GITLAB FOR MECHANICAL ENGINEERS

(V5, FEB 2022)

EXECUTIVE SUMMARY

ONBOARDING

1. Get your Gitlab credentials set up
2. Download GitHub Desktop
3. Create your Personal Access Token
4. Clone the repositories you need and initialize Git LFS

TYPICAL WORKFLOW

1. Fetch updates from the repo
2. Pull updates from the repo
3. Make a branch for your edits
4. Do work
5. Commit files that you worked on
6. Push updates to your branch, resolving conflicts as needed
7. Request to merge updates on your branch into main

TABLE OF CONTENTS

INTRODUCTION

We are moving away from Subversion (SVN) due to issues with hosting. Instead, we will now be using GitLab, a Git-based hosting service that is similar in principle to SVN. This guide details the process to get started with and use our new CAD versioning system and includes a snapshot of our slightly changed folder structure.

ONBOARDING

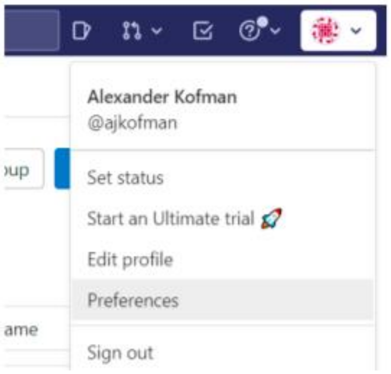
Invitations – The process begins when a team leader invites you to the GitLab workspace. With your invitation, you will also be assigned one of these permission levels:

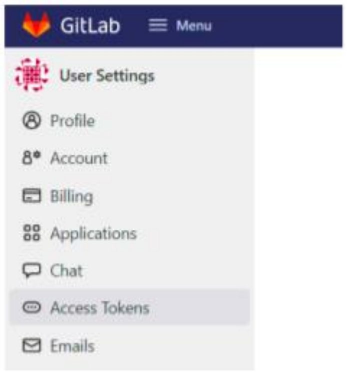
- Developer – Able to contribute new files and push to branches but cannot merge into master. Most mechanical members will have this role.
- Maintainer – Able to push to master and deploy to production. Most subteam leaders will have this role.
- Owner – Can change user permissions and has destructive and awesome powers. Only top team leaders (tech director / team principal) will have this power.

Set up your account from the invitation email, making sure to set your username to something recognizable. Also, don't forget to set your real name as well so we can recognize you.

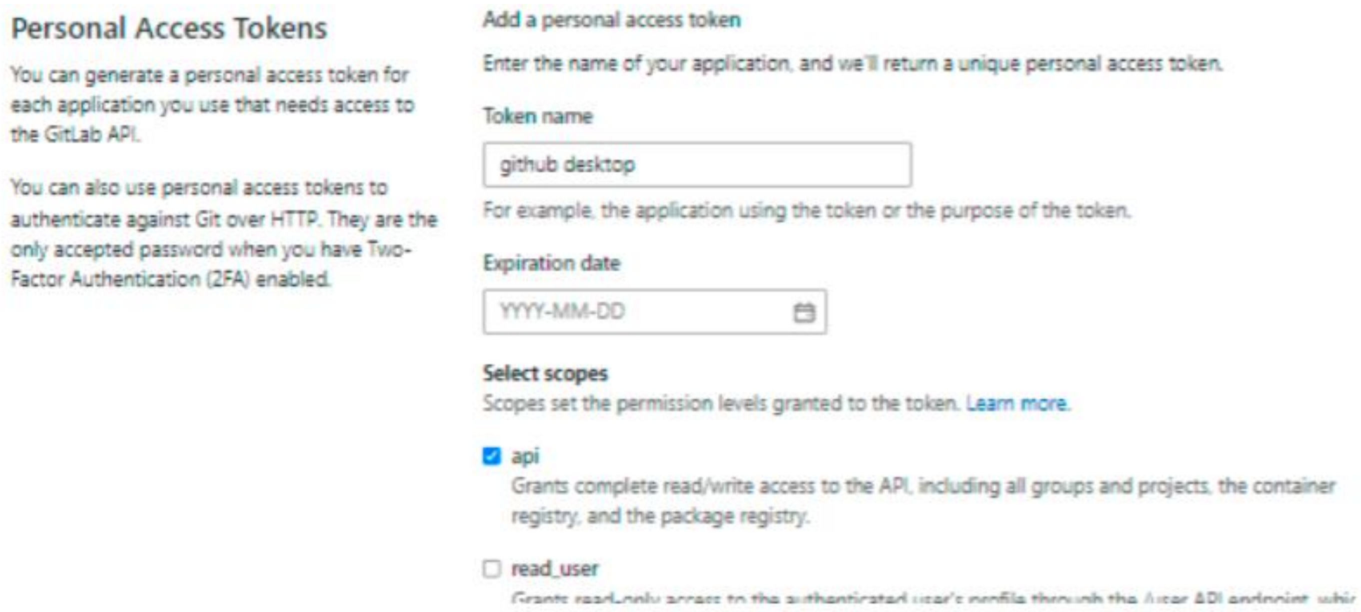
Client – We strongly recommend GitHub Desktop as the standard client for team members. Advanced users can pick their own client or use shell commands like a wizard, but you'll be on your own if you run into issues. GitHub Desktop can be downloaded here for Windows and Mac: desktop.github.com. When you install GitHub Desktop, skip the option to log in to GitHub. You will use other credentials when you clone repos instead.

Personal Access Key – This key will be your ticket to avoiding entering your username and password a lot. To create it, navigate to your settings in Gitlab by clicking Preferences in the top right corner dropdown, then choosing "Access Tokens" on the left.





Enter the name of your token ("github desktop" will do fine) and check the "api" box below. Leave the expiration field and other checkboxes empty.



Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Token name: github desktop

For example, the application using the token or the purpose of the token.

Expiration date: YYYY-MM-DD

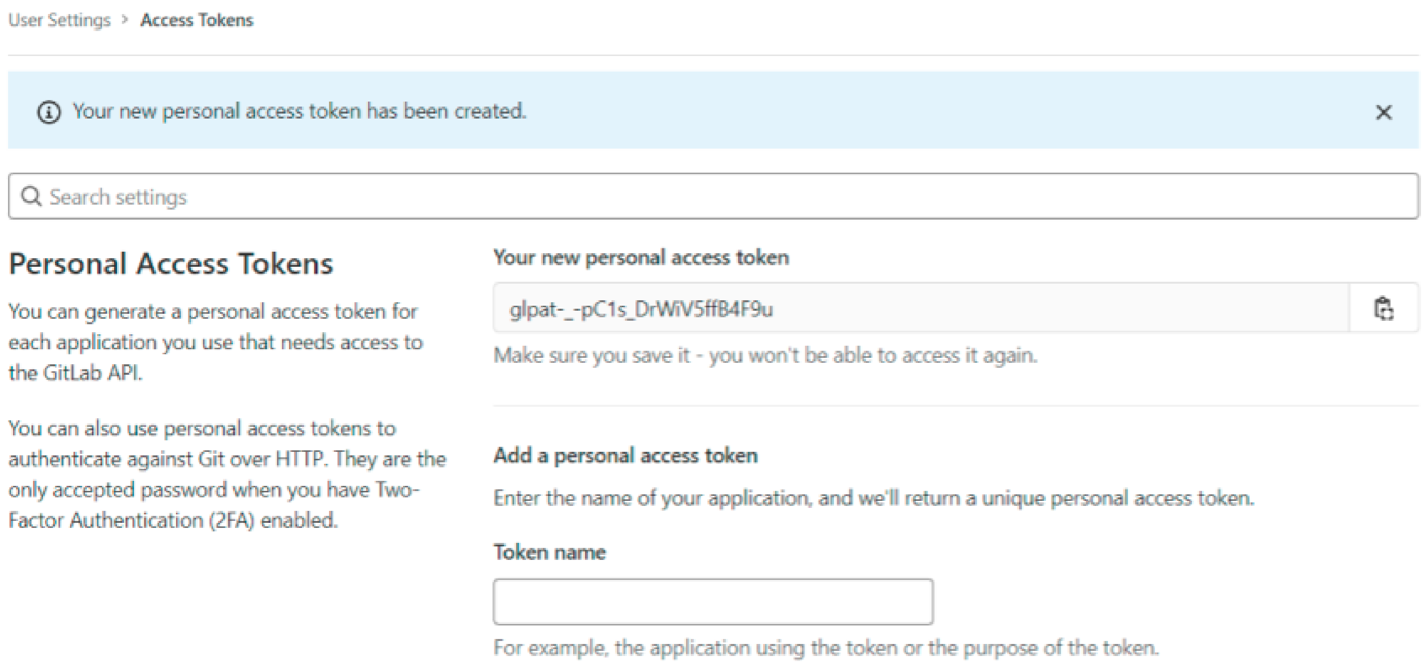
Select scopes

Scopes set the permission levels granted to the token. Learn more.

☑ api
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

☐ read_user

Click the blue "Create personal access token" at the bottom. You should then see your token appear at the top of the form. Copy this token and save it somewhere safe – it cannot be recovered if lost. You will need this token when you first clone a repo.



User Settings > Access Tokens

Your new personal access token has been created.

Search settings

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Your new personal access token

glpat-_-pC1s_DrWiV5ffB4F9u

Make sure you save it - you won't be able to access it again.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

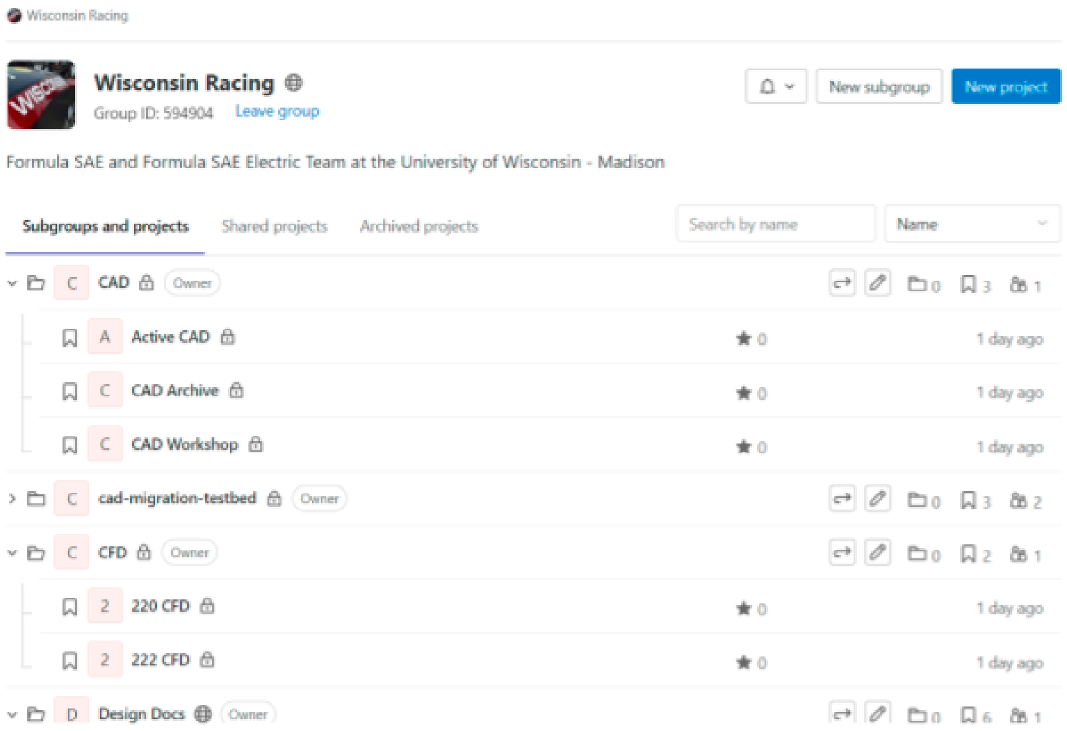
Token name

For example, the application using the token or the purpose of the token.

HOW TO USE GIT

Git is similar to Subversion: the repository (repo) is stored on the GitLab server, and you keep a "working copy" on your machine to make changes.

Cloning (Checking Out) the Repository – In Subversion, we used SVN Checkout to retrieve a working copy from the repo. In git, the same action is called cloning. In GitHub Desktop, go to File > Clone Repository (Ctrl + Shift + O on Windows). Select the URL tab and input the URL of the repository to clone. The easiest way to do this is to navigate to gitlab.com/wisconsinracing and find the folder you want.



Wisconsin Racing

Wisconsin Racing
Group ID: 594904 Leave group

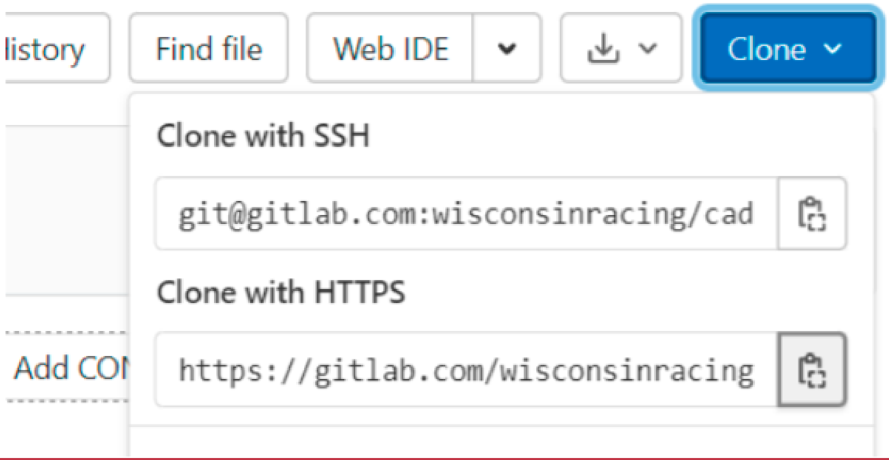
New subgroup New project

Formula SAE and Formula SAE Electric Team at the University of Wisconsin - Madison

Subgroups and projects Shared projects Archived projects Search by name Name

C CAD Owner	0 3 1		
A Active CAD		★ 0	1 day ago
C CAD Archive		★ 0	1 day ago
C CAD Workshop		★ 0	1 day ago
C cad-migration-testbed Owner	0 3 2		
C CFD Owner	0 2 1		
2 220 CFD		★ 0	1 day ago
2 222 CFD		★ 0	1 day ago
D Design Docs Owner	0 6 1		

When you click into the folder, hit the blue Clone button on the right. Then, hit the clipboard icon next to "Clone with HTTPS" to copy the correct URL to the clipboard.



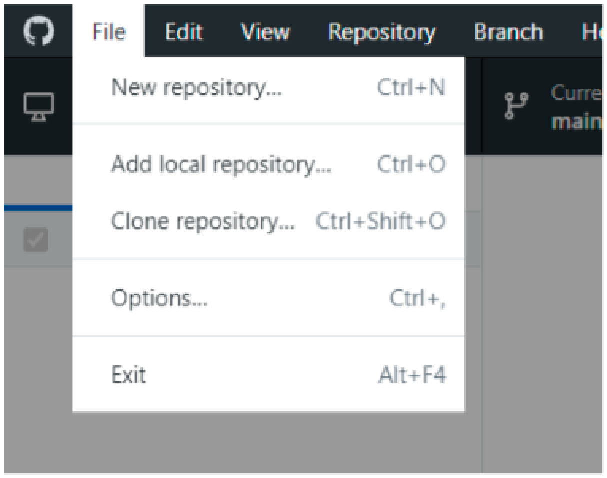
Clone with SSH

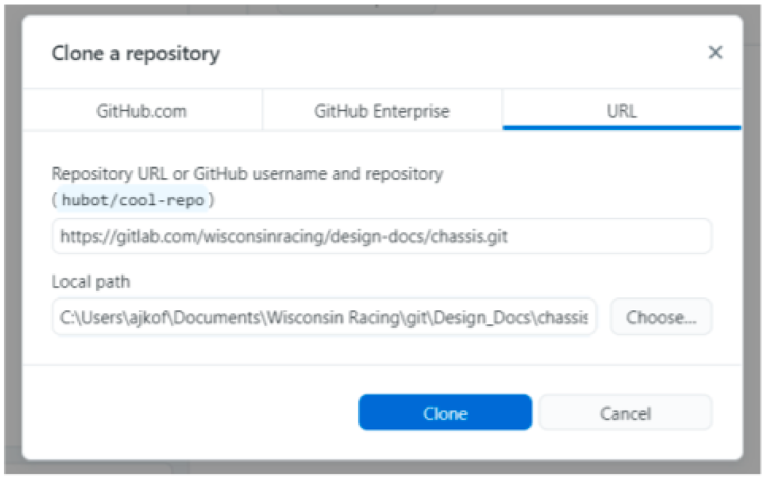
git@gitlab.com:wisconsinracing/cad

Clone with HTTPS

https://gitlab.com/wisconsinracing

After pasting the URL you just copied into GitHub Desktop, use the box marked "Local path" to specify where you are cloning the repo. We recommend making a folder called "git" right next to the folder labeled "svn" on your computer. Within that folder, we recommend making subfolders for each subgroup you are cloning repos from. This is clearer when looking at the organization scheme at the end of this guide. Do not change the name of the subfolder that you are creating by cloning this repo – it may cause problems for both git and Solidworks.




Clone a repository

GitHub.com | GitHub Enterprise | URL

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

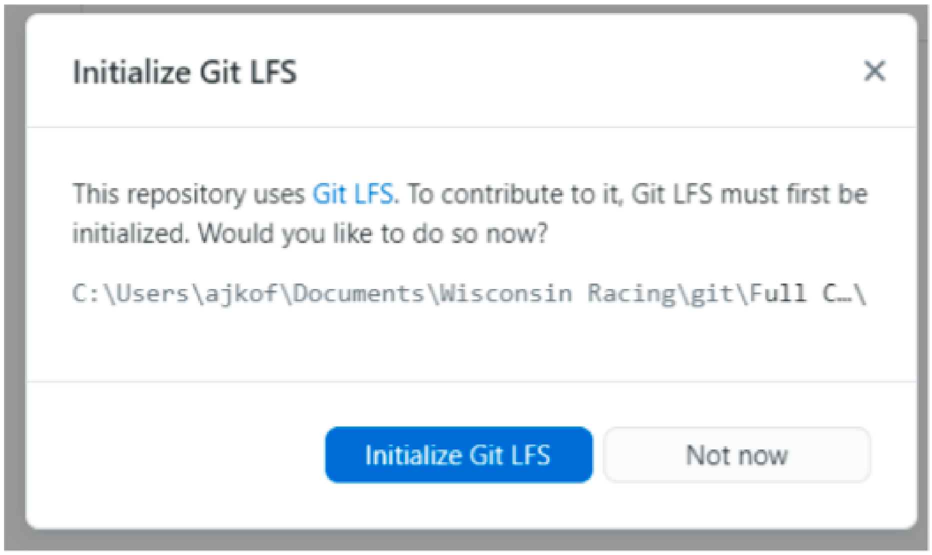
https://gitlab.com/wisconsinracing/design-docs/chassis.git

Local path

C:\Users\ajkof\Documents\Wisconsin Racing\git\Design_Docs\chassis

You will be prompted to enter a username and password. Your username is your GitLab username. Copy the Personal Access Key you created at the end of the last section and enter it as your password. The files should then begin to download.

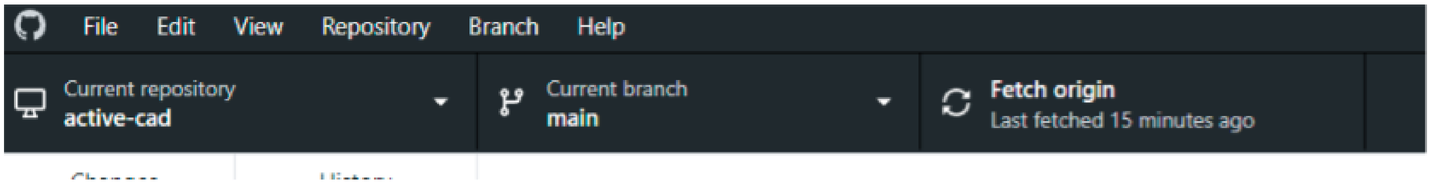
When the download is finished, you will be prompted to initialize Git LFS, a plugin that helps store binary files, including CAD files. Click Initialize Git LFS.


Initialize Git LFS

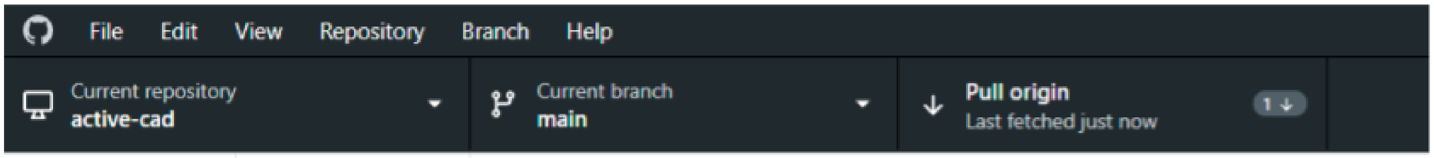
This repository uses Git LFS. To contribute to it, Git LFS must first be initialized. Would you like to do so now?

`C:\Users\ajkof\Documents\Wisconsin Racing\git\Full C…\`

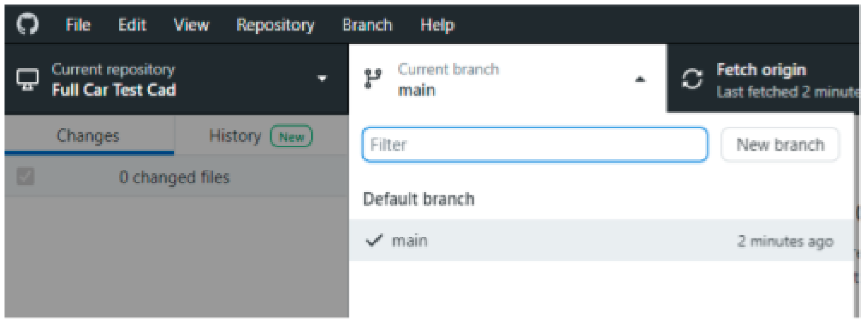
Pulling (Updating) the Working Copy – To receive updates to the repo, press the Fetch button at the top of the GitHub Desktop window.

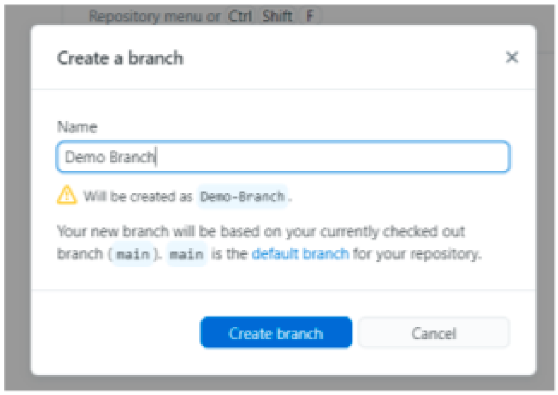


If there are new updates, the Fetch button will switch to a Pull button, which you should then press to pull updates.



Branches – To make edits, you will need to make your own branch. It is generally a bad idea to commit directly to main unless you know what you're doing. To make a branch, click on "Current branch" and hit the "new branch" button. Name the branch something descriptive.

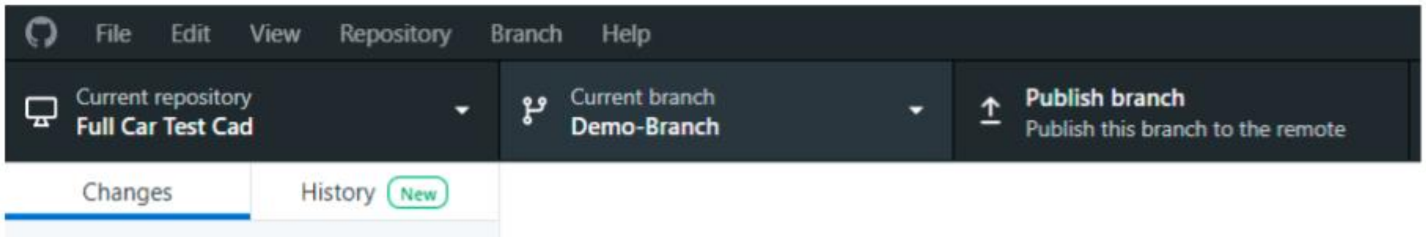




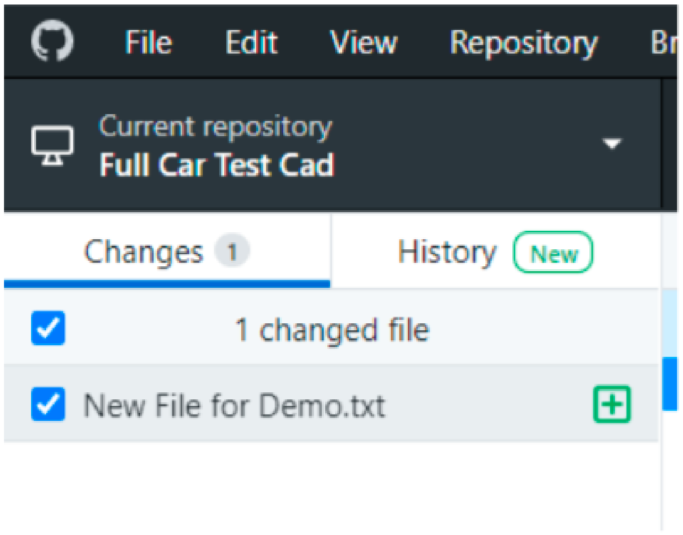
Will be created as Demo-Branch.

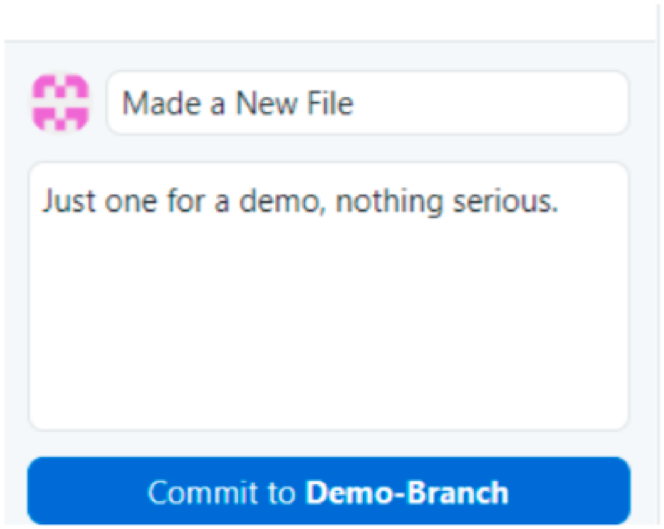
Your new branch will be based on your currently checked out branch (main). main is the default branch for your repository.

Your current branch should now be the branch that you just created. Hit "Publish Branch" to establish it for everyone else.



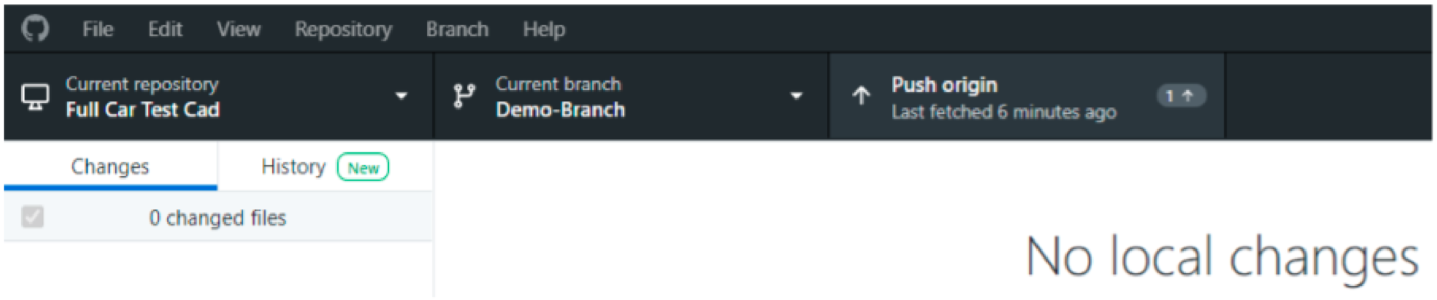
Now make your changes! Then, head back to GitHub Desktop and look at the Changes panel. Check the box for any files you changed. Git will ignore FEA-related files (.MFC, .LOG, etc.), but make sure not to include files you didn't actually work on, even if they are marked as changed because Solidworks forced you to save over them. See the troubleshooting section for how to discard these unwanted changes to files without committing them. Put in a title for your commit and a description if you'd like to be really detailed, then hit the big blue Commit button.





Just one for a demo, nothing serious.

Now the changes are committed, but the rest of the team still doesn't know you made changes. Hit the Push Origin button to push the commits on your branch to the repo.

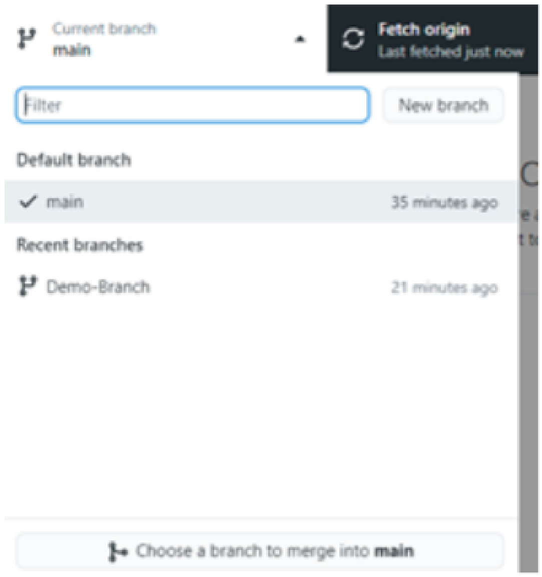


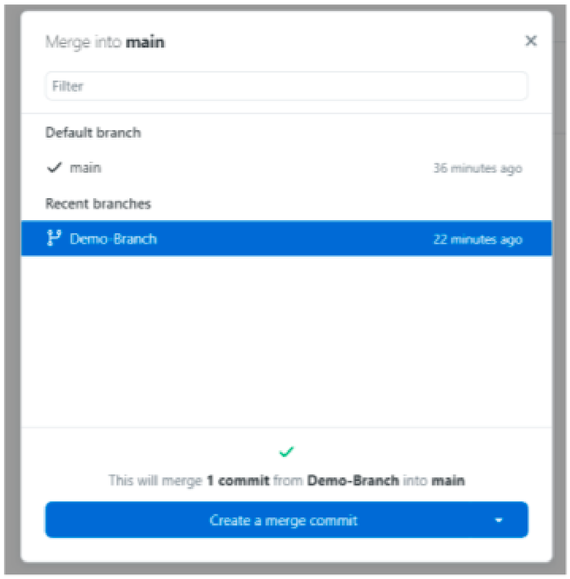
At this point, the website will reflect the changes you have made to your own branch. No changes will be made to the main branch of the repo, which protects it from accidents. You can and should make multiple commits to your own branch between merges to record small steps in your project.

MERGING BRANCHES

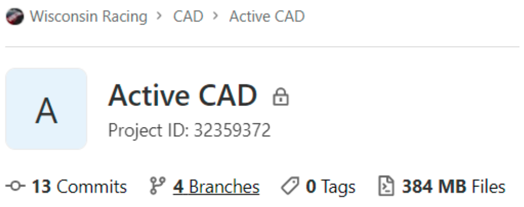
When you're ready to integrate your changes into the main repo, you will need to merge your branch into the main branch. Hit "current branch" and switch back to main.

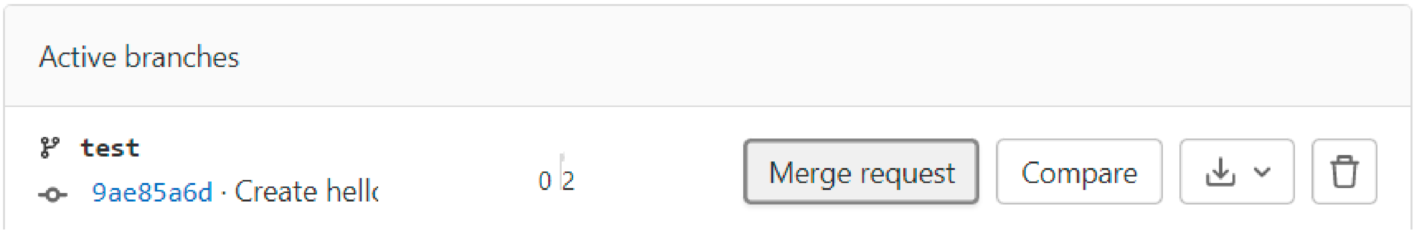
For Maintainers – Hit the button at the bottom of the branch menu to merge your branch into main. Select your branch and hit "Create a merge commit". Your merge commit will need to be approved by a team leader with Maintainer permissions.





For Developers – Navigate to your folder on gitlab.com. Hit branches, and then make a merge request on your branch.





When your pull request is approved, you've done it! Celebrate and close your branch or move on to your next round of edits.

ORGANIZATION OF THE NEW GIT REPOS

We're going to do things a little differently than we did in Subversion. The main change is that due to file storage limits, the mechanical files will be broken into smaller repos. It will be possible and frequent to build assemblies across different project folders, most notably the development area pulling models from the active cad area.

- CAD Repos
    - Active CAD
    - CAD Workshop
    - CAD Archive
- Design Docs
    - Architecture-LapSim
    - Chassis Docs
    - Powertrain Docs
    - Structures Docs
    - Suspension-Vehdyn
- CFD Repos
    - cfd-220
    - cfd-222
- ecu-controls
- 220e

active-cad – The most current car models and drawings are stored here. Any updates to this space should have passed a design review, and new part or drawing versions should update the revision letter in their name.

cad-workshop – For current and past projects. This folder can be a little looser about following the CAD naming rules. Any development parts that have not passed a design review and been finalized should be stored here.

cad-archive – A snapshot of the active-cad directory that is representative of the most recent car built will be saved here. Older versions are on the office hard drives for copy.

cfd –Full car sims are available on the office hard drives only.

design-docs – For storing any Matlab, Excel, or EES files that are worked on alongside a project. In general, presentations (including design slides) should be stored in Drive.

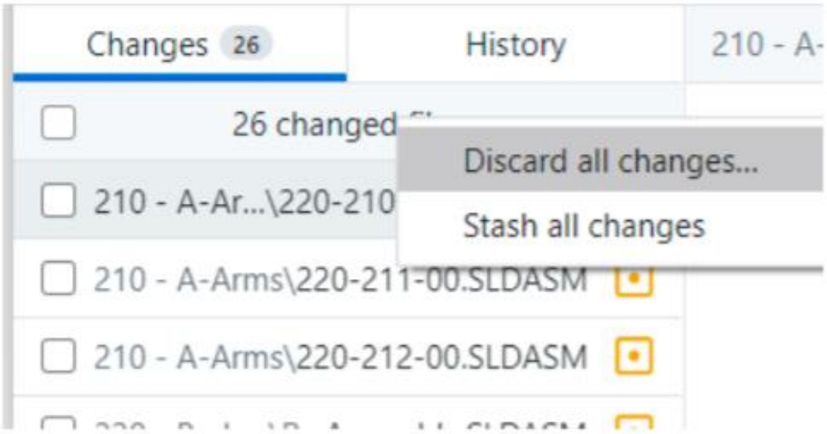
220e – For the 222 season only, the 220e folder will be kept as it was in svn.

Office Hard Drives – A pair of drives in the office have older files on them for reference and copying as needed. These files go back as far as 2006 and include vehicle CAD and team files. Make sure not to change the archival data stored on the drives!

TROUBLESHOOTING

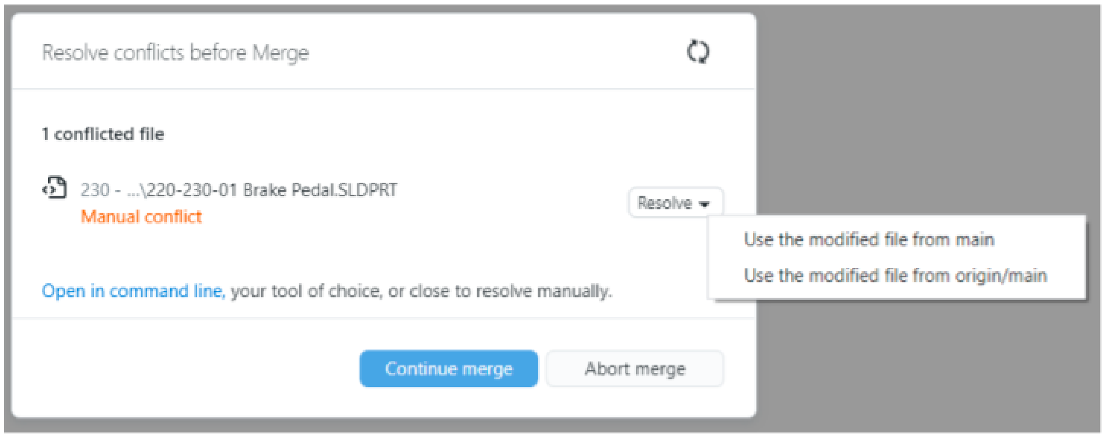
Sometimes things aren't going to go as planned. Here are some ways to resolve common errors.

Unwanted Files in Changed Files List – If you edit an assembly, you will be prompted to save all the dependent files. However, you should only commit the things you're working on. After you commit your real changes, right click at the top of the changelist and press "Discard all changes". This will revert the remaining files back to their original state.



Newer Commit on Remote – If there have been updates on your branch since you started working, you may be prompted to fetch from the remote repo. Just hit the Fetch button and then pull changes from the repo to clear the issue. If there is a conflict, see the next section.

Resolving Conflicts – As in svn, if someone edits the same file you worked on, only one version can be used. You will be prompted to choose a version from the dropdown. "from main" indicates the version on your computer, and "from origin/main" indicates the version being pulled from the repo. If you don't want to deal with the conflict, you can also abort the merge and make a new branch, and then figure out what to do from there.


Resolve conflicts before Merge

1 conflicted file

230 - ...\220-230-01 Brake Pedal.SLDPRT
Manual conflict

Open in command line, your tool of choice, or close to resolve manually.

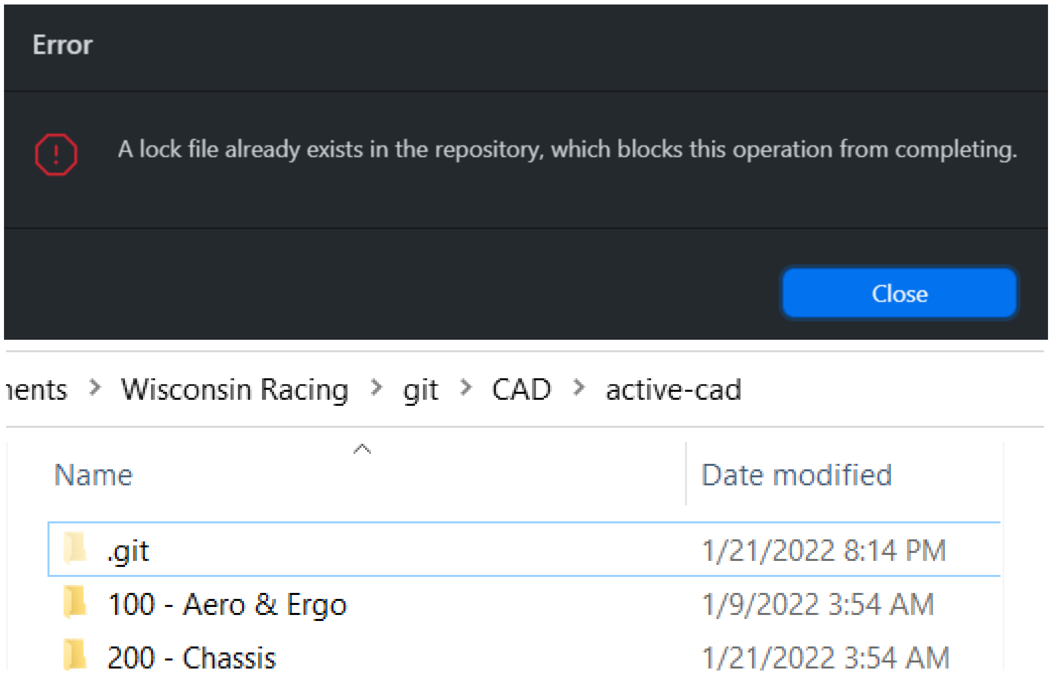
Resolve
Use the modified file from main
Use the modified file from origin/main
Continue merge
Abort merge

If GitHub Desktop seems to be hanging during this process, follow the next instructions.

GitHub Desktop Stuck While Resolving Conflicts – If the spinning arrows at the top right spin and spin, you should perform a reset of the app. Toggle the developer tools by pressing Ctrl + Shift + I (⌥ + ⌘ + I on Mac). Then, click inside the window that pops up on the right and press Ctrl + R (⌘ + R on Mac) to refresh. You may need to click back through the conflict choices but it should resolve itself promptly this time.

Can't find "[name of repo]" – This means you've probably moved the folder or disconnected the drive where the repo used to be. Use the Locate button to fix the problem.

Error Message About Lock Files Preventing Actions – If you see this message, you may need to delete an unexpected lock file. Navigate to the hidden .git folder at the top level of the repo you're trying to push to or pull from and delete the INDEX.LOCK file. You may need to enable showing hidden folders to find the .git folder.


Error

A lock file already exists in the repository, which blocks this operation from completing.

Name	Date modified
.git	1/21/2022 8:14 PM
100 - Aero & Ergo	1/9/2022 3:54 AM
200 - Chassis	1/21/2022 3:54 AM