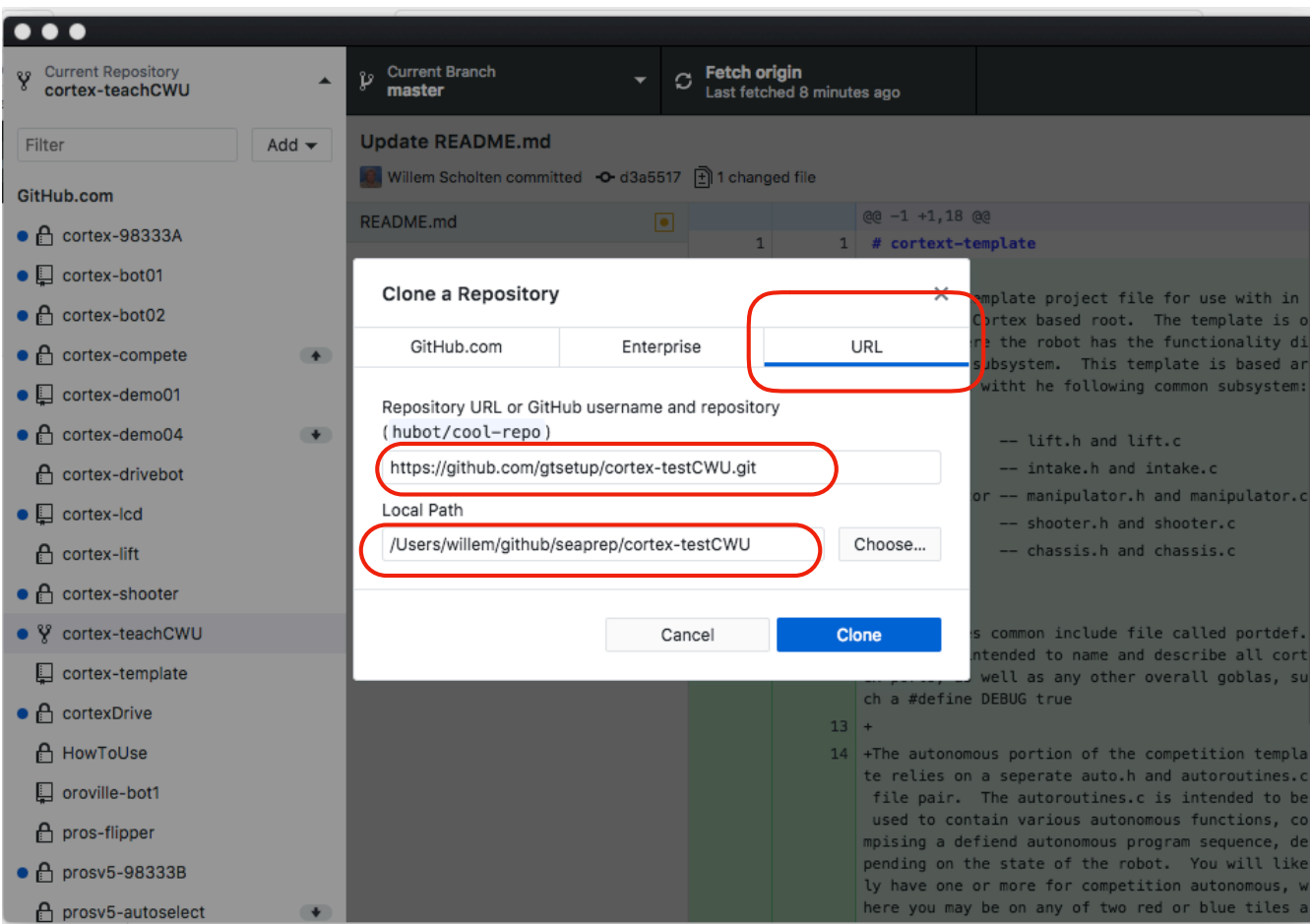
Clone A Forked Template

Step 1: Open gitHUB client on your workstation

Step 2: Goto File - Clone repository



Clone A Forked Template



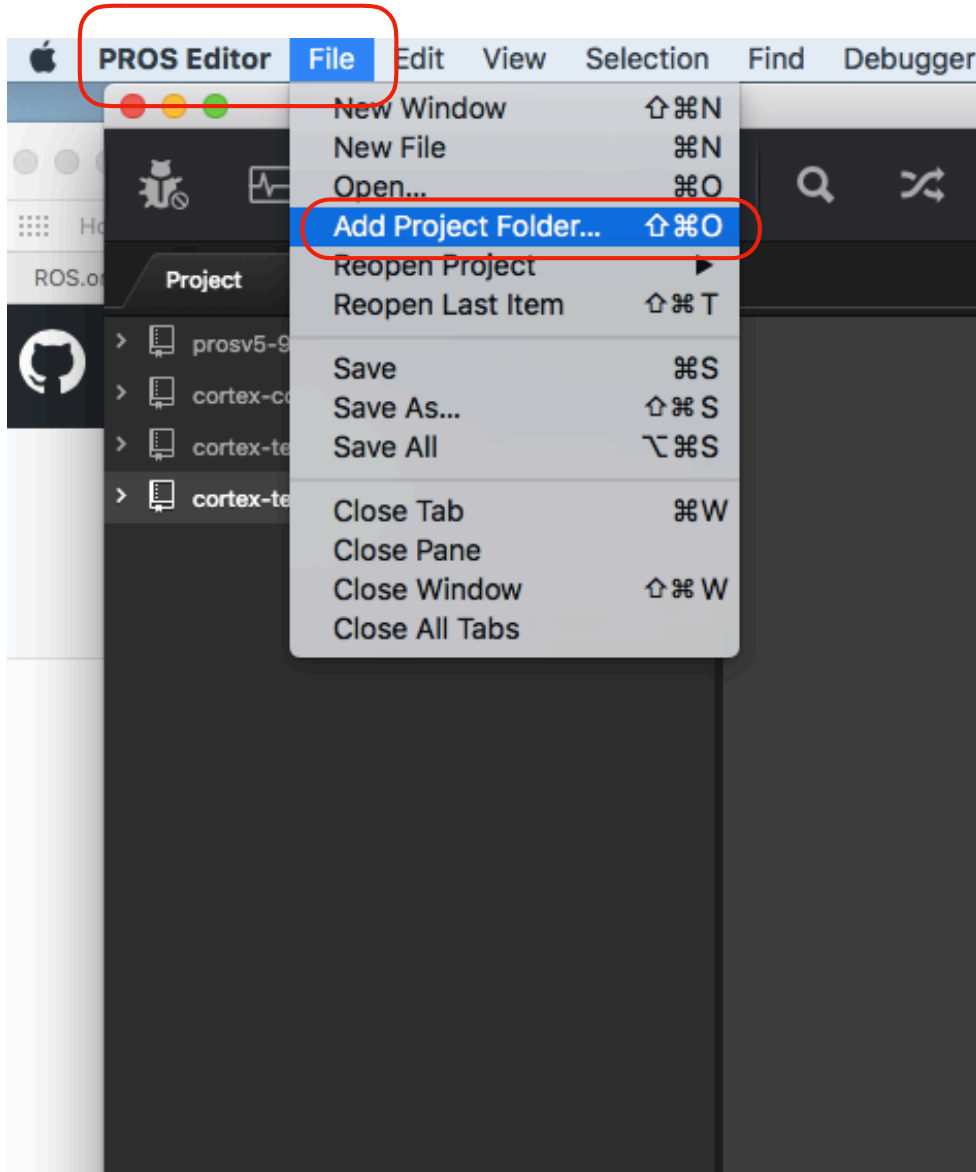
Step 1: Click on the Clone by “URL” tab

Step 2: Paste the URL from the clipboard in the repository URL box

Step 3: Decide where to put the repository on your local workstation

Step 4: Click Clone

Clone A Forked Template

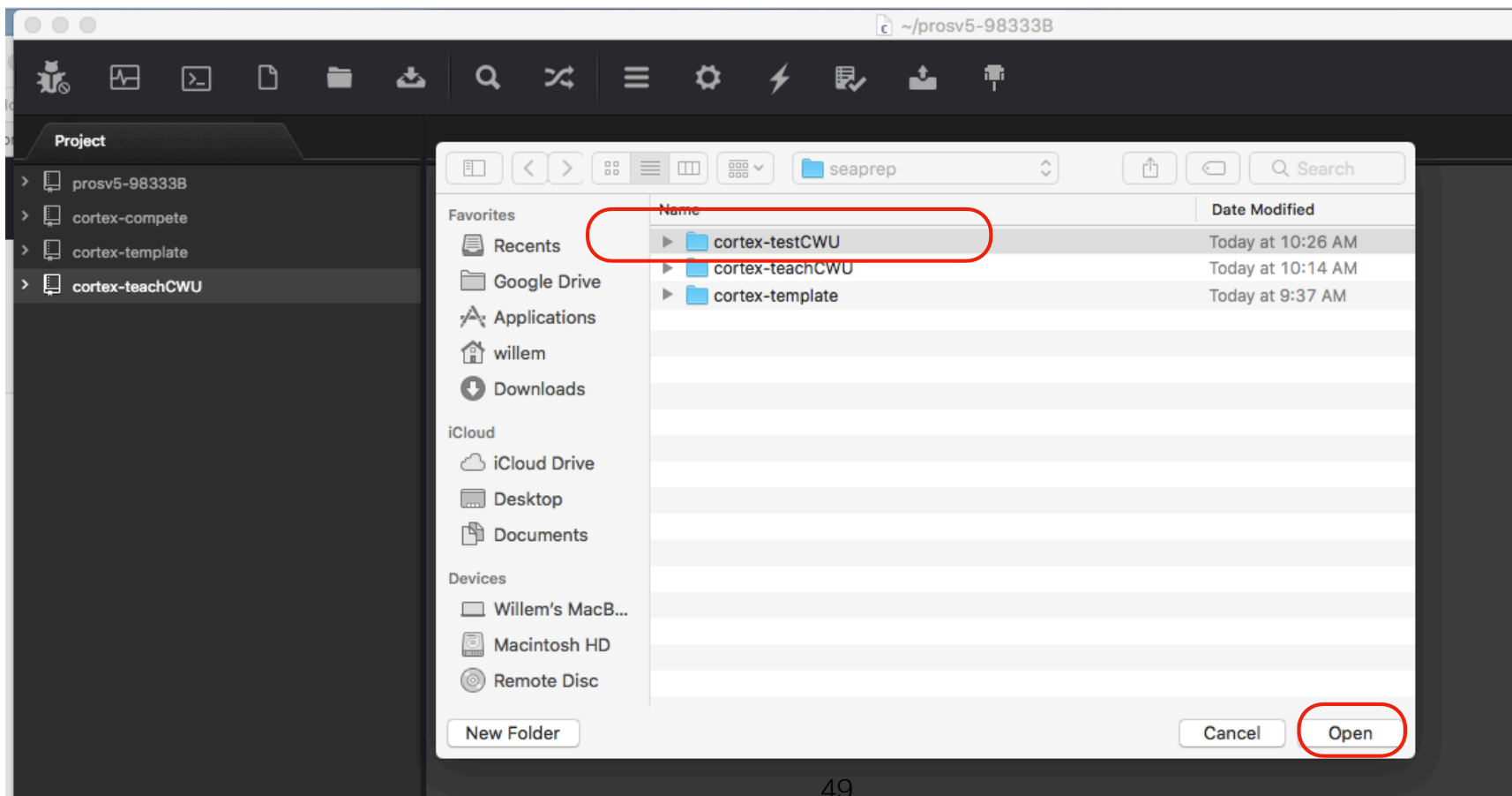


Step 1: Goto Atom PROS

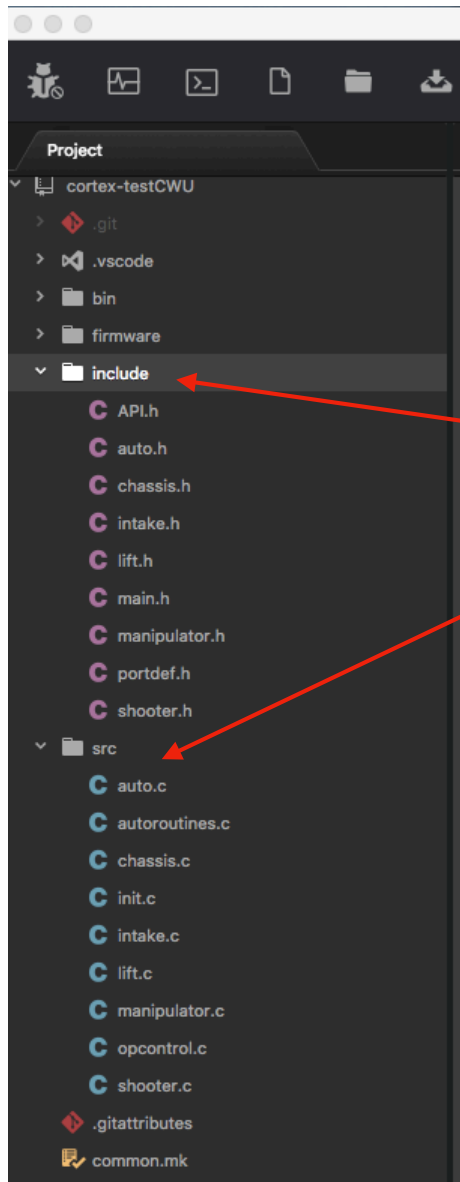
Step 2: Click on File - Add Project Folder

Clone A Forked Template

Step 1: Browse to the repository folder you just cloned and add it



Clone A Forked Template



When the project is added, you can expand the >src and >include tree and all the template field are there for you to modify, add-to, expand etc.

Keeping a FORK updated with upstream repository

FORK update

- Once you have a FORK'ed project repository, there maybe times you want your FORK to be synchronized with the original project you FORK'ed from
- This process must be undertaken with care, but can be useful in case a new library module or other significant improvements have been posted in the originating repository

FORK update

- FORK update process consists of three steps:
 - (1) — do a merge from original of FORK (source repository) to your FORK repository using github web client (<http://github.com>)
 - (2) — Using gitHUB desktop client sync your local development machine copy with the online repository
 - (3) — Make sure PROS has access to the newly updated project repository master

FORK update

Step 1: Goto your repository on gitHUB (on the web)

Step 2: Click on “compare” — this will start the process for you to compare your FORK against the source of your FORK

The screenshot shows the GitHub interface for a repository named 'gtsetup / cortex-teachCWU', which is a fork of 'sprobotics/cortex-teachCWU'. The repository has 0 Watchers, 0 Stars, and 2 Forks. The main navigation bar includes links for Code, Pull requests (0), Projects (0), Wiki, Insights, and Settings. The repository description is 'Cortex CWU MET class teampate project' with an 'Edit' button. Below this, statistics show 8 commits, 1 branch, 0 releases, and 2 contributors. A bar at the bottom indicates the current branch is 'master'. Action buttons include 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A status message states 'This branch is 1 commit ahead, 3 commits behind sprobotics:master.' To the right of this message are links for 'Pull request' and 'Compare', with the 'Compare' link highlighted by a red circle. A recent activity entry shows a merge pull request from 'sprobotics/master' by 'gtsetup' with the latest commit '7b909b4' made 18 hours ago.

FORK update

Step 1: Notice that it starts off with wanting to do a FORK compare in the wrong direction - comparing the original FORK against your FORK. We want to do the opposite - comparing your FORK with the original FORK

sprobotics / cortex-teachCWU

Watch 1

Star 0

Fork 2

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: sprobotics/cortex-teachCWU

base: master

head repository: gtsetup/cortex-teachCWU

compare: master

✓ Able to merge. These branches can be automatically merged.

Create pull request

Discuss and review the changes in this comparison with others.

1 commit

0 files changed

0 commit comments

1 contributor

Commits on Jan 21, 2019

gtsetup

Merge pull request #1 from sprobotics/master

Verified 7b909b4

FORK update

Step 1: Change the base repository to your FORK'ed repository

Step 2: Click the “compare across forks”

Step 3: Change the head repository to the source FORK

The screenshot shows a GitHub repository page for `gtsetup / cortex-teachCWU`, which is forked from `sprobotics/cortex-teachCWU`. The page includes navigation tabs for Code, Pull requests, Projects, Wiki, Insights, and Settings. The main heading is "Comparing changes", with a subtext that says "Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#)." Below this, there is a comparison bar with four dropdown menus: "base repository: gtsetup/cortex-teachCWU", "base: master", "head repository: sprobotics/cortex-teachCWU", and "compare: master". A green checkmark indicates "Able to merge. These branches can be automatically merged." Below the comparison bar is a yellow box with a green "Create pull request" button and the text "Discuss and review the changes in this comparison with others." At the bottom, there is a summary bar showing "3 commits", "19 files changed", "0 commit comments", and "1 contributor". Below this, a list of commits is shown, including "seaprep" with the commit message "fiexd PID turnign and drive functions" and "Minor tweak in autonomous call".

gtsetup / cortex-teachCWU
forked from sprobotics/cortex-teachCWU

Watch 0 Star 0 Fork 2

Code Pull requests 0 Projects 0 Wiki Insights Settings

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: gtsetup/cortex-teachCWU base: master head repository: sprobotics/cortex-teachCWU compare: master

✓ Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others.

3 commits 19 files changed 0 commit comments 1 contributor

Commits on Jan 21, 2019

seaprep fiexd PID turnign and drive functions ... 838e45a

seaprep Minor tweak in autonomous call ed2a830

FORK update

Step 1: You should now see “the create pull request” button to start a pull from the FORK source repository to be merged into your repository
Step 2: Click “Create Pull request”

The screenshot shows the GitHub interface for a pull request comparison. At the top, the repository is identified as `gtsetup / cortex-teachCWU`, forked from `sprobotics/cortex-teachCWU`. It has 0 watches, 0 stars, and 2 forks. The navigation bar includes links for Code, Pull requests (0), Projects (0), Wiki, Insights, and Settings.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

The comparison section shows the base repository as `gtsetup/cortex-teachCWU` with the `base: master` branch, and the head repository as `sprobotics/cortex-teachCWU` with the `compare: master` branch. A green checkmark indicates the branches are "Able to merge".

A prominent green button labeled "Create pull request" is highlighted with a red box. Below it, a yellow banner contains the text "Discuss and review the changes in this comparison with others." and a help icon.

Summary statistics at the bottom show 3 commits, 19 files changed, 0 commit comments, and 1 contributor.

The commit history for January 21, 2019, is listed below:

- Commit by `seaprep`: `fiexd PID turnign and drive functions` (hash `838e45a`)
- Commit by `seaprep`: `Minor tweak in autonomous cal` (hash `ed2a830`)

The number 57 is written at the bottom center of the page.

FORK update

Step 1: Give a short reason for the pull request for your records
Step 2: Click “Create Pull request”

gtsetup / cortex-teachCWU
forked from sprobotics/cortex-teachCWU

Watch 0 Star 0 Fork 2

Code Pull requests 0 Projects 0 Wiki Insights Settings

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: gtsetup/cortex-teachCWU base: master head repository: sprobotics/cortex-teachCWU compare: master

✓ Able to merge. These branches can be automatically merged.

Merge in chnages from MASTER template

Write Preview

Leave a comment

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

Styling with Markdown is supported

Create pull request

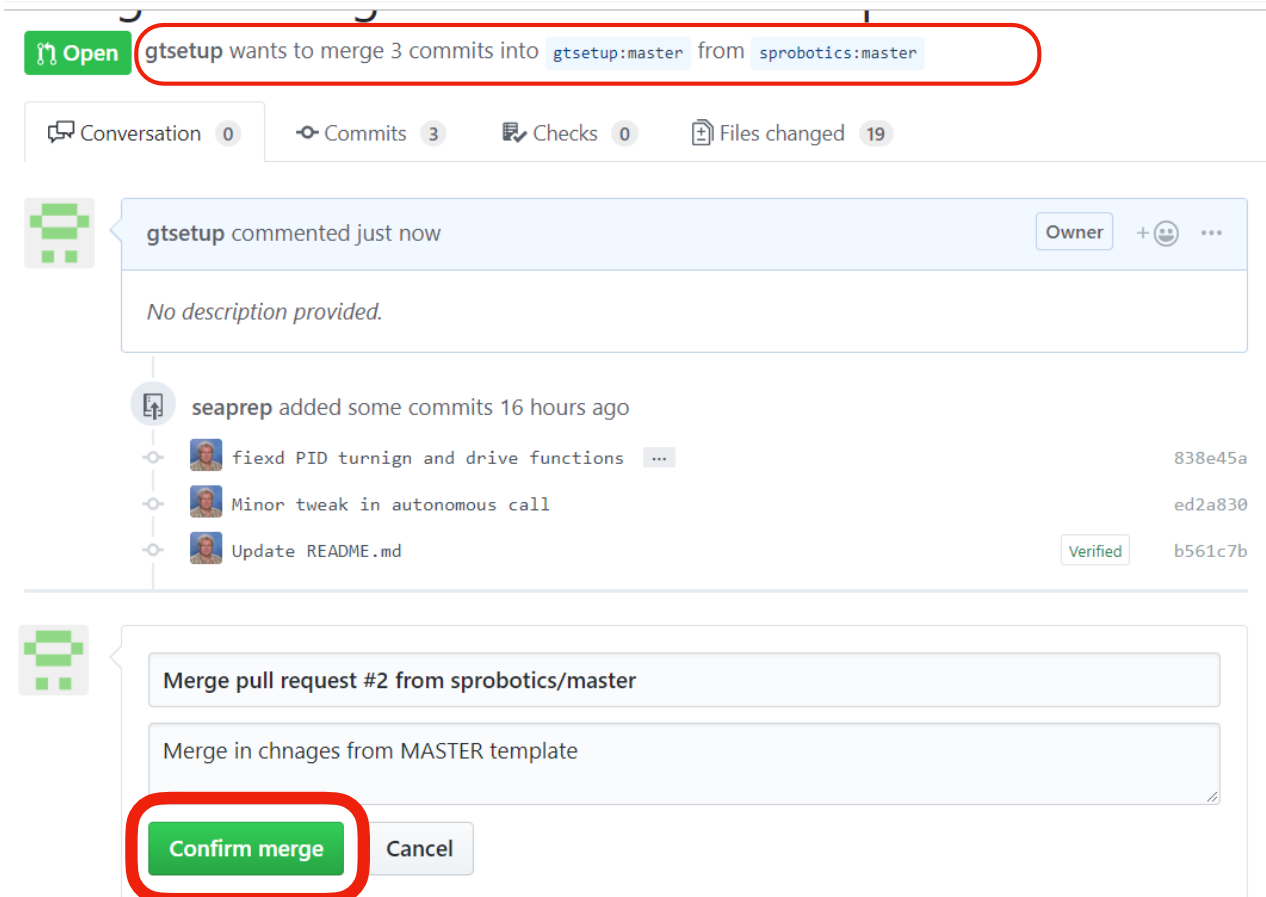
FORK update

Step 1: In this case 3 commits to the master repository - FORK original - are ready to be merged into your repository — in some case conflicts may occur they will be shown and you will have an opportunity to accept, edit or deny the changes
Step 2: Click “Merge Pull request”

The screenshot displays a GitHub pull request interface. At the top, a green 'Open' button is next to a summary: 'gtsetup wants to merge 3 commits into gtsetup:master from sprobotics:master'. Below this, navigation tabs show 'Conversation' (0), 'Commits' (3), 'Checks' (0), and 'Files changed' (19). A comment from 'gtsetup' is shown with the text 'No description provided.' Below the comment, a list of commits is displayed, including 'fiexd PID turnign and drive functions', 'Minor tweak in autonomous call', and 'Update README.md'. A green box highlights a message: 'Continuous integration has not been set up' and 'This branch has no conflicts with the base branch'. At the bottom, a green button labeled 'Merge pull request' is highlighted with a red box. To the right of the button, text reads: 'You can also [open this in GitHub Desktop](#) or view [command line instructions](#).'

FORK update

Step 1: You will be asked now to “Confirm Merge” by pressing the shown button.



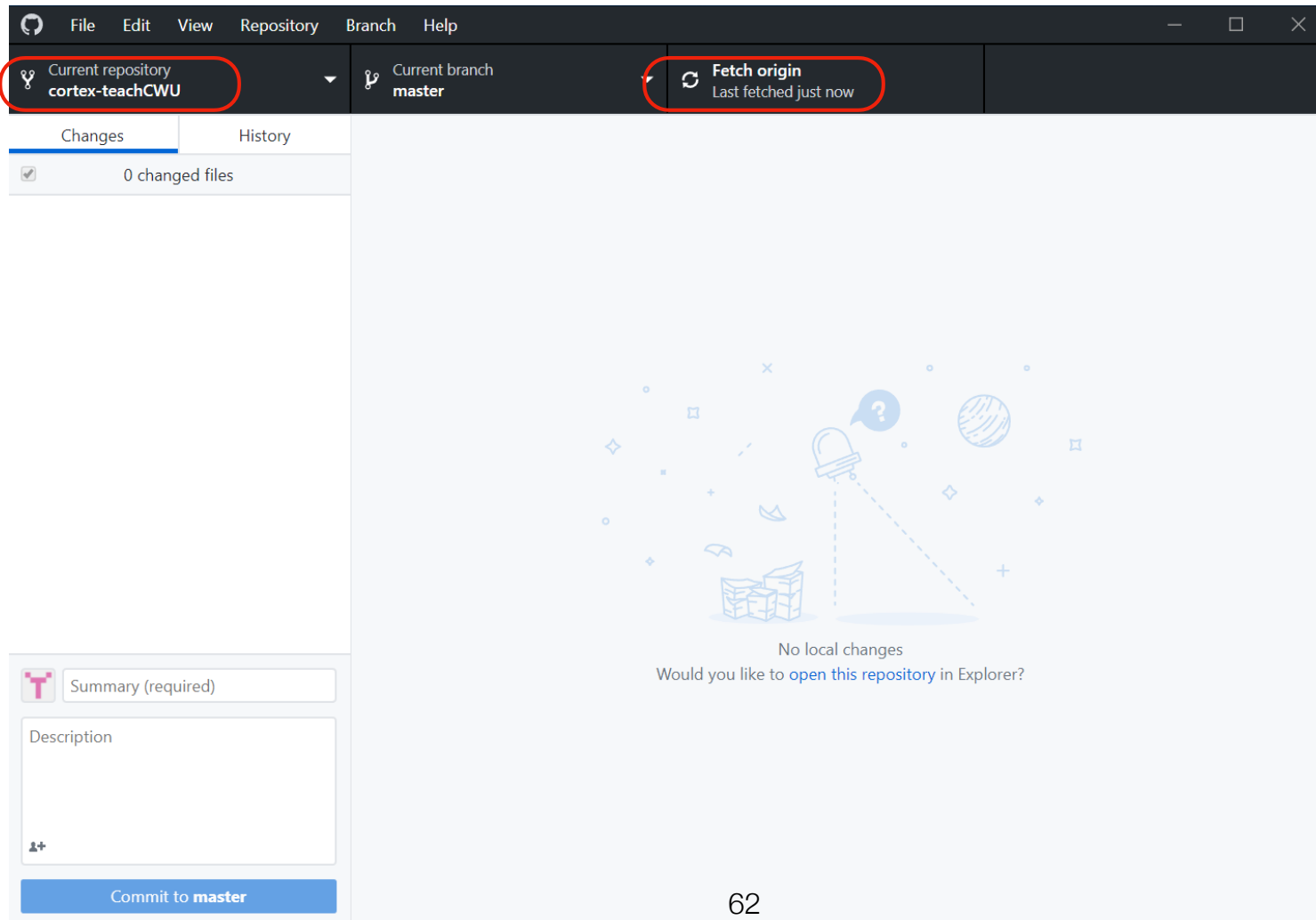
FORK update

- Before you refresh your local copy of the repository on your computer, (using local gitHUB client) you are advised to **ensure that PROS has none of the source files open in your current local repository copy.**

FORK update

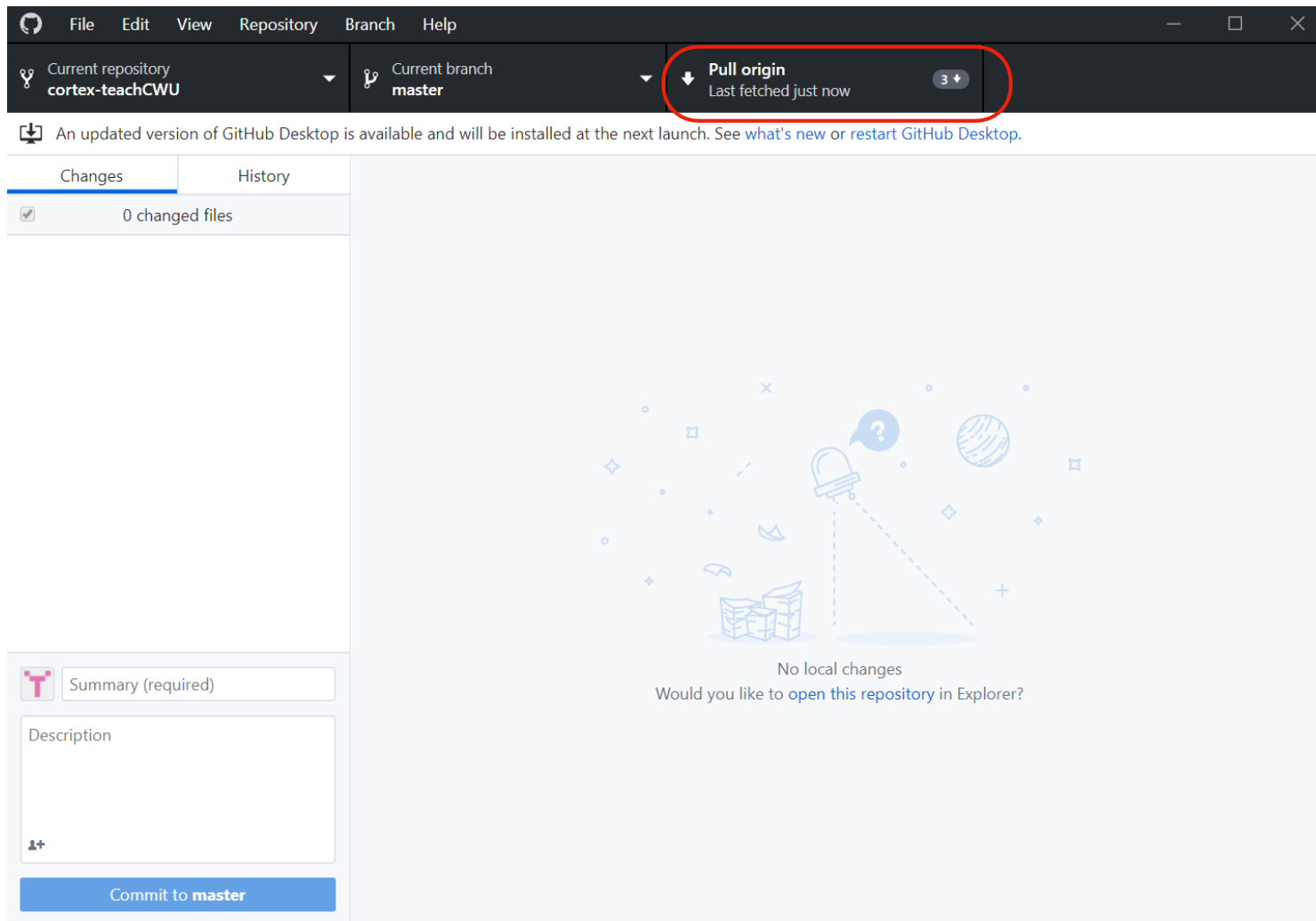
Step 1: Make sure you point to your local repository

Step 2: Refresh “Fetch from Origin”



FORK update

Step 1: You should see a “Pull origin” as your master repository has changed compared to your local copy
Step 2: Click “Pull Origin” to grab the updates for your local copy



FORK update

Step 1: Your changes should be pulled in, you can check this activity by clicking on the “History” tab.

You now will have a new refreshed master copy with reflects the pulled in changes from the upstream FORK original repository.

The screenshot shows the GitHub Desktop application interface. At the top, the menu bar includes File, Edit, View, Repository, Branch, and Help. Below the menu, the 'Current repository' is set to 'cortex-teachCWU' and the 'Current branch' is 'master'. A 'Fetch origin' button is visible, indicating the last fetch was 'just now'. A notification banner states: 'An updated version of GitHub Desktop is available and will be installed at the next launch. See [what's new](#) or [restart GitHub Desktop](#).' The main area is divided into two panes. The left pane shows the 'History' tab, listing recent commits and merges, including 'Merge pull request #2 from sprobotics/master' and 'Merge pull request #3 from sprobotics/master'. The right pane shows a diff view for the selected commit, comparing the current state with the previous one. The diff shows changes in several files, including .vscode\cquery_cache...\include@chassis.h, .vscode\cquery_cache...\src@auto.c, .vscode\cquery_cache...\src@chassis.c, .vscode\cquery_cache...\src@init.c, README.md, bin\auto.c.o, bin\chassis.c.o, bin\init.c.o, bin\output.bin, bin\output.elf, compile_commands.json, and include\chassis.h. The diff highlights changes in line numbers and content, such as the addition of a new line in the README.md file and changes in the .vscode\cquery_cache...\src@chassis.c file.

Current repository: cortex-teachCWU

Current branch: master

Fetch origin: Last fetched just now

Changes: History

Select branch to compare...

Merge pull request #2 from sprobotics/master

gtsetup committed 863f213 19 changed files

Merge in changes from MASTER template

File	Line	Diff
.vscode\cquery_cache...\include@chassis.h	5	@@ -5,7 +5,8 @@
.vscode\cquery_cache...\include@chassis.h	6	//
.vscode\cquery_cache...\src@auto.c	7	#define WHEEL_DIAMETER 4 // diameter of drive wheels
.vscode\cquery_cache...\src@auto.c	8	-#define WHEEL_BASE 10 // wheel base in inches left to right wheel
.vscode\cquery_cache...\src@chassis.c	8	+#define WHEEL_BASE 18 // wheel base in inches left to right wheel diagonal measured
.vscode\cquery_cache...\src@chassis.c	9	+ // of chassis.
.vscode\cquery_cache...\src@init.c	9	
.vscode\cquery_cache...\src@init.c	10	#define ARCADE_DRIVE false // when false it runs tank mode, if true
README.md	11	/// we will run ARCADE mode
bin\auto.c.o		@@ -37,4 +38,11 @@ void turnRight(int speed);
bin\chassis.c.o	37	// drive forward with PID tankControl
bin\init.c.o	38	void drivePID(int masterPower);
bin\output.bin	39	
bin\output.elf		41 +// drive with PID control for a given distance in inches
compile_commands.json		42 +void driveForDistancePID(int distance, int speed);
include\chassis.h		43 +
		44 +// make a pivot turn to the right or left for a given

FORK update

Step 1: Once the merge is confirmed your repository will be Merged and the changes applied

Next we will need to synchronize the repository from gitHUB onto your local computer using the gitHUB desktop client

The screenshot shows a GitHub pull request interface. At the top, the repository is identified as 'gtsetup / cortex-teachCWU', forked from 'sprobotics/cortex-teachCWU'. It has 0 Watchers, 0 Stars, and 2 Forks. The navigation bar includes links for Code, Pull requests (1), Projects (0), Wiki, Insights, and Settings. The main heading is 'Merge in changes from MASTER template #2'. Below this, a purple button with a merge icon and the text 'Merged' is highlighted with a red circle. The text next to it says 'gtsetup merged 3 commits into gtsetup:master from sprobotics:master just now'. Below the merge information, there are statistics: 0 Conversations, 3 Commits, 0 Checks, and 19 Files changed, with a net change of +338 -38 lines. At the bottom, a comment from 'gtsetup' is shown, stating 'No description provided.'.

gtsetup / cortex-teachCWU
forked from sprobotics/cortex-teachCWU

Watch 0 Star 0 Fork 2

Code Pull requests 1 Projects 0 Wiki Insights Settings

Merge in changes from MASTER template #2

Merged gtsetup merged 3 commits into gtsetup:master from sprobotics:master just now

Conversation 0 Commits 3 Checks 0 Files changed 19 +338 -38

gtsetup commented 2 minutes ago
No description provided.

Owner Reviewers
No reviews

FORK update

- After the update of the repository on your local computer (using the gitHUB desktop client) you can re-open it in PROS.
- In the PROS project view you should be able to see in specific source files changes which may have come in from the update of your FORK with upstream repository.

GIT Local Repo versus Remote Repo

GIT local V remote Repo

- GIT (and gitHUB) manages **two repositories**:
 - **Local repository** (repo) - the GIT maintained repository on your local development machine(s)
 - **Remote repository** (repo) - the GIT maintained in the cloud repository for central storage, backup and sharing

GIT local V remote Repo

- **Local repository** (repo):
 - Tracks all **changes** on your local development machine
 - Tracks local **commits** - i.e. milestones where you record the state of your local development tree
 - Allows for local changes to be **rolled back**
 - Local repository **can not be shared** with fellow developers unless synchronized with matching remote repository

GIT local V remote Repo

- **Local repository** (repo) - sample GIT actions:
 - **git commit** (commit changes to the repository)
 - **git stash** (*git-stash - Stash the changes in a dirty working directory away see: <https://git-scm.com/docs/git-stash>)*)
 - **git config** (*configure global options for your repositories such a user.name*)
 - **git revert** (*revert to a previous commit*)
 - **git checkout** (*checkout a file from a previous commit - restoring the file state*)
 - **git reset —hard** (*a true rollback of the state of your repo*)

GIT local V remote Repo

- **Remote repository (repo):**
 - Tracks all changes of **multiple** local development repositories
 - Tracks **project wide** commits - i.e. milestones where you record the state of various local development trees merged back into a single master copy for further development
 - Allows for local repositories to be **rolled back** in time, including changes committed by other developers
 - Remote repository **can be shared** with fellow developers as well as shared more broadly to the community

GIT local V remote Repo

- **Remote repository (repo):**
 - The main purpose of the remote repository is:
 - facilitate **team development** of code base
 - provide **backup** for the code base in the cloud
 - provide for **software release** to wider community

GIT local V remote Repo

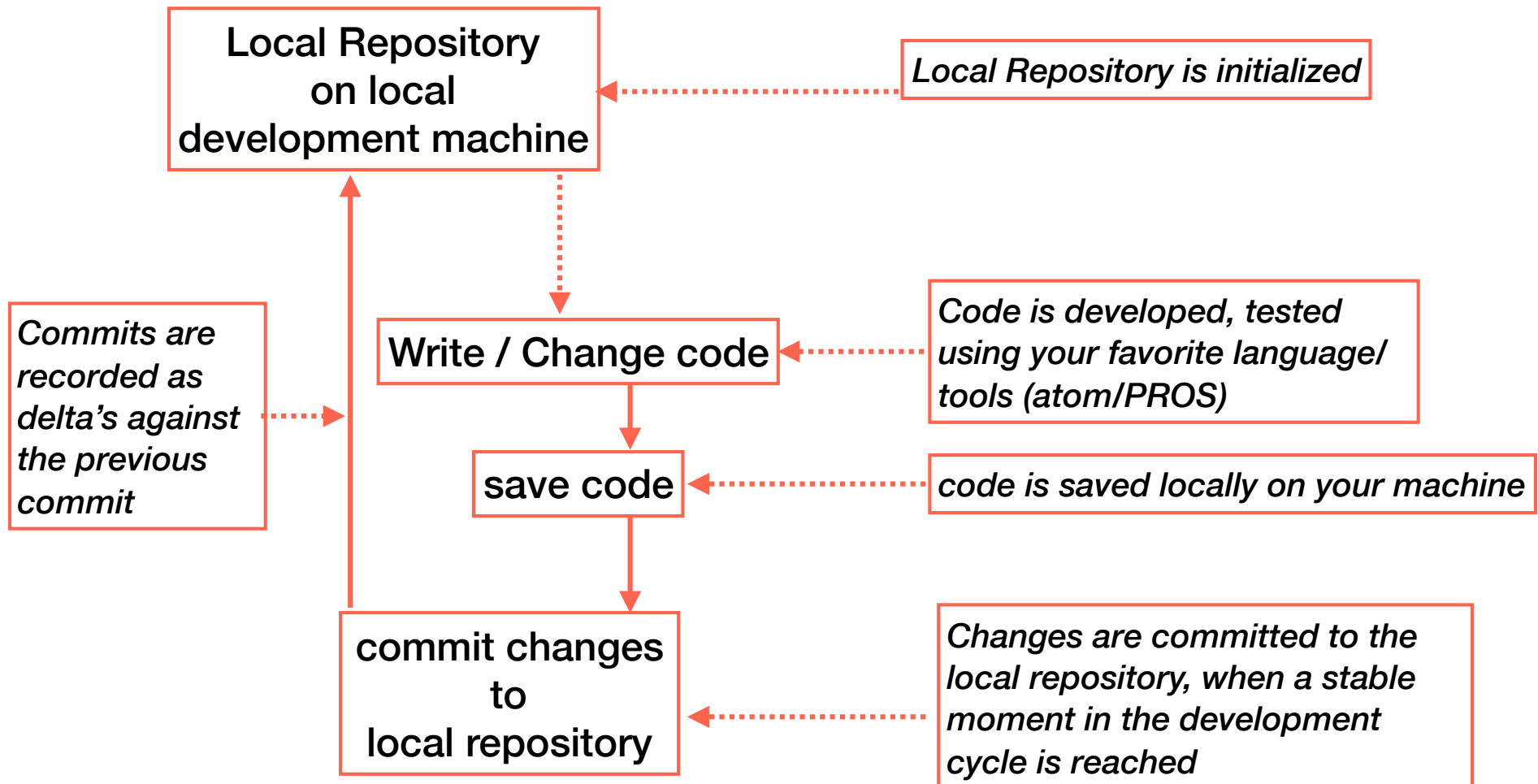
- **Remote repository (repo):**
 - The most common way for establishing a remote repository are:
 - **gitHUB** - a web based / cloud enabled remote repository management system, tightly integrated with GIT - most common platform
 - **Local GIT server** - most often done in larger enterprises or confidential development environments
 - For other options see: <https://www.git-tower.com/blog/git-hosting-services-compared/>

GIT local V remote Repo

- **Remote repository (repo)** - sample GIT actions:
 - **git commit** (commit changes to the repository)
 - **git stash** (*git-stash - Stash the changes in a dirty working directory away see: <https://git-scm.com/docs/git-stash>)*)
 - **git config** (*configure global options for your repositories such a user.name*)
 - **git revert** (*revert to a previous commit*)
 - **git checkout** (*checkout a file from a previous commit - restoring the file state*)
 - **git reset —hard** (*a true rollback of the state of your repo*)

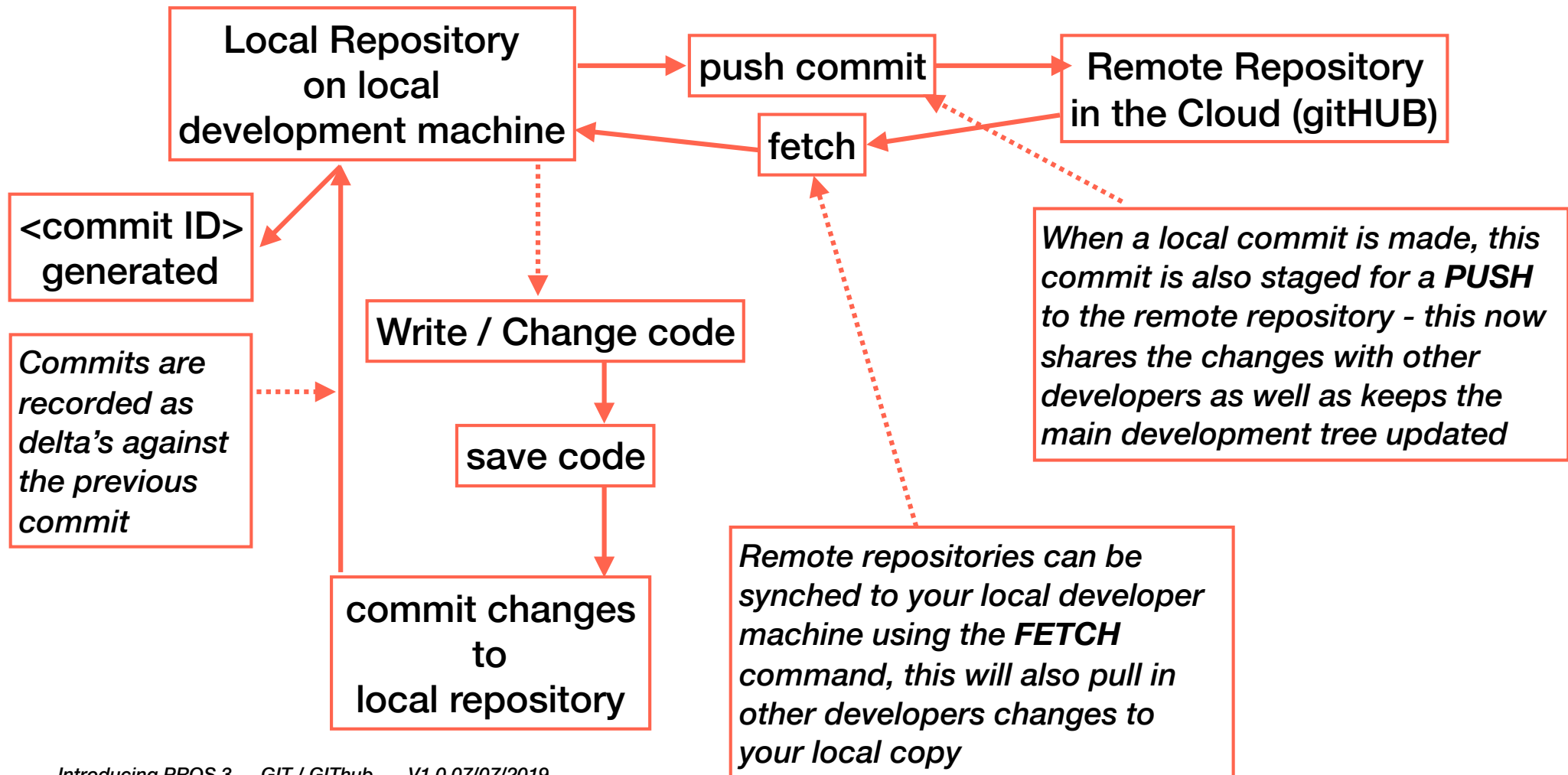
GIT local V remote Repo

Local Machine Development Process



GIT local V remote Repo

Synching local repository with remote repository



GIT local V remote Repo

- It is **important to understand the interaction** between your **local repository** and the **remote repository**
- **Local repository** (repo) - is **your local copy** of your development tree and all changes are recorded but are only visible to you
- **Remote repository** (repo) - your and other developer changes are merged and **made visible to all the developers** on a project.

Dropping a local repository and starting fresh

Dropping Repository

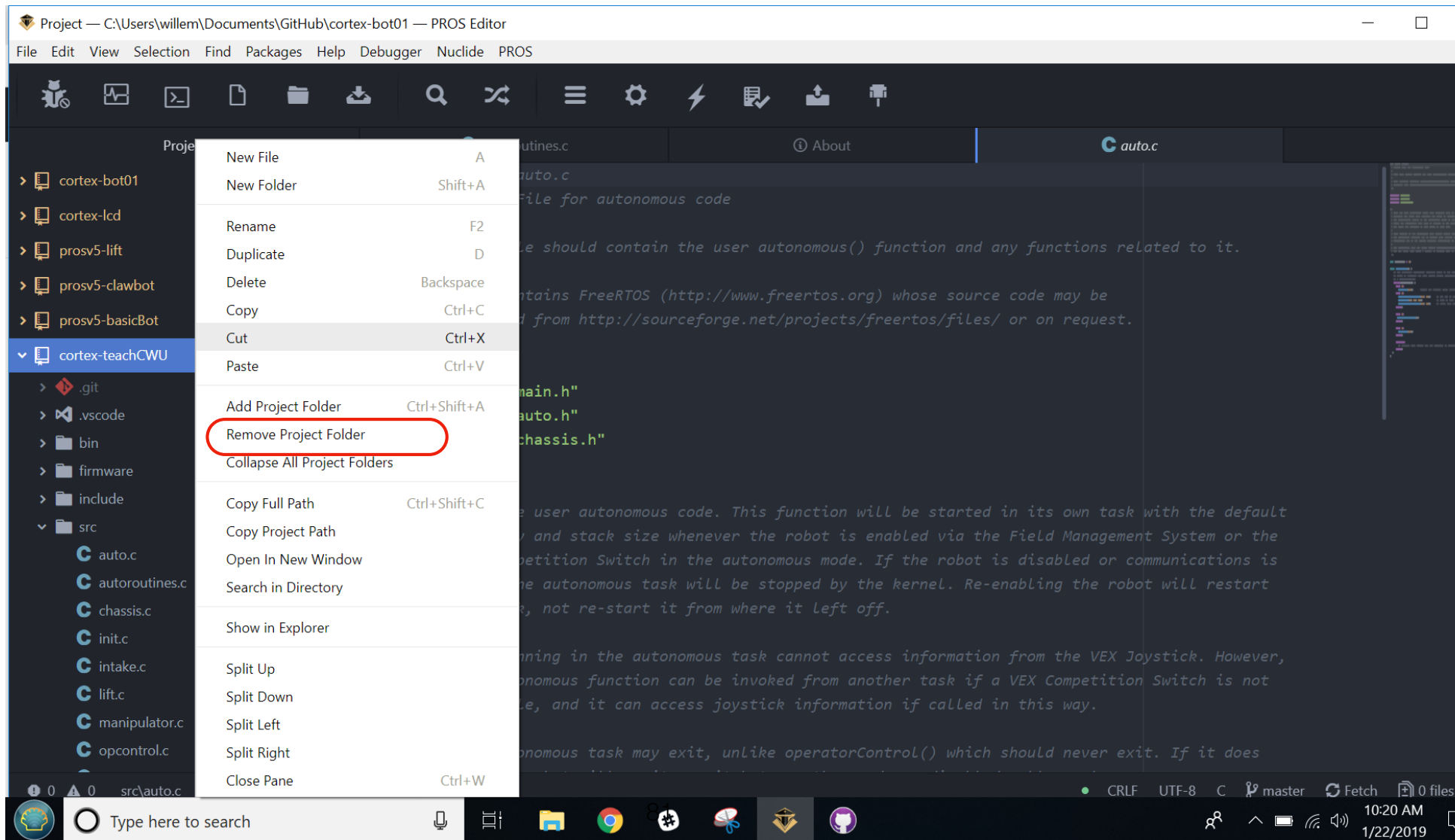
- There are circumstances where you want to drop the repository from the local development machine and / or your gitHUB repository.
- To **refresh your local development copy** of your repository, may occur when you are getting to out of sync, their is a new release posted and you want to sync to it etc
- Sometimes you want to drop everything - particularly in a class room environment where you may want to drop your local FORK, and **start over from the instructor provided MASTER repository**

Dropping Repository

- Starting clean from an instructor provided master repository, involves a few steps:
 - (1) — remove your local development machine repository from PROS and from your local machine
 - (2) — remove the repository from your gitHUB space
- After these steps you are able to start all over, creating a new FORK, synching your local development machine etc.

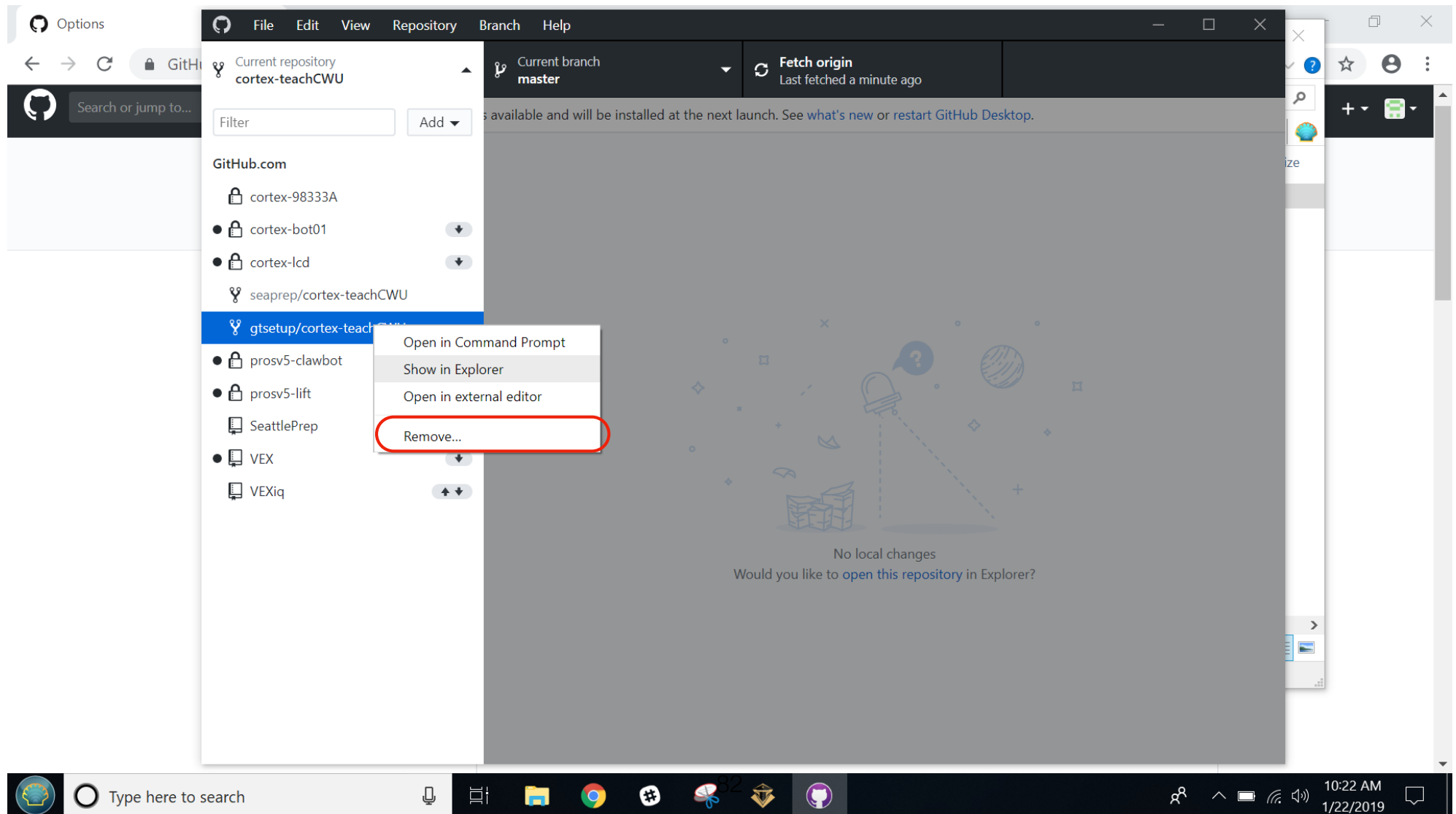
Dropping Repository

Step 1: In PROS - right click on the project to drop, then select "Remove Project Folder"



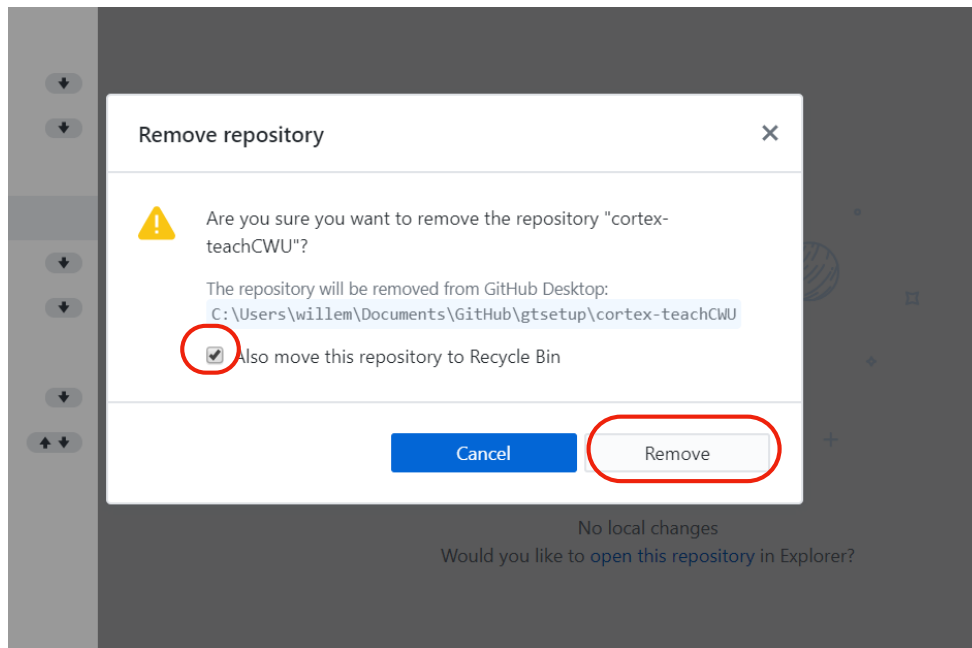
Dropping Repository

Step 1: Go to your local desktop gitHUB application, select the repository to remove, right click, and select 'Remove...'



Dropping Repository

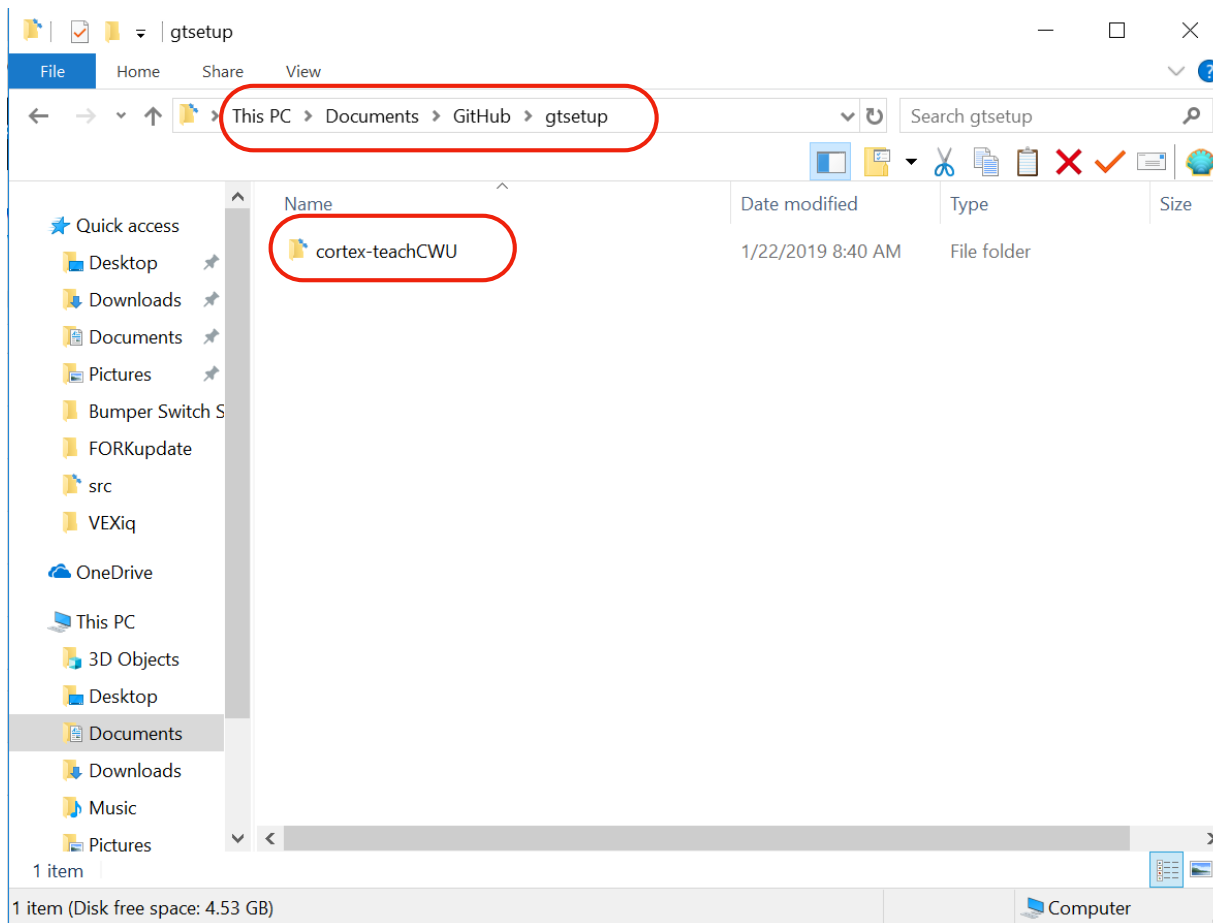
Step 1: Follow the prompts - make sure to check “Also move the repository to Recycling Bin”
Step 2: Click “Remove”



Dropping Repository

Step 1: You can check if the repository is removed from your local machine by going not the location where the repositories are stored, most often in Documents\GitHub

Step 2: If you still have a folder there for the repository, go ahead and delete it.



Dropping Repository

Step 1: Now go to the web - github.com - and go to your repositories, click on the one to remove, and click on “Settings” tab.

The screenshot shows the GitHub interface for a repository named 'gtsetup / cortex-teachCWU', which is a fork of 'sprobotics/cortex-teachCWU'. The repository name is circled in red. In the top navigation bar, the 'Settings' tab is also circled in red. The left sidebar contains a list of settings categories: Options (highlighted), Collaborators, Branches, Webhooks, Integrations & services, and Deploy keys. Below these is a 'Moderation' section with 'Interaction limits'. The main content area is titled 'Settings' and includes a 'Repository name' field with the value 'cortex-teachCWU' and a 'Rename' button. Below this is a 'Features' section with a checked checkbox for 'Wikis' and a description of the feature.

gtsetup / cortex-teachCWU
forked from sprobotics/cortex-teachCWU

Watch 0 Star 0

Code Pull requests 0 Projects 0 Wiki Insights Settings

Options

- Collaborators
- Branches
- Webhooks
- Integrations & services
- Deploy keys

Moderation

- Interaction limits

Settings

Repository name

cortex-teachCWU Rename

Features

☒ Wikis

GitHub Wikis is a simple way to let others contribute content. Any GitHub user can create and edit pages to use for documentation, examples, support, or anything you wish.

Dropping Repository

Step 1: Scroll to the bottom of the Settings Page and find the “Danger Zone”

Step 2: Click “Delete this repository”

Danger Zone

Make this repository private

Public forks can't be made private. Please [duplicate the repository](#).

Make private

Transfer ownership

Transfer this repository to another user or to an organization where you have the ability to create repositories.

Transfer

Archive this repository

Mark this repository as archived and read-only.

Archive this repository

Delete this repository

Once you delete a repository, there is no going back. Please be certain.

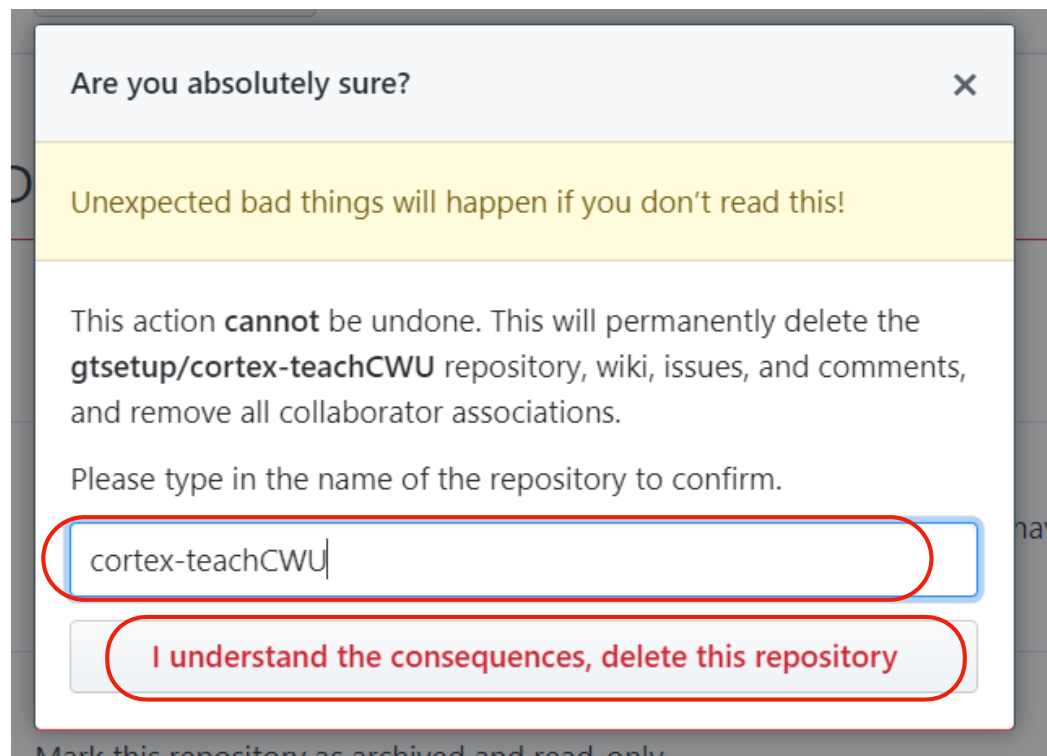
Delete this repository

Dropping Repository

Step 1: In the conformation box, retype in the name of the repository you are about to delete

Step 2: If this matches - lick the “I understand the consequences, delete this repository”

At this point you are back at the same place you original started, and you can follow again the steps to create a new FORK, synch the FORK to your local development machine and go forward.



A screenshot of a GitHub confirmation dialog box titled "Are you absolutely sure?". The dialog has a light blue header with a close button (X) in the top right corner. Below the header is a yellow warning bar with the text "Unexpected bad things will happen if you don't read this!". The main body of the dialog contains the following text: "This action **cannot** be undone. This will permanently delete the **gtsetup/cortex-teachCWU** repository, wiki, issues, and comments, and remove all collaborator associations." followed by "Please type in the name of the repository to confirm." Below this text is a text input field containing "cortex-teachCWU". At the bottom of the dialog is a red button with the text "I understand the consequences, delete this repository". The entire dialog is framed by a red border.

Ignoring some file in the GIT repository

GIT .gitignore setup

- Git sees every file in your working copy as one of three things:
 - **tracked** - a file which has been previously staged or committed;
 - **untracked** - a file which has not been staged or committed; or
 - **ignored** - a file which Git has been explicitly told to ignore.

GIT .gitignore setup

- Ignored files are usually build artifacts and machine generated files that can be derived from your repository source or should otherwise not be committed. Some common examples are:
 - **dependency caches**, such as the contents of /node_modules or /packages
 - **build output directories**, such as /bin, /out, or /target
 - **compiled code**, such as .o, .pyc, and .class files


GIT **.gitignore** setup

- Ignored files are tracked in a special file named **.gitignore** that is checked in at the root of your repository.
- There is no explicit git ignore command: instead the **.gitignore** file must be edited and committed by hand when you have new files that you wish to ignore.
- **.gitignore** files contain patterns that are matched against file names in your repository to determine whether or not they should be ignored.

GIT .gitignore setup

```
#Things to ignore in repository  
.vscode  
bin  
firmware
```

Typical .gitignore file for a PROS V5 / Cortex project would look like this. It will ignore the cache, firmware and the various build stage and binaries.



See here for more details: <https://www.atlassian.com/git/tutorials/saving-changes/gitignore>

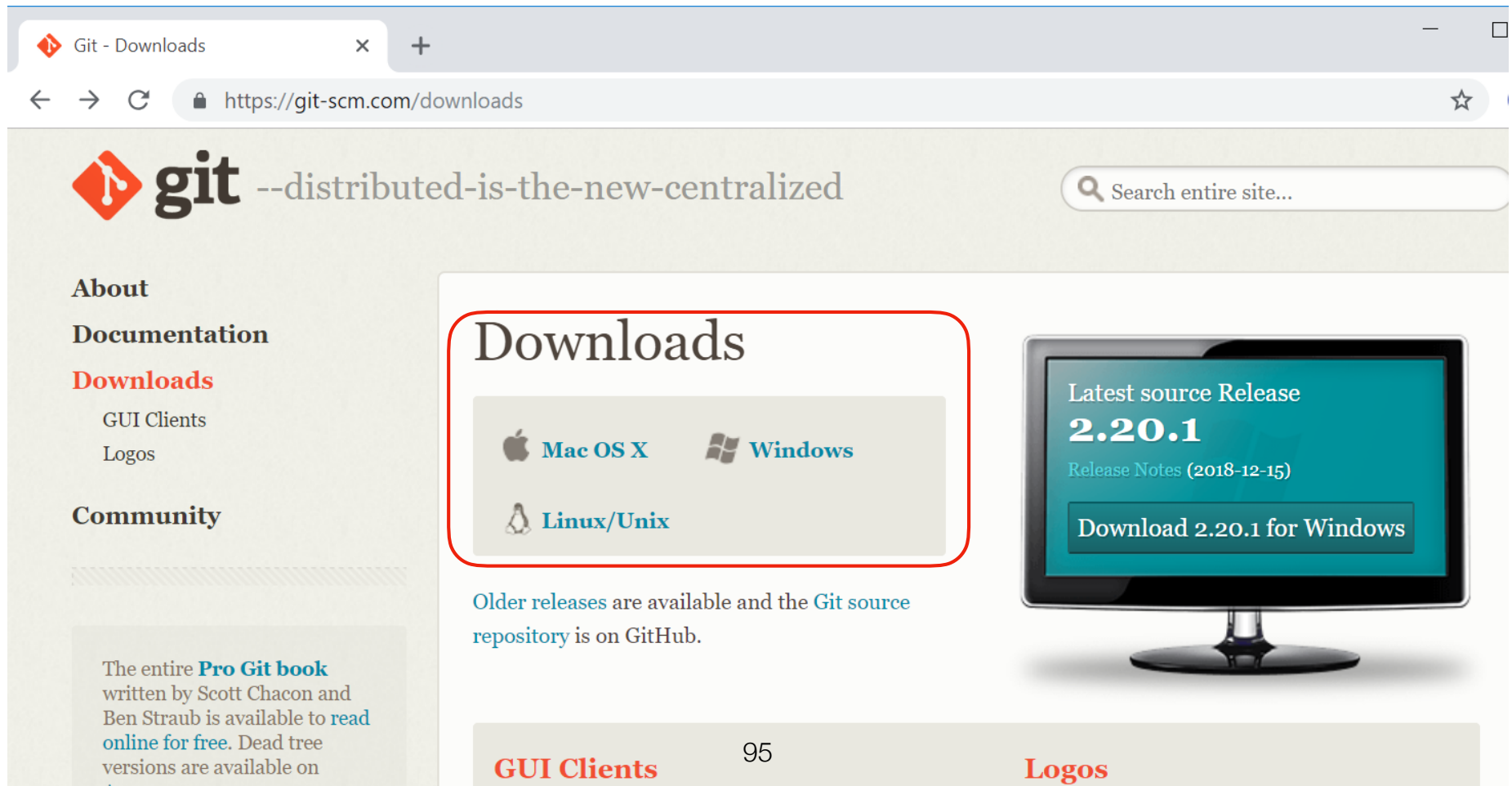
GIT utility and Fixing Common Errors

GIT utilities

- For more advanced management of your local repository, it is advised you install the GIT command line utilities.
- Some advanced use, requiring the GIT command line utility are:
 - **git stash** (*git-stash - Stash the changes in a dirty working directory away see: <https://git-scm.com/docs/git-stash>)*)
 - **git config** (*configure global options for your repositories such a user.name*)
 - **git revert** (*revert to a previous commit*)
 - **git reset —hard** (*a true rollback of the state of your repo*)

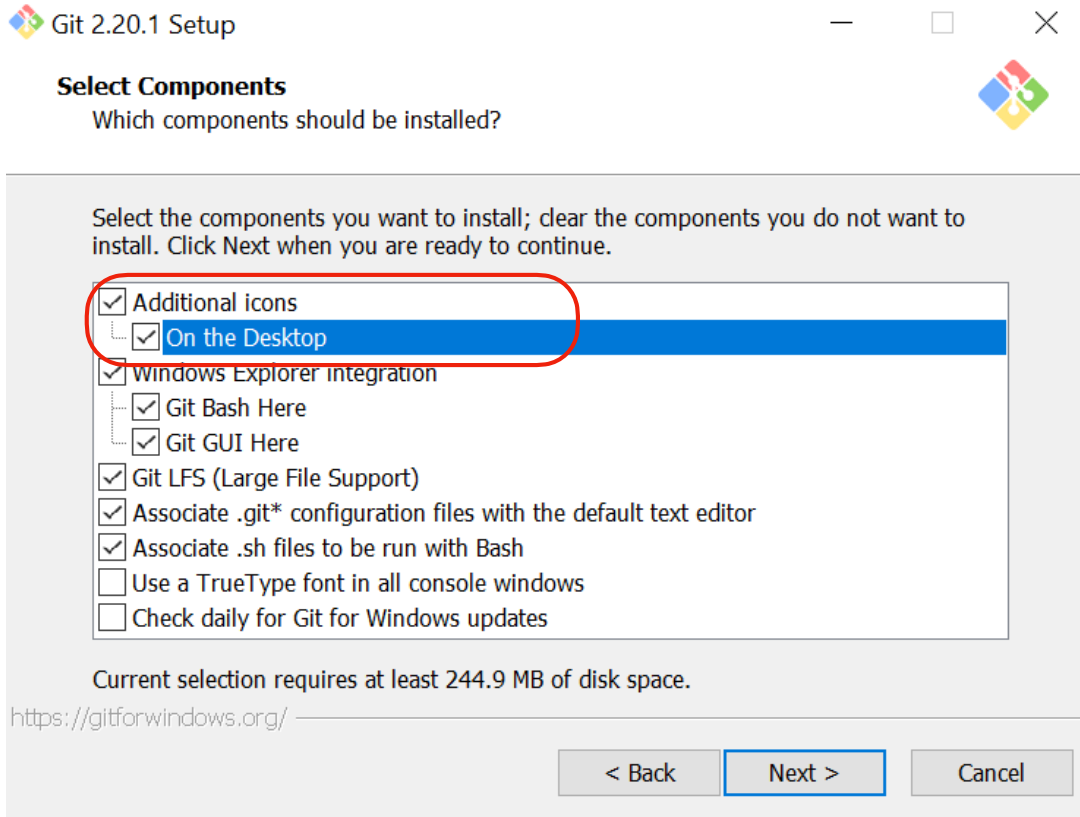
GIT utilities

Step 1: Install the git command line utilities from the following URL: <https://git-scm.com/downloads>



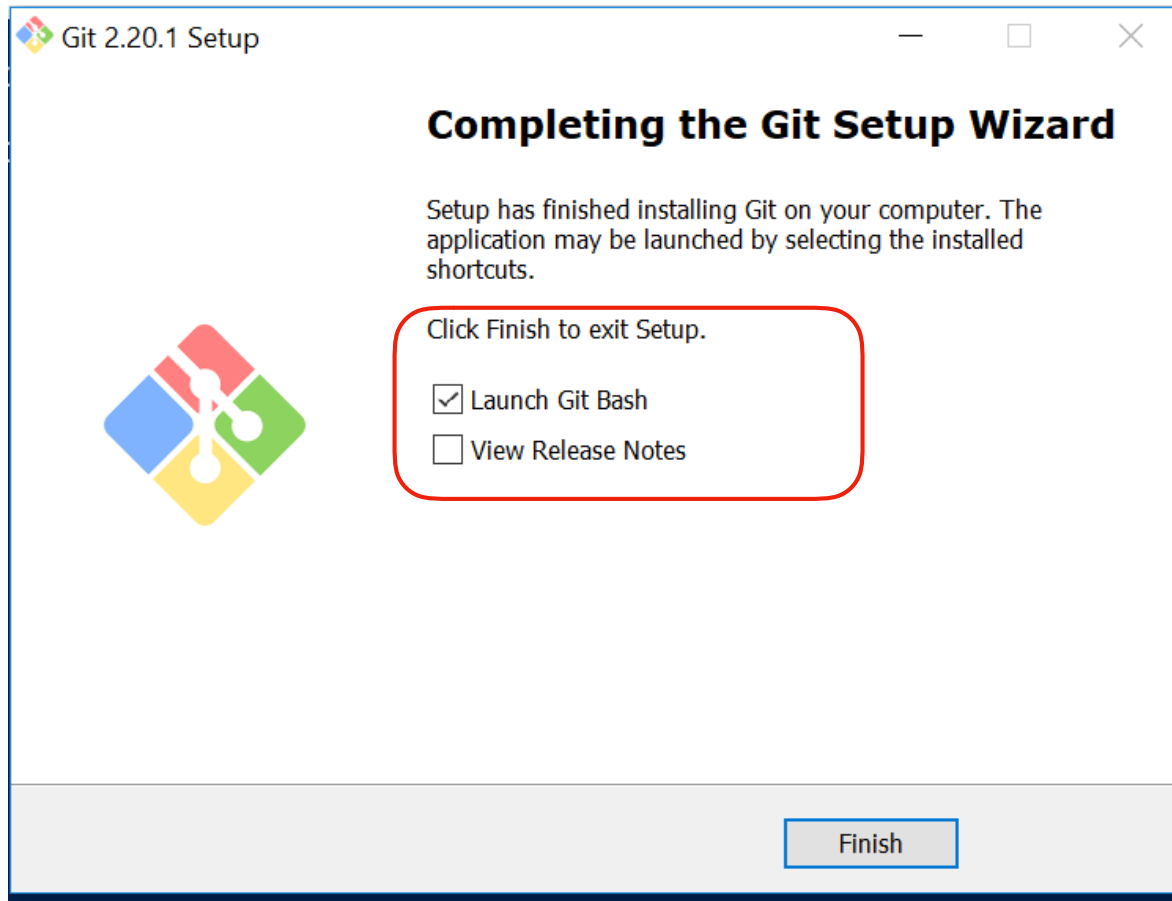
GIT utilities

Step 1: During install select “On the Desktop” — leave all other options as they are.



GIT utilities

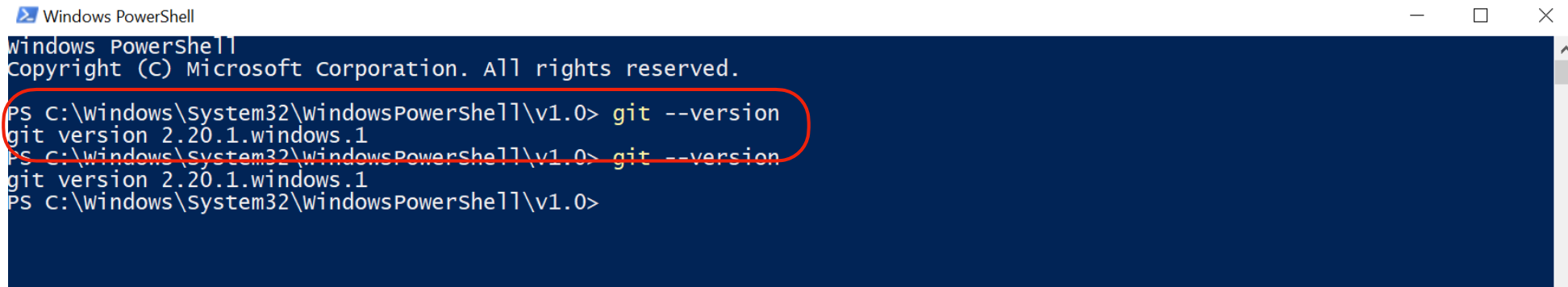
Step 1: On the final installation screen — Check “Launch Git Bash” and Uncheck “View Release Notes”



GIT utilities

Step 1: Once GIT is installed open the PowerShell and type in: `git --version`

This should respond with the installed version in this case: `git version 2.20.1.windows.1`



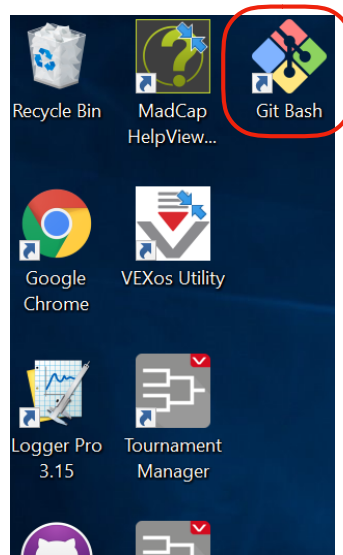
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\System32\WindowsPowerShell\v1.0> git --version
git version 2.20.1.windows.1
PS C:\Windows\System32\WindowsPowerShell\v1.0> git --version
git version 2.20.1.windows.1
PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

GIT utilities

Step 1: Once GIT is installed an icon is added to the Desktop - Git Bash

In the GIT bash shell, you can now interact with your repositories, for example setting core.longpaths to true



```
MINGW64:/c/Users/willem

willem@DESKTOP-HGQQHN2 MINGW64 ~
$ git --version
git version 2.20.1.windows.1

willem@DESKTOP-HGQQHN2 MINGW64 ~
$ git --global user.name wscholten
unknown option: --global
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      <command> [<args>]

willem@DESKTOP-HGQQHN2 MINGW64 ~
$ git config --global core.longpaths true

willem@DESKTOP-HGQQHN2 MINGW64 ~
$ |
```

For more GIT options and tricks see: <https://git-scm.com/doc>

GIT Long PATH error

It is possible that your local gitHUB client will complain about being unable to clone your repository due to a “Long File Name Path Error”

This can be fixed by telling GIT to be bale to use the long paths as follows:

Step 1: **Window Machine:** open Powershell **MAC or Linux:** Open Terminal

Step 2: Issue the following command: *git config --global core.longpaths true*

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\windows\system32\windowspowershell\v1.0> git config --global core.longpaths true
PS C:\windows\system32\windowspowershell\v1.0>
```


How to undo things in GIT?

How to undo things in GIT

- There are times when you (or a teammate) make a change that is not desired or causes your robot code to not work.
- Other times you forgot to include a certain change, or made a mistake in your commit message.
- In these situations, it's common to want to **rollback**, or **undo**, a change made by you or your team.

How to undo things in GIT

- **How do I remove all uncommitted changes in my working directory?**
 - One possible action to take in Git is to **undo changes you made locally**, but have not yet committed or pushed up to your remote repo.
 - **Note:** *An important distinction with uncommitted changes is that you cannot recover the changes you discard with the commands below. As they have not been committed, Git has no record of the changes.*

How to undo things in GIT

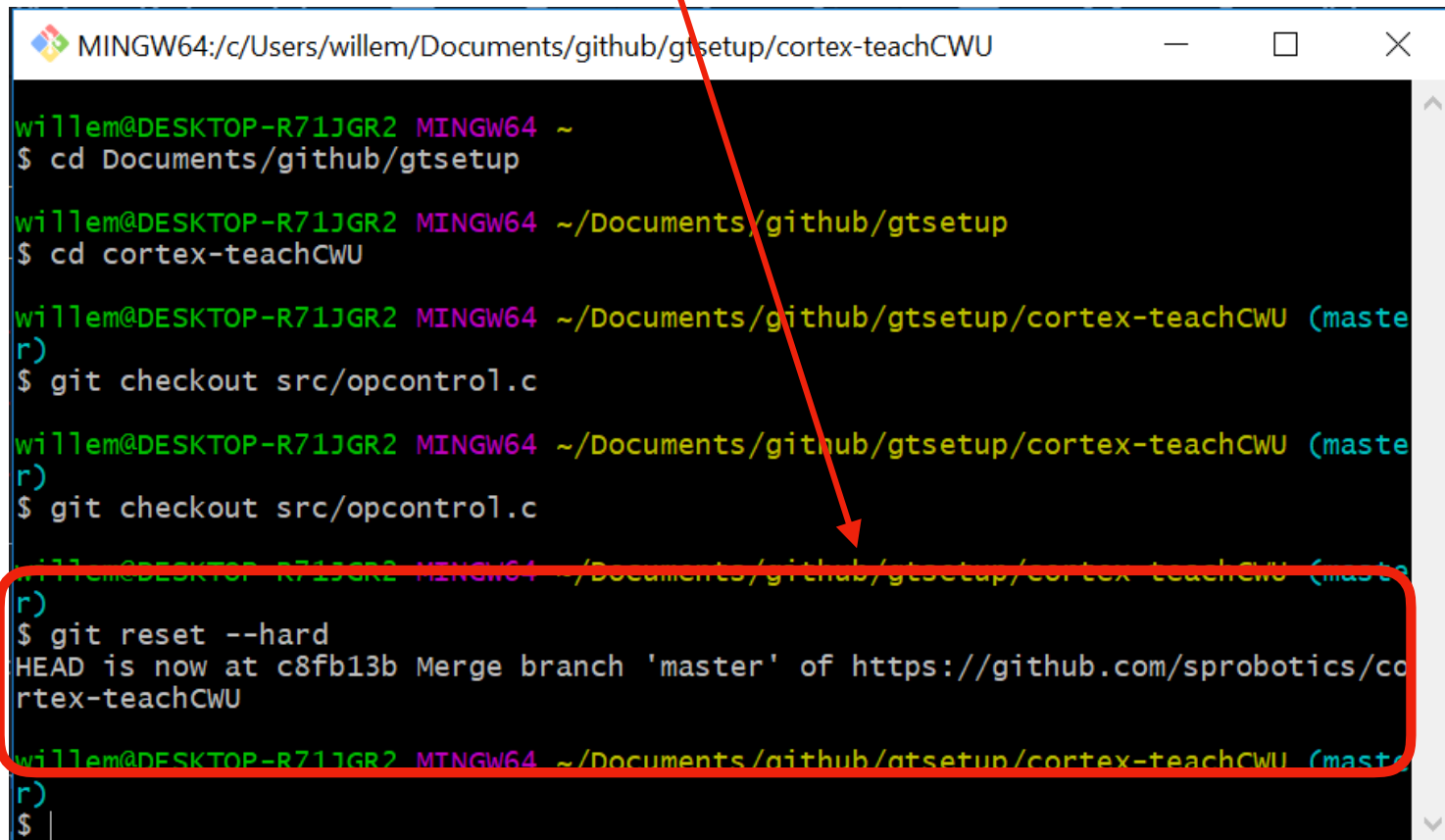
- To **undo** all the changes you've introduced **since the last commit**, use the following command — **Use with care:**

```
git reset --hard
```

- This command **reverts the repo** (*local repo on your computer*) to the state of the HEAD revision, which is the last committed version. **Git discards all the local changes** you made since that point.

How to undo things in GIT

Any changes made in your **local copy** are undone, and your local repository is back at the last committed and synchronized master version



A terminal window titled 'MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU' showing a series of commands and their outputs. A red arrow points from the text box above to the 'git checkout' command. A red box highlights the 'git reset --hard' command and its output.

```
MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~
$ cd Documents/github/gtsetup

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup
$ cd cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git checkout src/opcontrol.c

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git checkout src/opcontrol.c

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git reset --hard
HEAD is now at c8fb13b Merge branch 'master' of https://github.com/sprobotics/cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$
```

How to undo things in GIT

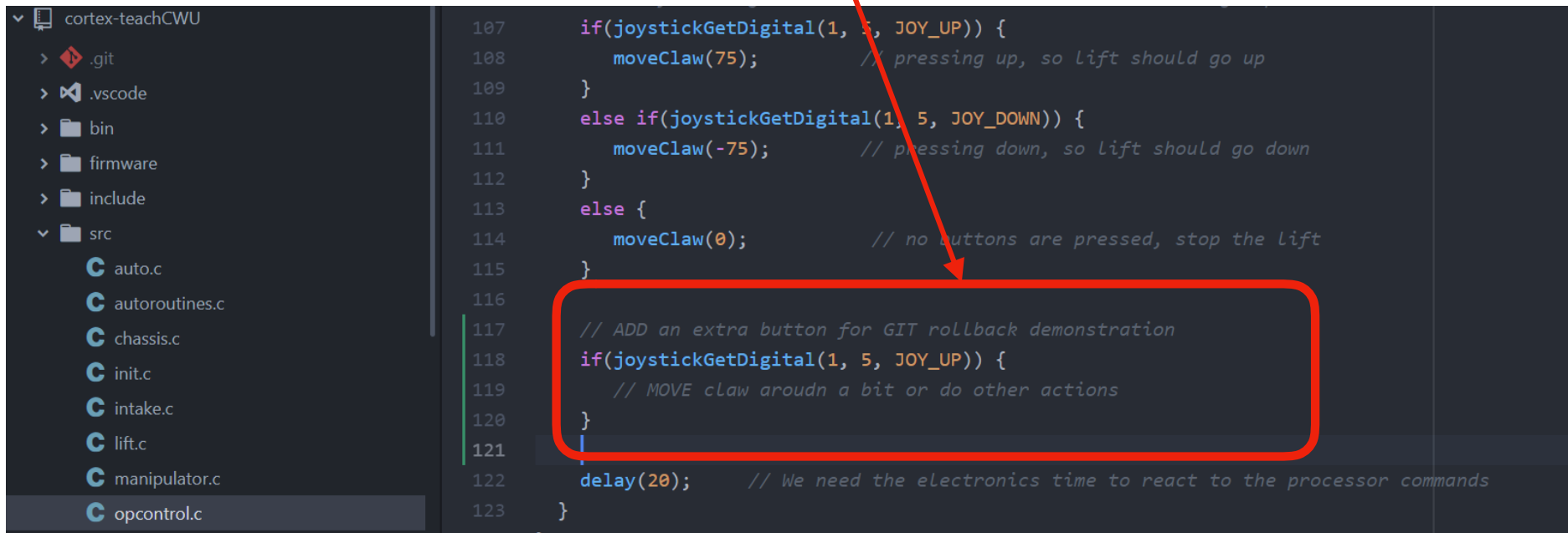
- More common, you may want to only **discard the changes to one file in the local repo**. You can do this with the checkout command:

```
git checkout -- path/to/the/file.txt
```

- This will **reset** the specified file to the last committed version in **your local repo**.

How to undo things in GIT

Assume that the **local** file `opcontrol.c` has changes in it you want to discard and roll back for just that file to a previous version as checked into the repo.

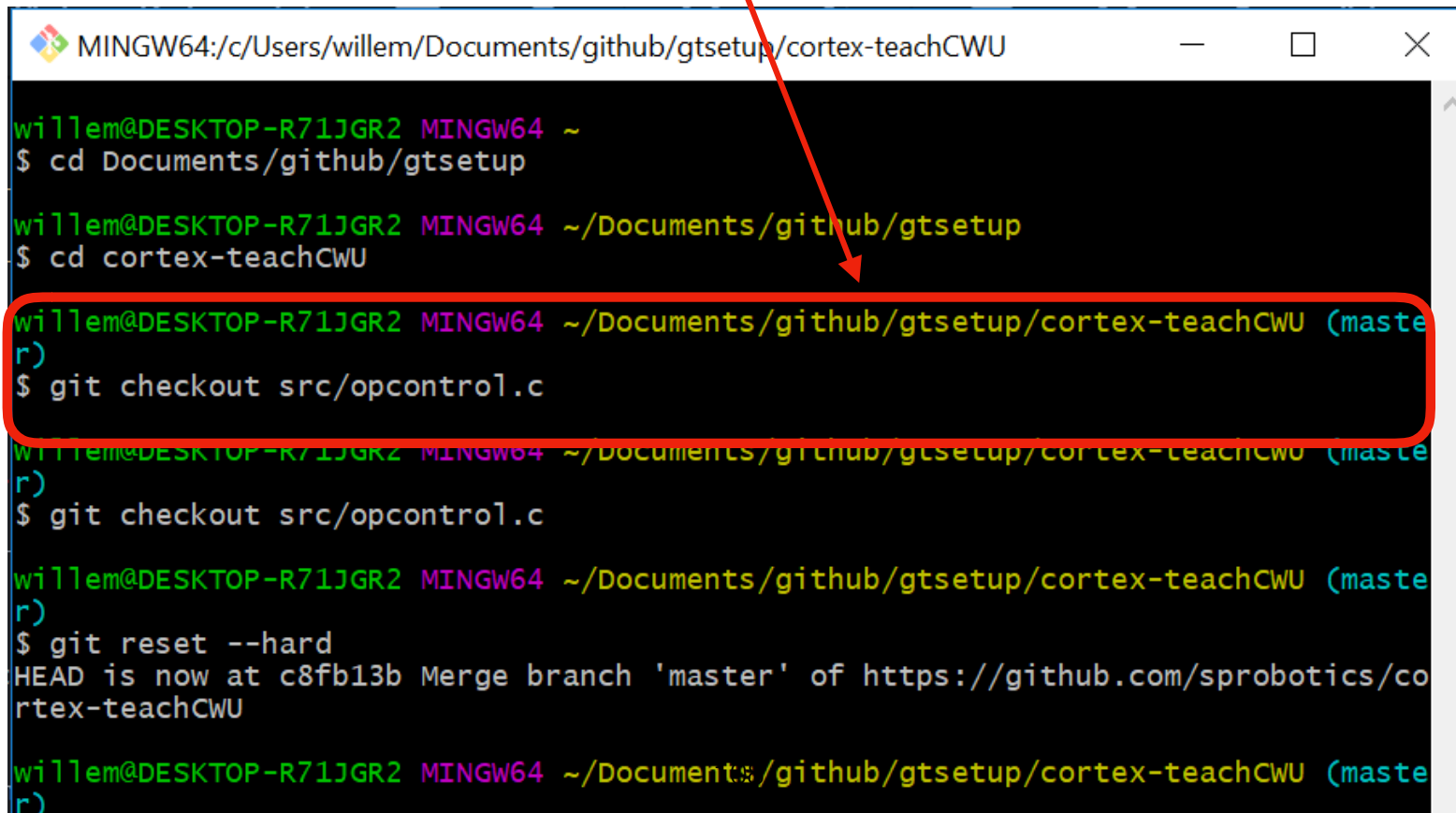


```
107     if(joystickGetDigital(1, 5, JOY_UP)) {
108         moveClaw(75);           // pressing up, so lift should go up
109     }
110     else if(joystickGetDigital(1, 5, JOY_DOWN)) {
111         moveClaw(-75);         // pressing down, so lift should go down
112     }
113     else {
114         moveClaw(0);           // no buttons are pressed, stop the lift
115     }
116
117     // ADD an extra button for GIT rollback demonstration
118     if(joystickGetDigital(1, 5, JOY_UP)) {
119         // MOVE claw around a bit or do other actions
120     }
121
122     delay(20);                 // We need the electronics time to react to the processor commands
123 }
```

How to undo things in GIT

Issue the: ***git checkout <path\filename>*** command to revert the specified file in the **local repo** back to the last version in the MASTER repo.

git checkout src\opcontrol.c

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU'. The terminal shows a series of commands and their outputs. A red arrow points from the text 'git checkout src\opcontrol.c' in the box above to the same command in the terminal. The command is highlighted with a red rounded rectangle. The terminal output shows the user navigating to the correct directory and then executing the checkout command, followed by a git reset --hard command and its output.

```
MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~
$ cd Documents/github/gtsetup

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup
$ cd cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git checkout src/opcontrol.c

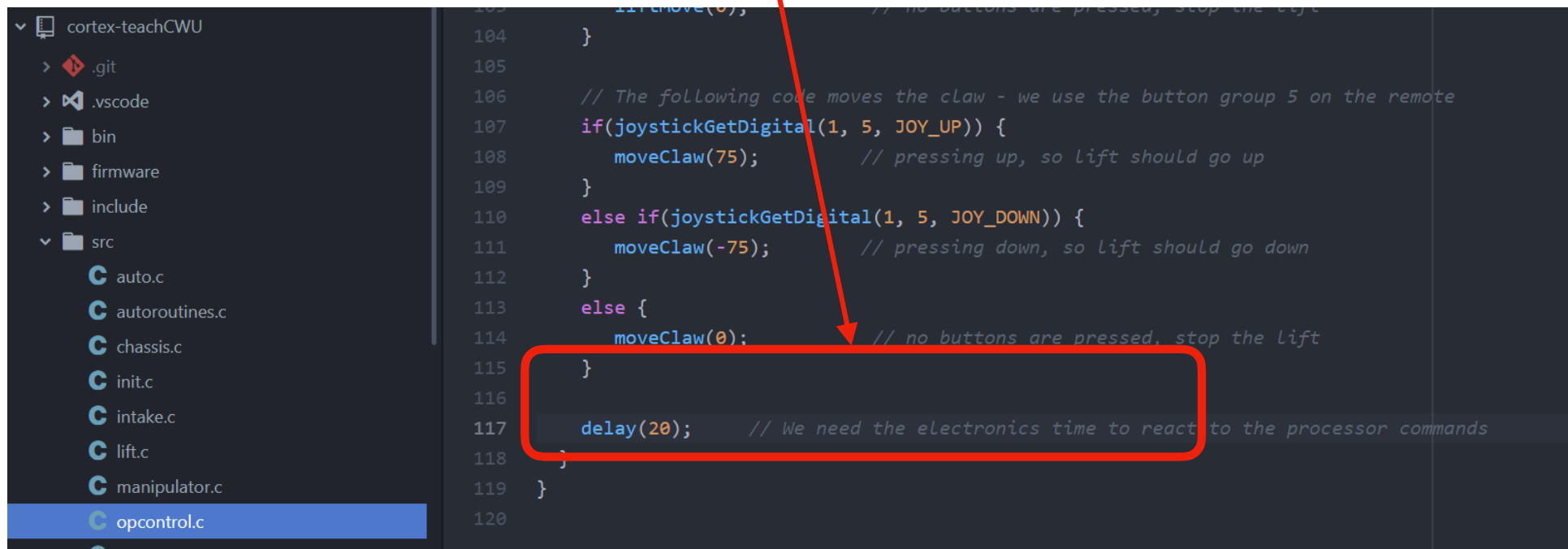
willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git checkout src/opcontrol.c

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git reset --hard
HEAD is now at c8fb13b Merge branch 'master' of https://github.com/sprobotics/cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
```


How to undo things in GIT

After the particular file “rollback” the opcontrol.c file shows the previous code locally added removed



```
103     if(move(0); // no buttons are pressed, stop the lift
104     }
105
106     // The following code moves the claw - we use the button group 5 on the remote
107     if(joystickGetDigital(1, 5, JOY_UP)) {
108         moveClaw(75); // pressing up, so lift should go up
109     }
110     else if(joystickGetDigital(1, 5, JOY_DOWN)) {
111         moveClaw(-75); // pressing down, so lift should go down
112     }
113     else {
114         moveClaw(0); // no buttons are pressed, stop the lift
115     }
116
117     delay(20); // We need the electronics time to react to the processor commands
118 }
119 }
120
```

How to undo things in GIT

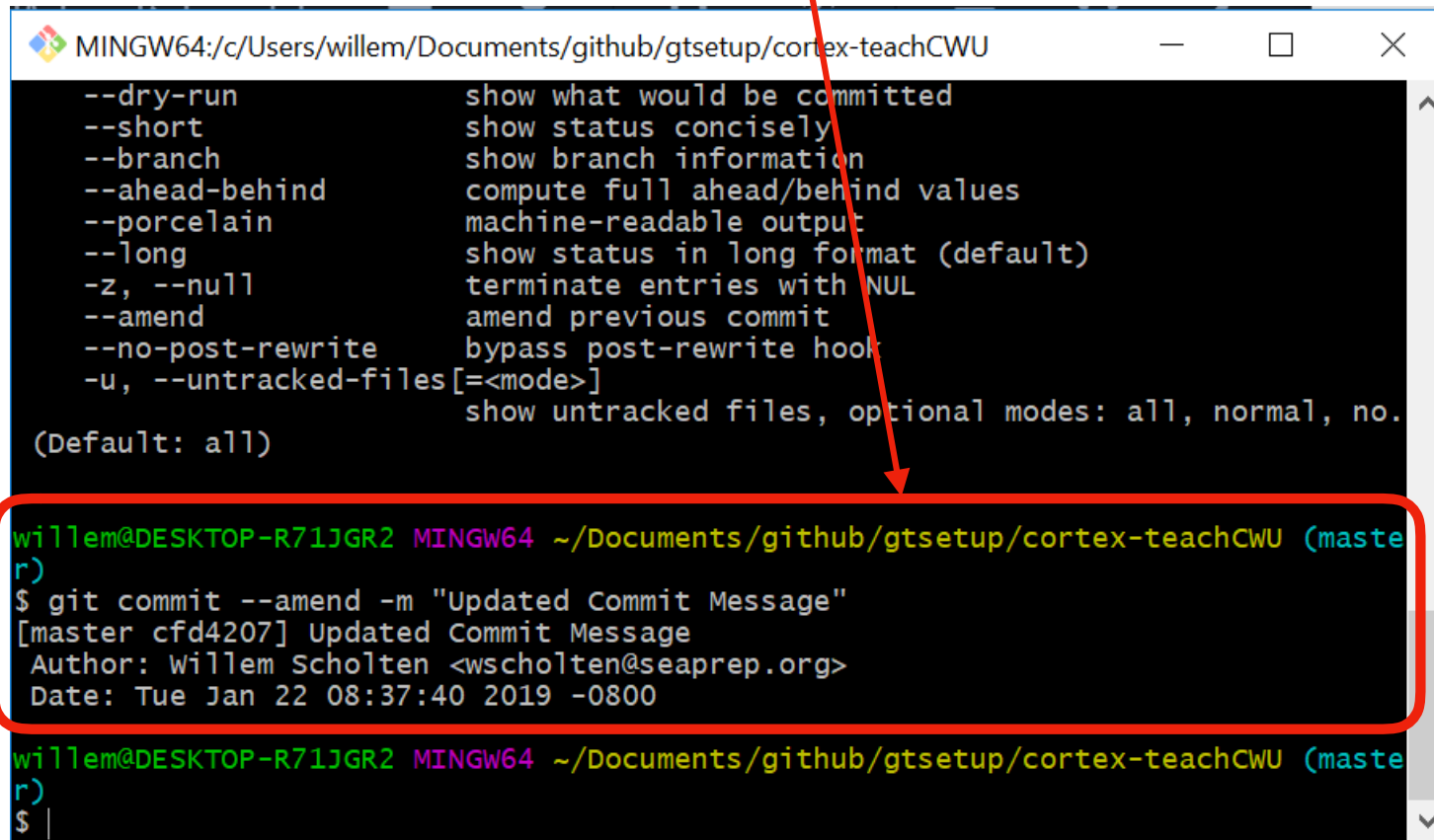
- **How do I fix a message of a commit I just made?**
 - First, Git includes the ability to **amend** the most recent commit message.
 - **Note** that this is not a specific commit in your history, but simply the very last commit. The usage is straight forward:

```
git commit --amend -m "Add your correct commit message here."
```

How to undo things in GIT

Issue the: **git commit --amend -m “*Add your correct commit message here.*”** command to update/adjust your previous commit message

Note: can only be done for the last commit you made.



A terminal window titled "MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU" showing the output of the `git commit --amend` command. The output lists various git commit options and their descriptions. Below this, the command `$ git commit --amend -m "Updated Commit Message"` is executed, resulting in the commit message "Updated Commit Message" and the author "Willem Scholten". The commit hash is "cf4207". The terminal window has a red box around the command and its output, and a red arrow pointing from the note above to the command.

```
--dry-run          show what would be committed
--short           show status concisely
--branch          show branch information
--ahead-behind    compute full ahead/behind values
--porcelain       machine-readable output
--long            show status in long format (default)
-z, --null        terminate entries with NUL
--amend           amend previous commit
--no-post-rewrite bypass post-rewrite hook
-u, --untracked-files[=<mode>] show untracked files, optional modes: all, normal, no.
(Default: all)
```

```
willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git commit --amend -m "Updated Commit Message"
[master cf4207] Updated Commit Message
Author: Willem Scholten <wscholten@seaprep.org>
Date: Tue Jan 22 08:37:40 2019 -0800

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$
```

How to undo things in GIT

Commits · gtsetup/cortex-teachCWU

GitHub, Inc. [US] | https://github.com/gtsetup/cortex-teachCWU

Why GitHub? Enterprise Explore Marketplace Pricing

gtsetup / cortex-teachCWU
forked from sprobotics/cortex-teachCWU

Code Pull requests 0 Projects 0 Insights

Branch: master

Commits on Feb 8, 2019

Merge branch 'master' of https://github.com/gtsetup/cortex-teachCWU
wscholten committed 2 minutes ago

Updated Commit Message
seaprep authored and wscholten committed 17 days ago

Commits on Jan 22, 2019

Merge branch 'master' of https://github.com/sprobotics/cortex-teachCWU
seaprep committed 17 days ago

on gitHUB the repository now reflects the updated commit message

How to undo things in GIT

- **How do I rollback a single file to a certain commit in history?**
 - This scenario is also straightforward. You can use the ***git-checkout*** command to change a specific file back to its state at a specific commit

```
git checkout <commit_ID> path/to/the/file
```

- Once completed, you would then **commit the change** to this file, returning it to its earlier state.

How to undo things in GIT

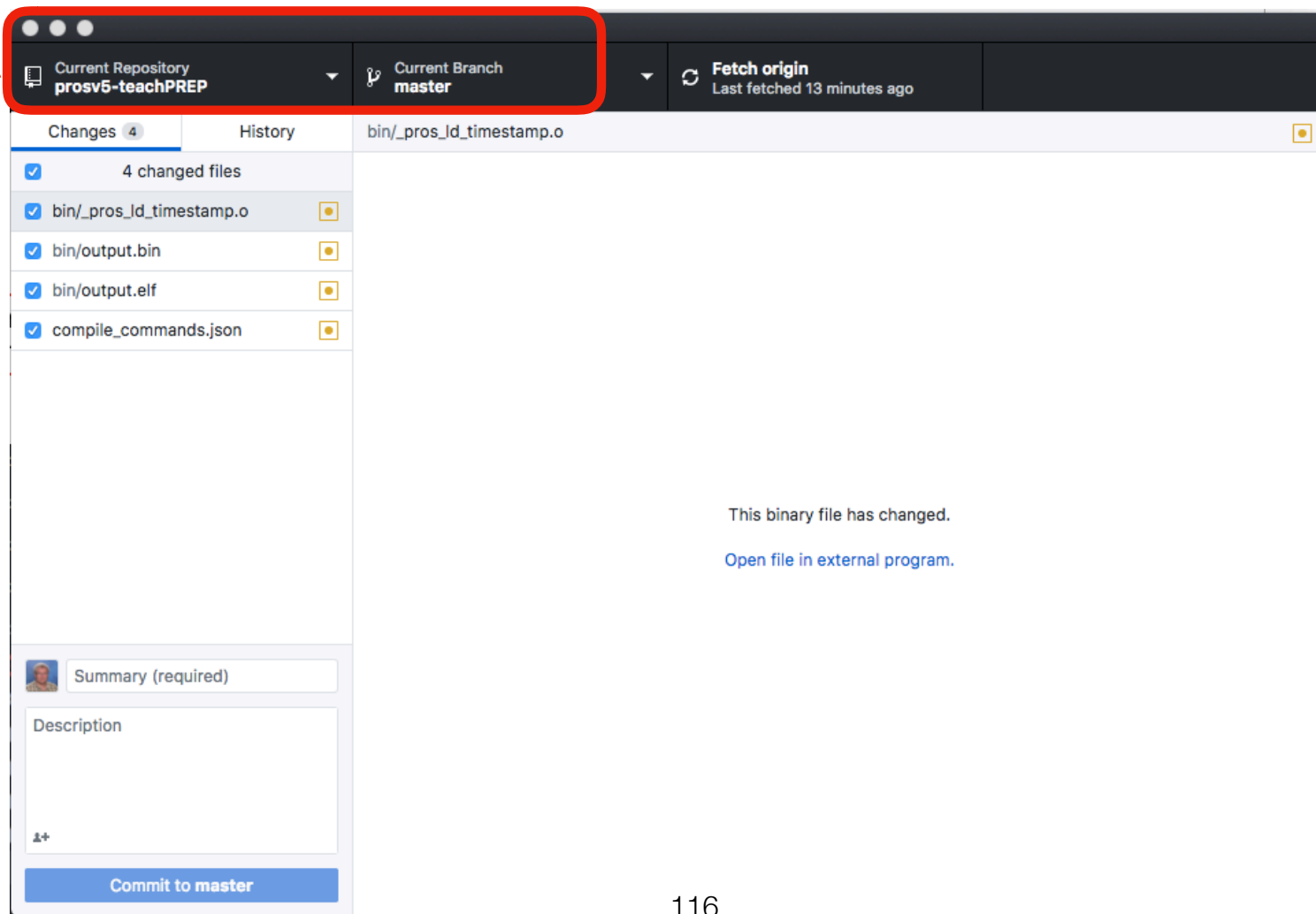
- **What is the commit ID, where do I find it?**
 - Every time you commit your local changes to the repository, your commit is assigned a unique *<commit ID>*
 - The *<commit ID>* can be found a in a number of places, in the gitHUB desktop client, using the gitHUB web client, using a local **git -log** command.

How to undo things in GIT

- **Using gitHUB desktop client to find *<commit ID>***
 - You can use the gitHUB desktop client to both find the *<commit ID>* you potentially want to roll back to
 - Find a particular file version to roll back too
 - Explore the changes made as part of a particular commit

How to undo things in GIT

in the gitHUB desktop client open the local repository for your project - in this case prosv5-teachPREP



How to undo things in GIT

Click on the the 'History' tab, to see a chronological list of all commits.

Chronological list of all commits.

The screenshot shows the Git GUI interface. At the top, there's a header bar with 'Current Repository: prosv5-teachPREP', 'Current Branch: master', and a 'Fetch origin' button. Below this, there are tabs for 'Changes' (4) and 'History'. The 'History' tab is selected and highlighted with a red box. A red arrow points from the instruction box to this tab. Below the tabs, there's a section titled 'Tweaked lift settings' showing a commit by Willem Scholten. A red box highlights the list of commits on the left side of the interface, which is a chronological list of all commits. A red arrow points from the 'Chronological list of all commits' box to this list. The list includes commits like 'Added a TODO', 'Update TODO', 'Added TODO', 'fine turned autonomous', 'Tweaked lift settings', 'Remote Buton PDF updated', 'New manual Lift Move Functions', 'updated encoder settings', 'Update gear cartidge colors', 'Button reassignment', and 'Update README.md'. To the right of the commit list, there's a diff view showing changes to files like .vscode/cquery_cac.../include@lift.h, .vscode/cquery.../include@lift.h.json, .vscode/cquer.../src@opcontrol.cpp, .vscode/c.../src@opcontrol.cpp.json, bin/_pros_id_timestamp.o, bin/lift.cpp.o, bin/opcontrol.cpp.o, bin/output.bin, bin/output.elf, compile_commands.json, include/lift.h, and src/opcontrol.cpp. The diff view shows line numbers and the changes made in each commit.

Commit	Author	Message	Files
Added a TODO	Willem Scholten	committed a day ago	
Update TODO	Willem Scholten	committed 2 days ago	
Added TODO	Willem Scholten	committed 2 days ago	
fine turned autonomous	Willem Scholten	committed 2 days ago	
Tweaked lift settings	Willem Scholten	committed 2 days ago	
Remote Buton PDF updated	Willem Scholten	committed 2 days ago	
New manual Lift Move Functions	Willem Scholten	committed 2 days ago	
updated encoder settings	Willem Scholten	committed 3 days ago	
Update gear cartidge colors	Willem Scholten	committed 3 days ago	
Button reassignment	Willem Scholten	committed 3 days ago	
Update README.md			

117

How to undo things in GIT

Click on the the 'History' tab, to see a chronological list of all commits.

<commit ID>

The screenshot shows the Git GUI interface for a repository named 'prosv5-teachPREP'. The 'History' tab is selected, displaying a list of commits by Willem Scholten. The commit 'Tweaked lift settings' is highlighted. A red box highlights the commit ID '1829586'. A red box highlights the file 'src/opcontrol.cpp' in the list of changed files. A red box highlights the diff view for 'src/opcontrol.cpp', showing line-by-line changes with additions in green and deletions in red.

Current Repository: prosv5-teachPREP

Current Branch: master

Fetch origin: Last fetched 14 minutes ago

Changes: 4

History

Select Branch to Compare...

Willem Scholten committed 2 days ago

1829586 12 changed files

Set the lift middle, high and low encoder settings

Added a TODO

Update TODO

Added TODO

fine turned autonomous

Tweaked lift settings

Remote Buton PDF updated

New manual Lift Move Functions

updated encoder settings

Update gear cartidge colors

Button reassignment

Update README.md

src/opcontrol.cpp

```
@@ -149,16 +149,16 @@ void opcontrol() {
149 149
150 150     // Control lift movement
151 151     if(master.get_digital(DIGITAL_R1)) {
152 152     - liftRaise(100, 1);
153 153     + liftRaise(150, 1);
154 154     // raise to middle pole
155 155     } else if(master.get_digital(DIGITAL_R2))
156 156     {
157 157     - liftRaise(100, 2);
158 158     + liftRaise(150, 2);
159 159     // raise to high pole
160 160     } else {
161 161     // stay put lock it
162 162     // lift retract to zero
163 163     if (master.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_DOWN)) {
164 164     - liftRaise(100, 0);
165 165     + liftRaise(150, 0);
166 166     }
167 167     pros::delay(20);
168 168 }
```

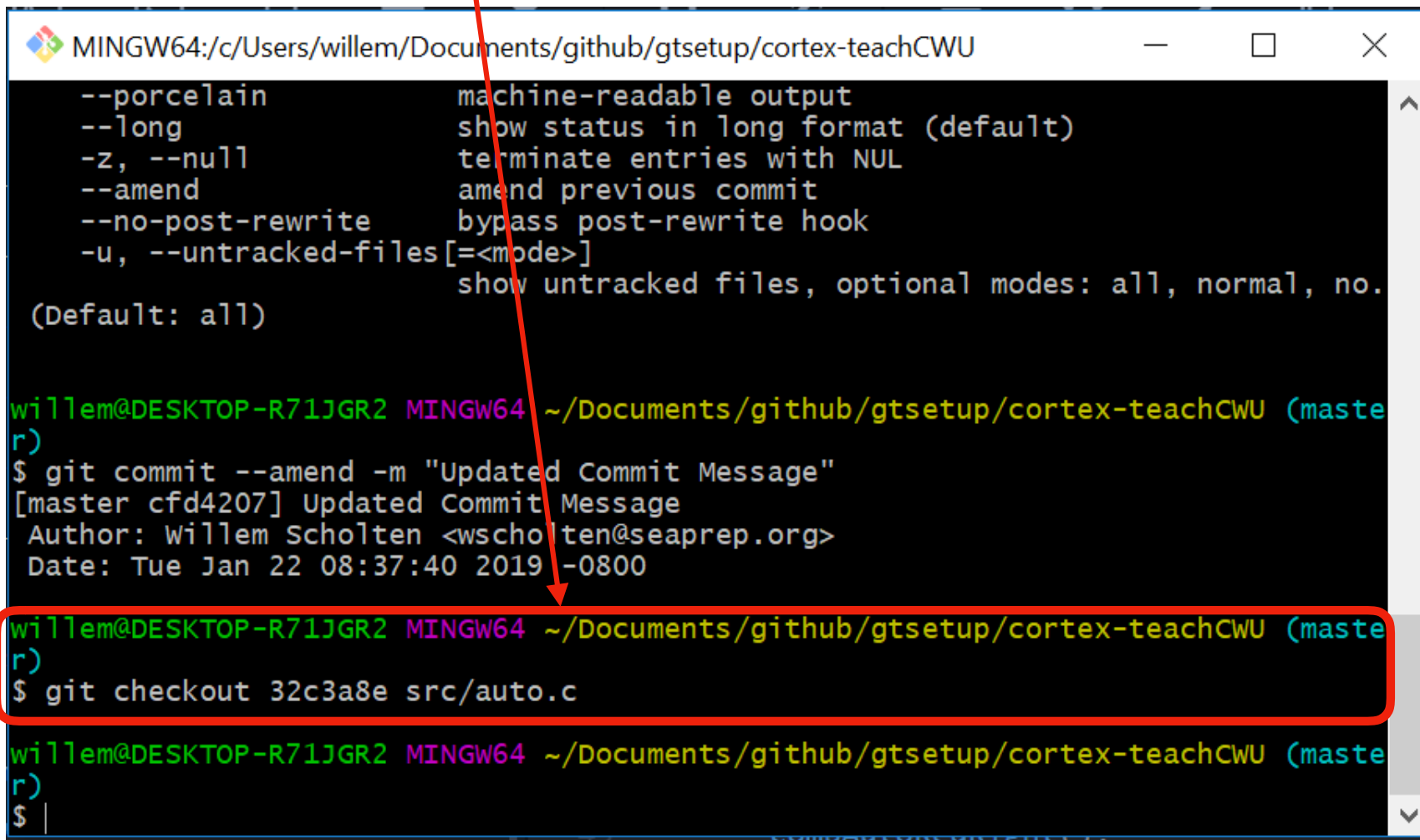
Select a commit for deeper analysis

Select a changed source file for inspection

Changed source code - note add (+) and subtractions (-)

How to undo things in GIT

Once the file to roll back is identified, issue the ***git checkout <commit ID> filename*** command. In this case ***git checkout 3283a8e src/auto.c*** which will rollback the auto.c file to the one in the repo committed with the the commit ID 32c3a8e

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU'. The terminal shows a list of git options like --porcelain, --long, etc. Then, a commit is made with 'git commit --amend -m "Updated Commit Message"'. The commit message is '[master cfd4207] Updated Commit Message' with author 'Willem Scholten' and date 'Tue Jan 22 08:37:40 2019 -0800'. The next command, '\$ git checkout 32c3a8e src/auto.c', is highlighted with a red box. A red arrow points from the text in the first block to this command. The prompt '\$' is visible at the bottom.

```
MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU

--porcelain      machine-readable output
--long           show status in long format (default)
-z, --null       terminate entries with NUL
--amend          amend previous commit
--no-post-rewrite bypass post-rewrite hook
-u, --untracked-files[=<mode>]
                  show untracked files, optional modes: all, normal, no.
(Default: all)

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git commit --amend -m "Updated Commit Message"
[master cfd4207] Updated Commit Message
Author: Willem Scholten <wscholten@seaprep.org>
Date: Tue Jan 22 08:37:40 2019 -0800

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git checkout 32c3a8e src/auto.c

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$
```

How to undo things in GIT

- Using GIT command line utility to find **<commit ID>**
 - You can use the GIT desktop client command line utility to find the **<commit ID>** you potentially want to roll back to

```
git log --pretty=format:"%h - %an, %ar : %s"
```

<commit ID>

```
Initial Project Commit
[Willems-MacBook-Air:prosv5-gitTest willem$ git log --pretty=format:"%h - %an, %ar : %s"
0da3e6f Willem Scholten, 20 hours ago : Minor tweak to opcontrol.c
a4f5244 Willem Scholten, 20 hours ago : Merge branch 'master' of https://github.com/sprobotics/prosv5-gitTest
32cacb7 Willem Scholten, 20 hours ago : Initial Commit
e22ce20 Willem Scholten, 20 hours ago : Initial Project Commit
Willems-MacBook-Air:prosv5-gitTest willem$
```

How to undo things in GIT

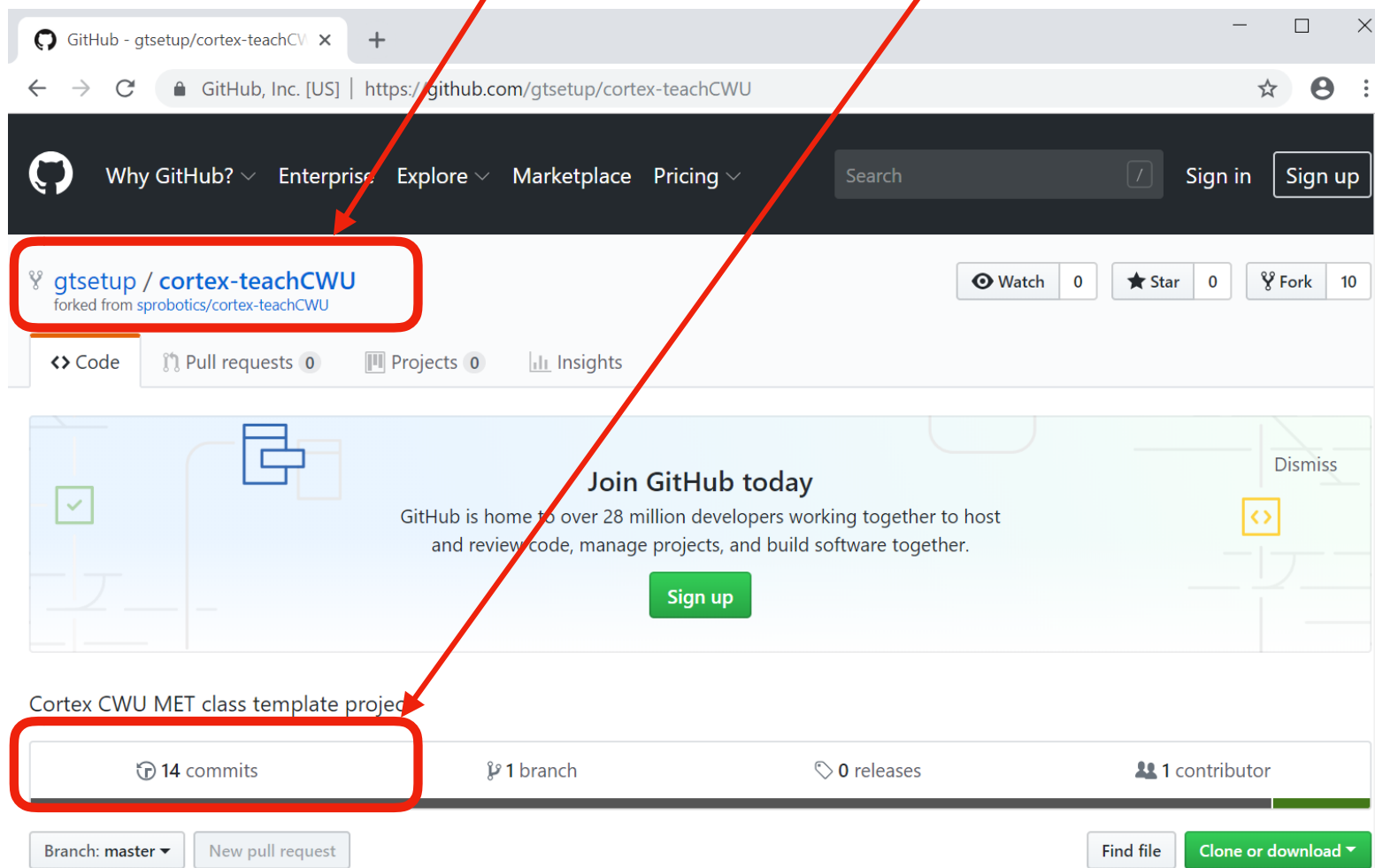
- One additional way to find the <commit ID> is using the **GitHub web interface** for the remote repository
- This option naturally **only works if you have established a remote repository** on gitHUB or some other cloud based repository server

How to undo things in GIT

Finding the <commit ID> on gitHUB:

Step 1: browse to the repository for example: github.com/gtsetup/cortex-teachCWU

Step 2: click on the tab/button showing the number of commits for the repository



How to undo things in GIT

Commit history in chronological order with the commit message

<commit ID> for each commit to the repository

Click to browse the files/state of the repository at a particular <commit ID> moment in time

Assume we want to see the commit history of this state to determine if this is the place to roll back to.

The screenshot shows the GitHub interface for the repository `gtsetup / cortex-teachCWU`. The page displays a list of commits in chronological order. A red box highlights the commit history section, which includes the following details:

- Commits on Feb 8, 2019**
 - Merge branch 'master' of <https://github.com/gtsetup/cortex-teachCWU>
wscholten committed 11 minutes ago
 - Updated Commit Message
seaprep authored and wscholten committed 17 days ago
- Commits on Jan 22, 2019**
 - Merge branch 'master' of <https://github.com/sprobotics/cortex-teachCWU>
seaprep committed 17 days ago
 - Added one more autonomous move

At the bottom of the commit list, there is a pagination control showing "123".

On the right side of the page, a list of commit IDs is shown, each with a file icon and a code icon. The commit IDs are:

- c844901
- cf4207
- c8fb13b
- ab6de1f

Red arrows point from the text boxes to the corresponding elements in the screenshot:

- From the first text box to the commit history section.
- From the second text box to the commit IDs.
- From the third text box to the code icon next to the commit ID `c844901`.
- From the fourth text box to the code icon next to the commit ID `ab6de1f`.

How to undo things in GIT

When we clicked on the **<commit ID>** we will be brought to the repository state as of that moment in time. We can now browse any of it's files.

sprobotics / cortex-teachCWU

Unwatch 1 Star 0 Fork 10

Code Issues 0 Pull requests 0 ZenHub Projects 0 Wiki Insights Settings

Cortex CWU MET class template project Edit

Manage topics

10 commits 1 branch 0 releases 1 contributor

Tree: ab6de1f15c New pull request Create new file Upload files Find file Clone or download

		Latest commit ab6de1f 18 days ago
vscode/cquery_cached_index	Added one more autonomous move	18 days ago
bin	Added one more autonomous move	18 days ago
firmware	Initial commit	20 days ago
include	fiexd PID turnign and drive functions	19 days ago
src	Added one more autonomous move	18 days ago
.DS_Store	Initial commit	20 days ago
.gitattributes	Initial commit	20 days ago
Makefile	Initial commit	
README.md	Update README.m	20 days ago

Sign in now to use

How to undo things in GIT

In this case we opened `src/auto.c` for the given **<commit ID>** and we can look if this is the state we want to recover too.

The screenshot shows the GitHub interface for the repository `sprobotics / cortex-teachCWU`. The file `src / auto.c` is selected, and the commit `ab6de1f15c` is chosen. The commit message is "seaprep Added one more autonomous move" from 18 days ago. The file content is displayed, showing a C program for autonomous code. The file has 57 lines (49 sloc) and is 2 KB in size. The code includes headers for `main.h`, `auto.h`, and `chassis.h`, and contains a comment describing the autonomous task.

```
1  /** @file auto.c
2   * @brief File for autonomous code
3   *
4   * This file should contain the user autonomous() function and any functions related to it.
5   *
6   * PROS contains FreeRTOS (http://www.freertos.org) whose source code may be
7   * obtained from http://sourceforge.net/projects/freertos/files/ or on request.
8   */
9
10 #include "main.h"
11 #include "auto.h"
12 #include "chassis.h"
13
14 /**
15  * Runs the user autonomous code. This function will be started in its own task with the default
16  * priority and stack size whenever the robot is enabled via the Field Management System or the
17  * VEX Competition Switch in the autonomous mode. If the robot is disabled or communications is
18  * lost, the autonomous task will be stopped by the kernel. Re-enabling the robot will restart
19  * the task, not re-start it from where it left off.
```

How to undo things in GIT

Commit history in chronological order with the commit message

<commit ID> for each commit to the repository

Click to browse the files/state of the repository at a particular <commit ID> moment in time

click here to see the commit history - i.e what field changed and what changed within it

The screenshot shows the GitHub interface for the repository 'gtsetup/cortex-teachCWU'. The page displays a list of commits in chronological order. The top commit is a merge of branch 'master' from the same repository, committed 11 minutes ago by 'wscholten'. Below it is an 'Updated Commit Message' by 'seaprep' and 'wscholten' from 17 days ago. The page is grouped by date, with 'Commits on Feb 8, 2019' and 'Commits on Jan 22, 2019'. The bottom commit is a merge of branch 'master' from 'sprobotics/cortex-teachCWU' by 'seaprep' from 17 days ago, followed by a commit 'Added one more autonomous move'. On the right side, there is a list of commit IDs with corresponding file icons and a '<>' button to view the commit details. The commit IDs shown are c844901, cfd4207, c8fb13b, and ab6de1f. The commit 'ab6de1f' is highlighted with a red box. The page number '126' is visible at the bottom right.

Commit ID	File Icon	<> Button
c844901		<>
cfd4207		<>
c8fb13b		<>
ab6de1f		<>

How to undo things in GIT

One of the files changed is src/auto.c

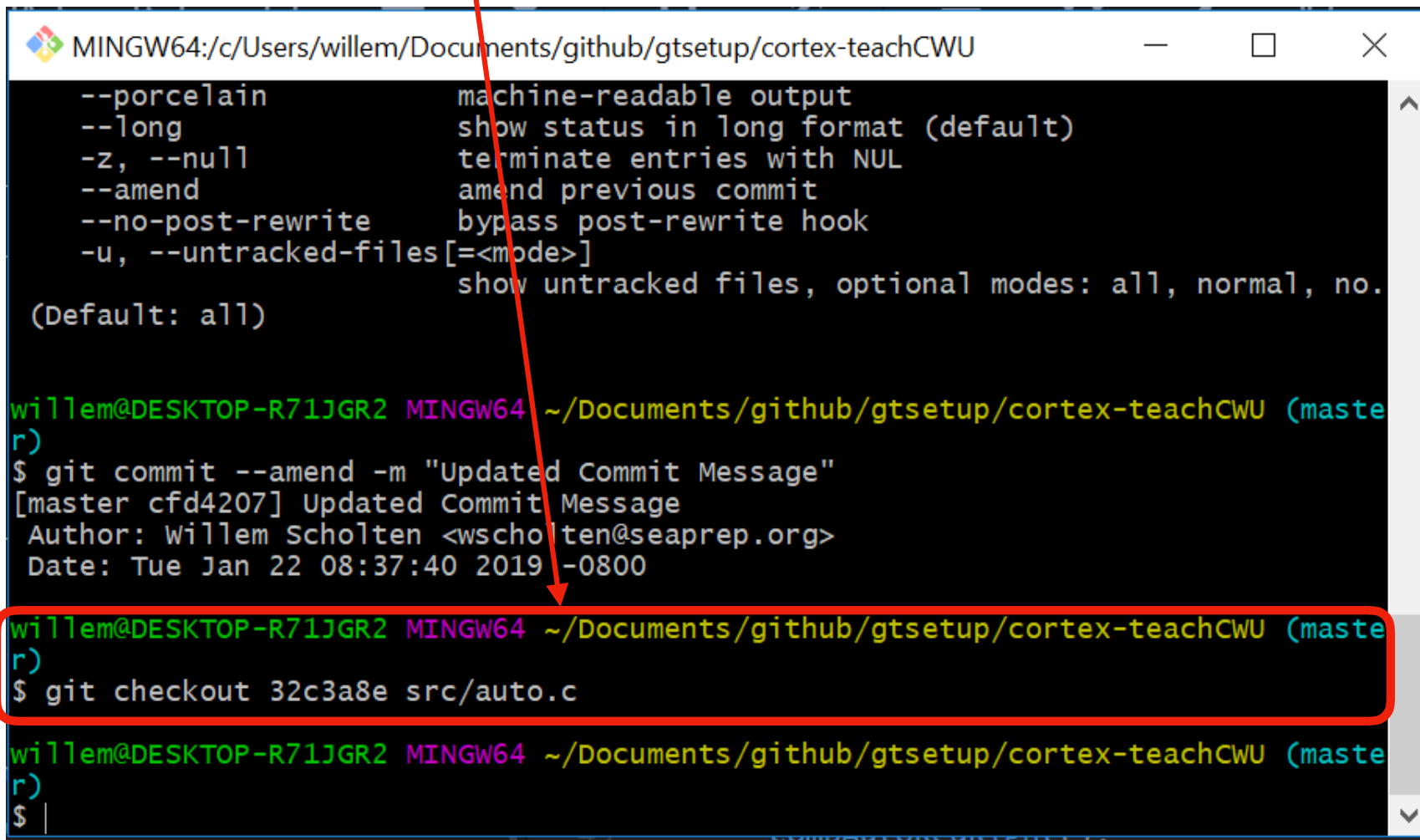
Changed lines are highlighted, a line starting with a + means it is added, one starting with a - means it has been removed since the last commit.

The screenshot shows a GitHub repository page for 'sprobotics / cortex-teachCWU'. The commit message is 'Added one more autonomous move' by user 'seaprep', committed 18 days ago. The commit hash is 'ab6de1f15c40aa47ca84068f9a4983b10ee2bab9'. Below the commit information, it states 'Showing 7 changed files with 24 additions and 22 deletions.' The file list shows '.vscode/cquery_cached_index/@Users@willem@cortex-teachCWU/src@auto.c' as the selected file. The diff view shows changes to 'auto.c'. Line 41 is highlighted in green and marked with a '+' sign, indicating it was added. The code snippet shows a C function 'void autonomous()' with a 'case 1:' block containing 'driveForDistancePID(12, 50);' and a comment '// After turn drive a bit more forward'. Other lines in the diff are marked with '-' indicating deletions.

```
1 .vscode/cquery_cached_index/@Users@willem@cortex-teachCWU/src@auto.c
@@ -38,6 +38,7 @@ void autonomous() {
38 38     case 1:
39 39         driveForDistancePID(36, 60);    // for 36" at speed 60
40 40         pivotTurn(0, 30, 00);          // turn at speed 30 for 90 degree angle right turn
41 +    driveForDistancePID(12, 50);        // After turn drive a bit more forward
42 42         break;
43 43
44 44     case 2:
45 45
```

How to undo things in GIT

Once the file to roll back is identified, issue the ***git checkout <commit ID> filename*** command. In this case ***git checkout 3283a8e src/auto.c*** which will rollback the auto.c file to the one in the repo committed with the the commit ID 32c3a8e



```
MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU

--porcelain      machine-readable output
--long           show status in long format (default)
-z, --null       terminate entries with NUL
--amend          amend previous commit
--no-post-rewrite bypass post-rewrite hook
-u, --untracked-files[=<mode>]
                  show untracked files, optional modes: all, normal, no.
(Default: all)

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git commit --amend -m "Updated Commit Message"
[master cfd4207] Updated Commit Message
Author: Willem Scholten <wscholten@seaprep.org>
Date: Tue Jan 22 08:37:40 2019 -0800

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git checkout 32c3a8e src/auto.c

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$
```

How to undo things in GIT

- **IMPORTANT:** after you have rolled back a file or as shown next a whole repo, and make a subsequent change to the rolled back project, **you must commit these changes** in order for them to be recorded and ensure the repo reflects at your desired change.

How to undo things in GIT

- **How do I revert changes introduced by a specific commit?**
 - When you need to undo something you've committed, you have a couple of good options.

```
git revert <commit_ID>
```

```
git reset --hard <commit_ID>
```

- Both of the options will only affect the current HEAD, so be sure to confirm that you have the intended branch checked out.

How to undo things in GIT

- **Revert:**

- As it sounds, the revert command changes all the files for a specific commit back to their state before that commit was completed.

```
git revert <commit_ID>
```

- It's important to note the mechanics of this command. The **reverted commit is not deleted**. Rather, Git creates a new commit with the included files reverted to their previous state.
- So your **version control history moves forward** while the state of **your files moves backwards**.

How to undo things in GIT

- **Reset:**

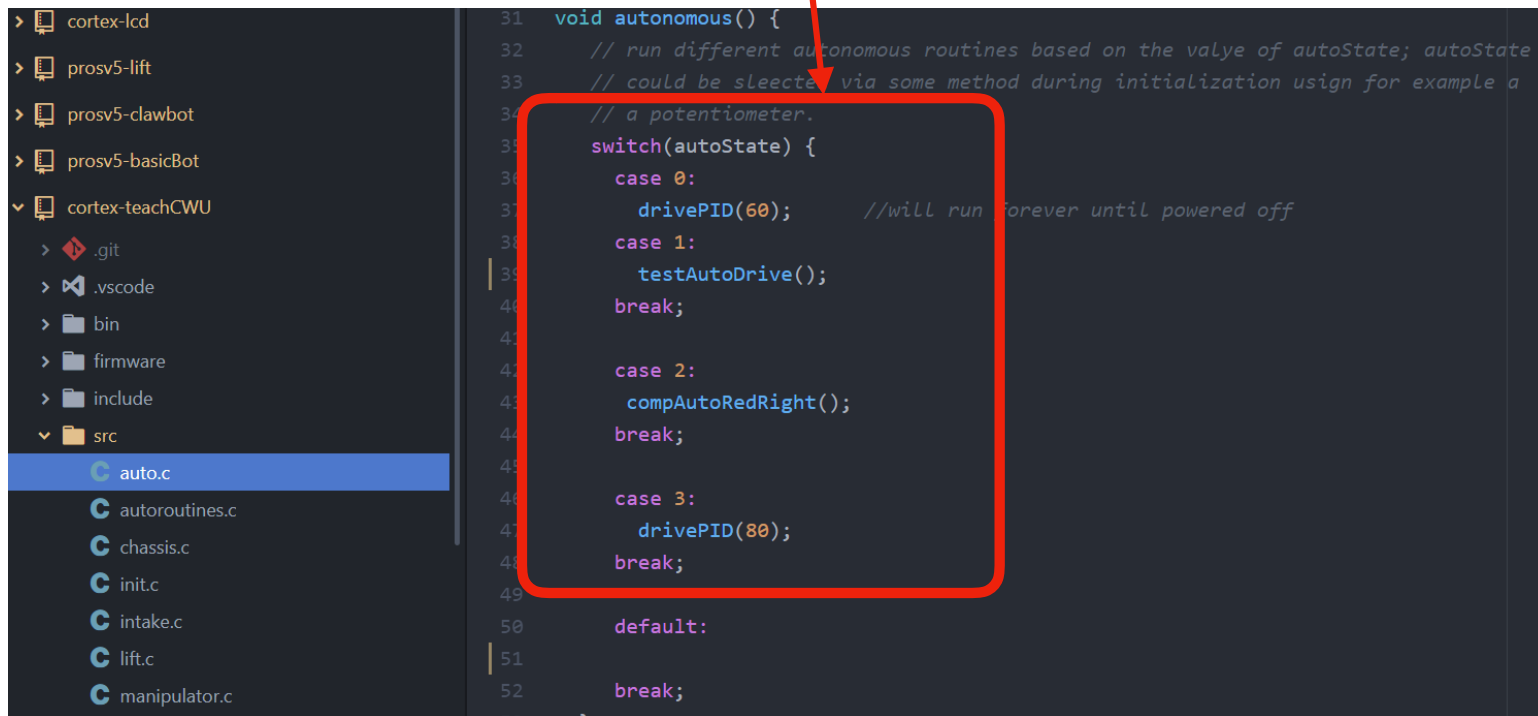
- This option is a little different than a revert. It **resets the status of your repo** (working HEAD) to an older revision. It's a **true rollback** of the state of your repo.

```
git reset --hard <commit_ID>
```

- When you use this option, **Git discards any commits between the current state of the repo and the target commit.** The branch will then appear to stop at the commit you reset the HEAD to.
- **Note:** *Although the commits no longer appear to be a part of your branch history, they are not deleted. They are still stored in Git.*

How to undo things in GIT

Assume that the current state of the repository (last commit) is reflected by the following src/auto.c file in your project.

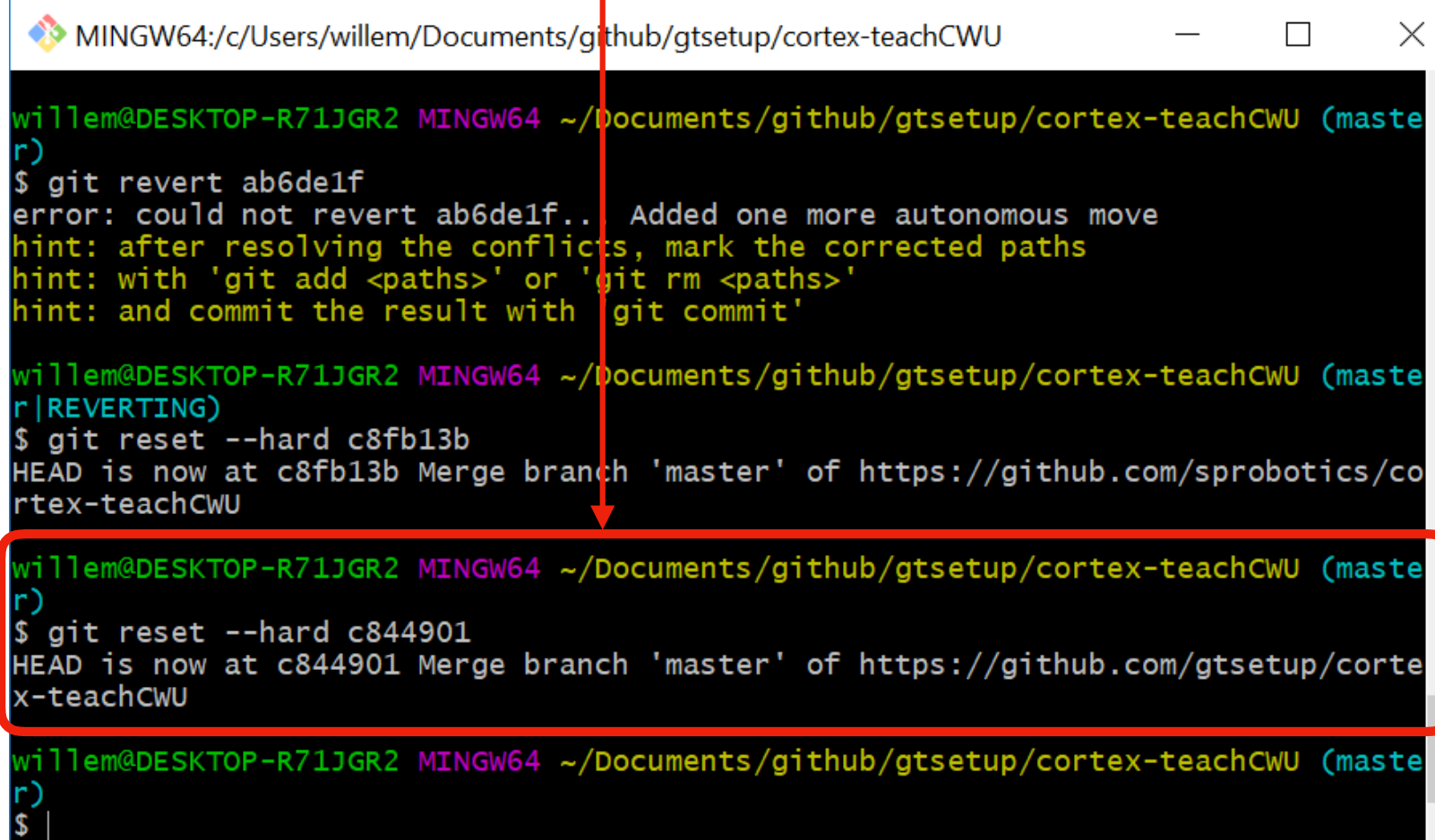


```
31 void autonomous() {
32     // run different autonomous routines based on the value of autoState; autoState
33     // could be selected via some method during initialization using for example a
34     // a potentiometer.
35     switch(autoState) {
36         case 0:
37             drivePID(60); //will run forever until powered off
38         case 1:
39             testAutoDrive();
40             break;
41         case 2:
42             compAutoRedRight();
43             break;
44         case 3:
45             drivePID(80);
46             break;
47         default:
48             break;
49     }
50 }
51
52 break;
```

How to undo things in GIT

Now we decide rollback (revert) the earlier commit with commit ID = c844901 using:

`git reset --hard c844901` this will now move your HEAD back to this committed state and effectively have rolled back to that point in time. Any changes going forward will be based off of this state of your repository.



```
MINGW64:/c/Users/willem/Documents/github/gtsetup/cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git revert ab6de1f
error: could not revert ab6de1f.. Added one more autonomous move
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'

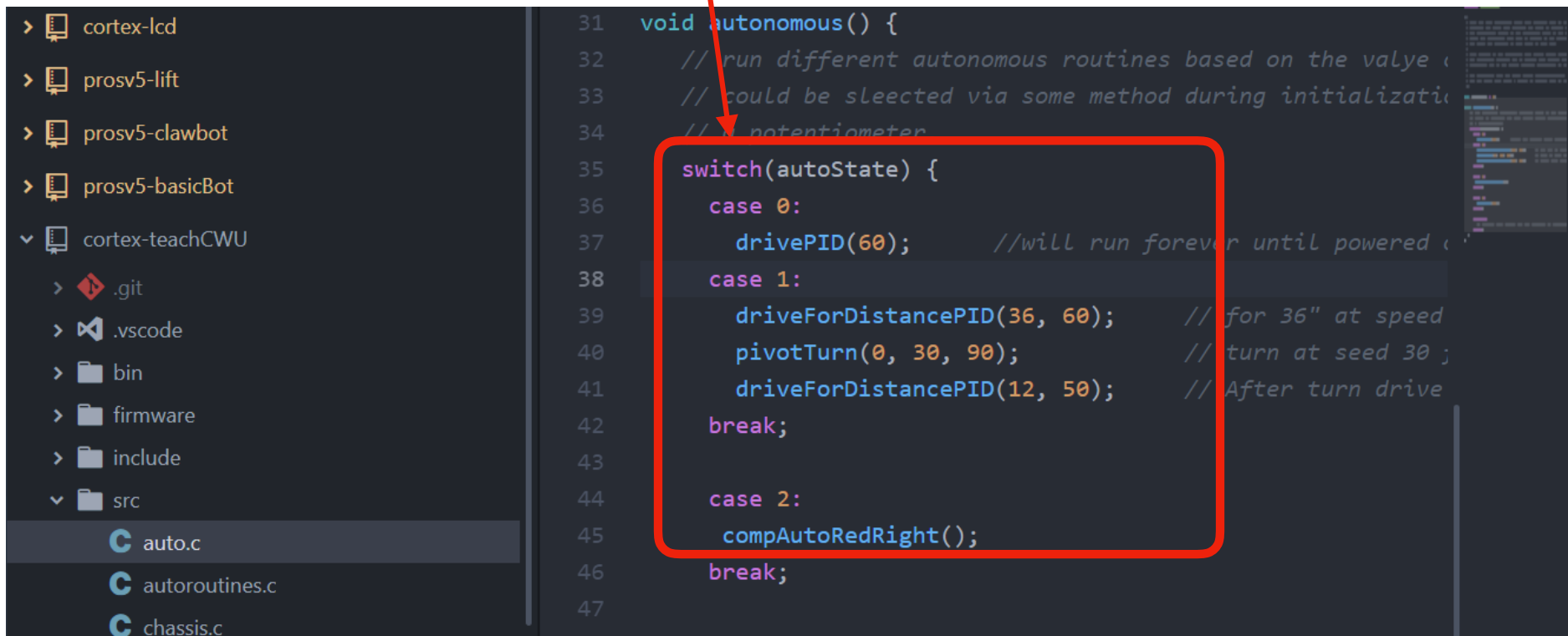
willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master|REVERTING)
$ git reset --hard c8fb13b
HEAD is now at c8fb13b Merge branch 'master' of https://github.com/sprobotics/cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$ git reset --hard c844901
HEAD is now at c844901 Merge branch 'master' of https://github.com/gtsetup/cortex-teachCWU

willem@DESKTOP-R71JGR2 MINGW64 ~/Documents/github/gtsetup/cortex-teachCWU (master)
$
```

How to undo things in GIT

The new current state of the repository (after reset —hard) is reflected by the following src/auto.c file in your project.



```
31 void autonomous() {
32     //run different autonomous routines based on the valye (
33     //could be sleected via some method during initializati
34     //potentiometer
35     switch(autoState) {
36         case 0:
37             drivePID(60); //will run forever until powered c
38         case 1:
39             driveForDistancePID(36, 60); // for 36" at speed
40             pivotTurn(0, 30, 90); // turn at seed 30 ;
41             driveForDistancePID(12, 50); // After turn drive
42             break;
43
44         case 2:
45             compAutoRedRight();
46             break;
47     }
```

GIT integration with PROS / Atom

PROS and GIT integration

- GIT is **fully integrated with Atom** and does the PROS programming environment.
- When initially creating a new PROS project, you will need to use gitHUB desktop client to initialize the REPO on gitHUB

PROS and GIT integration

When a new project is created in PROS - there will be no GIT repository until that is separately created. This can be done right in the RPOS / Atom interface.

The screenshot displays the Atom IDE interface with the PROS project 'prosv5-gitTest' open. The left sidebar shows the project tree with folders like '.vscode', 'bin', 'firmware', 'include', and 'src'. The main editor window shows the 'opcontrol.cpp' file with the following code:

```
1 #include "main.h"
2
3 /**
4  * Runs the operator control code. This function will be started in its
5  * with the default priority and stack size whenever the robot is enabled
6  * the Field Management System or the VEX Competition Switch in the oper
7  * control mode.
8  *
9  * If no competition control is connected, this function will run immedi
10 * following initialize().
11 *
12 * If the robot is disabled or communications is lost, the
13 * operator control task will be stopped. Re-enabling the robot will res
14 * task, not resume it from where it left off.
15 */
16 void opcontrol() {
17     pros::Controller master(pros::E_CONTROLLER_MASTER);
18     pros::Motor left_mtr(1);
19     pros::Motor right_mtr(2);
20     while (true) {
21         pros::lcd::print(0, "%d %d %d", (pros::lcd::read_buttons() & LCD_BTN
22             (pros::lcd::read_buttons() & LCD_BTN_CENTER) >> 1,
23             (pros::lcd::read_buttons() & LCD_BTN_RIGHT) >> 0);
24         int left = master.get_analog(ANALOG_LEFT_Y);
25         int right = master.get_analog(ANALOG_RIGHT_Y);
26
27         left_mtr = left;
28         right_mtr = right;
29         pros::delay(20);
30     }
31 }
32
```

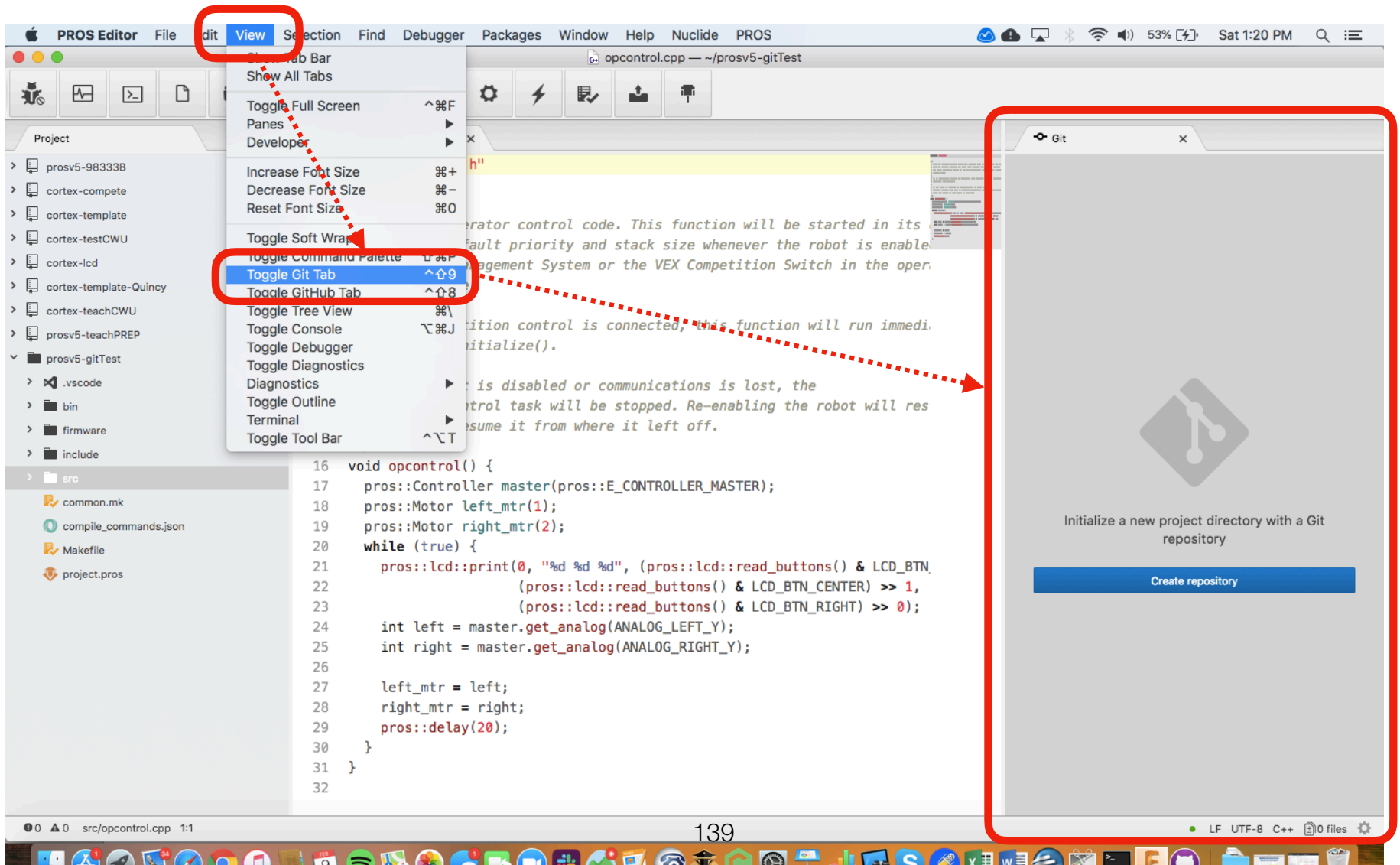
On the right side, the Git extension interface is visible, showing a 'Create repository' button and the text 'Initialize a new project directory with a Git repository'.

Page number: 138

Bottom status bar: 0 0 ▲ 0 src/opcontrol.cpp 1:1 | LF UTF-8 C++ 0 files

PROS and GIT integration

If the GIT pane does not show initially, it can be toggled by going to View -> ToggleGit Tab



PROS and GIT integration

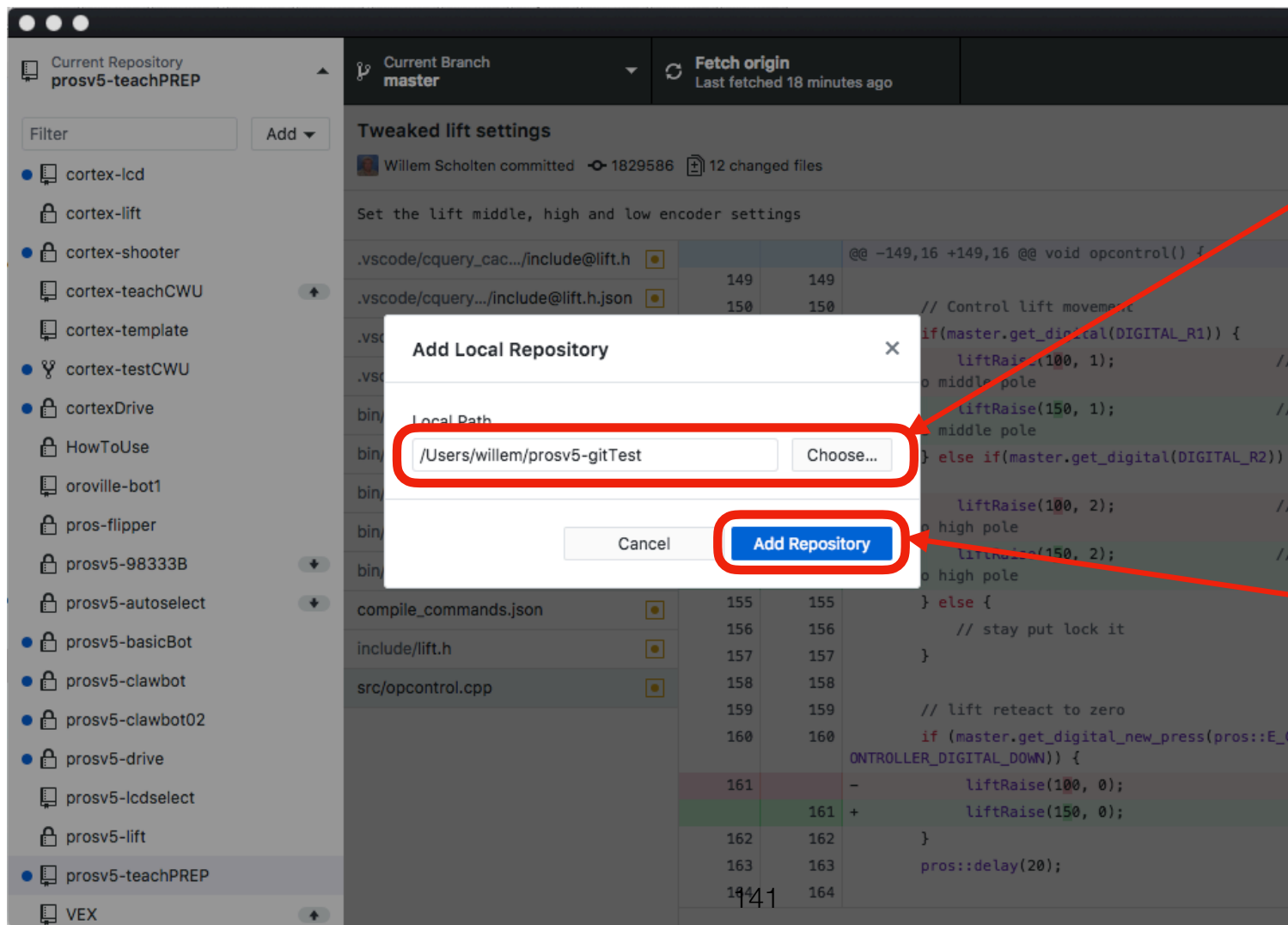
To initialize the repository:

Click the **Create repository** button then point to the path where your repository is located and click **+init**



PROS and GIT integration

To create the remote repository - on gitHUB - open the gitHUB desktop client, and select **File -> Add Local Repository** — then choose the local path of your PROS project where also your local git repository lives.

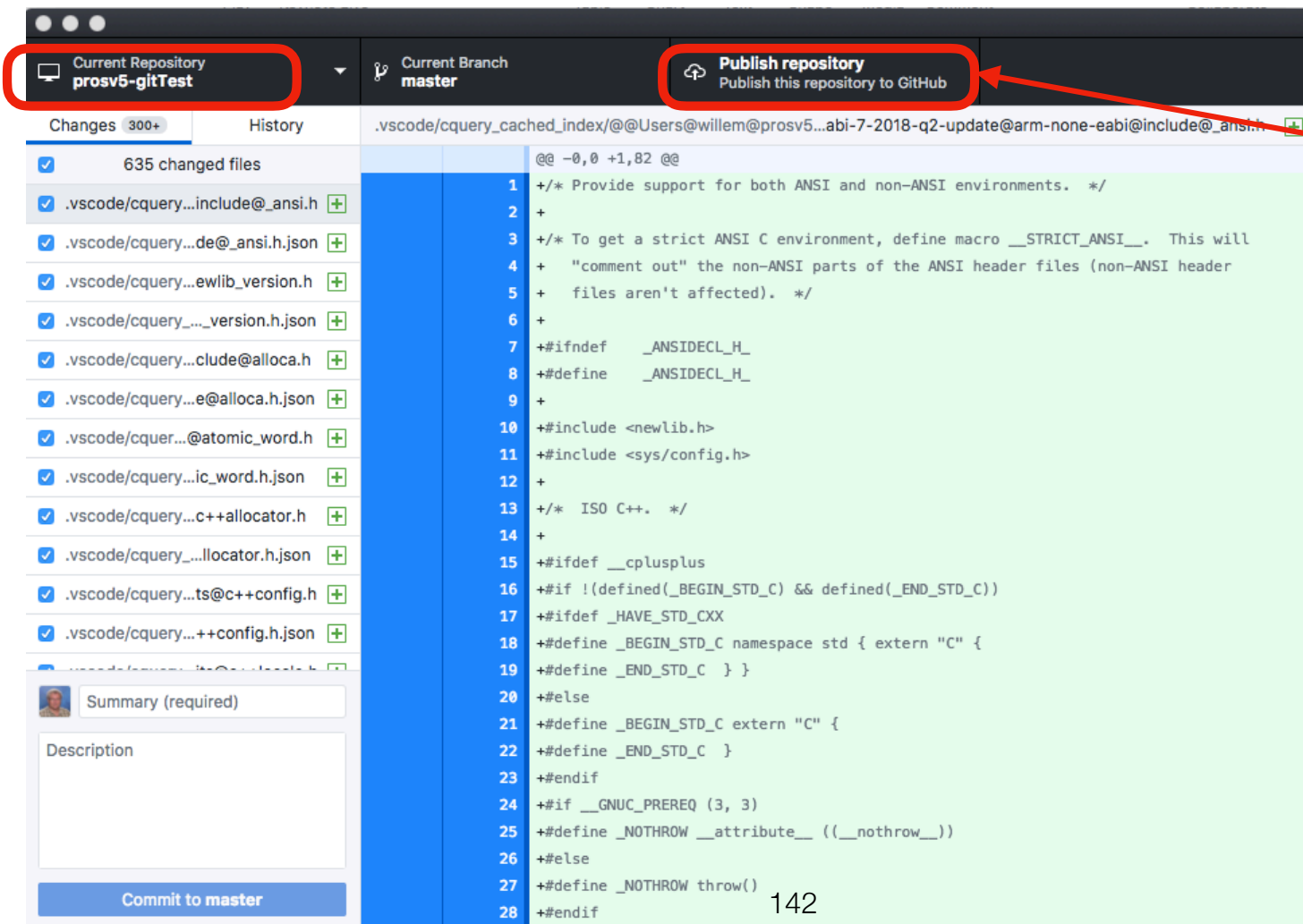


Point to the PROS project we just created and for which we initialized a local repository

Then 'Add Repository'

PROS and GIT integration

To create the remote repository - on gitHUB - open the gitHUB desktop client, and select **File -> Add Local Repository** — then choose the local path of your PROS project where also your local git repository lives.



Now publish the Repository - this will initialize it on gitHUB

PROS and GIT integration

Now we have added the local repository to the gitHub desktop system, and we are ready to initialize the remote repository by clicking 'Publish repository'

The screenshot shows the GitHub Desktop application interface. At the top, the 'Current Repository' is 'prosv5-gitTest' and the 'Current Branch' is 'master'. A 'Publish repository' button is visible in the top right. The main area shows 'No local changes'. A 'Publish Repository' dialog box is open, with a red rectangle highlighting its content. Red arrows point from text boxes to specific elements in the dialog:

- An arrow points from the 'Description' field to the text: 'Add brief description of purpose'.
- An arrow points from the 'Keep this code private' checkbox to the text: 'Select **optional Organization** - depends on type of gitHub account you have, same with the ability to **keep repository private**'.
- An arrow points from the 'Publish Repository' button to the text: 'Publish the repository'.

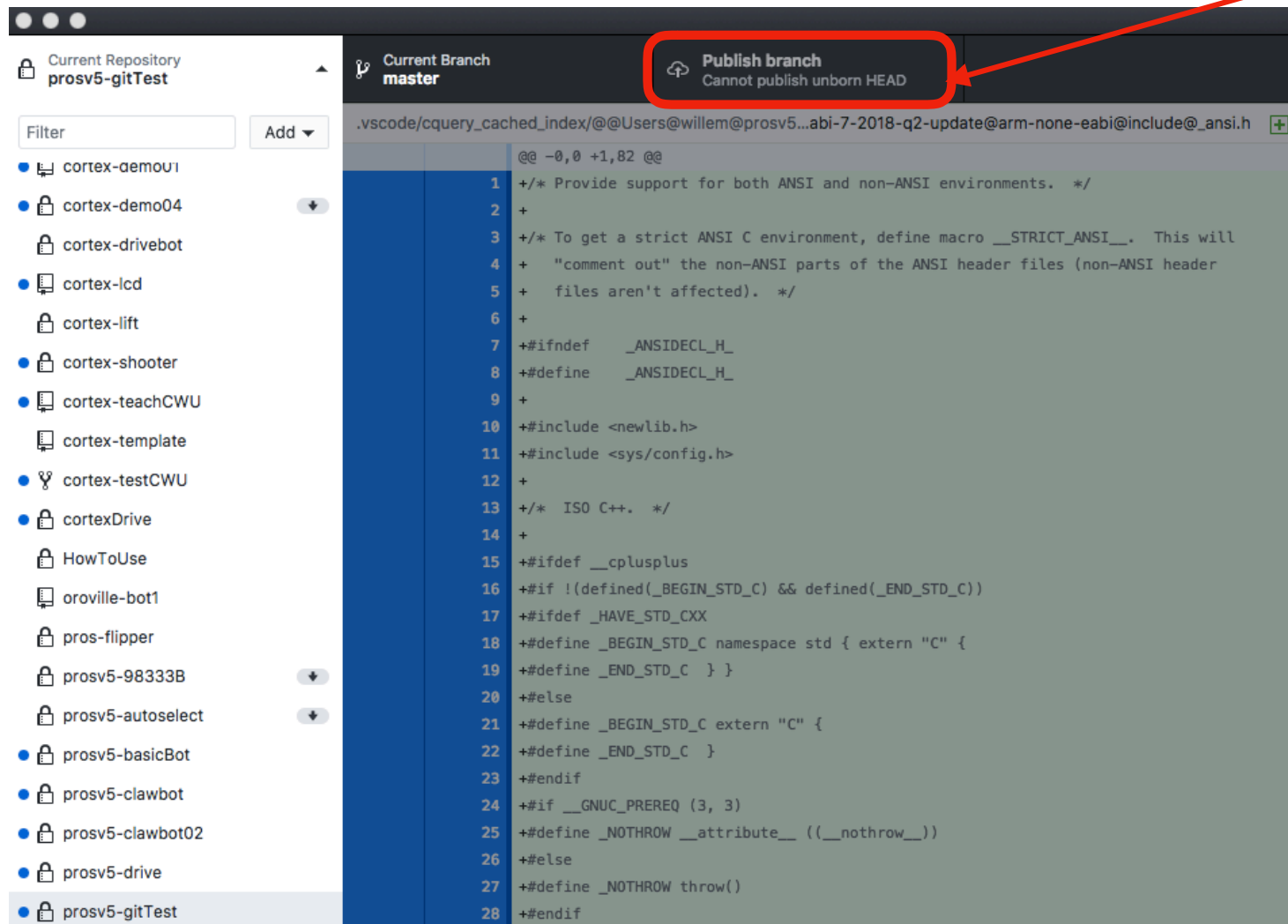
The dialog box contains the following fields and options:

- Repository type: GitHub.com (selected) or Enterprise.
- Name: prosv5-gitTest
- Description: Test Repository for PROS git testing
- ☒ Keep this code private
- Organization: sprobotics
- Buttons: Cancel, Publish Repository

At the bottom of the screen, the status bar shows 'Committed 5 minutes ago' and 'Initial Project Commit'.

PROS and GIT integration

When the repository is published you will receive a status of: **Cannot publish unborn HEAD**



We now **MUST** first make an initial commit to the local repository in PROS

PROS and GIT integration

The screenshot shows the Git interface in Visual Studio Code. The 'Unstaged Changes' section is highlighted with a red box and contains three files. The 'Staged Changes' section is empty and labeled 'No changes'. The 'Commit message' section is visible below. The 'Git' status bar at the bottom is also highlighted with a red box.

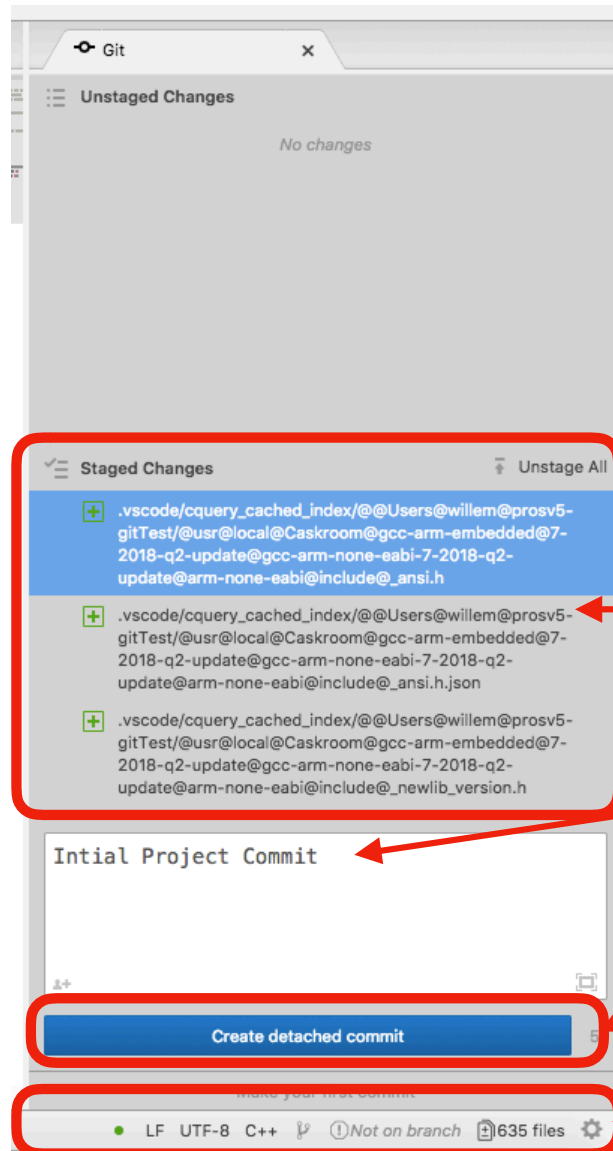
Click 'Stage All' to stage files for Commit.

Once initialized, you will likely see a number of unstated files

Files must be stage prior to be committed to the repository

Git status bar

PROS and GIT integration



NOTE: we are committing field only to our local repository at this point, we do not yet have a repository setup on gitHUB to share our projects and track/protect our changes in the cloud

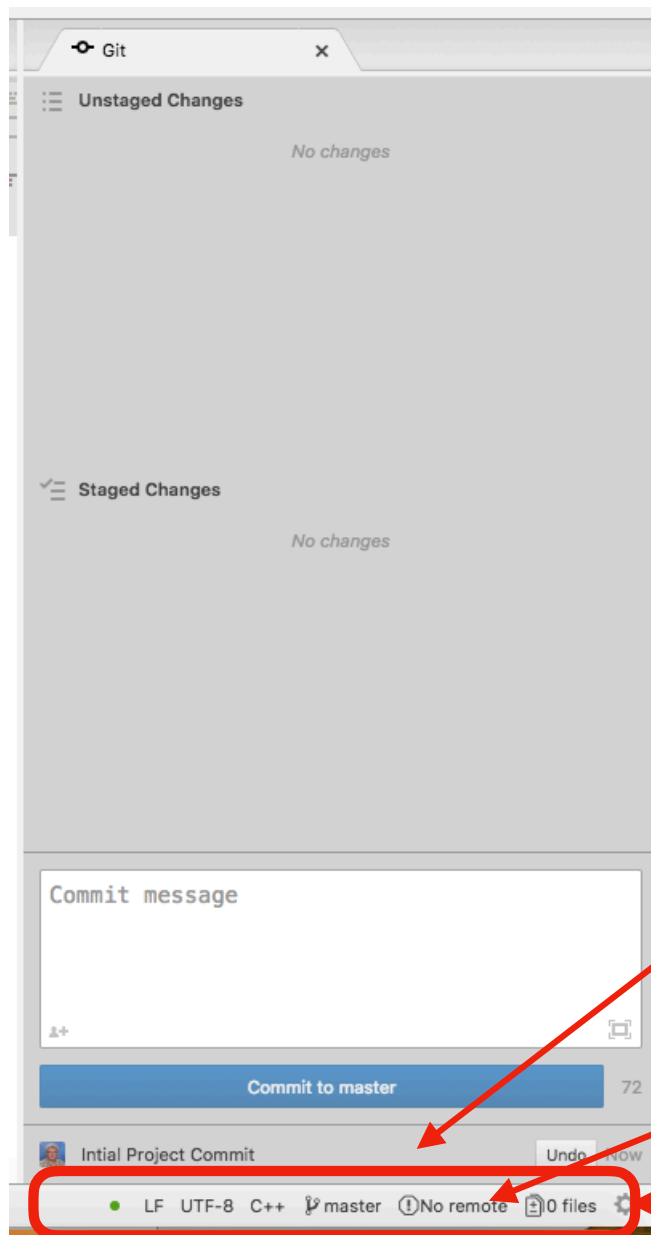
Files ready to be committed to the repository

Write a Commit Message

Commit to the local repository

Git status bar

PROS and GIT integration



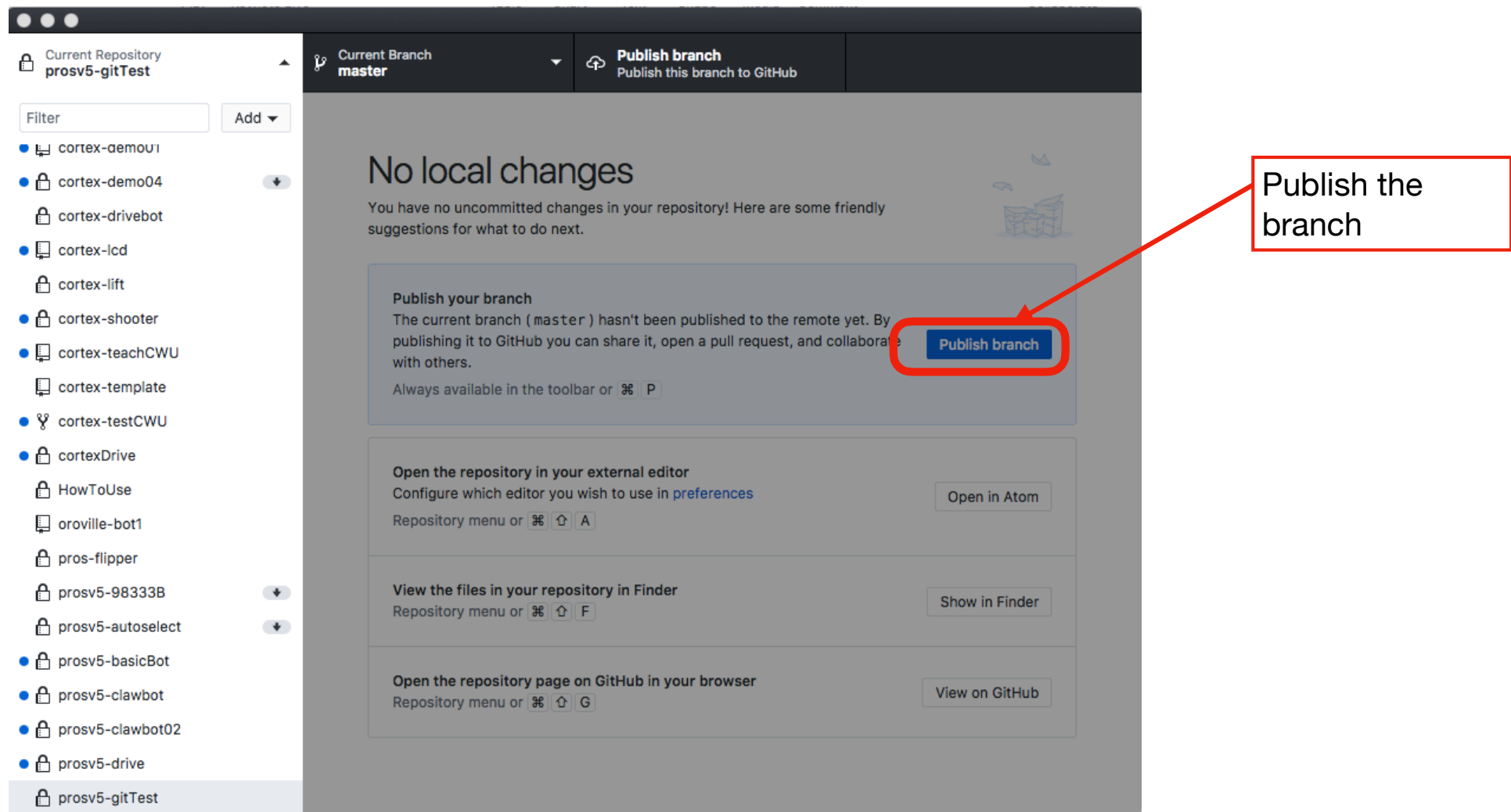
We have our first local commit to the repository

NOTICE: (!) No remote — we need to link our local repository next to the a remote repository on gitHUB

Git status bar

PROS and GIT integration

Now go back to the gitHUB client and publish the just committed local repo



PROS and GIT integration

Now in PROS the git pane should have your repo committed and pushed to gitHUB, going forward you can now use the build in git pane in PROS to manage your push/pull actions of the repo.

The screenshot shows the PROS IDE interface. On the left is a project tree for 'prosv5-98333B'. The center pane shows the code for 'opcontrol.cpp'. On the right is the 'Git' pane, which displays 'Unstaged Changes' and 'Staged Changes'. Below these is a 'Commit message' field and a 'Commit to master' button. At the bottom of the Git pane, there is a list of commits, including 'Initial followup commit' and 'Initial Commit'. A red circle highlights the 'Pull 1' button, with a red arrow pointing to it from a text box on the right.

Sometimes the local repository maybe still 'disconnected' from the remote repo.

You can fix this by right-click on the 'error' and selecting Force Push

PROS and GIT integration

When we publish initially the repository remotely, we may receive a message stating:

(!) Unable to merge unrelated histories in this repository.

This can be fixed by issuing a special git command:

git pull origin master --allow-unrelated-histories

Open a local terminal, change to your git local repository, in our case cd prosv5-gitTest and issue the above command to resolve the problem and have the local history synchronized with the remote repository.

```
drwxr-xr-x 5 willem staff 160 Dec 29 17:53 src
Willems-MacBook-Air:~ willem$ cd prosv5-gitTest
Willems-MacBook-Air:prosv5-gitTest willem$ git pull origin master --allow-unrelated-histories
From https://github.com/sprobotics/prosv5-gitTest
* branch      master      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
Willems-MacBook-Air:prosv5-gitTest willem$ git pull origin master --allow-unrelated-histories
From https://github.com/sprobotics/prosv5-gitTest
* branch      master      -> FETCH_HEAD
Already up to date.
Willems-MacBook-Air:prosv5-gitTest willem$
```

Issue git
command to
resolve

git Branches in PROS

- It is often best to code for your project using **branches**, this is especially the case when more than one programmer is working on the project - most likely scenario
- Each programmer would go ahead and create / checkout a branch of the current master.
- These branches are worked on and committed and pushed to the remote repository
- At some point in time the branches are reviewed and merged into the master, at which point the process repeats itself.

git Branches in PROS

- The PROS integration into Atom and does integrated with git makes creating and managing your local branch painless if you use the following simple steps:
 - Step 1: make sure you have fetched the most uptodate master branch
 - Step 2: create your local branch
 - Step 3: write code, update / commit /push your local branch
 - Step 4: code review - merge branches into master and repeat.

git Branches in PROS

With your current project open and the latest master fetched, click on the 'master' branch button, to go ahead and create your own local branch.

The screenshot shows the PROS IDE interface with the Git panel on the right. The 'master' branch button is highlighted with a red circle and an arrow pointing to it from a text box that says "Click on 'master' to open the branch management dialogue".

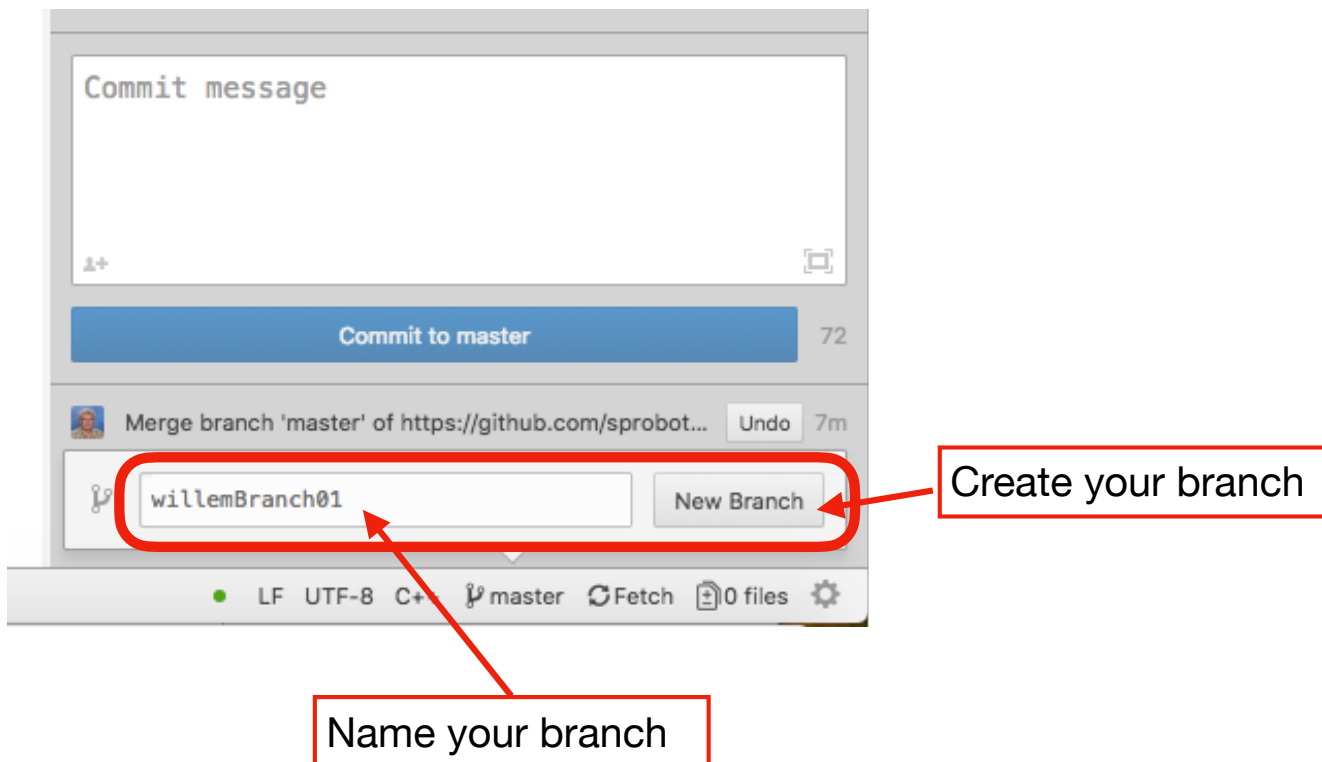
The Git panel shows the following sections:

- Unstaged Changes: No changes
- Staged Changes: No changes
- Commit message: (empty text area)
- Commit to master: 72
- Initial followup commit: 4m
- Initial Commit: 10m

The 'master' branch button is located at the bottom of the Git panel, next to the 'Pull 1' button and the '0 files' indicator.

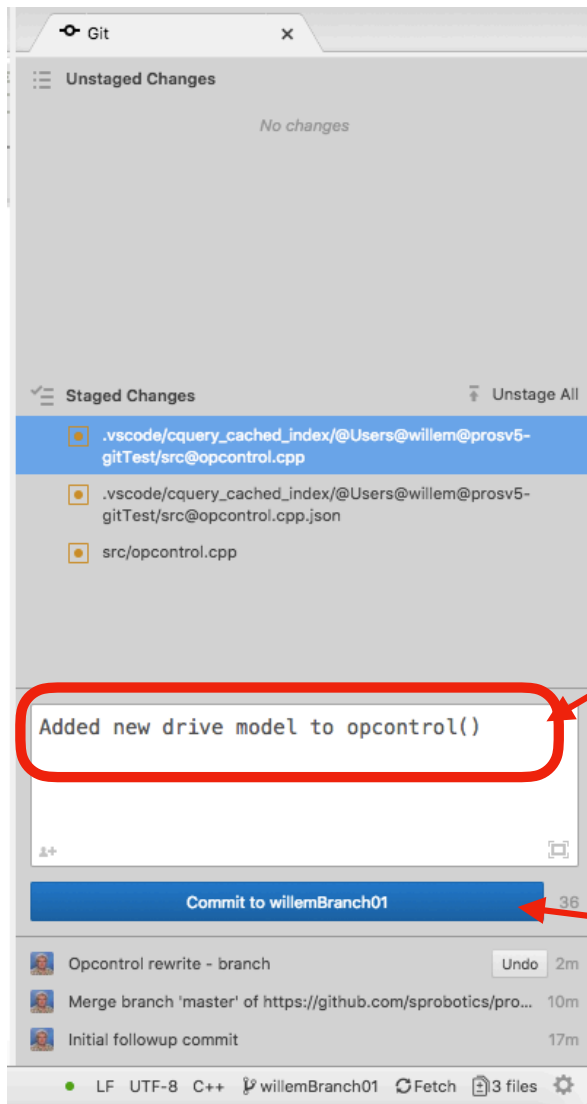
git Branches in PROS

Creating your own code branch



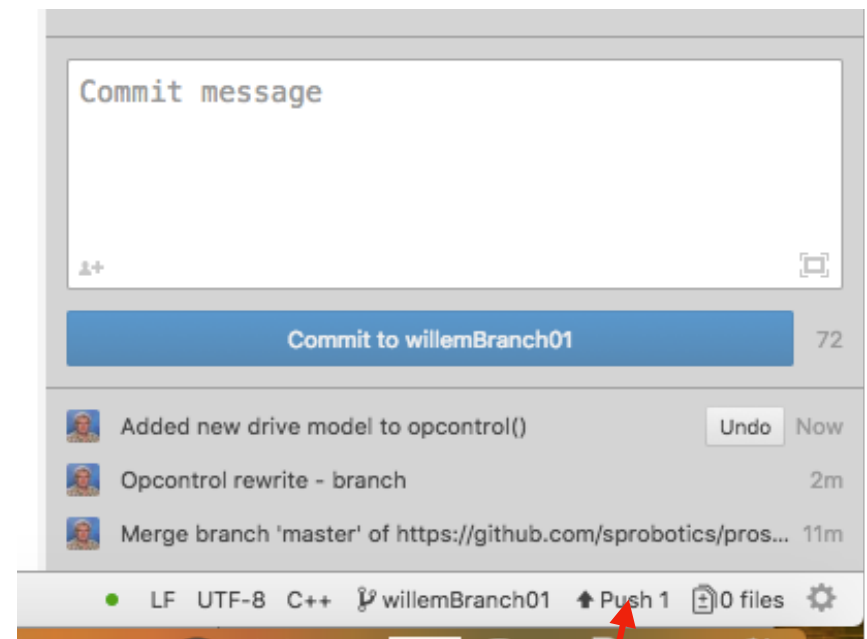
git Branches in PROS

Write your code / fix code, save it, and then Stage All — now you are ready to commit your changes to your repo, and subsequently push it up to the remote repo



Add commit message

Commit changes to your branch



Push your local changes to the remote repo

git Branches in PROS

If you now look at the gitHUB page for the repo - the remote repo storage on gitHUB - you will notice that there are branches and various commits

The screenshot shows the GitHub interface for the repository 'sprobotics / prosv5-gitTest'. A red box highlights the repository name and its private status. Another red box highlights the commit and branch statistics: '4 commits' and '2 branches'. A red arrow points from a text box on the right to the '2 branches' text. Below the statistics, a section titled 'Your recently pushed branches:' shows a branch named 'willemBranch01' pushed 1 minute ago. At the bottom, a table lists recent commits and file changes.

Commit Message	Time Ago
seaprep Merge branch 'master' of https://github.com/sprobotics/prosv5-gitTest	Latest commit 50767b2 13 minutes ago
.vscode/cquery_cached_index	Initial followup commit 17 minutes ago
bin	Initial Commit 26 minutes ago
firmware	Initial Commit 26 minutes ago
include	Initial Commit 26 minutes ago
src	Initial followup commit 17 minutes ago
Makefile	Initial Commit 26 minutes ago
common.mk	Initial Commit 26 minutes ago

Commit and Branch commit status

git Branches in PROS

If you now look at the gitHUB page for the repo - the remote repo storage on gitHUB - you will notice that there are branches and various commits

sprobotics / prosv5-gitTest Private

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 ZenHub Projects 0 Wiki Insights Settings

Overview Yours Active Stale All branches Search branches

Default branch

master Updated 34 minutes ago by seaprep Default Change default branch

Your branches

willemBranch01 Updated 23 minutes ago by seaprep 0 | 2 New pull request

Active branches

willemBranch01 Updated 23 minutes ago by seaprep 0 | 2 New pull request

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

MASTER branch against we will later apply the branch changes

your committed branch

git Branches in PROS

Now that we have branches we can go ahead and decide to merge them into the master. We do this by **comparing** the branch with the master and requesting a **pull** from the branch into the master

The screenshot shows the GitHub interface for the repository 'sprobotics / prosv5-gitTest'. The repository is private and has 1 watch, 0 stars, and 0 forks. The 'Code' tab is selected. Below the repository name, there are statistics: 4 commits, 2 branches, 0 releases, and 1 contributor. A section titled 'Your recently pushed branches:' shows a branch named 'willemBranch01' pushed 41 minutes ago. A green button labeled 'Compare & pull request' is highlighted with a red box. An arrow points from a text box containing 'initiate a compare & pull' to this button. Below the button, there are options to 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. At the bottom, a table shows the commit history for the 'master' branch, with the latest commit being 'Merge branch 'master' of https://github.com/sprobotics/prosv5-gitTest' by 'seaprep'.

File	Commit Message	Time
.vscode/cquery_cached_index	Initial followup commit	an hour ago
bin	Initial Commit	an hour ago
firmware	Initial Commit	an hour ago
include	Initial Commit	an hour ago
src	Initial followup commit	an hour ago

git Branches in PROS

Step 1: check if we can merge - if not resolve the conflicts

Step 2: write reason for merge and short description

Step 3: create the pull request to initiate the merge

sprobotics / prosv5-gitTest Private

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 ZenHub Projects 0 Wiki Insights Settings

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master compare: willemBranch01

✓ Able to merge. These branches can be automatically merged.

Willem branch01 being merged and resolved

Write Preview

The proposed `opcontrol()` changes are accepted for merging into the master

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

Styling with Markdown is supported

Create pull request

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Sign in now to use ZenHub

Check if able to merge or resolve conflicts

Name and explain merge reason

initiate merge through pull request

git Branches in PROS


When the pull request is made, you need to complete one more step, if fully merging the pull request into the master

Willem branch01 being merged and resolved #1


Edit

 Open seaprep wants to merge 2 commits into master from willemBranch01

 Conversation 0

 Commits 2

 Checks 0

 Files changed 3

+5 -1 




seaprep commented just now

+ 😊 ...

The proposed opcontrol() changes are accepted for merging into the master

 seaprep added some commits an hour ago

 Opcontrol rewrite - branch

718969d

 Added new drive model to opcontrol()

d185911

Add more commits by pushing to the willemBranch01 branch on sprobotics/prosv5-gitTest.



Continuous integration has not been set up

Several apps are available to automatically catch bugs and enforce style.



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also open this in [GitHub Desktop](#) or view [command line instructions](#).

Reviewers



No reviews

Assignees



No one—assign yourself

Labels



None yet

Projects

None yet

Milestone

No milestone

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

Double check their are no conflicts, if so resolve them.

Complete merge pull request

git Branches in PROS

When the merge completes, you will be given a view of it's current status, including the ability to remove the just merged in branch

Willem branch01 being merged and resolved #1

Merged seaprep merged 2 commits into master from willemBranch01 just now

Conversation 0 Commits 2 Checks 0 Files changed 3 +5 -1

seaprep commented 2 minutes ago

The proposed opcontrol() changes are accepted for merging into the master

seaprep added some commits an hour ago

- Opcontrol rewrite - branch
- Added new drive model to opcontrol()

seaprep merged commit 672d33a into master just now

Merge completed

Pull request successfully merged and closed
You're all set—the willemBranch01 branch can be safely deleted.

Delete branch

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Revert

Optional delete Branch

git Branches in PROS

gitHUB shows that the branch which was successfully merged into the master was deleted.

Willem branch01 being merged and resolved #1

Edit

Merged seaprep merged 2 commits into master from willemBranch01 2 minutes ago

Conversation 0

Commits 2

Checks 0

Files changed 3

+5 -1



seaprep commented 4 minutes ago

+ 😊 ...

The proposed opcontrol() changes are accepted for merging into the master

seaprep added some commits an hour ago

Opcontrol rewrite - branch

Added new drive model to opcontrol()

seaprep merged commit 672d33a into master 2 minutes ago

seaprep deleted the willemBranch01 branch 10 seconds ago

Revert

Restore branch

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

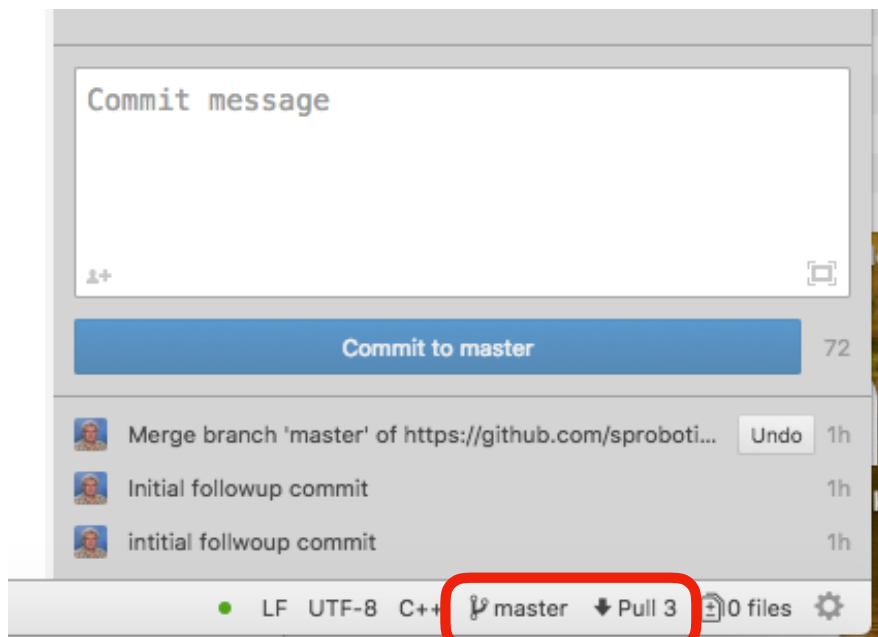
Milestone

Branch has been deleted

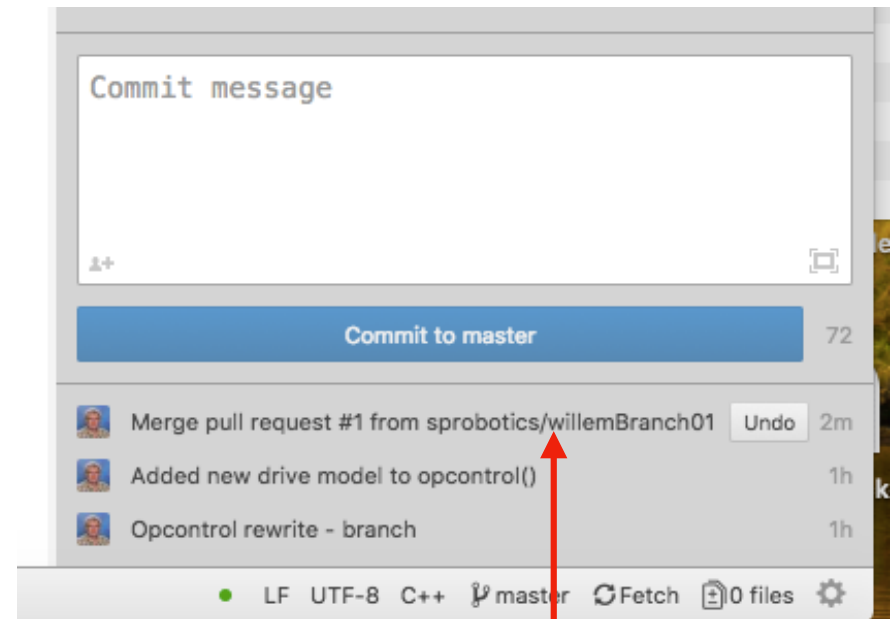
Notice even now you can still recover.

git Branches in PROS

Now in PROS in your project, change back to the 'master' branch, and initiate a pull/fetch to make sure that the local master matches the remote repo master.



Merge completed — change back to master and issue fetch pull request



Master is now synched locally