GitHub Developer Guide

1. Purpose 5
2. GitHub Overview 5
3. Submit BEARS Requests 6
   3.1. Request GitHub 6
4. After BEARS Approval 6
   4.1. Confirm GitHub Access 6
   4.2. Request Access to Git Organization 7
   4.3. Add the GitHub user to a team 7
   4.4. Validate Your Access to your repository on GitHub 8
   4.5. Install Git 8
   4.6. Install TortoiseGit 8
   4.7. Setup Local Home Drive to “C:\Users\SEID”: 8
5. Setup Public Key and Private Key 9
   5.1. Generate Public Key 9
   5.2. Add Public Key to GitHub 9
   5.3. Configuring Tortoise to use Key 11
6. ITPE Repo and branches 14
7. Eclipse setup 14
   7.1. Setup SSH Keys 14
   7.2. Configure Git 15
   7.3. Clone repository 21
   7.4. Importing Project using Maven 25
   7.5. Updating Project 26
   7.6. Importing Project using Git 27
8. Gitflow 30
9. Creating Local Team Branch 31
   9.1. Adding Pre-Existing Remote Branches 31
   9.2. Creating Local Team Branch 34
   9.3. Creating Remote Tracking Branch 36
10. Creating Freature Branches 39

11. Pulling Changes and keeping branches up to date 42
11.1. Pulling Changes 42
12. Committing Changes 47
12.1. Steps for Committing Changes 47
12.1.1. Updating Local Team Branch Before Checking in Changes 47
12.1.2. Checking In Code Changes 49
12.2. Troubleshooting Commit Steps 50
12.3. Create a Pull Request 52
12.4. Updating After a Pull Request is Approved 53
13. Deleting a Branch 53
13.1. Troubleshooting Steps 57
14. Tortoise Basics 60
14.1. Overview 60
14.2. Tortoise Folder Icons 60
15. Tortoise Show Log 60
15.1. Overview 60
15.2. Steps to Show Log 61
16. Resolving Conflicts with Tortoise 63
16.1. Steps for Resolving Conflicts 63
16.2. Overview of Conflict Editor 67
16.3. Options Available for Resolving Conflicts 67
16.3.1. Overview 67
16.3.2. Selecting “Use this text block” from left or right frame 68
Choosing Left Line Over Right Line 68
Choosing Right Line Over Left Line 69
16.3.3. Selecting “Use this whole file” from left or right frame 69
Choosing Left File Over Right File 70
Choosing Right File over Left Life 71
16.3.4. Selecting Use One Block Before the Other 71
Selecting Right Line Before Left 72
Selecting Left Line Before Right 73
Selecting an Entire Block of Lines To Use One Before the Other 73

16.3.5. Using Preview Frame 74
16.3.6. Go to Next Conflict 76
16.3.7. Saving File and Resolving Conflict Status on File 77
17. Aborting a Merge with Tortoise 79
18. Special Instructions for the ALC Baselines Repo 80
19. Running a Personal Build 81
20. ITPE Repo Links Team Branches and other Resources 83

GitHub Developer Guide

This Document was created based on this EWM to GitHub Transaction document

EWM to GitHub Transition.docx (sharepoint.com)

Document Change Control

The table below includes the revision number, the date of update/issue, the author responsible for the changes, and a brief description of the context and/or scope of the changes in that revision.

SharePoint Revision Number	Date	Author(s)	Summary of Change(s)
1.0	8/28/2024	Zain Kazi	Initial
1.1	4/21/2025	Dan Tran	Updated for the RAPID team
1.2	5/8/2025	Balki Dharmalingam	Updated for ITPE
3.0	6/2/2025	Jason Gardner	Combined this guide with another guide Creating Branches and Committing Changes
4.0	6/6/2025	Jason Gardner	Continuing to make updates based of scenarios that come up and standards we apply
4.1	6/11/2025	Jason Gardner	Update comitmsg Template Added additional instruction to section 7.2 for GIT Configuration added additional information to BEARS request for Contractor
5.0	6/13/2025	John Allen	Added sections for Tortoise. Rearranged some of the sections and updated numbering

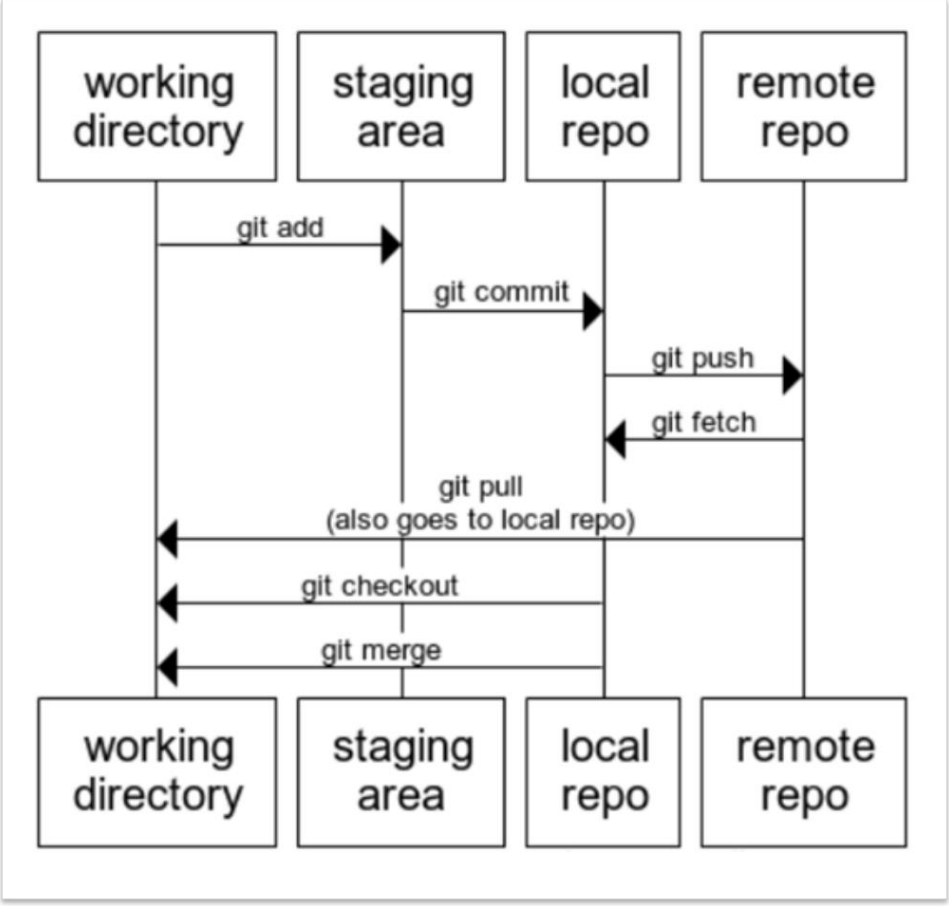
GitHub Developer Guide

1. Purpose

This document provides guidelines and instructions for using GitHub for Source Code Management of ITPE.

2. GitHub Overview

GitHub is a web-based platform for version control and collaboration. GitHub is based on Git. These are the typical git commands performed on GitHub.



Note: Working directory, staging area, and local repo reside on your local computer. Remote repo lives on remote GitHub server

3. Submit BEARS Requests

3.1. Request GitHub

- Access BEARS
- Select Manage My Access
- Enter "PROD USER GITHUB ENTERPRISE (GITHUB ENTERPRISE GHE)"
- Click the circle to the left of the selection
- Complete request process
    - Add a comment like: "**Access is needed to fulfill duties as part of the XXX Development Team**"
- Wait for approval

4. After BEARS Approval

4.1. Confirm GitHub Access

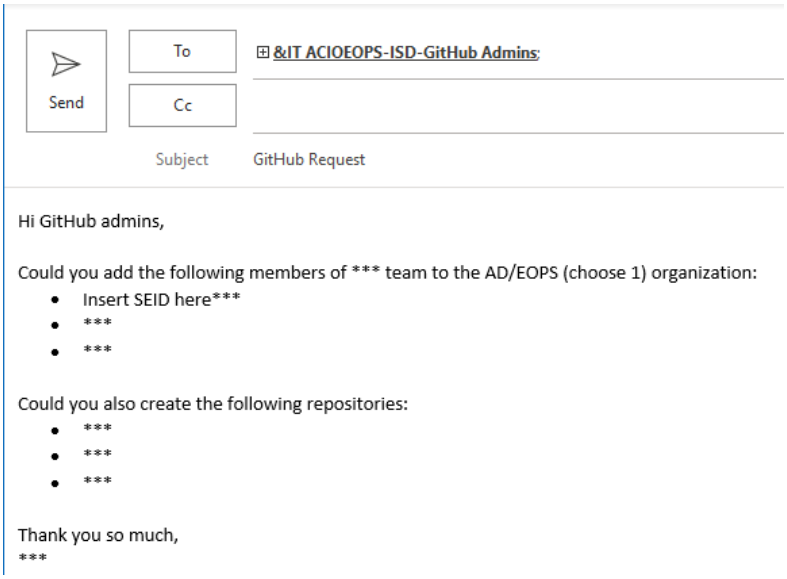
Once your request has been approved by GitHub Administrator, you will receive an email from BEARS. Log back in to BEARS and accept to acknowledge the approval. You should now have access to IRS GitHub Enterprise: cl

You will be prompted to Login - **Use Sign in with SAML option**

4.2. Request Access to Git Organization

Send email to GitHub Administrator **(&IT ACIOEOPS-ISD-GitHub Admins**) and request to be added to the appropriate Organization: **AD** (clicking on the mail will populate the message)

GitHub Access Request.msg



You **need not create repositories**. So, you can **remove** that part of the email.

If your BEARS request is delayed or NOT Approved, please email **&IT ACIOEOPS-ISD-GitHub Admins** for assistance.

4.3. Add the GitHub user to a team

After getting access to GitHub, and added to the AD organization, the user needs to be assigned to proper team. We have following teams defined for ITPE:

Name	Access
ITPE Dev	Read, Clone, Pull from team stream
ITPE Dev Lead	Read, Clone, Push, manage Pull
ITPE Admin	Read, Clone, Push, manage Pull, Add collaborators
ITPE DM	Read, Clone
ITPE CM	Read, Clone, Push, manage Pull

Inform your IRS manager to email Project Managers at &IT ITPE PM Support. Your manager should include your assigned role and team name in the email. Project Managers at &IT ITPE PM Support will add you to the appropriate teams.

4.4. Validate Your Access to your repository on GitHub

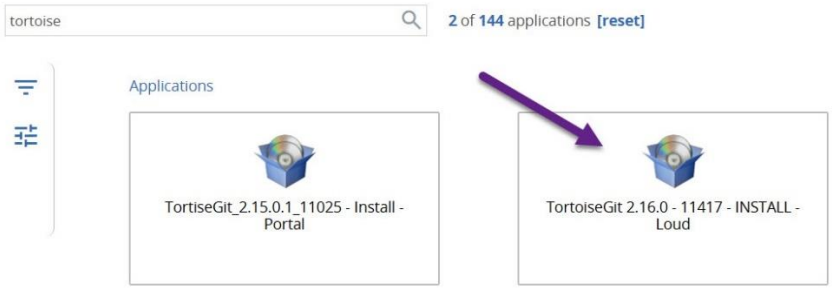
Make sure you can access this link -> https://github.enterprise.irs.gov/YourSEID

4.5. Install Git

- Access Symantec Software Portal (irsnet.gov)
- At top of screen, in Show Filter field, type in "Git" and hit enter
- Select latest Git version to install (Git v2.45.2 as of August 2024)

4.6. Install TortoiseGit

- Access Symantec Software Portal (irsnet.gov)
- At top of screen, in Show Filter field, type in "tortoise" and hit enter
- Select latest TortoiseGit version to install (TortoiseGit 2.16.0 as of June 2025)
- After Tortoise has finished installing



4.7. Setup Local Home Drive to "C:\Users\SEID":

For best performance update your configuration so that Git uses your local home directory on the C: drive rather than the I: drive network share.

1. From the **Start Menu** open **Edit environment variables for your account**

2. Press the **New…** button

3. Set the **Variable name** to **HOME** (or it may be **HOMEDIR** in either case it would be set to the same home directory) and set **Variable value** to your local home directory. For example, *C:\Users\SEID*

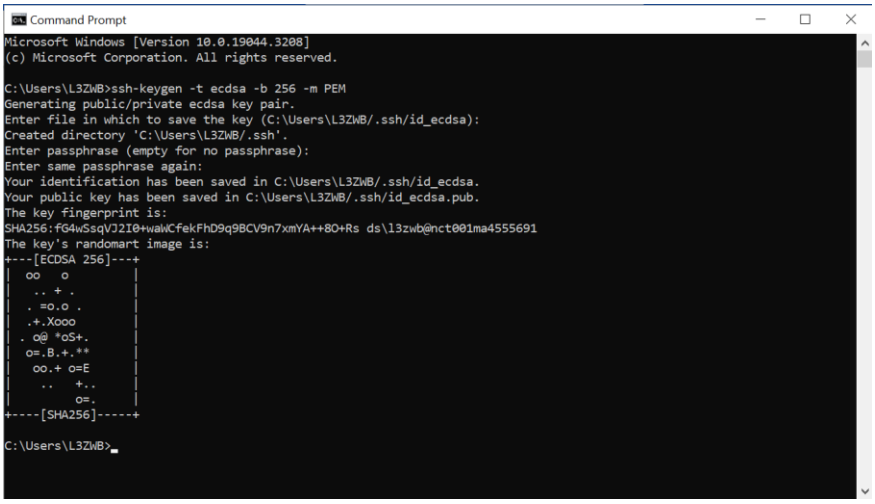
4. Press the **OK** button to create the environment variable

5. Press the **OK** button on the make window

5. Setup Public Key and Private Key

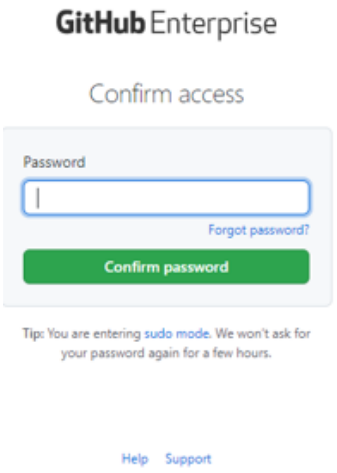
5.1. Generate Public Key

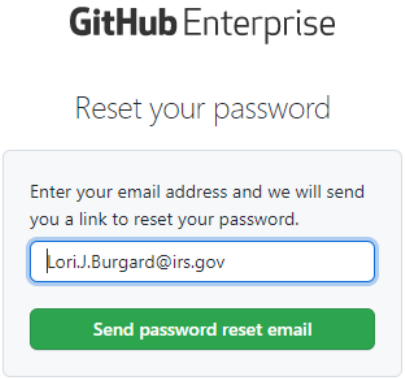
1. Open a **cmd** window and type 'ssh-keygen -t ecdsa -b 256 -m PEM'
2. For the responses
    a. Press Enter for "Enter file in which to save the key 'C:\Users\[YOUR SEID]/.ssh/id_ecdsa'
    b. Press Enter for "Enter passphrase (empty for no passphrase):", though optionally you can add passphrase which will be used during authentication
    c. Press Enter for "Enter same passphrase again:", though optionally you can confirm a passphrase



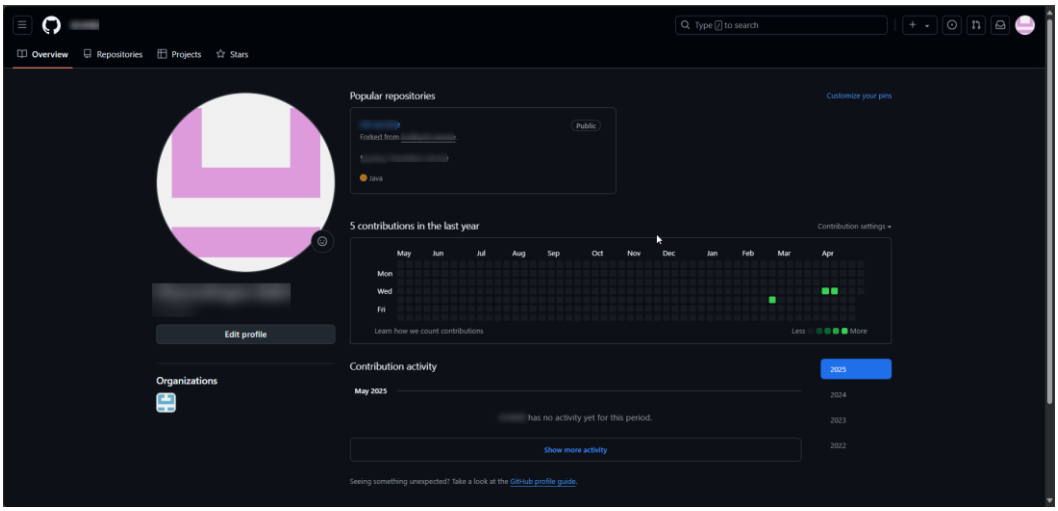
5.2. Add Public Key to GitHub

1. Log into your IRS enterprise GitHub account. If it is your first-time login, you may need to reset your password.

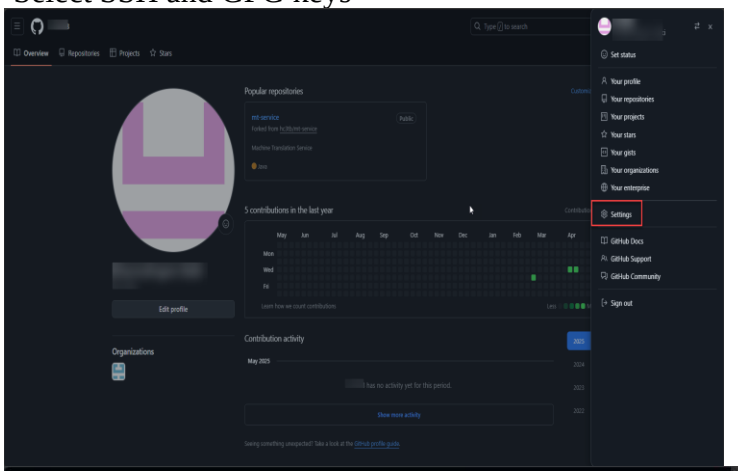


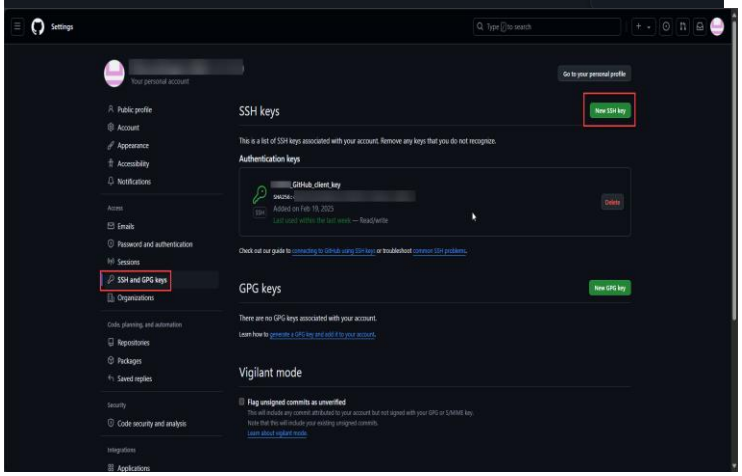


2. If you successfully login, you will see as follows: https://github.enterprise.irs.gov/YourSEID

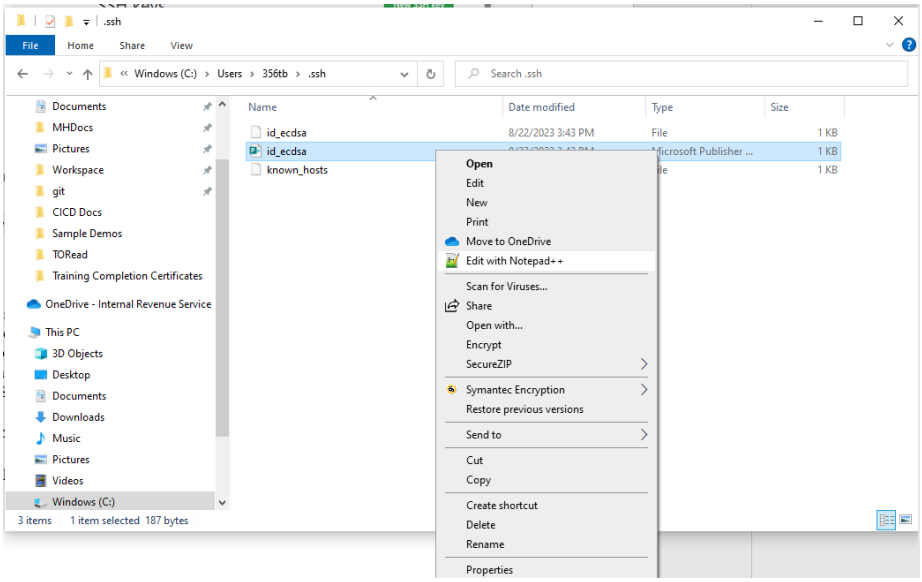


3. Click on Profile circle in upper right corner
4. Select Settings
5. Select SSH and GPG keys

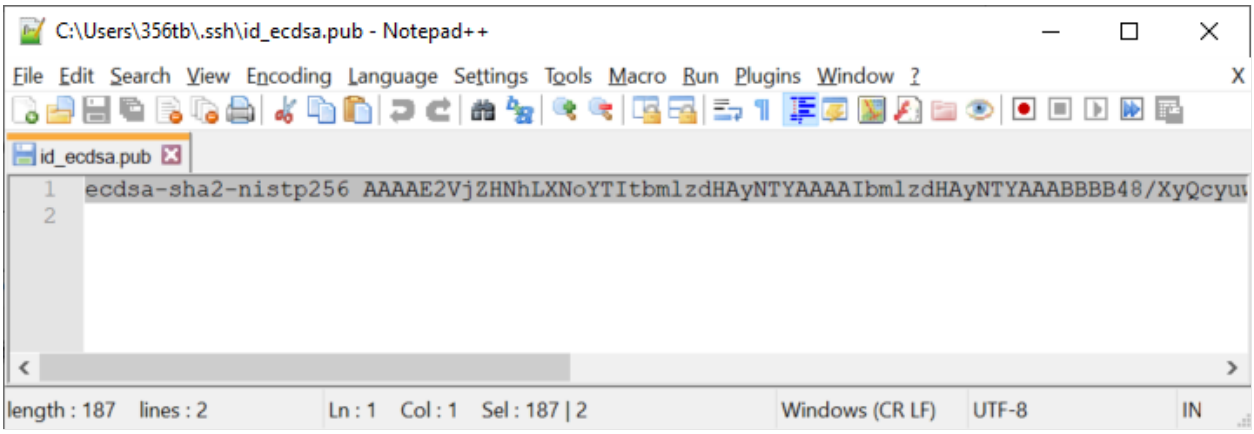




6. Remove any existing Keys that you do not need.
7. Click New SSH Key
8. Set Title to something like: [SEID]_GitHub_client_key
9. Open the C:\Users\SEID\.ssh\id_ecdsa.pub file with Notepad or any text editor.



10. Copy and Paste the entire contents of the id_ecdsa.pub file into the key field, except the extra line. Make sure there isn't an extra line at the end of the field.



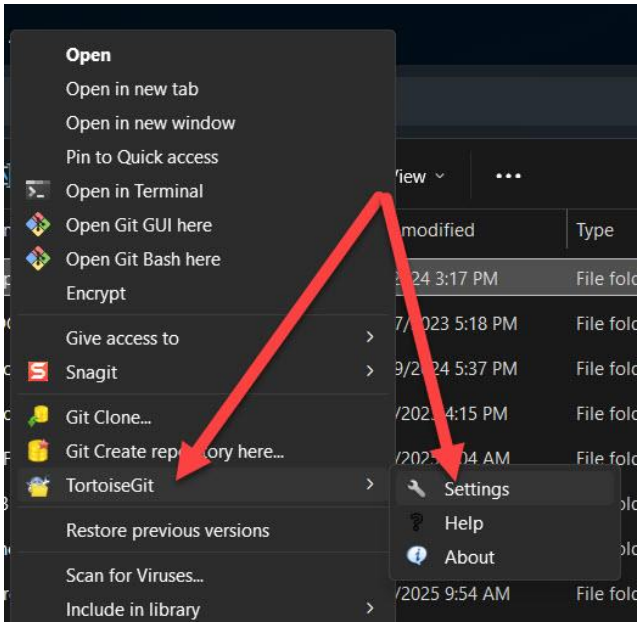
11. Click Add SSH key
12. After "Add SSH Key", GitHub "confirm password" screen may appear.
    a. If you don't know your password, click Forgot password.
    b. Box will pop up prepopulated with IRS email address.
    c. Click "Send Password Reset email"

5.3. Configuring Tortoise to use Key

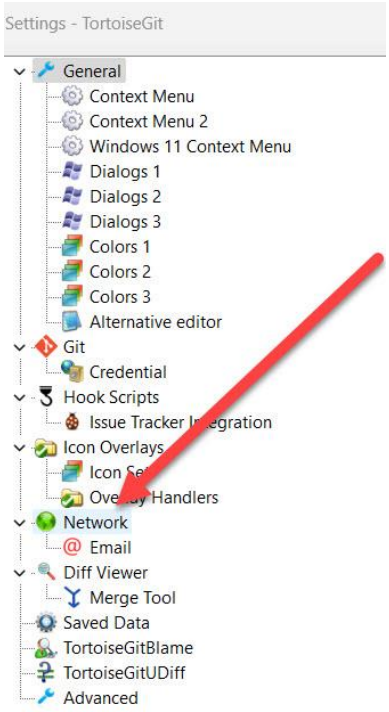
By default, tortoise comes packaged with an SSH program that isn't set up to find the SSH key created in the previous steps. By changing the SSH program Tortoise uses to match the SSH

program utilized by Git, it will automatically look in the location the key was saved to in the Generate Public Key step.

1. Open Windows Explorer and navigate to any directory that contains another directory or file Right click on any file or folder. If using Windows 11, click on "Show More Options.". Select the "Tortoise Git" submenu and click on "Settings"

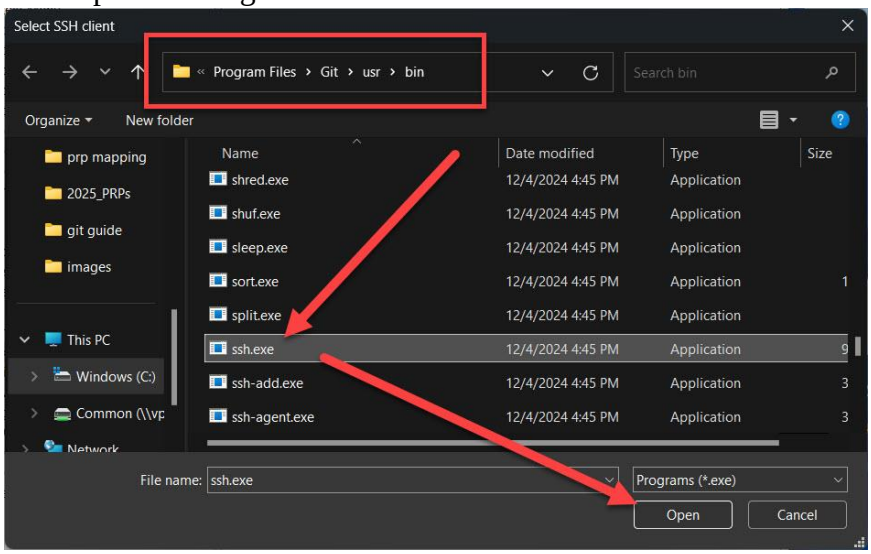


2. In the Settings dialogue that pops up, select the "Network" settings in the left frame.

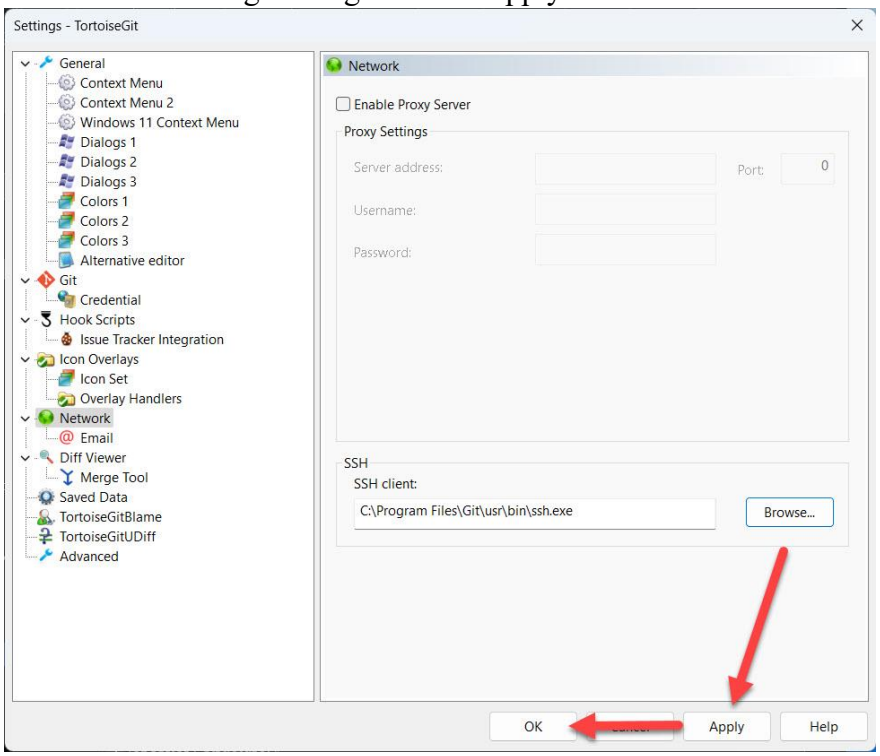


3. Under the SSH heading, click on “Browse” to change the SSH client.

4. Navigate to the path C:\Program Files\Git\usr\bin. Select “ssh.exe” and click “Open”

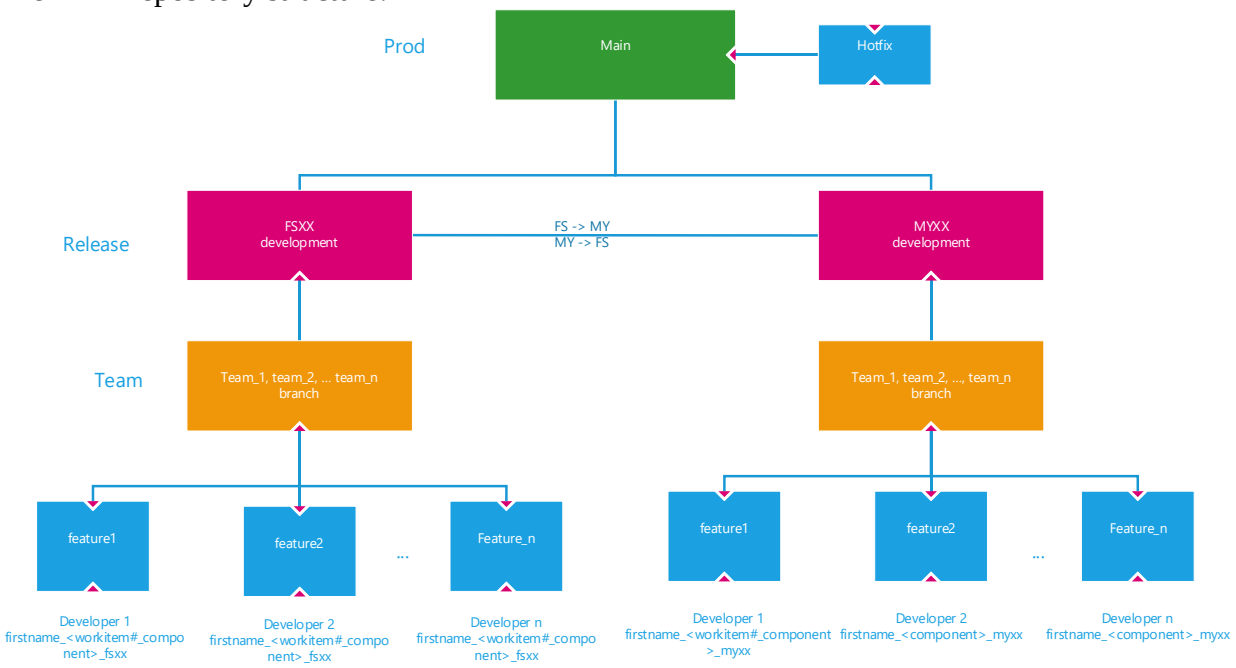


5. Back on the Tortoise Settings dialogue click “Apply” and then “Ok”



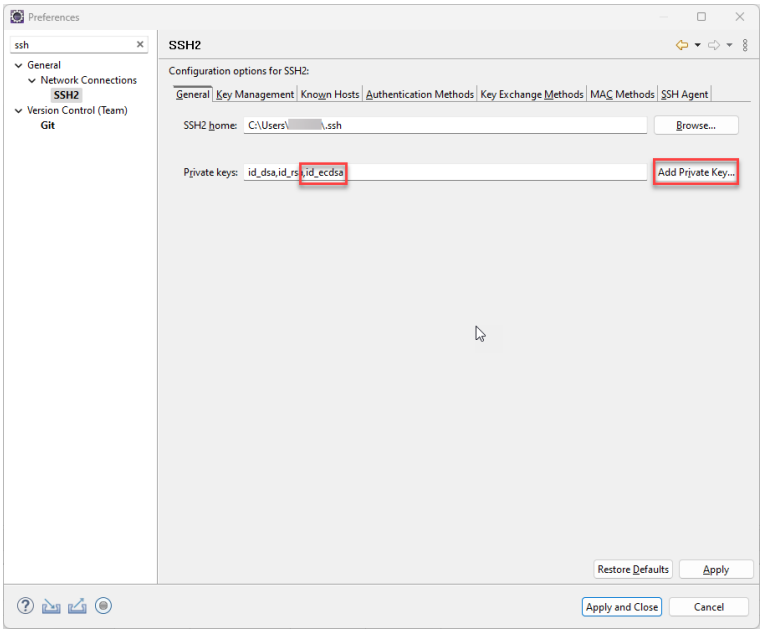
6. ITPE Repo and branches

The ITPE repository structure:



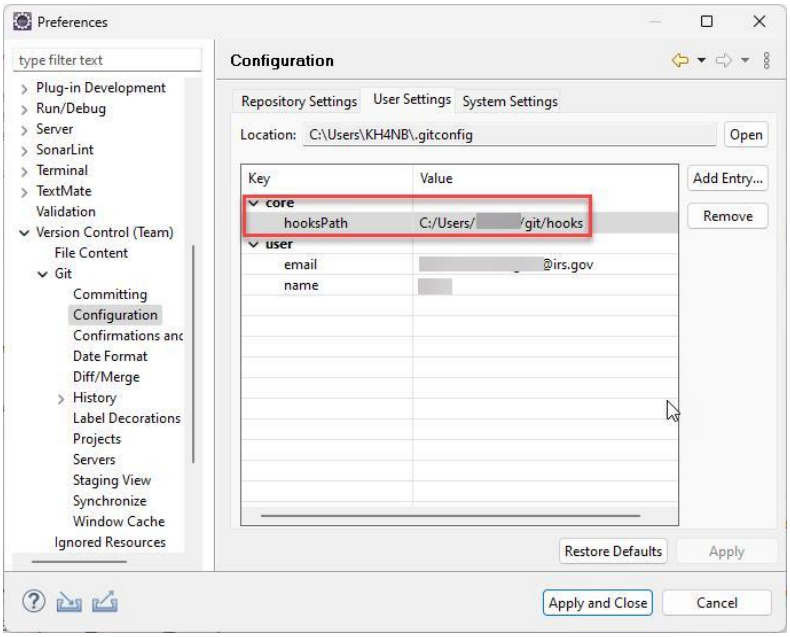
7. Eclipse setup

7.1. Setup SSH Keys

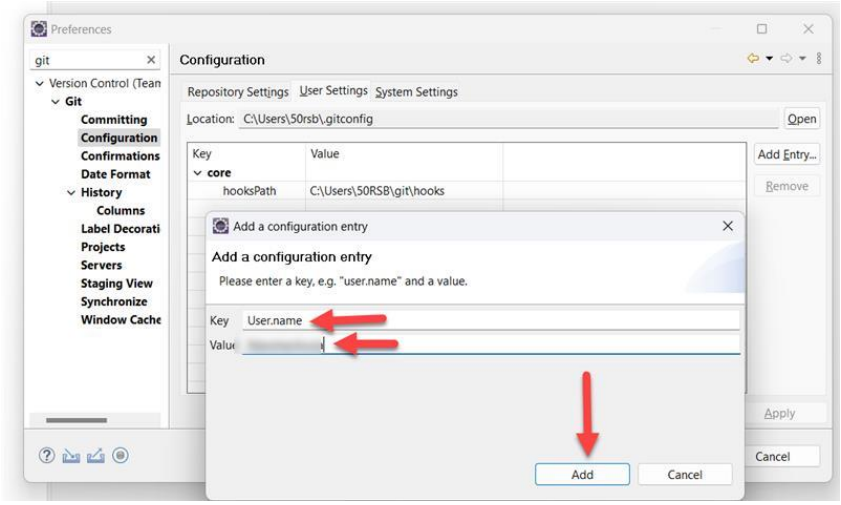


7.2. Configure Git

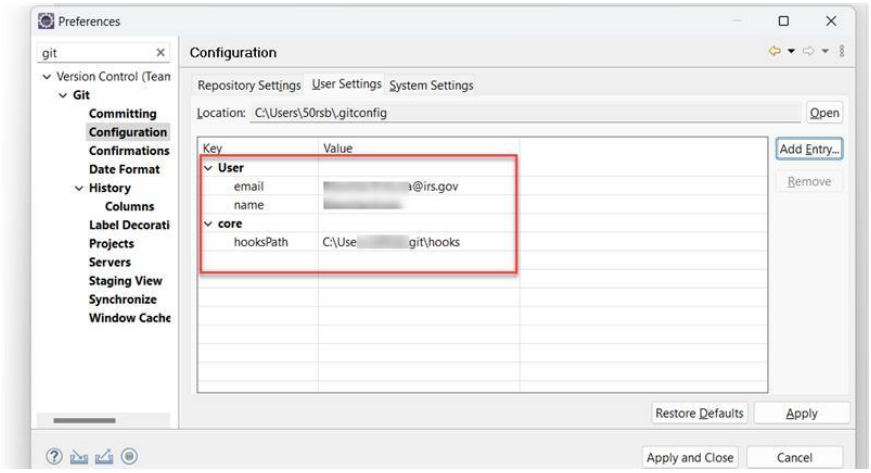
1. Select “Window -> Preferences ->Version Control> Git -> Configuration” and Add Entry add key “core.hooksPath” and value “C:\Users\<SEID>\git\hooks”.



2. In the same Configuration Window Add Entry add key “user.name”and value “<Name/SEID>”. Add another Entry with key “user.email” and value “<Your Email>”



3. Your Configuration should look like this when completed

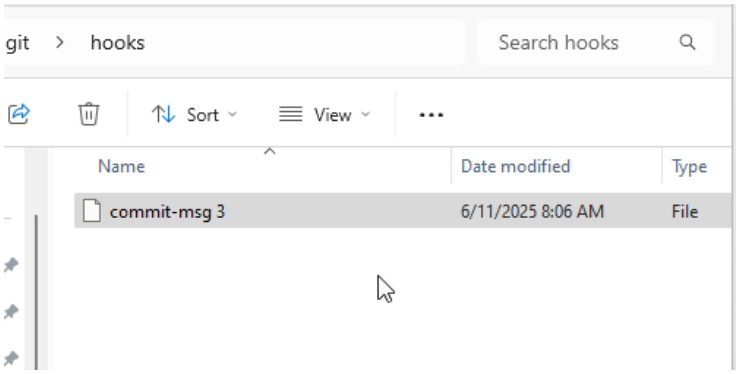


4. Close Eclipse.
5. Open eclipse.ini in the folder C:\Eclipse_IDE_2022.12 (or wherever Eclipse is installed) and add the following line:

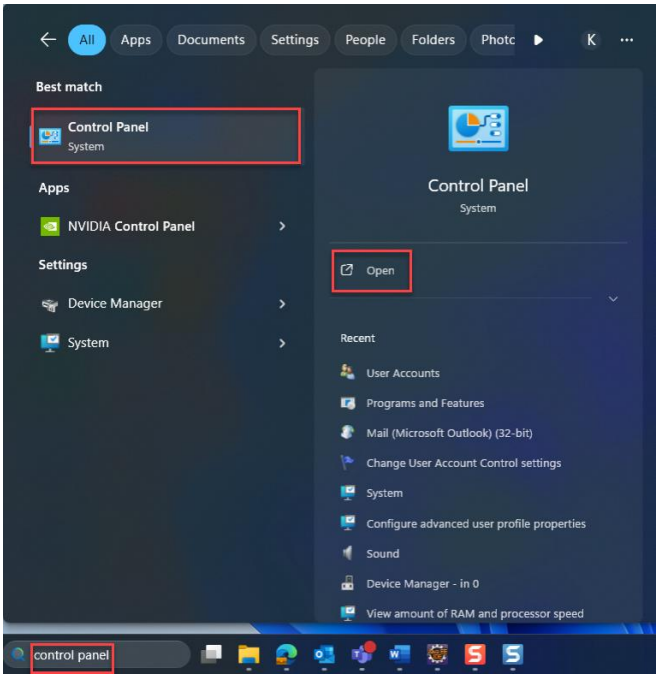
```
--add-modules=ALL-SYSTEM
-Djava.library.path=C:/Program Files/Git/usr/bin
```

6. Create 'hooks' directory under C:\Users\<SEID>\git and copy the attached commit-msg file to that folder

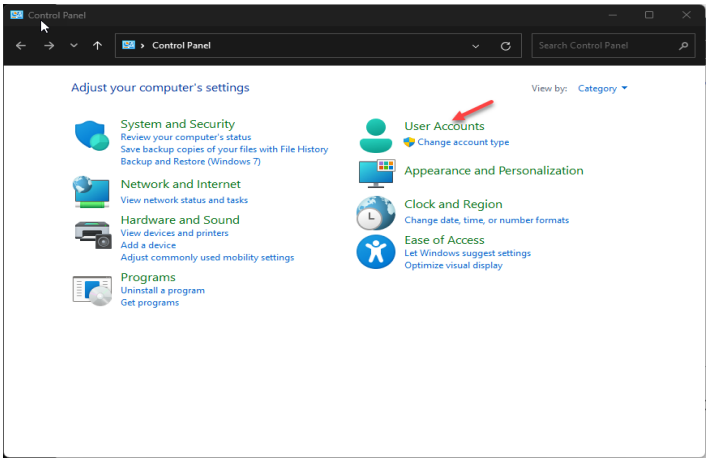
commit-msg 3



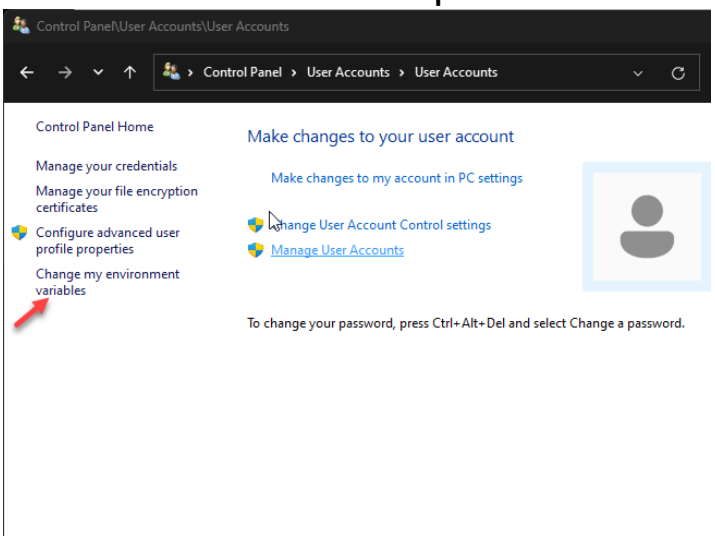
7. In Windows Search, type “Control Panel” and click “Open”.



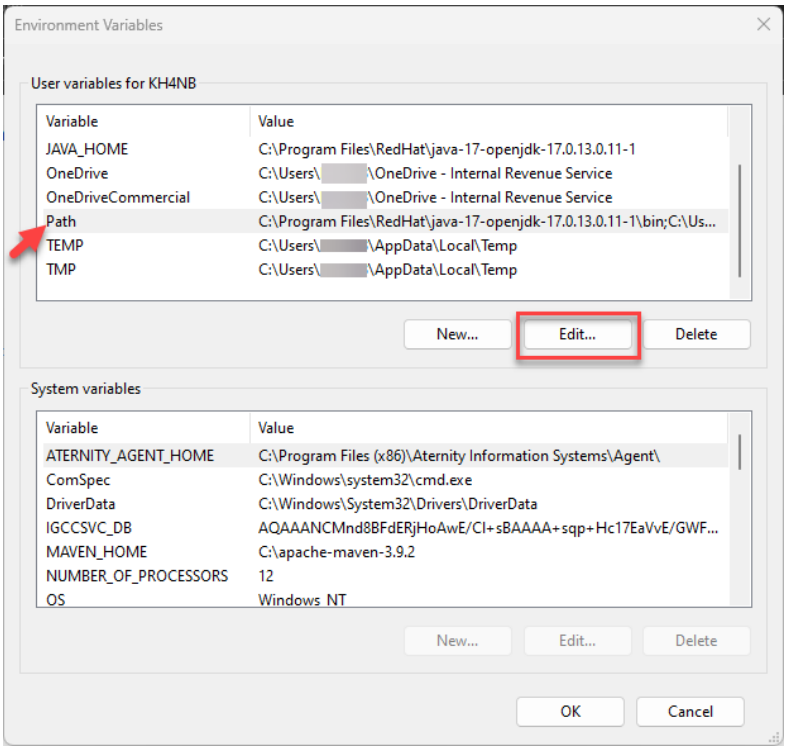
8. Click on “User Accounts”.



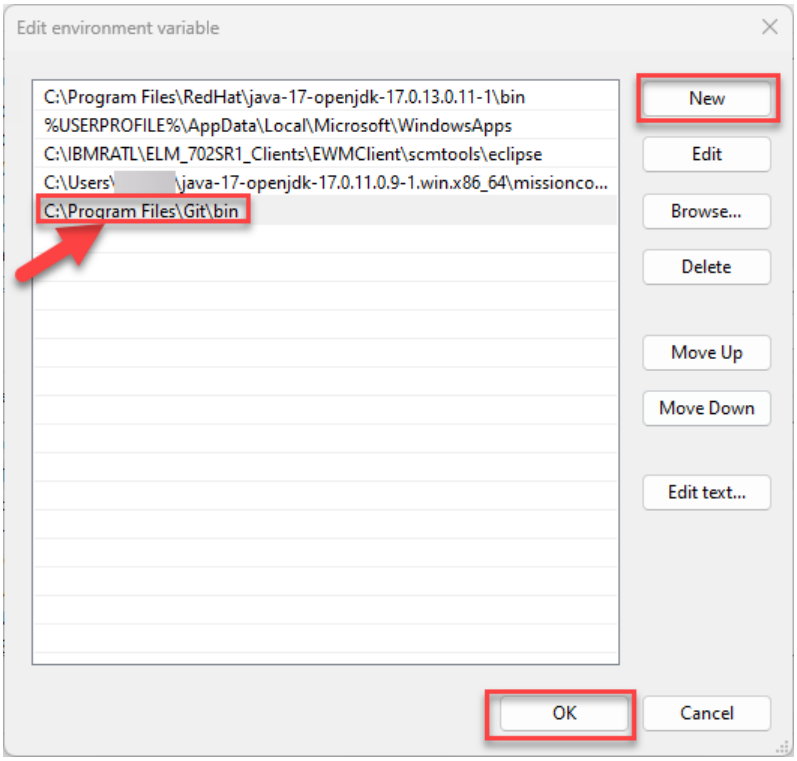
9. Click on “Change my environment variables”



10. Click on Edit

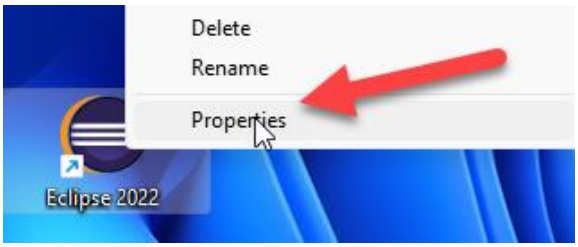


11. Click on New and add “C:\Program Files\Git\bin” to the path and click “OK”.



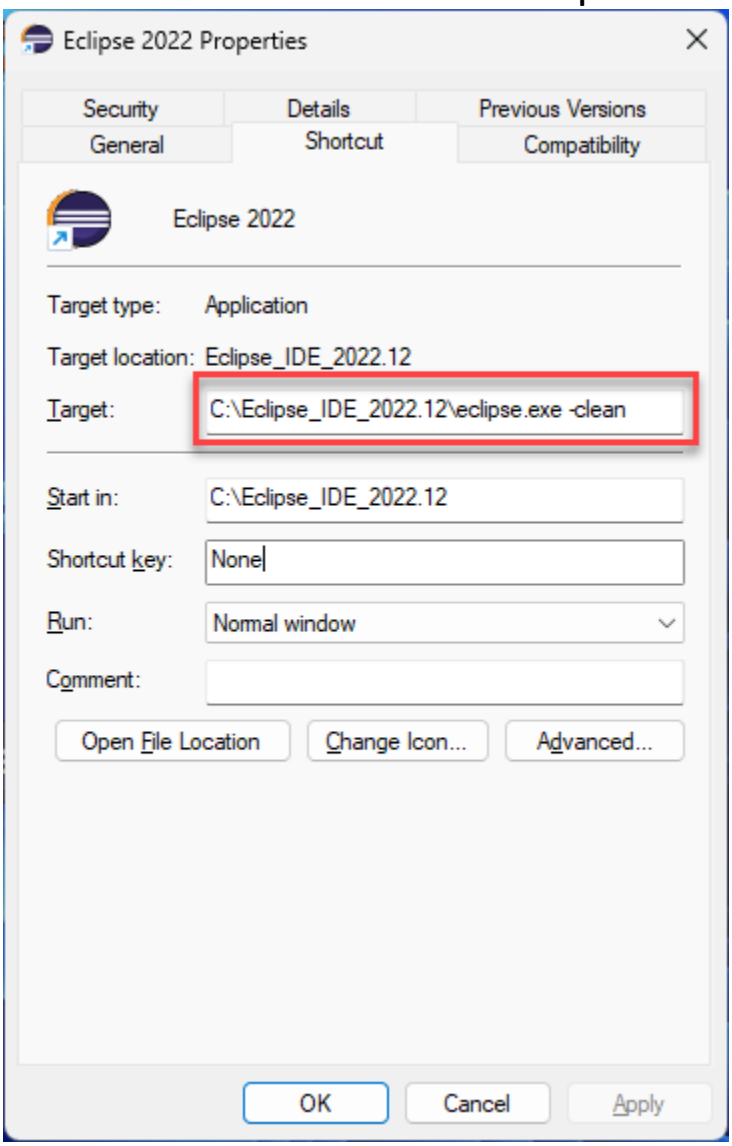
12. Launch Eclipse with “-clean” option, to do this you need to right click on shortcut icon and select properties.

Note: You may have to select show more options when you right click on icon



13. In the properties window append “-clean” to the target path.

Note: there is a space between eclipse.exe and -clean

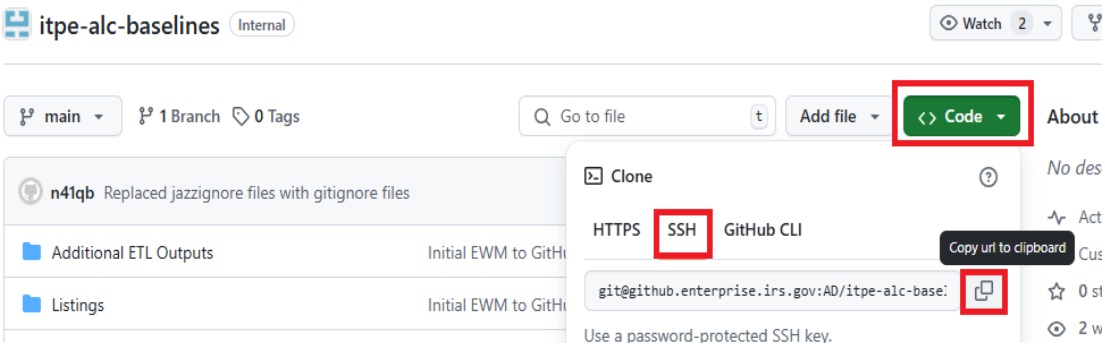


14. After you have started Eclipse one time with this option, go back into properties and remove the "-clean" from the Target so, Eclipse will start quickly next time.

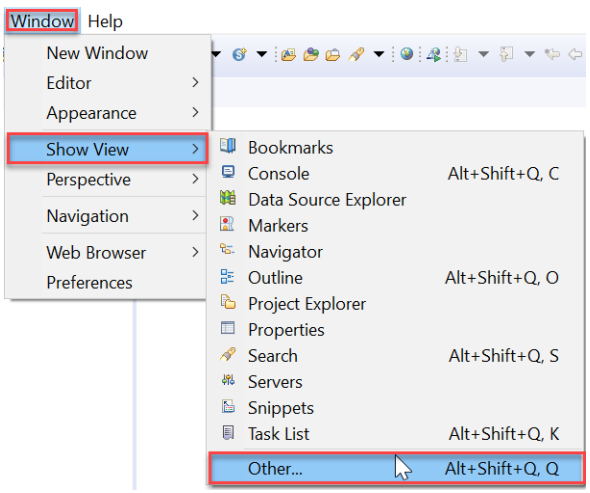
7.3. Clone repository

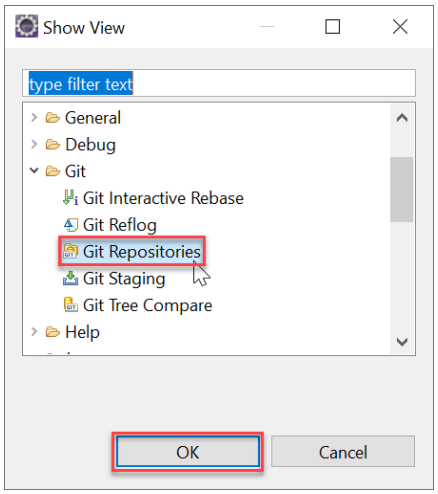
1. Copy the Git repo URL. The repo URLs for ITPE repositories are in Section 16.

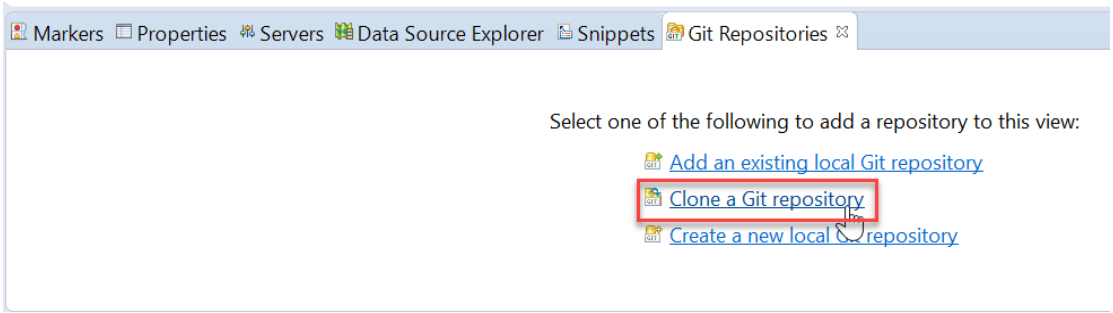
a. If on the GitHub Repository on the web you can find the URL by clicking Code>SSH>Copy to Clipboard as depicted below:



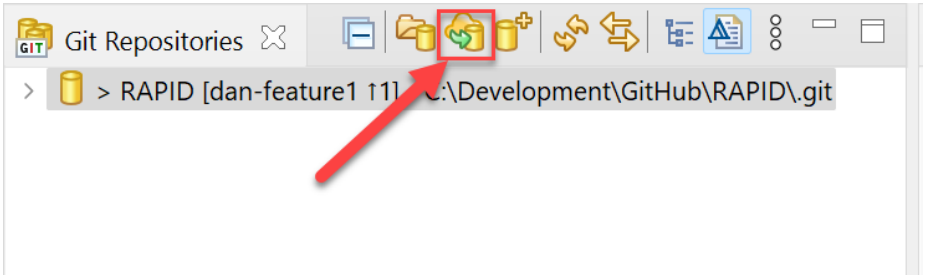
2. Select Window> Show View > Other; Select Git Repositories; then click Clone a Git repository



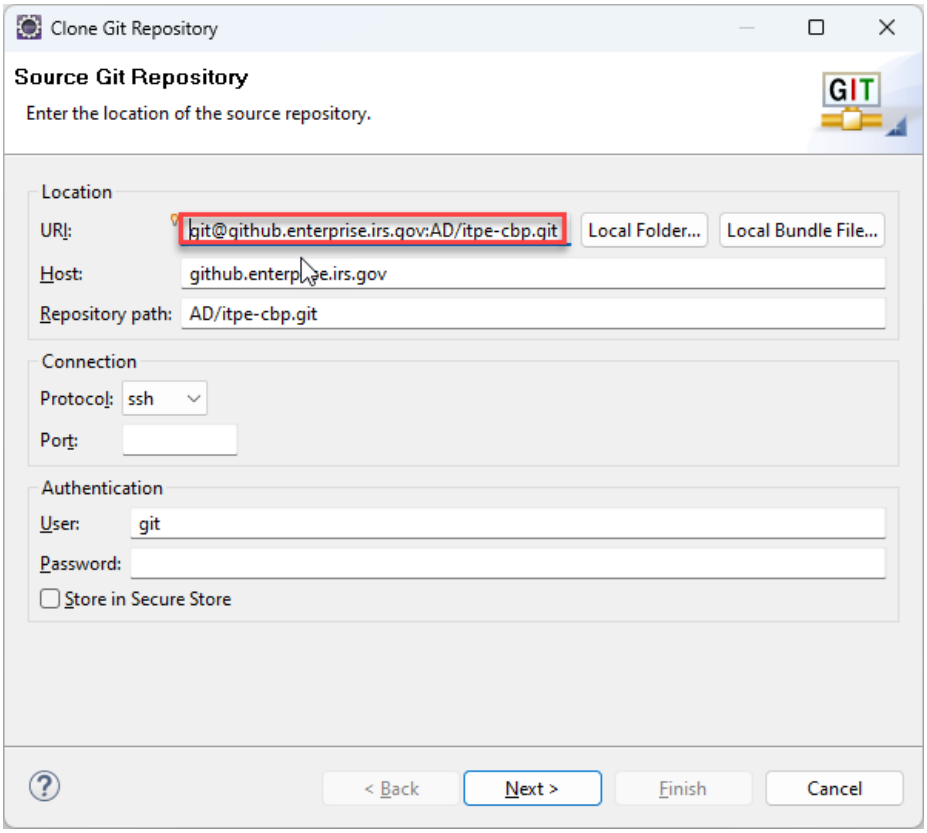




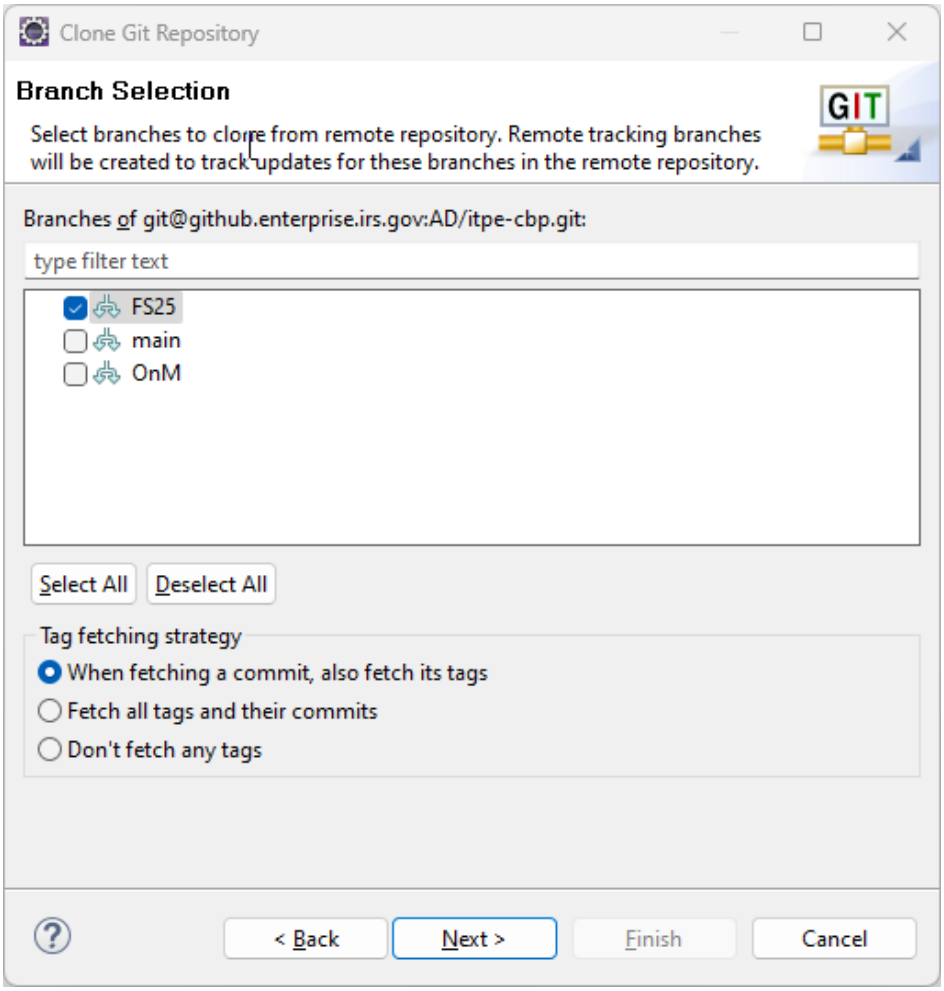
If you don't see the "Clone a Git repository" link, you can click on the icon below to clone a Git repository



3. Paste the SSH URL and keep all other values as is. Click "Next"



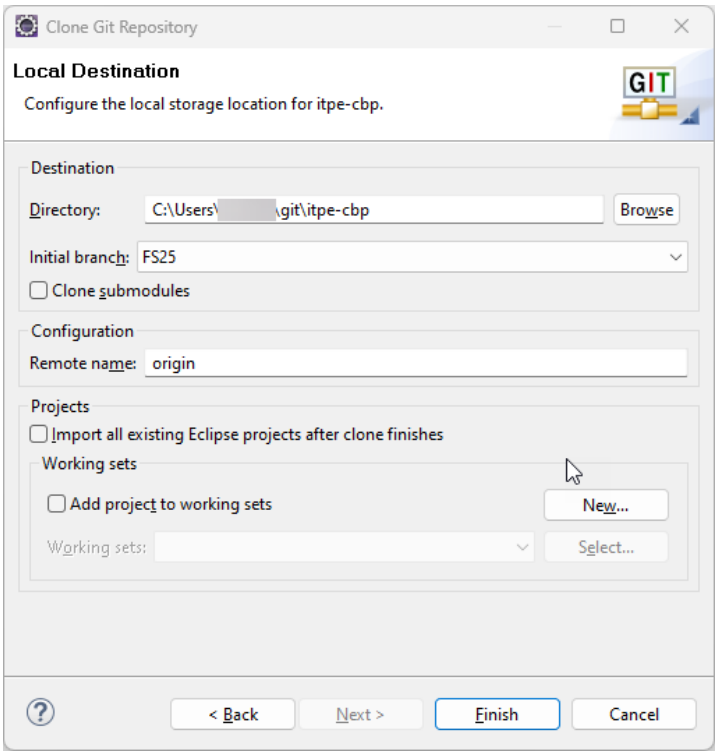
4. Select the branch you want to clone, click Next



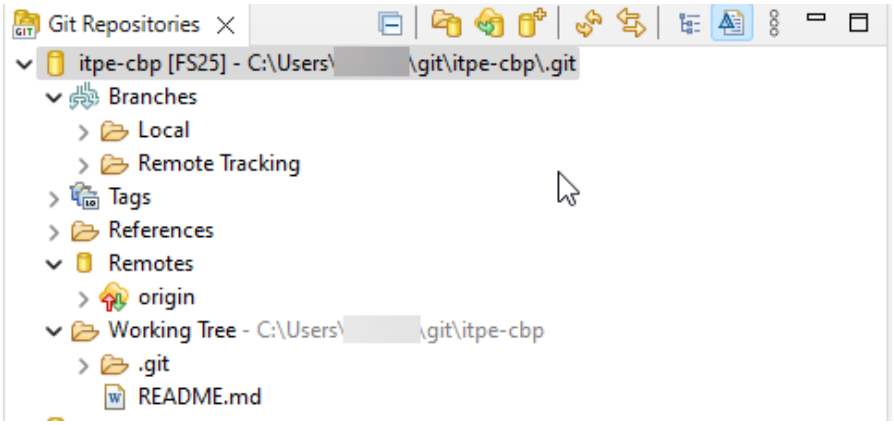
5. Review the directory and initial branch and Click Finish.

   Note: Please accept the default directory. Changing the directory may cause an issue.

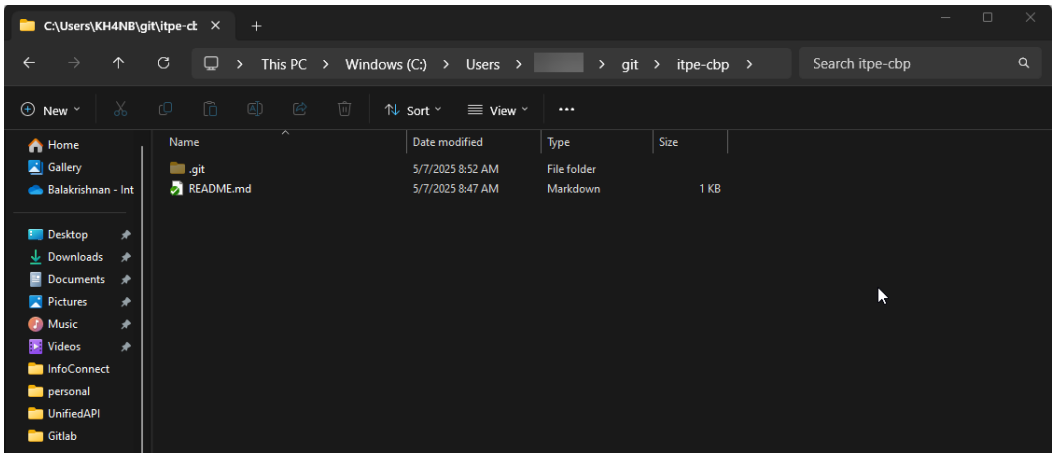
   Note: If you do not have an I: drive, then your default will likely be C:\Users\Your_SEID



6. If successfully cloned, you will see the cloned repositories as follows.

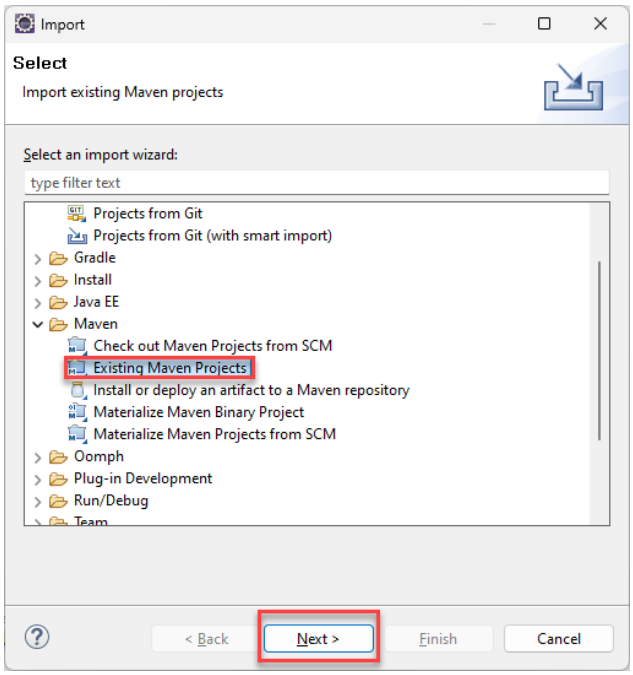


7. Open File Explorer, navigate to the folder location (in this case C:\Users\<SEID>\git\itpe-cbp)

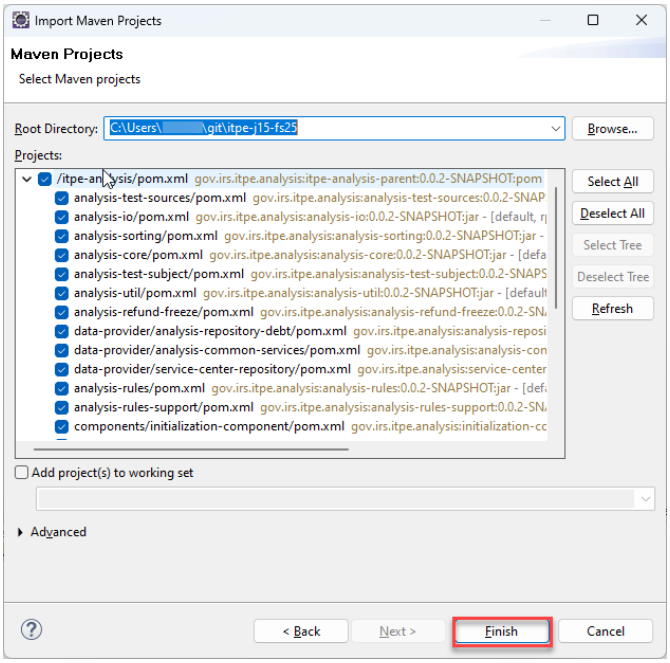


7.4. Importing Project using Maven

1. In the Project Explorer Tab select Import Project

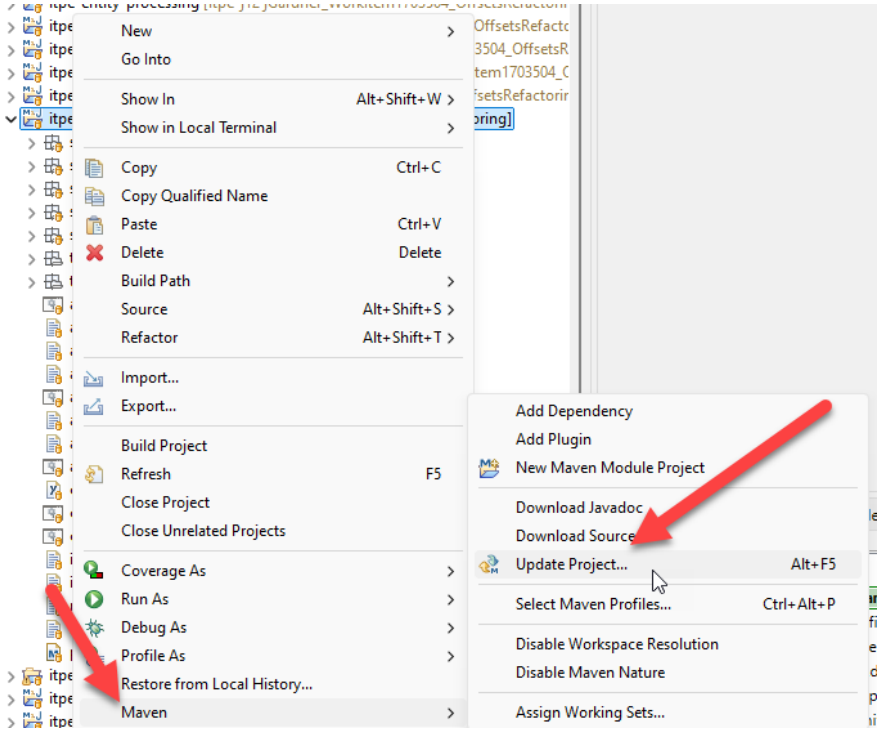


2. Navigate to your projects folder in your user directory

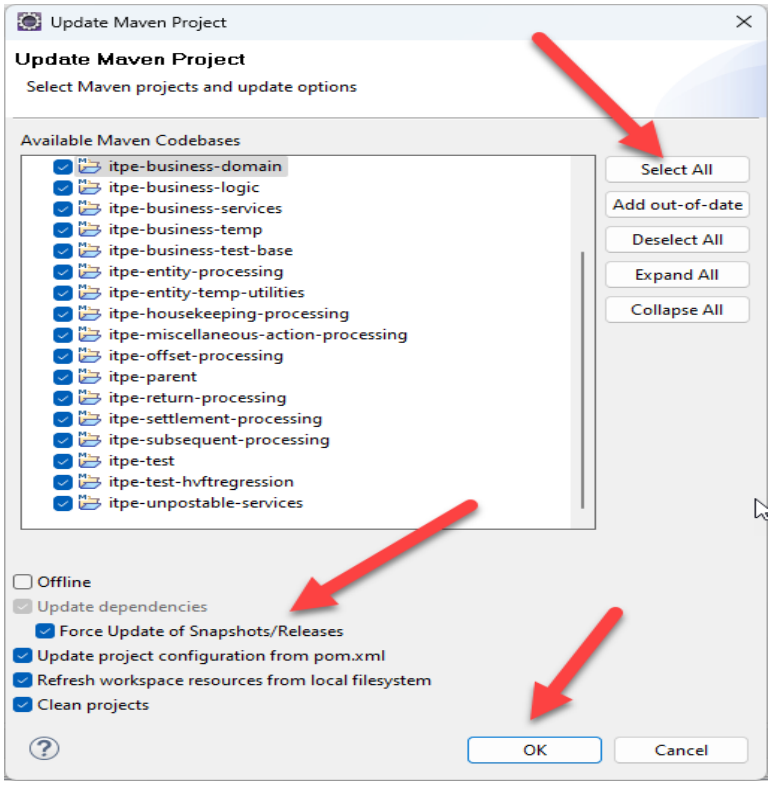


7.5. Updating Project

1. Right click on a project Maven>Update Project to make sure the projects are compiled clean.



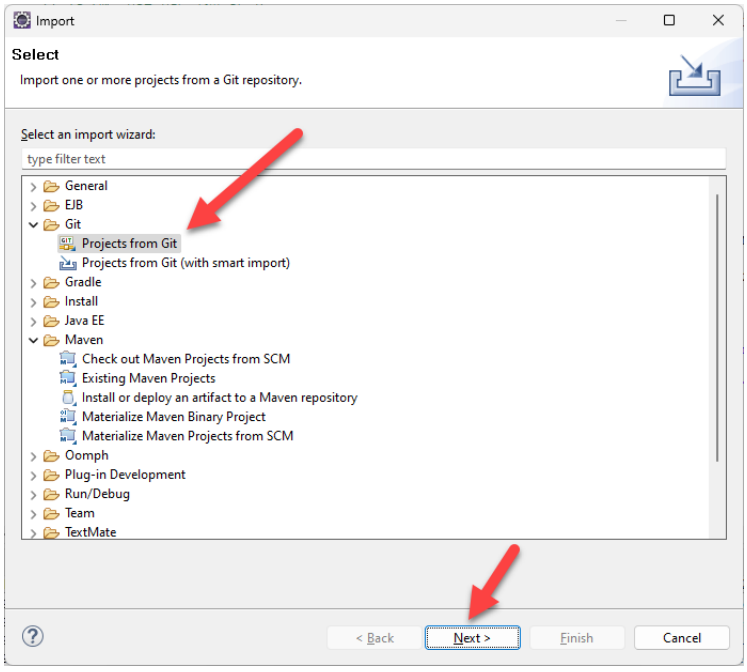
2. Click Select All Projects>check the box “Force Update of Snapshots/Releases”>Select OK



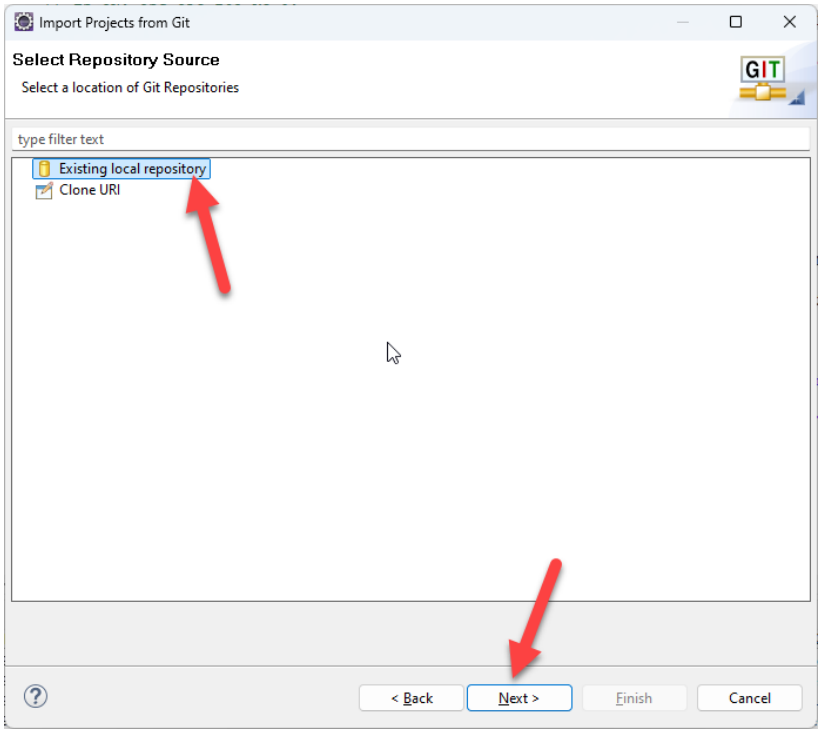
7.6. Importing Project using Git

Note: You need to use this Import Process to get the BB Documentation

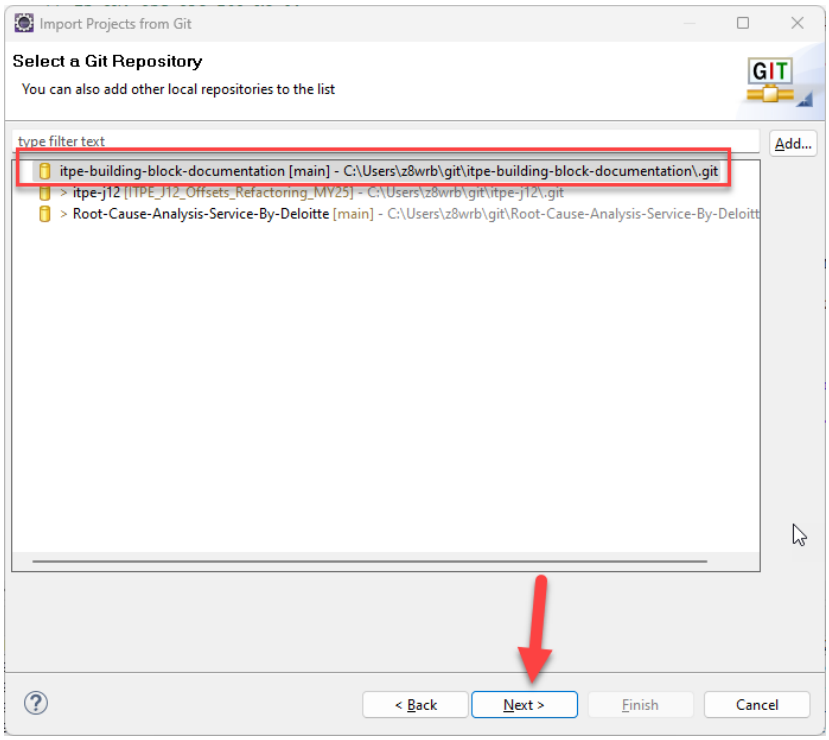
1.Select “Projects from Git” and Click “Next”



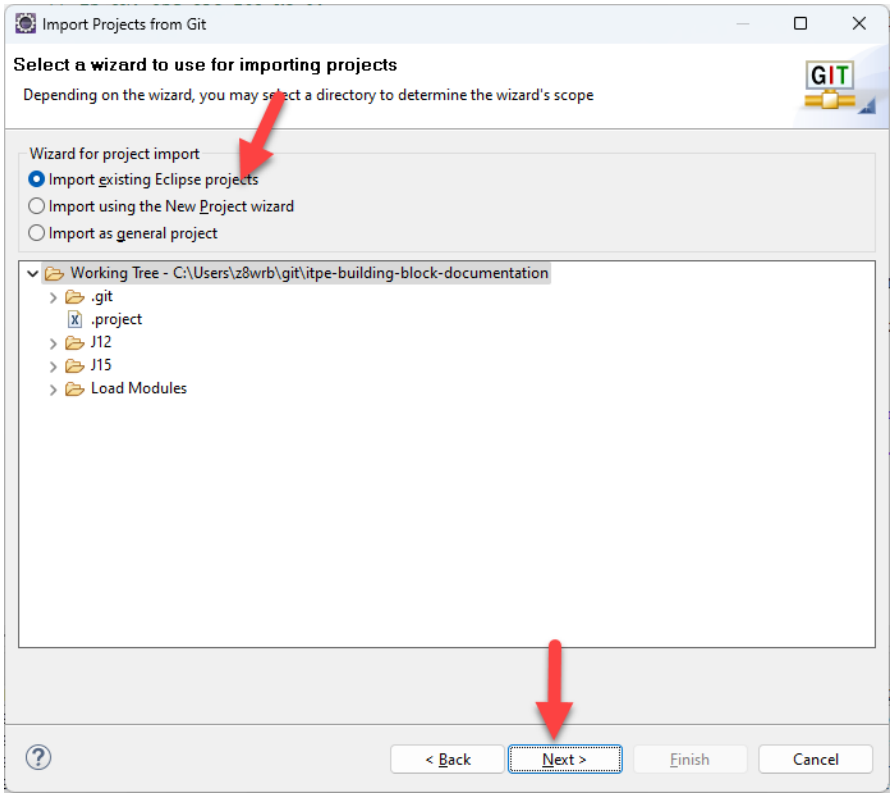
2. Highlight “Existing Local Repository” and Click “Next”



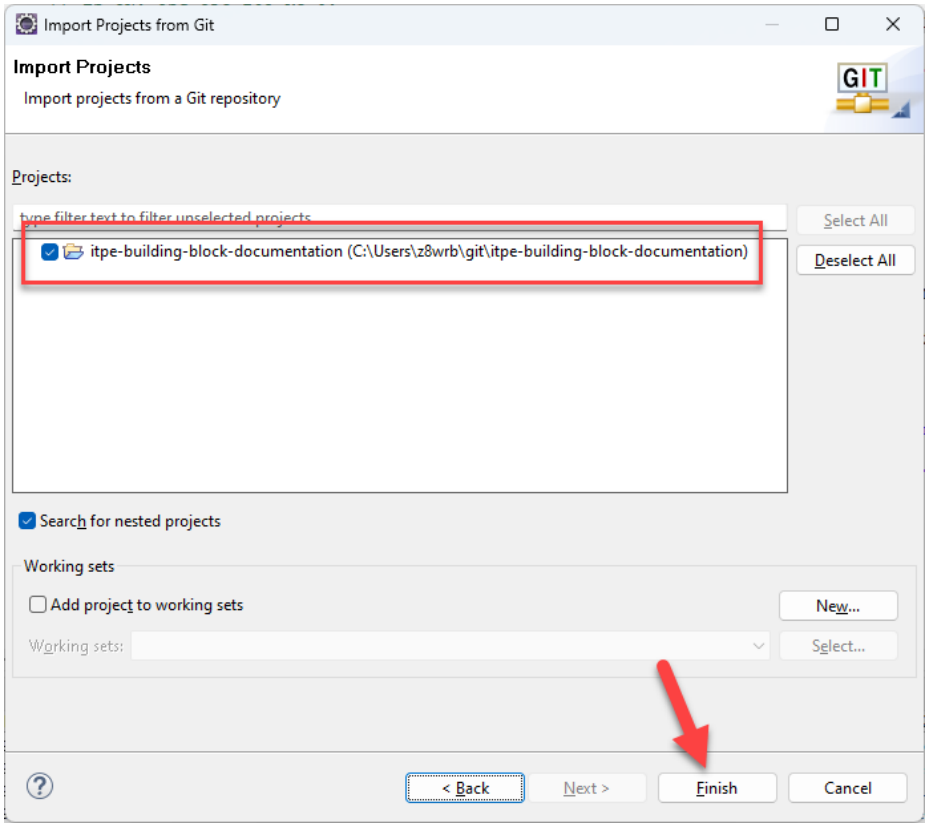
3.Select the project you want to import and click “Next”



4.Toggle “Import existing Eclipse projects” and click “Next”



5. Review project to be imported and click “Finish”



8. Gitflow

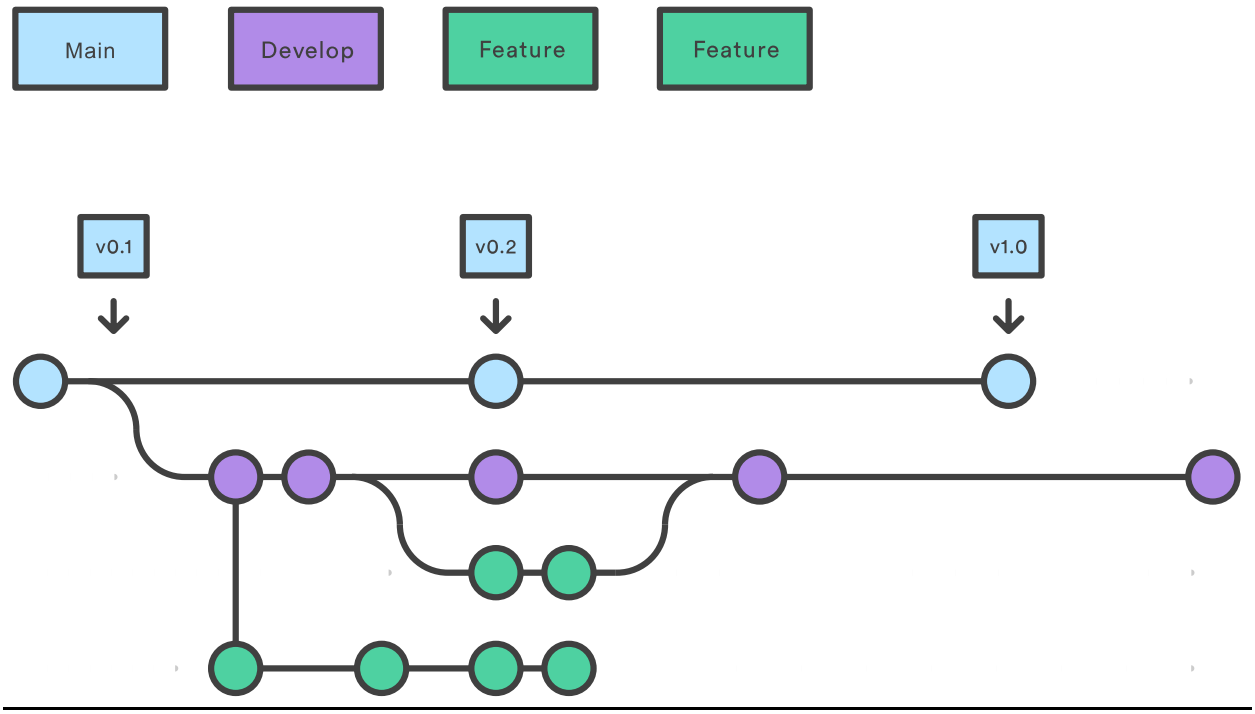


Diagram from www.atlassian.com

Git flow is a branching and merging process designed around project releases. It's recommended to be used when projects and teams have formal releases.

1. The ITPE uses 4 formal branches: **main**, **fs, my, team,** and **feature** branches. The **main** branch is the stable production-ready branch. Only users with repository Admin access can update the **main** branch. The **fs, and my** branches are the integration branches where new features are added and tested. **Team** branch is for the team that works on the same features. **Feature** branches are created off the **team** branches and merged back into them when complete.
2. To work on a new feature, developer creates a **feature** branch from the **team** branch. To distinguish your feature branches with others, it's recommended to include your name in the names of your feature branches. The naming convention of feature branch is <firstLetterofFirstNameLastName>_workitem# -fs25.
3. When a feature is complete, the **feature** branch is merged into the **team** branch.
    a. To merge from a feature branch to the **team** branch, developer needs to create a **Pull Request**
    b. When creating a **Pull Request**, developer specify **approvers/reviewers** to review the changes
    c. Once reviewers approve the changes, developer can merge the change to the **team** branch
4. Once the changes are merged to team branch, the team lead will merge the team branch to **fs/my** branch.

5. When all features related to a release are implemented, and **fs/my** branch is tested, repository admin will merge the **fs or my** branch to the **main** branch
6. DM team will deploy the **fs or my** branch code to production
7. If an issue is detected in production, create a **Hotfix** branch from the **main** branch
8. Once the hotfix is complete, **Hotfix** is merged to **fs, my, and team** branches

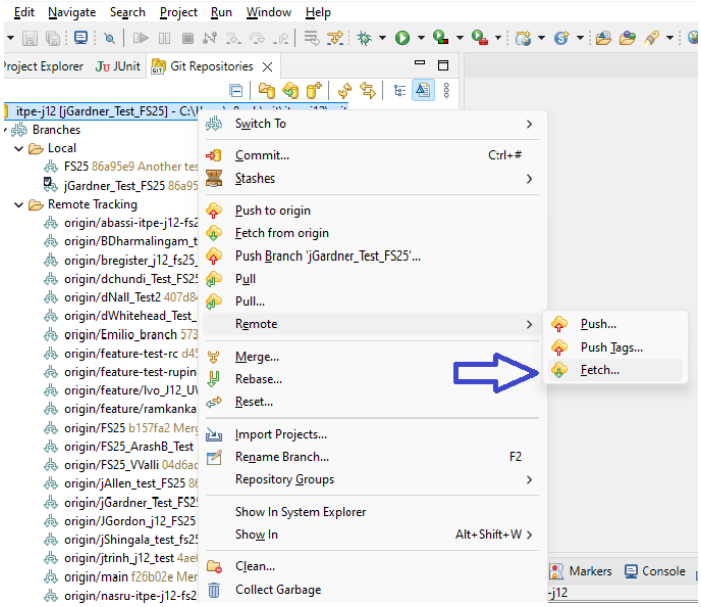
9. Creating Local Team Branch

If stating from a new workspace; the following steps should be done in order. This guide is intended to create a standardized process for creating the appropriate branches and using feature branches to make code changes.

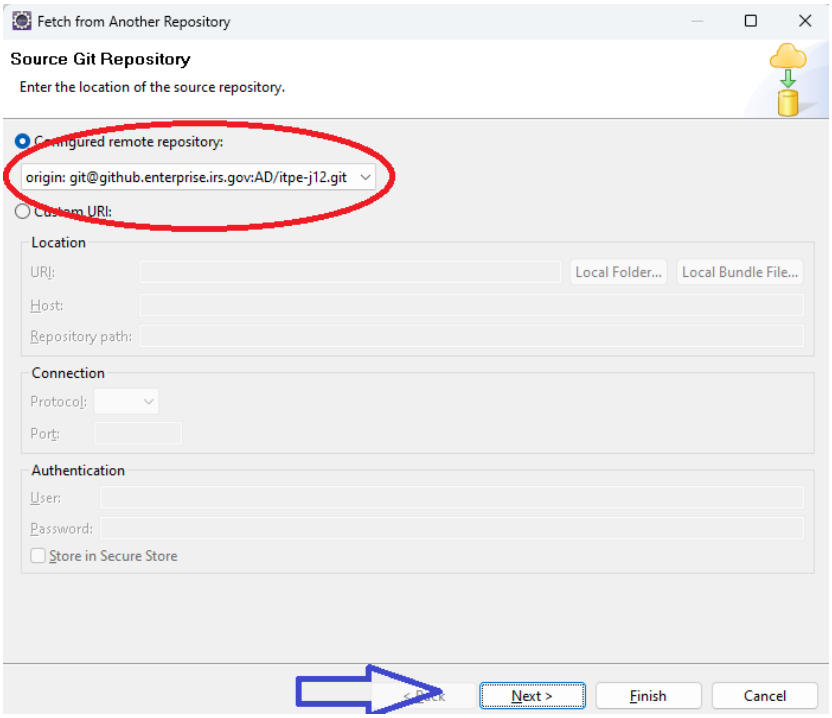
9.1. Adding Pre-Existing Remote Branches

Team lead will notify you when they have created a new Team branch. If you don't know the name of it reach out to your Team Lead.

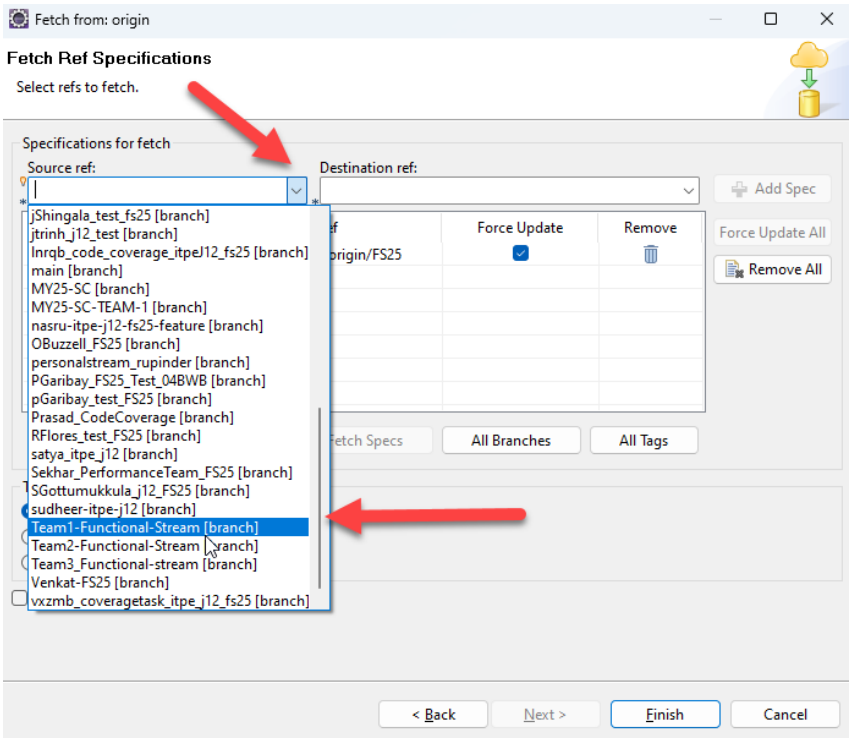
1. Right Click on Repository Icon>Remote>Fetch



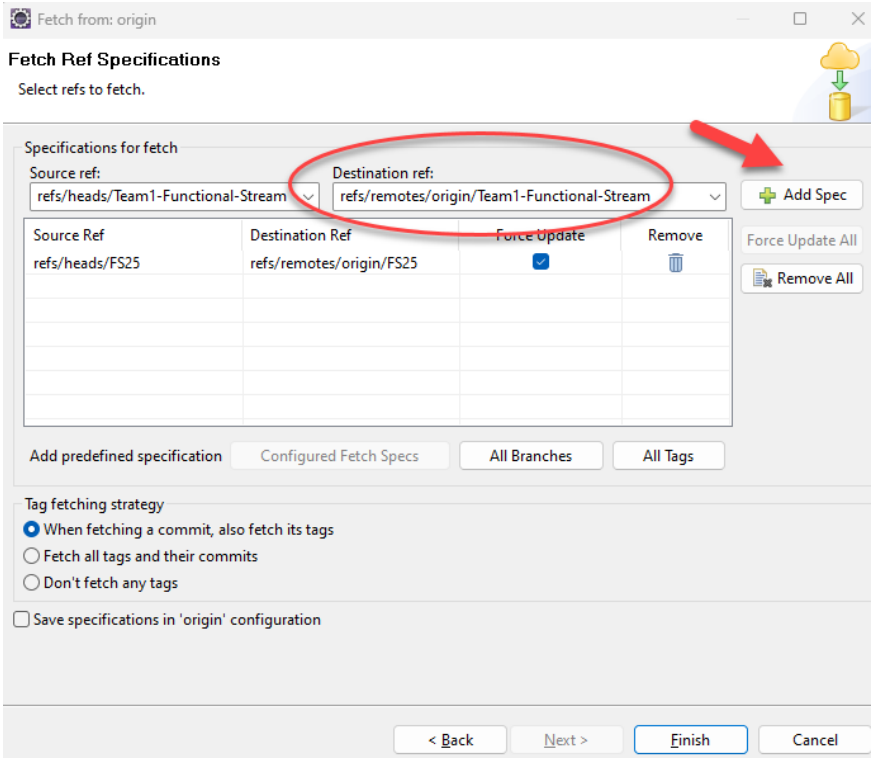
2. Confirm you are pulling from the correct repository (e.g. itpe-j12, itpe-j15, Root-Cause-Analysis-Service-By-Deloitte, etc) and hit next.



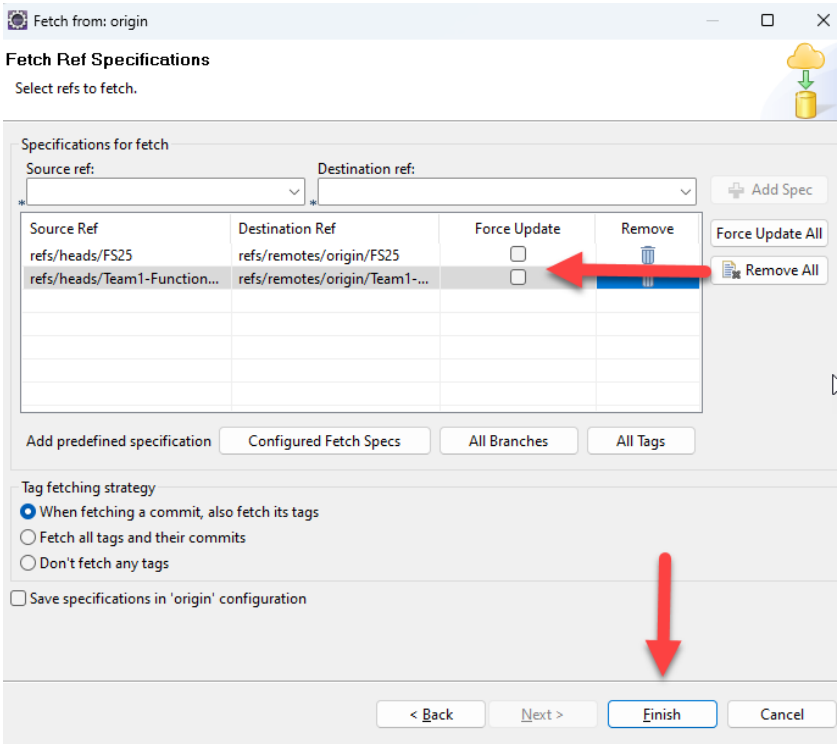
3. Click the source dropdown and select the appropriate remote branch (e.g. Team1-Functional-Stream, Team2-Functional-Stream, etc.)



4. Ensure that the destination ref is autopopulated correctly and click Add Spec



5. Ensure that the checkboxes for Force Update are unchecked and click finish.

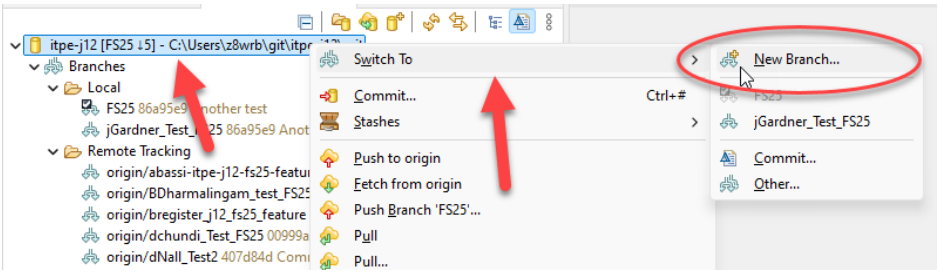


6. After clicking finish, you should now see the branch under your remote tracking folder.

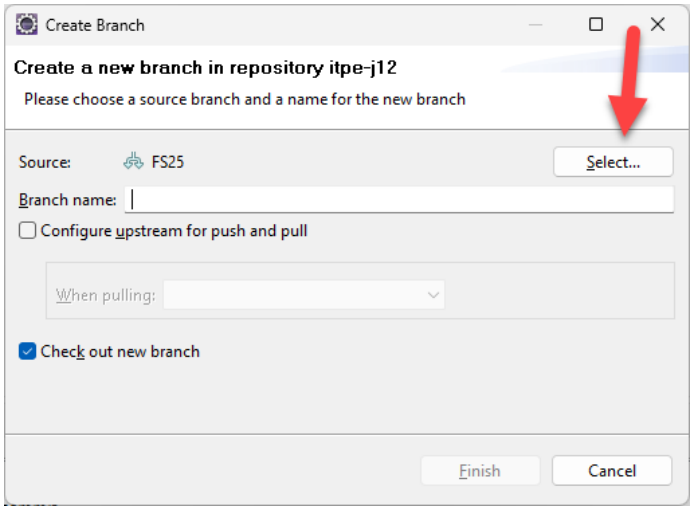
9.2. Creating Local Team Branch

After adding the team branch to remote branches folder, you need to create a local copy of the Team Branch.

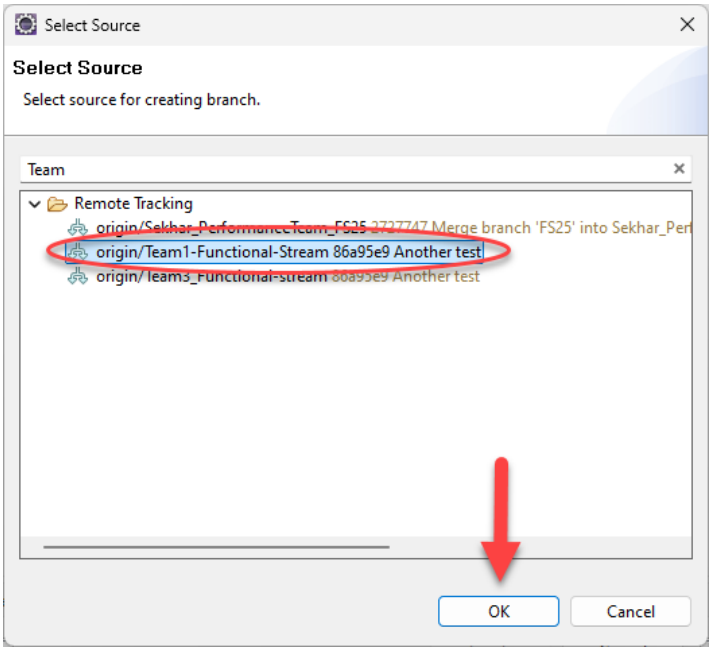
1. Right Click on the Repository>Select Switch To> New Branch



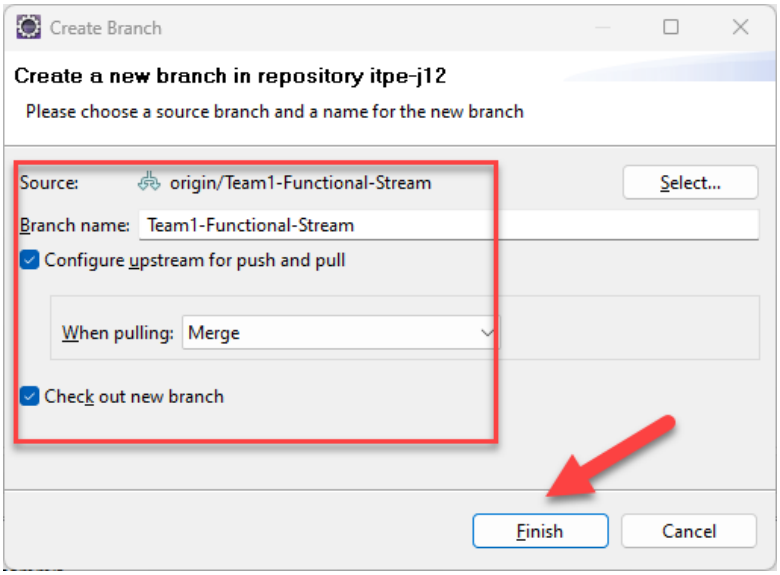
2. On the dialog box that opens click on Select… for the Source Branch.



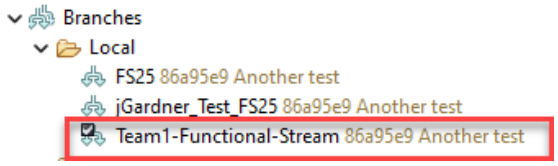
3. Select the appropriate Team Branch and click OK.



4. On the next Dialog Box, keep all the options to the default (e.g. Branch Name, Configure upstream for push and pull, checkout new branch) and click Finish.

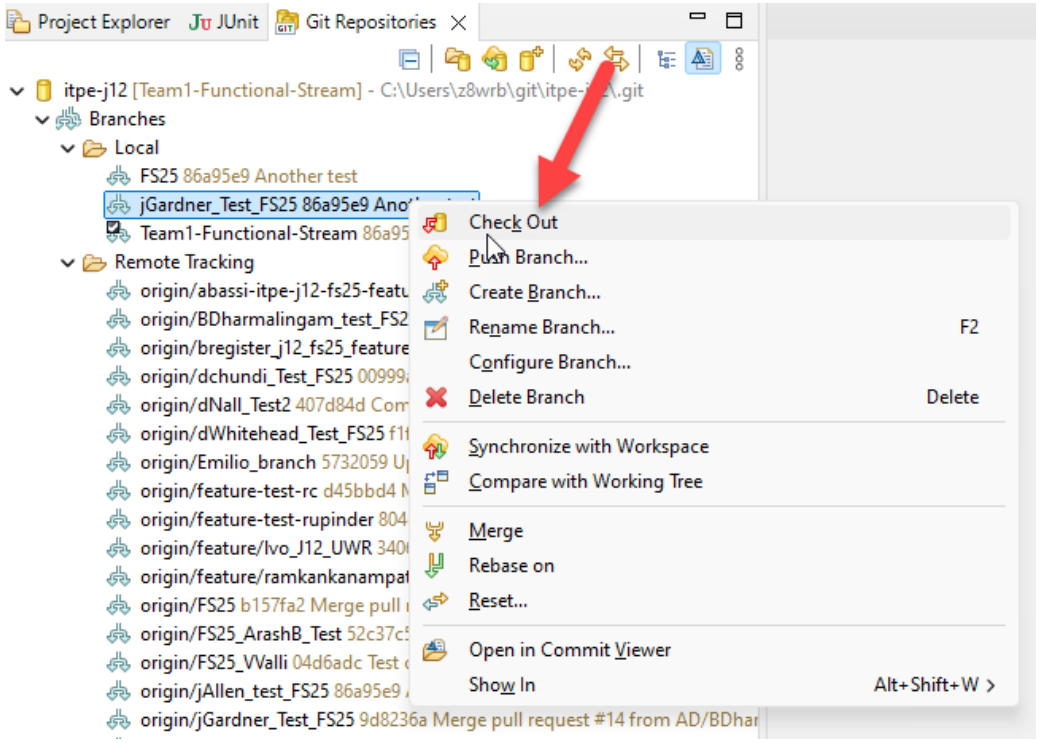


5. Under Branches>Local you should now see the Team Branch with a checkbox on it.

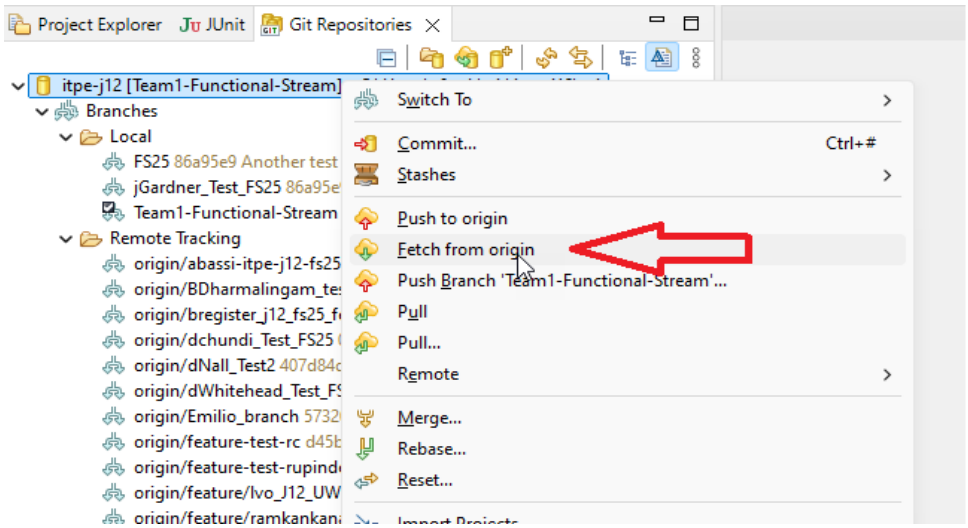


9.3. Creating Remote Tracking Branch

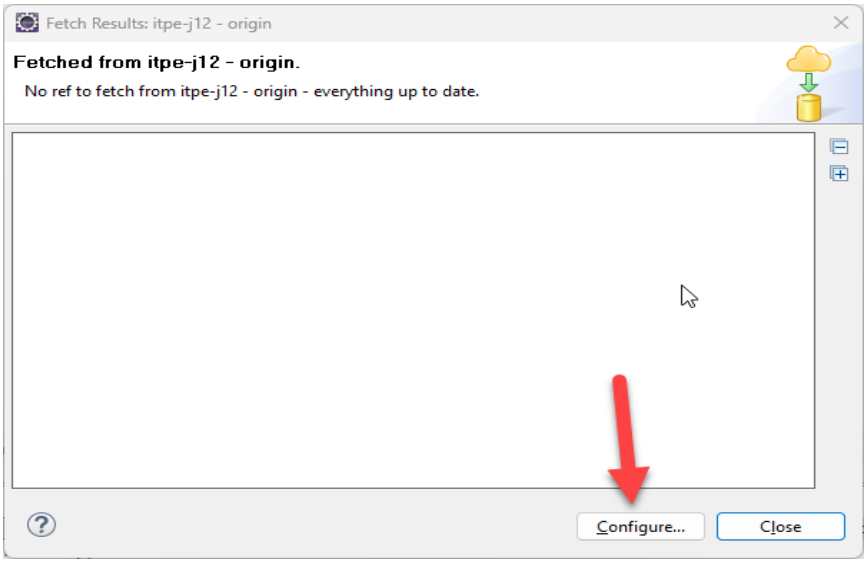
1. Check out the branch that you want to create a Remote Tracking Branch.



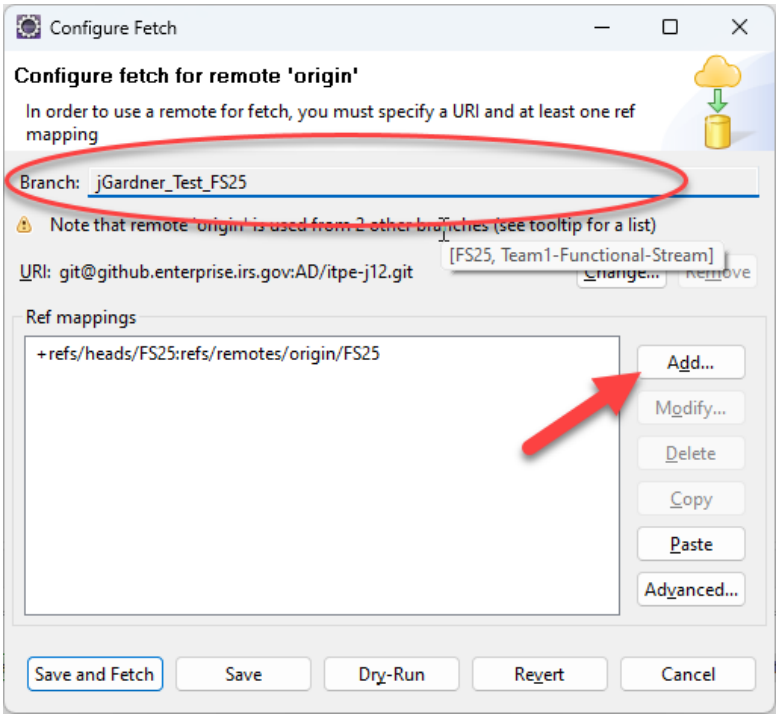
2. Right click at the repository level and click "Fetch from origin", keeping the ensuing dialog box open.



3. In the dialog box that pops up, click "Configure" at the bottom right.

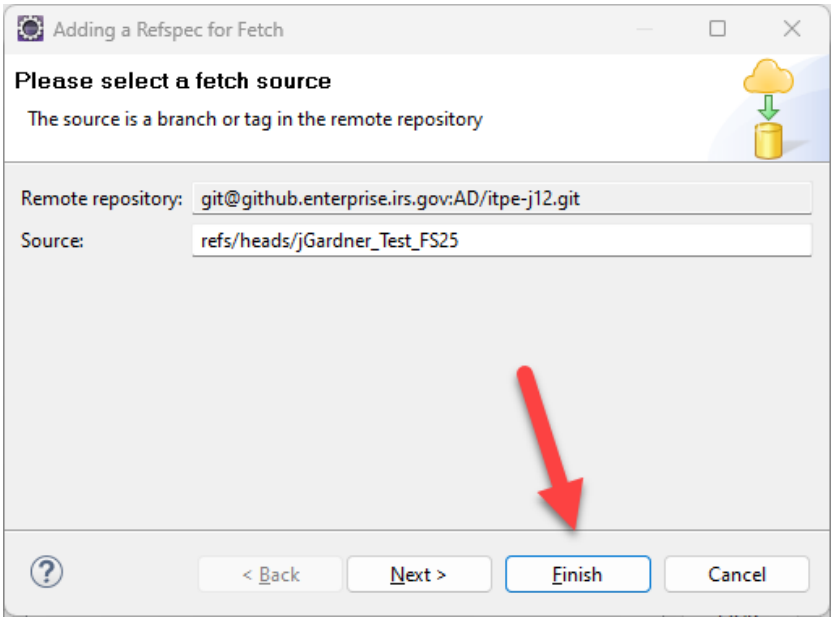


4. A second dialog box will appear, click the "Add" button on the right



5. A third dialog box will appear, start typing the name of the new branch until a dropdown appears, select the name of your new branch in the dropdown.

   this should populate the text box with something like refs/heads/<your new branch name> for example refs/heads/jAllen_WorkItem196541_MY26



6. Click finish. Under the Remote Tracking folder, you should now see origin/<name of new branch>

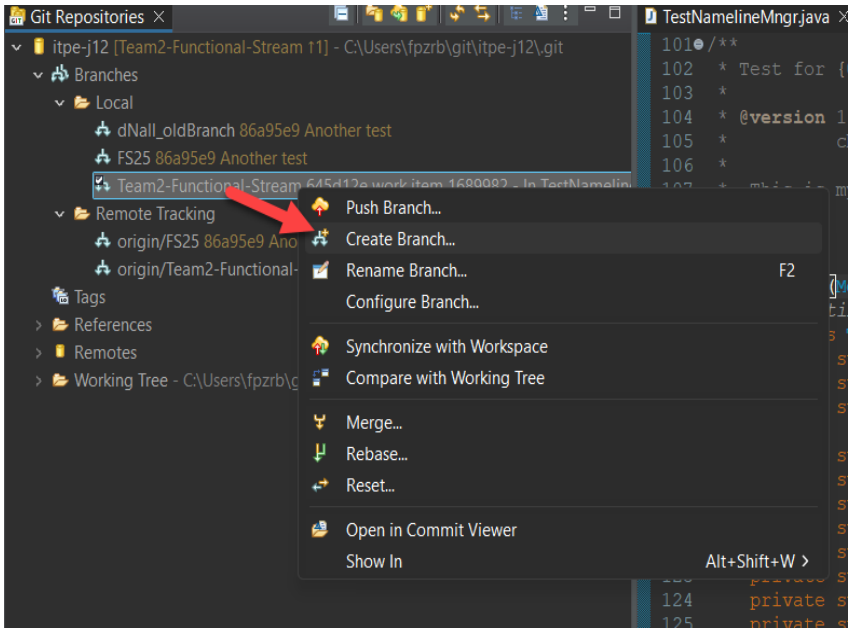
7. ***. Every branch you want to pull from or push to will need a Remote Tracking branch***

10. Creating Freature Branches

If this is your first feature branch you can perform the instructions as follows.

Note: If this is not your first feature branch you need to follow the instructions in Section 12, ensuring the Team Branch is up to date before creating your feature branch. It's important to create the new branch from the current state of the source branch to minimize merge conflicts when submitting the final pull request for the feature branch to merge back into the team branch

1. Right click on the source branch and select "create branch"

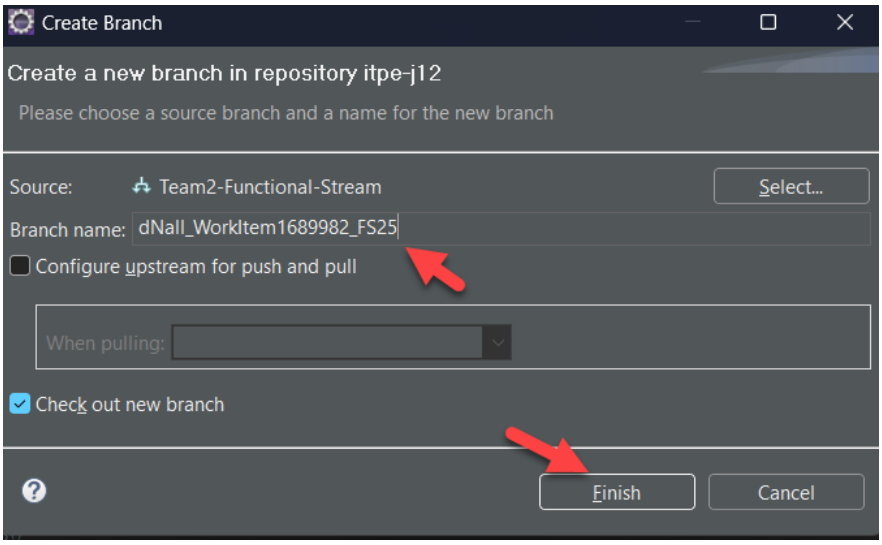


2. Enter the name for your new feature branch. Follow naming convention according to your team and task. Ex.

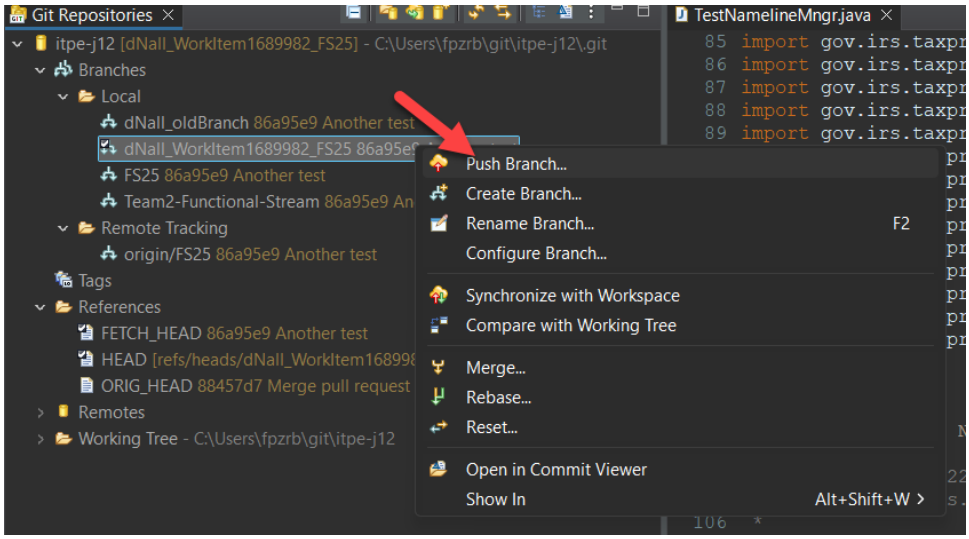
<lower case first Initial><Camel case Last Name>_<Short Description>_WorkItem<work item #>_<FSYY, MYYY, or RefactoringEffort>,

For example, if John Allen wanted to create a feature branch for Defect 196541 during Midyear season of 2026, the correct name for the branch would be ***jAllen_actionCode54Defect_WorkItem1234_FS25***

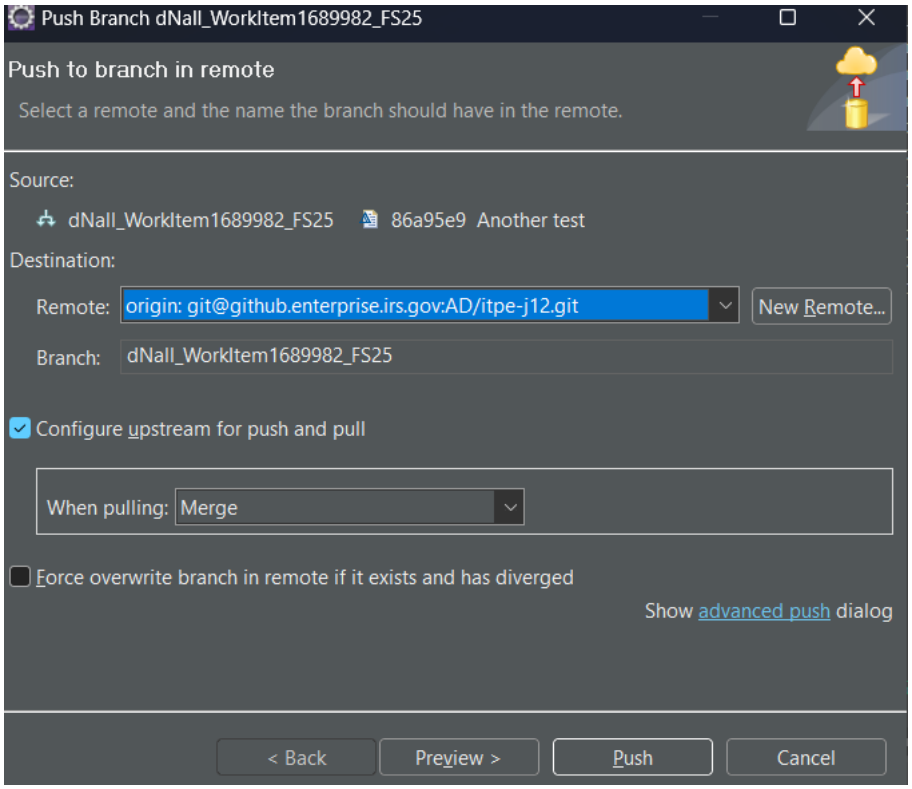
Select “Check out new branch”



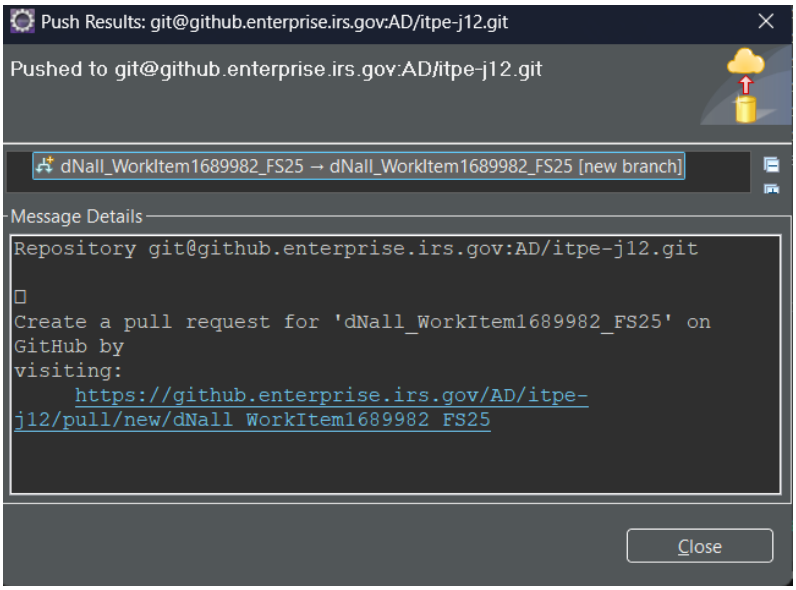
3. Right click on the new branch and select "Push Branch"



4. Ensure your Feature Branch name was populated correctly and “Configure upstream for push and pull” is selected, then click “Push”



5. After pushing you will get a confirmation dialog. You can copy the GitHub link from here for reference. Click Close

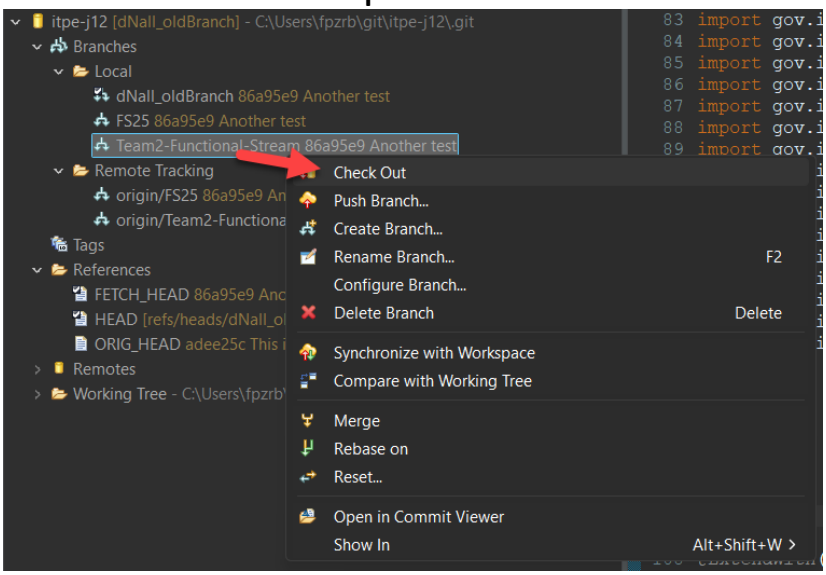


11. Pulling Changes and keeping branches up to date

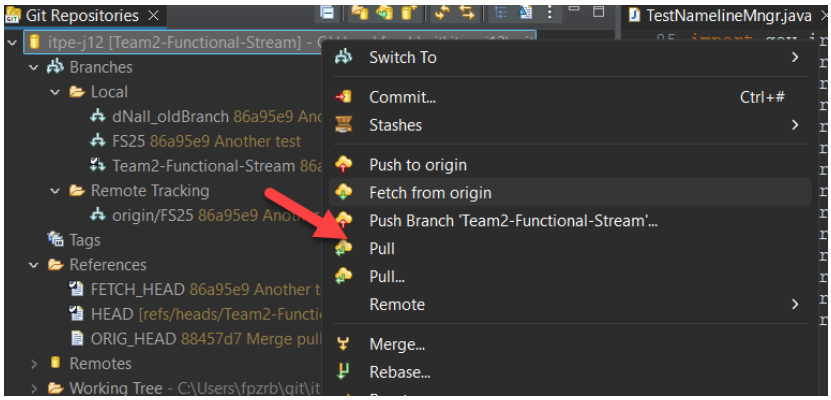
Note: If you have just created your Team Branch, you can skip the Pulling Changes section because your Team Branch will be up to date. If this is not your first Feature branch you will want to follow the steps for pulling changes.

11.1. Pulling Changes

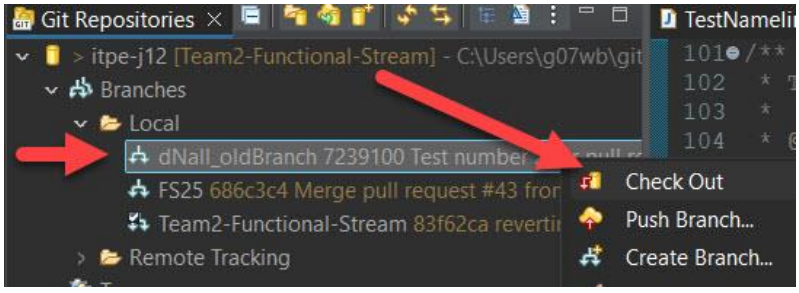
1. Check out your source branch (e.g. Team1-Functional-Stream, Team2-Functional-Stream). Which branch this is will depend on your team and your task. Check with your lead. Uncommitted changes may prevent you from checking out a different branch; commit, stash or undo these changes as is appropriate.



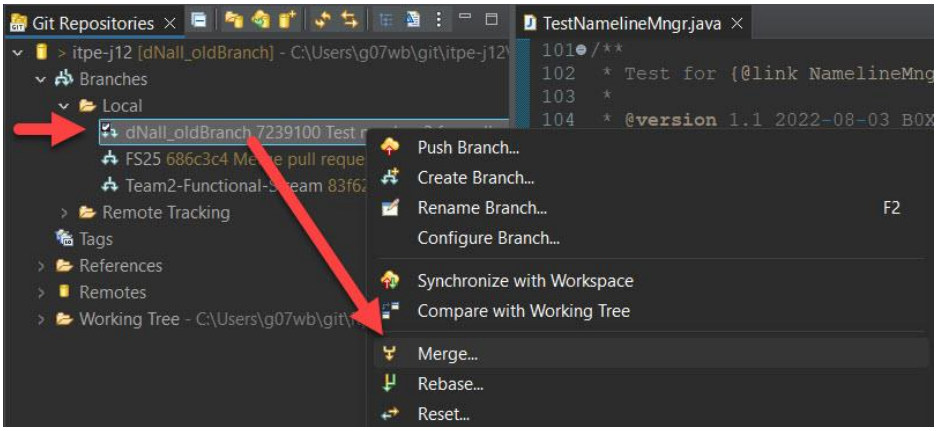
2. At the repository level (the top level with the cylinder icon), right click and select "Pull" to bring your local copy of the source branch up to date



3. Check out your local feature branch so it can be updated with the changes pulled into the team branch

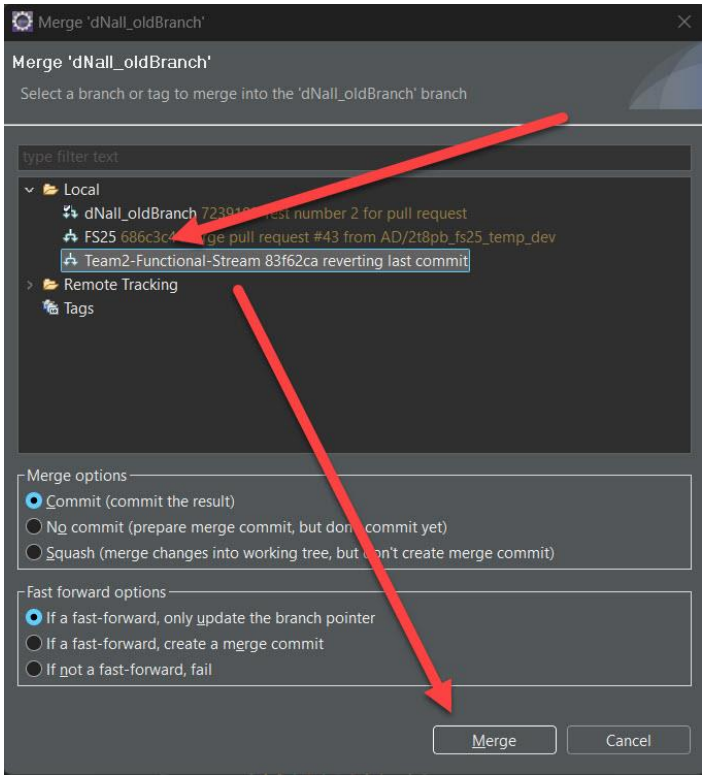


4. Right click on your local feature branch and select “Merge…”

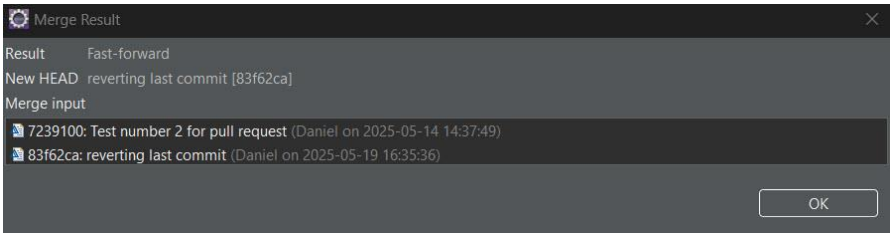


*****IMPORTANT: If the icon says “Merge” without the ellipsis (…) then you likely have a different branch checked out, and the merge will occur in the wrong direction. Be sure to have your feature branch checked out so that you are given the “Merge…” option*****

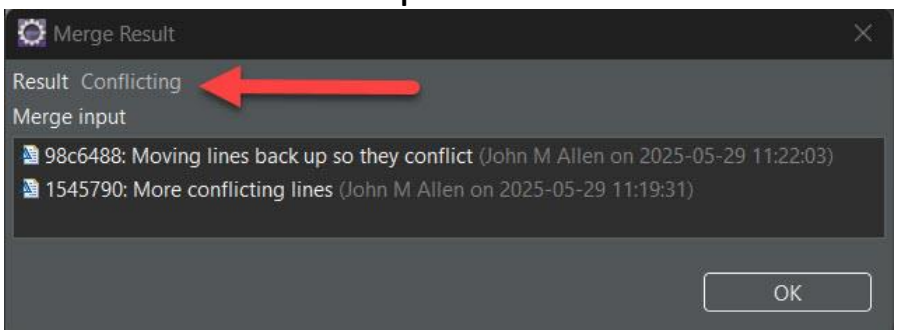
5. Select your source branch (the branch that you pulled in step 1) in the merge dialogue. Ensure that the “Commit” radio button is checked for “Merge Options” and the “If a fast-forward, only update the branch pointer” radio button is checked for “Fast Forward Options.” Click the Merge button



6. If the merge is successful, the Merge Result dialog should specify a result indicating so, such as “Fast Forward.” If there merge conflicts, the dialog will indicate so. Any conflicts will need to be resolved before moving to the next step. See the Resolving Conflicts with Tortoise section for instructions.



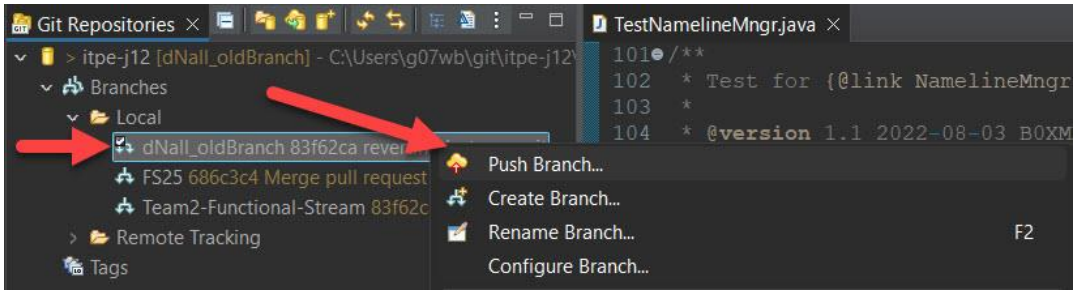
Successful Merge Result



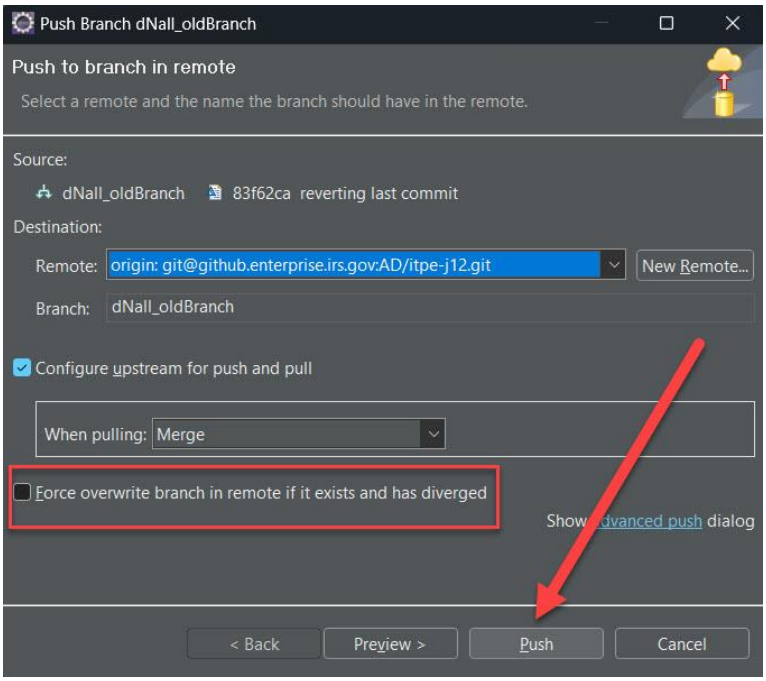
Failed Merge Due to Conflicts

7. If you had any conflicts in the previous step, ensure that they have all been resolved (See , and the result has been committed (See "Committing and Pushing Changes" for instructions on how to commit)

1. Right click on your local feature branch and select "Push Branch" to push the changes from your local feature branch to your remote feature branch on GitHub



2. In the dialogue that appears, ensure that the “Force overwrite” checkbox remains **unchecked**. Click the “Push” button



****Important: In order to minimize the possibility of merge conflicts, it’s a good idea to pull changes to the source branch at least every few days to ensure that everything is up to date. If upon pulling, no updates have been made to the source branch, then there is no need to perform the steps for merging into the feature branch****

12. Committing Changes

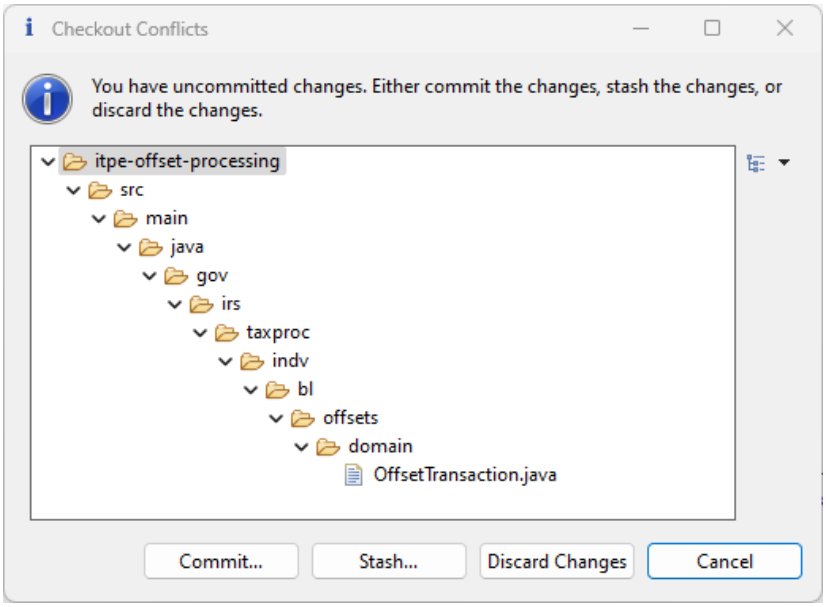
12.1. Steps for Committing Changes

12.1.1. Updating Local Team Branch Before Checking in Changes

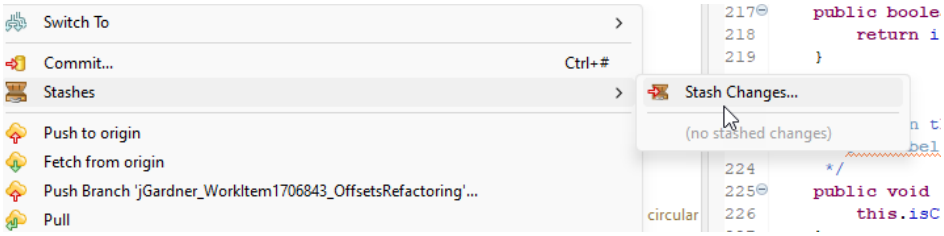
Before you commit any of your changes you will want to make sure your local team branch is up to date. During that process you will most likely need to stash your local changes before updating the branch.

1. When you check out your local team branch to update you will most likely have this dialog box appear, and you will want to stash your

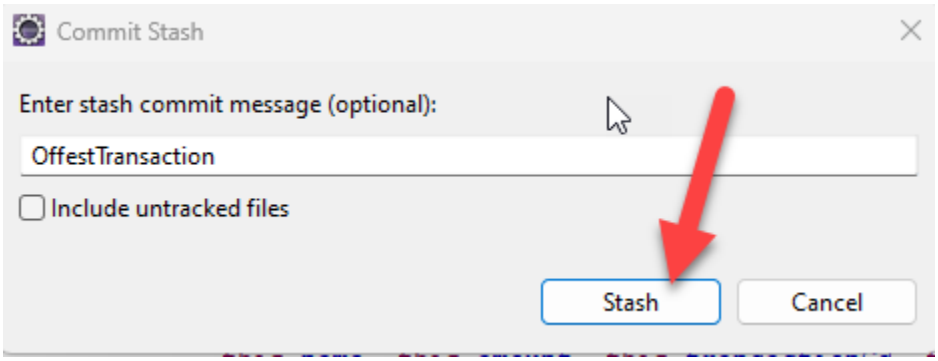
changes.



2. Click Cancel and go to the top of the repository and Right Click, you will be given a number of options including Stashes, hover over it and select “Stash Changes”

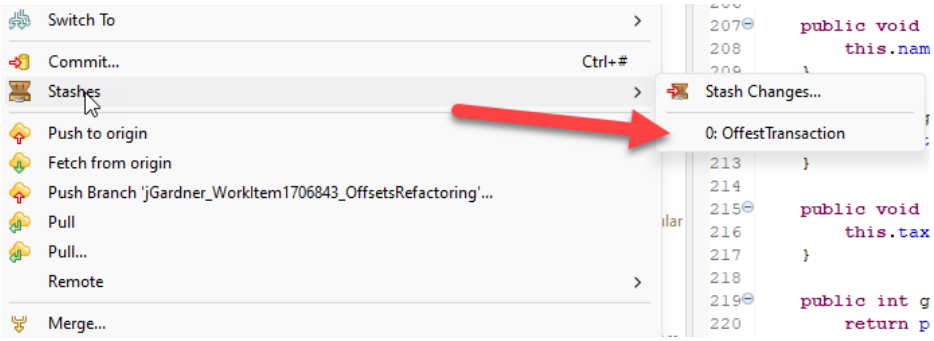


3. On the next dialog box, you will need to give it a name and click Stash, this will save all of your uncommitted changes:



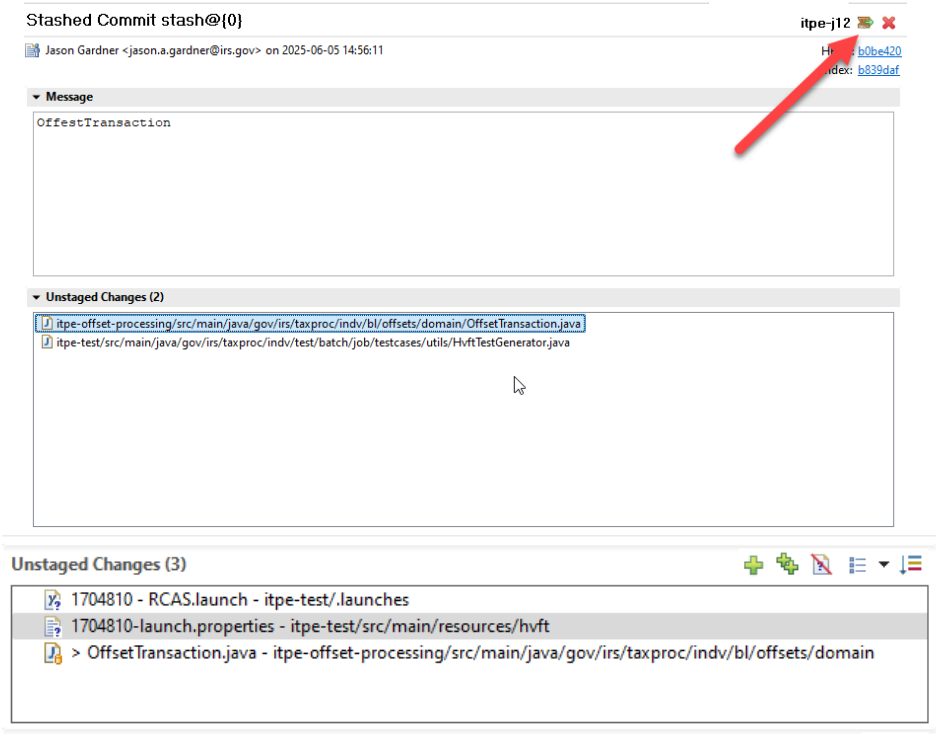
4. After going through the steps to Pulling Changes you will want to apply your stashed changes back to your feature branch, navigate back to stashes and you should see the Stah you created in the

previous step:



12.1.2. Checking In Code Changes

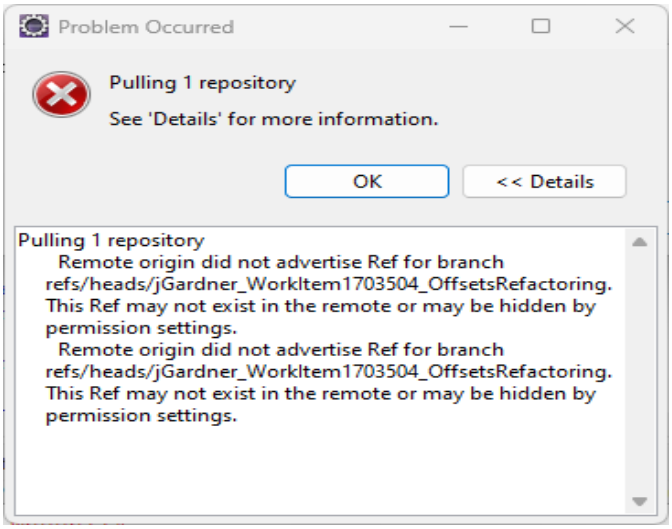
5. Click on it and this diaglog will appear, click the green arrow box and all of your uncommitted changes will appear in the Git Staging view



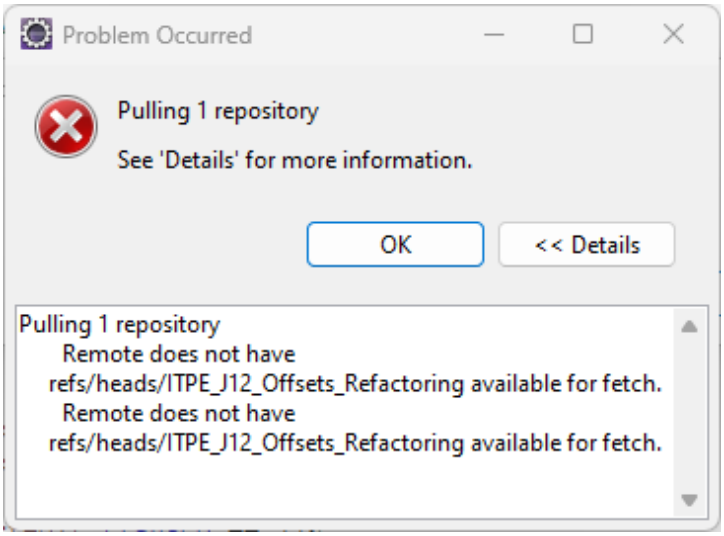
6. Now you will be able to Commit your changes as normal.

12.2. Troubleshooting Commit Steps

1. If you receive this error while trying to pull you need to create a remote branch



2. There is a chance you can run into this error as well, I found that this happens when the Local branch you are pulling from has changed.

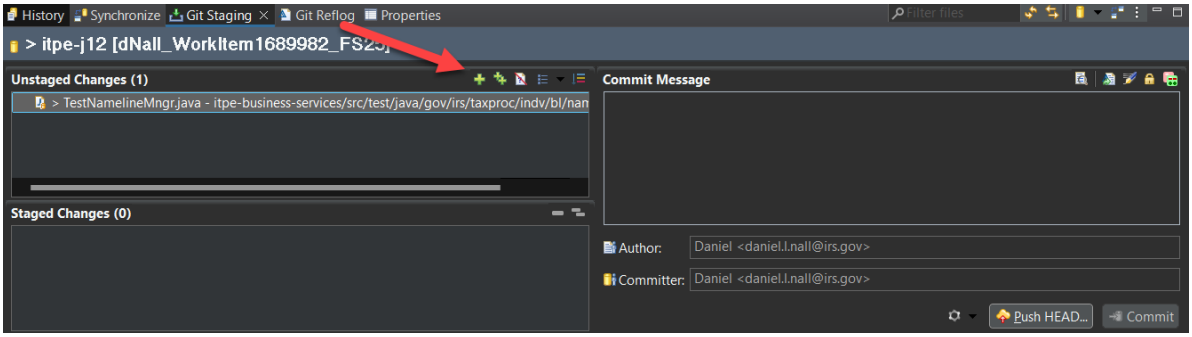


3. You can fix this by following the instructions in the Deleting Branch Section to delete all the associated branches (Local, Tracking, Feature)

4. After that you will have to add all the branches(Local, Tracking, Feature) based on the new Branch name.

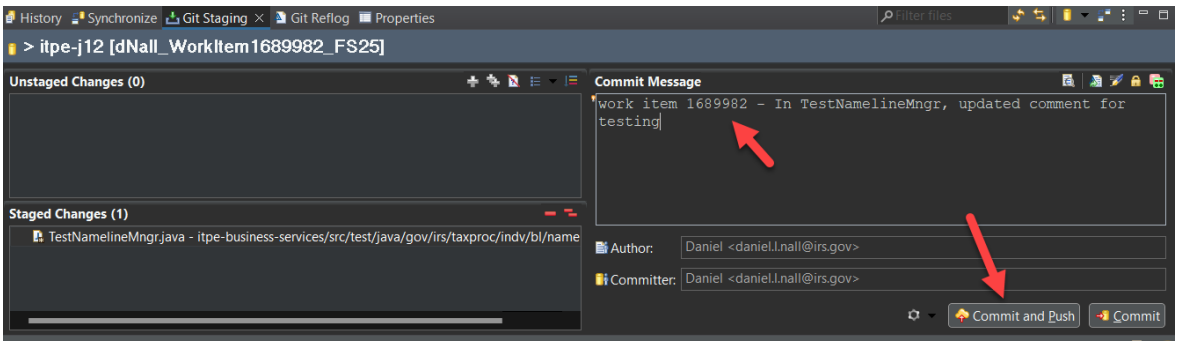
1. Make a change and save.

2. The changed file will show in the 'Unstaged Changes' section of the 'Git Staging' view. Select the changed files you want to stage and click 'Add' . The files will be moved to the 'Staged Changes section.



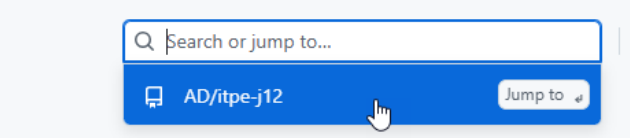
3. Add a commit message starting with "Work Item <EWM Task ID> commit description...". Adding a comment in this format will allow EWM to track the change with the associated ticket. ***Note: do not put '#' after Work Item***

   Click "Commit and Push" <u>after reviewing changes</u>.

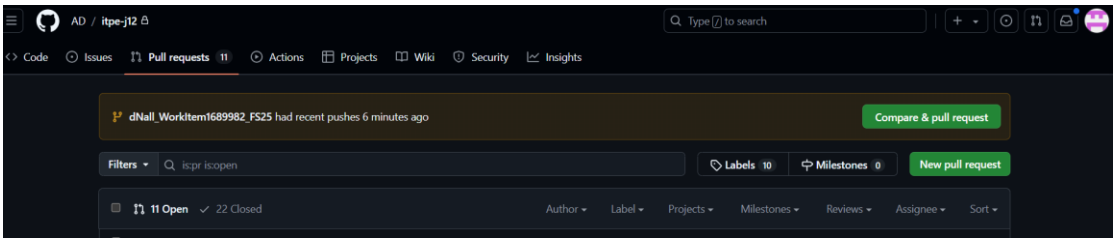


12.3. Create a Pull Request

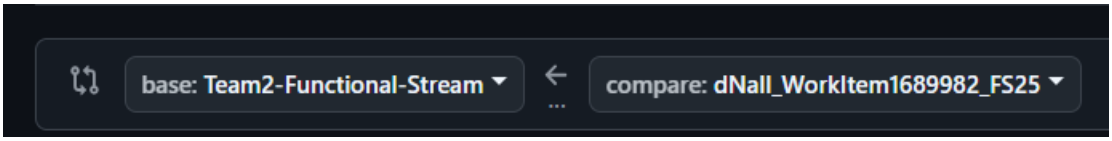
1. On GitHub, navigate to your repository.(e.g itpe-j12,itpe-j15)



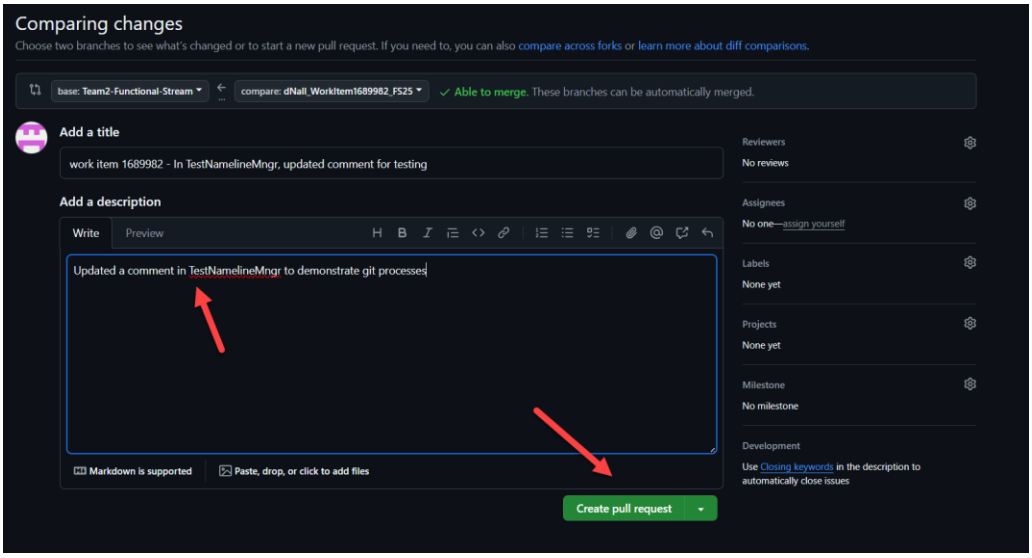
2. Select the ‘Pull requests’ tab and click ‘Compare & pull request’ or ‘New pull request’



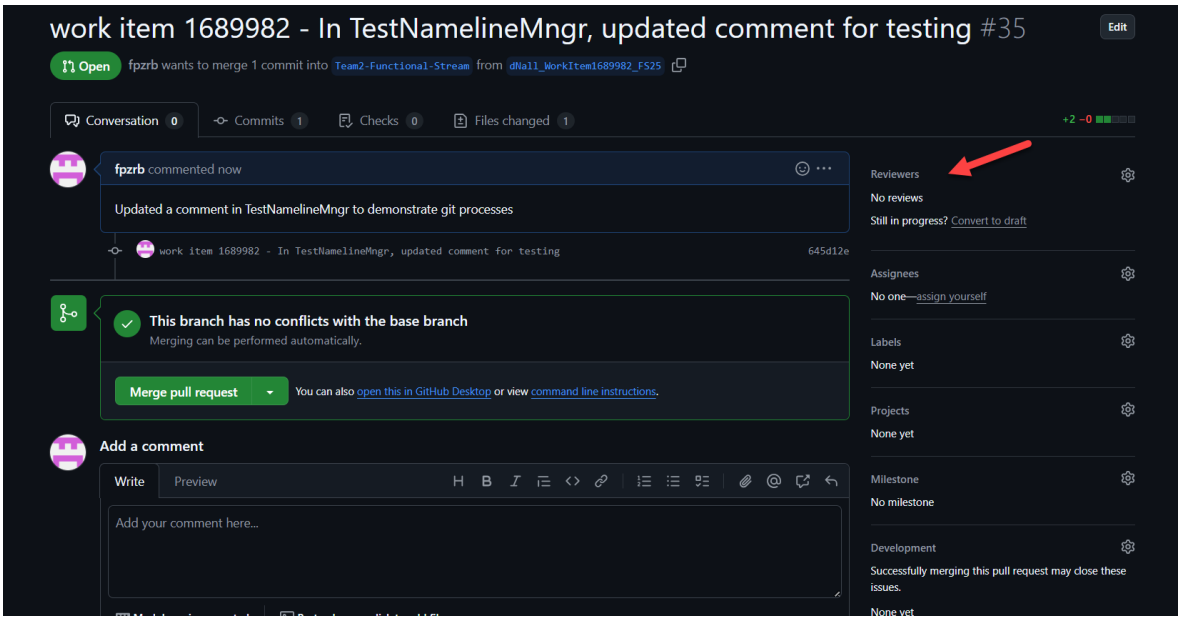
3. Ensure you are pointing to the correct branches (left: target, right: feature branch)



4. Add a description and click “Create pull request”



5. Set Reviewers by clicking on the gear icon on the right side of the page.

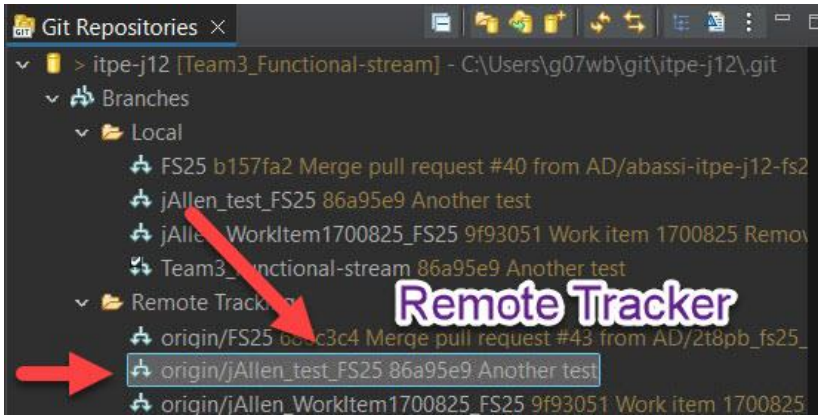


12.4. Updating After a Pull Request is Approved

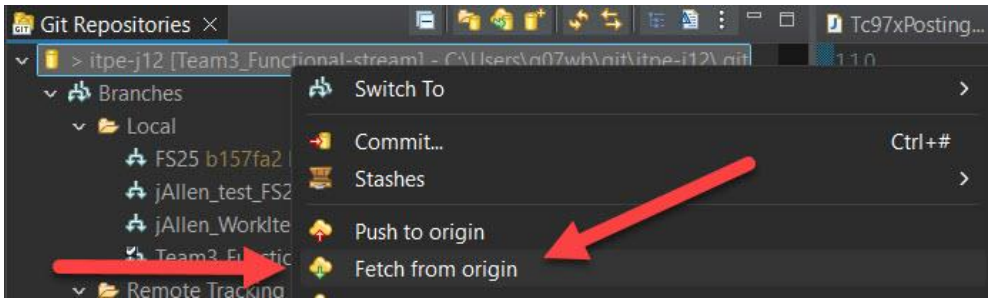
After a pull request is approved, it's a good idea to pull changes on all source branches (see "Pulling Changes" section for details. This helps ensure that any feature branches created in the future will be up to date and will minimize the possibility of merge conflicts.

13. Deleting a Branch

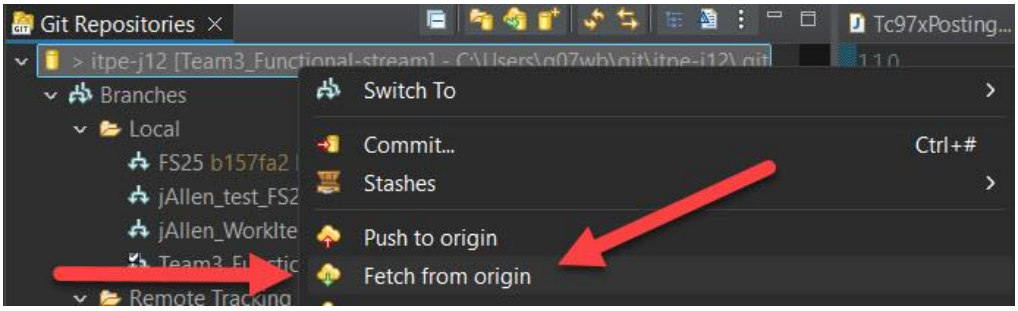
1. Remove from fetch configuration for the remote tracker



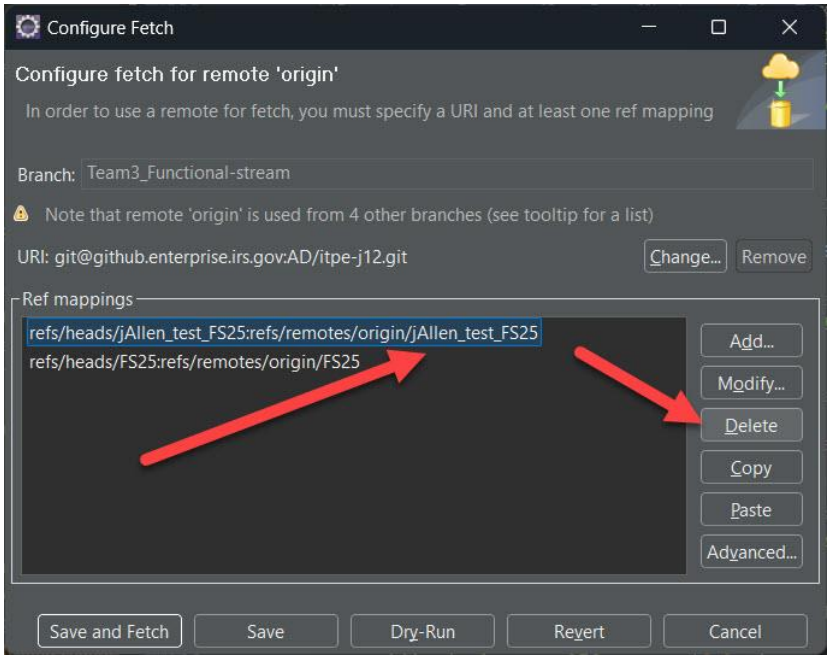
a) Right click on at the top-level repository and select "Fetch from Origin" in the dropdown



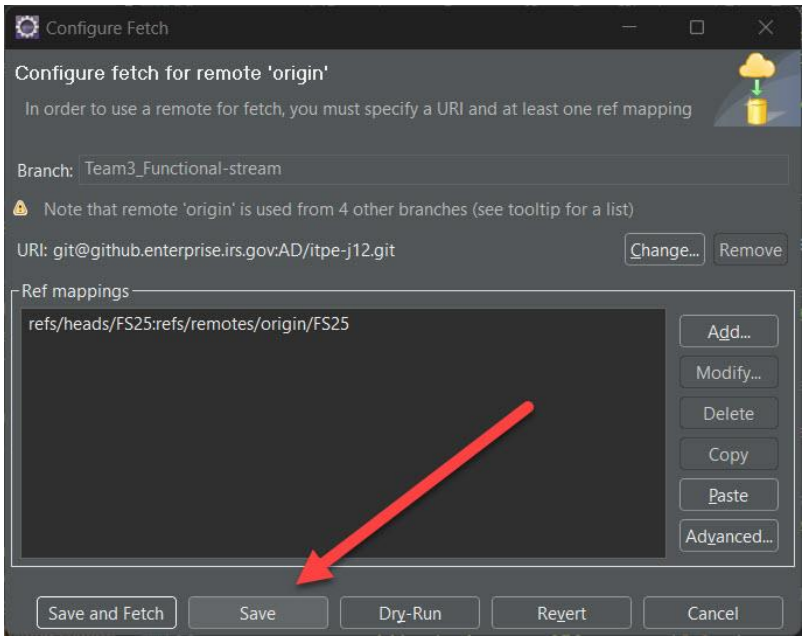
b) Click "Configure" in the in dialogue that appear



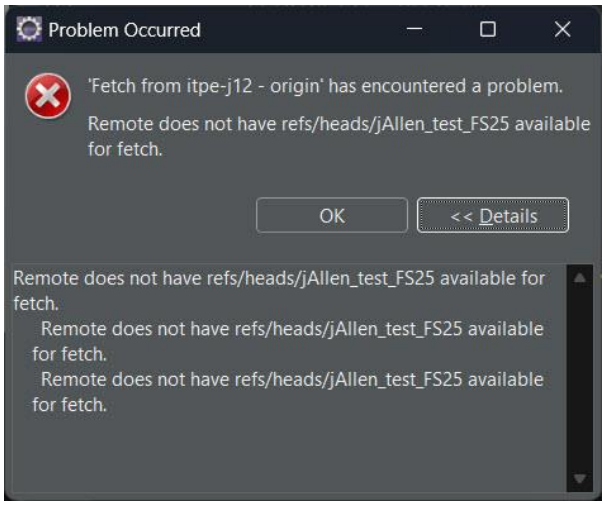
c) Select the line corresponding to the branch you want to delete and click the "Delete" button on the right



d) Click "Save" to close the configuration Dialogue

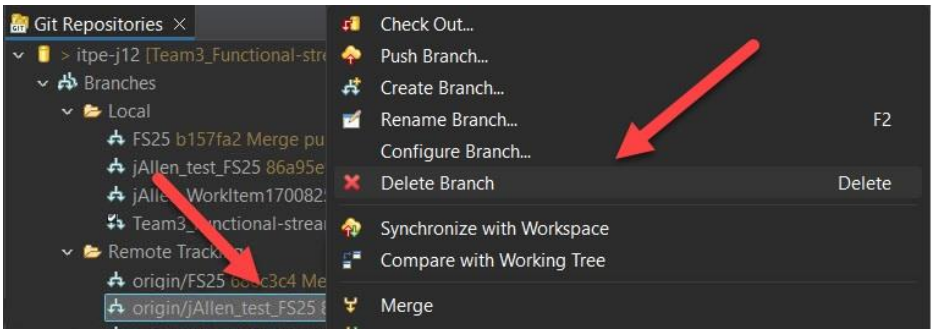


***** IMPORTANT: If you fail to update the configuration before deleting the branch on GitHub, you will receive an error like the following when you try to pull or fetch:**

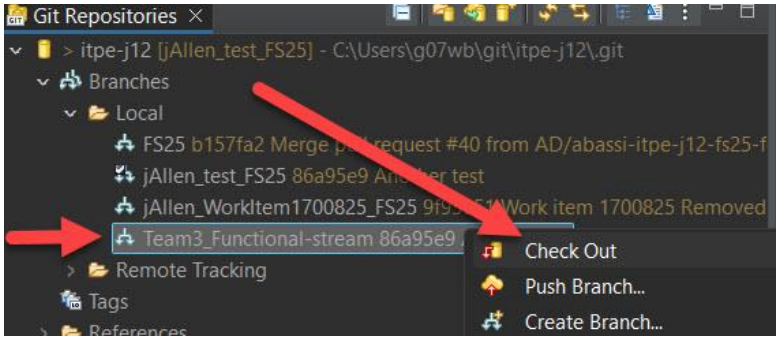


If you receive this error, you will need to follow the trouble shooting steps at the bottom of this section to fix it. ***

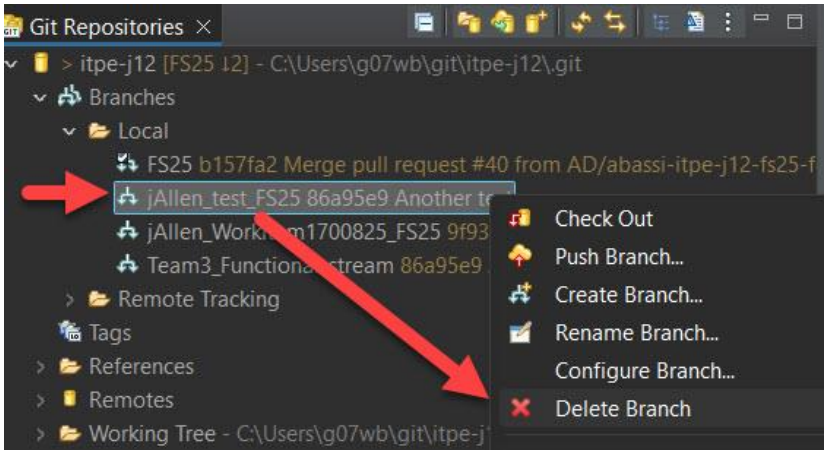
2. Delete your feature branch under the Remote Tracking folder, right click on it and select "Delete Branch"



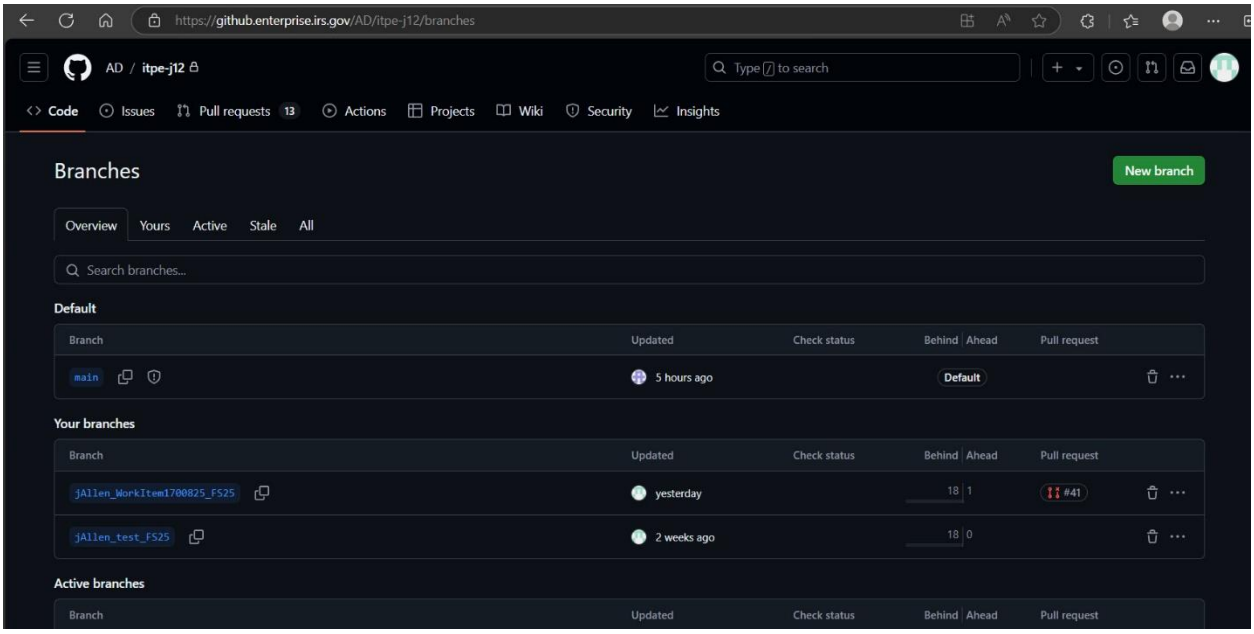
3. Right click on your local team branch and select "Check Out Branch" if you don't currently have it checked out. This is the branch that you used to create your feature branch



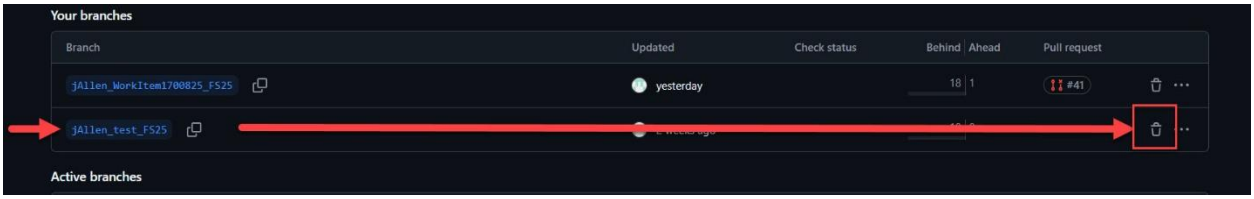
4. Right click on the feature branch you created and select delete branch



5. In your web browser, navigate to the branches page of your repository in GitHub. For example, if your repository is the itpe-j12 repository, you would navigate to GitHub ITPE J12 Branches. If you receive an error, you may need to sign in first by navigating to the base directory first - GitHub Homepage. After signing in, navigate to the branches page.



6. Under the section labeled "Your Branches," find the branch you want to delete and click on the trash can icon to delete it from GitHub. The Branch should now be safely deleted

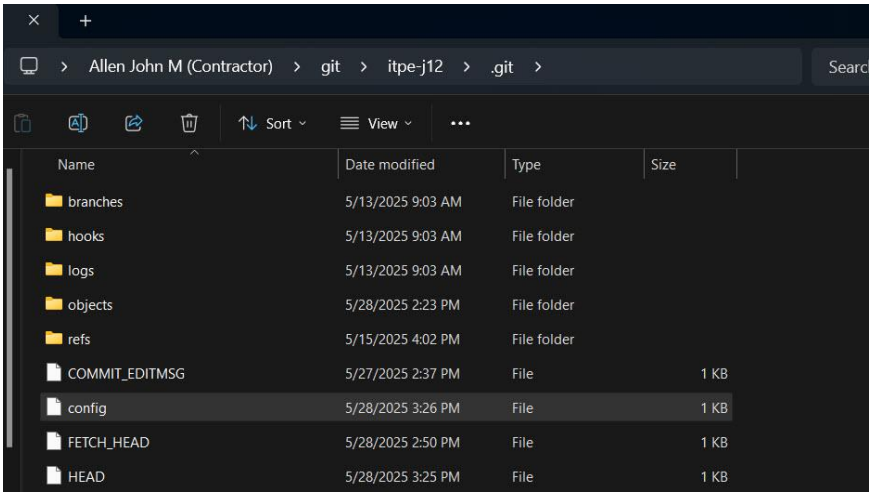


13.1. Troubleshooting Steps

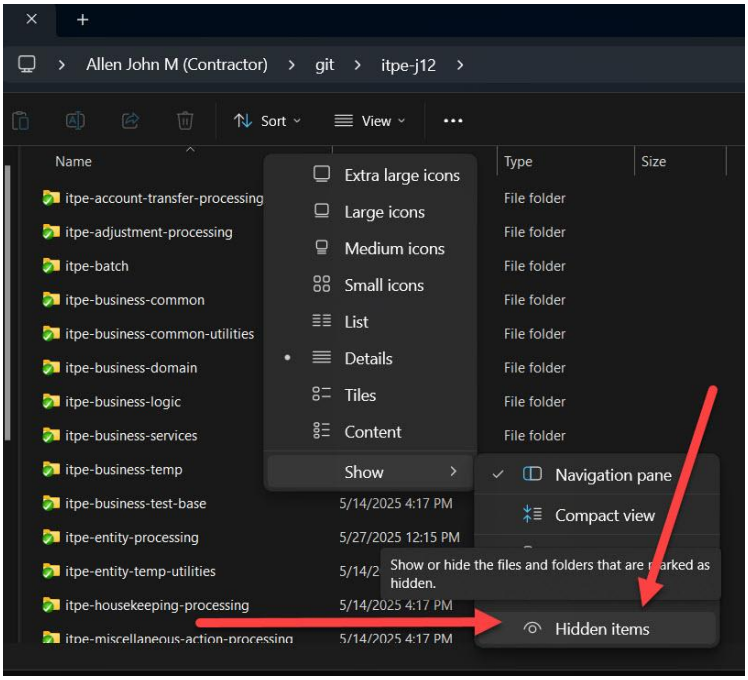
If you received the fetch error depicted in step 1d for deleting a branch, you will need to execute the following steps to fix Eclipse so it can continue pulling/fetching from GitHub:

1. Open windows explorer and navigate to the hidden git folder for your repository. It should be found in C:\Users\<seid>\git\<repository name>\.git. For example if your seid

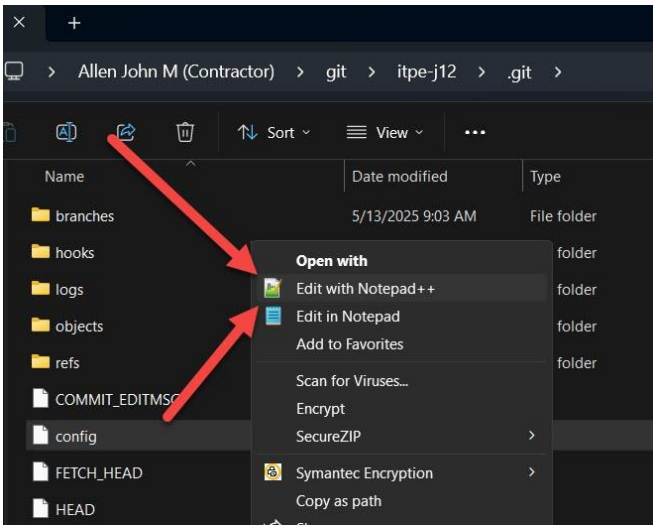
is m4bwy and your repository is itpe-j15, the appropriate path would be C:\Users\m4bwy\git\itpe-j15\.git



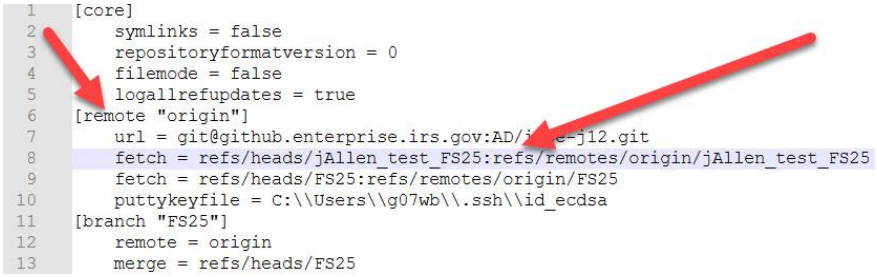
****Note: In order to see this folder, you will have to enable "Show hidden items." In Windows Explorer, click on the "View" menu just below the address bar, expand the "Show" menu at the bottom, and click "Show Hidden Items" if it doesn't already have a check next to it****



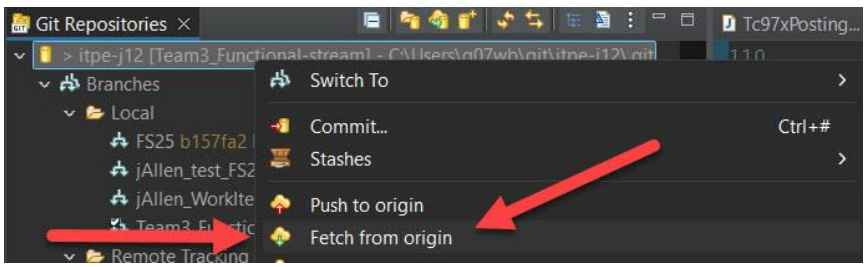
2. Right click on the file named "Config" and open in Notepad or Notepad++ (whichever text editor you prefer)



3. Under the "[remote "origin"]" header, find the line starting with "fetch = " that contains the name of the branch that you deleted from GitHub. Delete this line and save the file



4. In Eclipse, right click on the top-level repository and select "Fetch from Origin". You should no longer see the error. If you've followed these steps properly and still see the error, reach out to a coworker for help



14. Tortoise Basics

14.1. Overview

Tortoise is a graphical user interface for GIT that integrates directly with the Windows Explorer context menu. Whenever you right click on a folder that is part of a git repository (in other words, contains a hidden ".git" folder), you will be presented with numerous git operations within the Tortoise Git submenu. Additionally, tortoise git will normally display one of three status icons on a folder to indicate whether any uncommitted changes are still outstanding against the checked out branch within that folder or sub-folders, or whether there are any conflicted files in that folder or subfolders

There are several activities which are more convenient to execute using tortoise than with Eclipse. These include:

a. Viewing the history of commits and status of remote branches ("Show Log")
b. Resolving Conflicts
c. Aborting a merge
d. Viewing the history of a given file ("Blame")

14.2. Tortoise Folder Icons

Tortoise uses the following icons to represent the status of a folder and it's subfolders:

- **Green checkmark** – present working directory matches branch (no uncommitted changed)

- Red Exclamation mark – present working directory has uncommitted changes

- Yellow Triangle and Exclamation Mark – present working directory has merge conflicts
-

NOTE: Windows 11 might not display these icons.

15. Tortoise Show Log

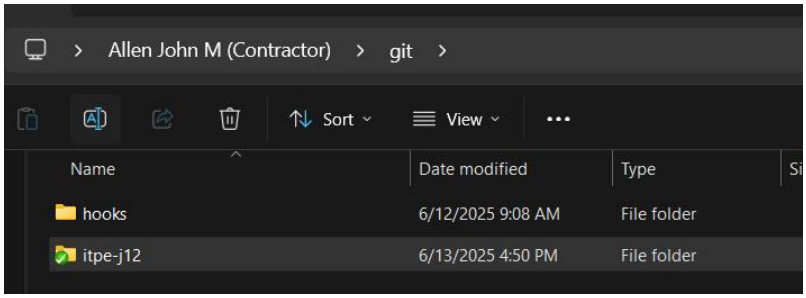
15.1. Overview

Tortoise's log provides much more functionality than Eclipse. So long as the "All Branches" check box is checked, it enables you to see the state of all fetched remote branches and compare them with their local counterparts to see if you need to pull changes. It also allows to

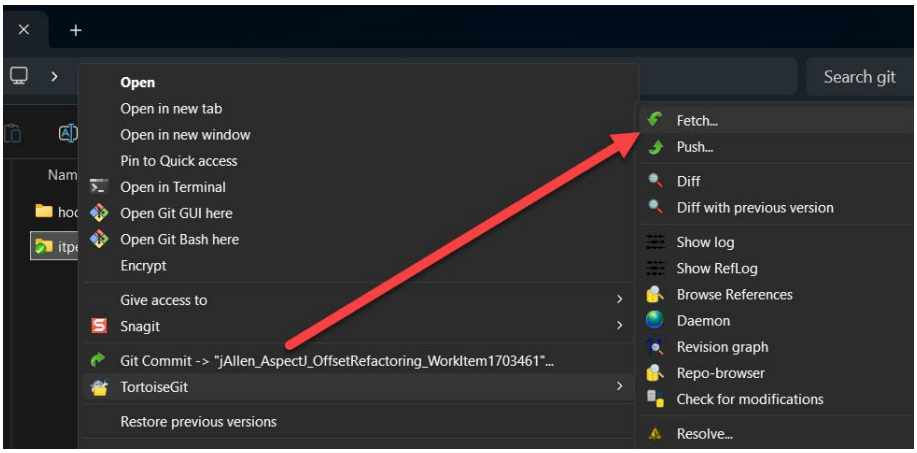
perform searches on author, branch name, or commit message. You can further restrict that search by constraining it to a range of dates

15.2. Steps to Show Log

1. Open Windows Explorer and navigagte to directory above your git repository, such that the top-level folder of your git repository is displayed. For most developers, this will be the directory C:\Users\<seid>\git

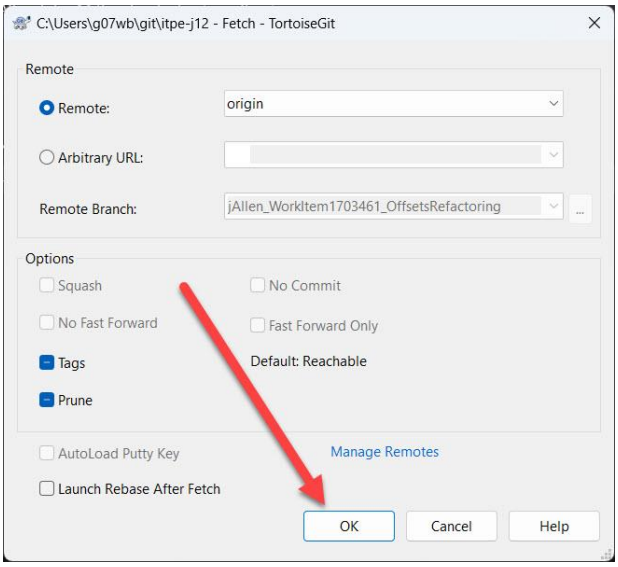


2. Right click on the top-level folder of your git repository. Hover over the “Tortoise Git” submenu and then select the “Fetch” option. If you’re using Windows 11, you might need to click “Show More Options” before the “Tortoise Git” submenu appears.

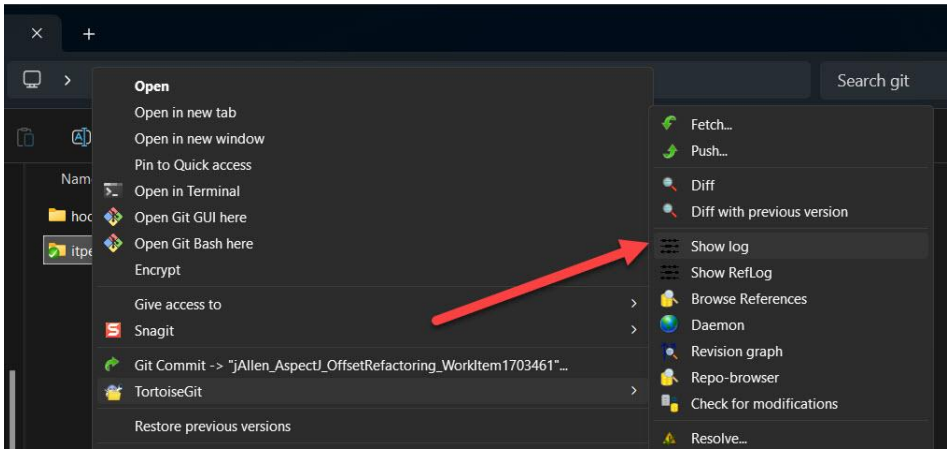


3. The Tortoise Fetch dialogue will appear. Click ok. This tells git to reach out to the remote servers to check if any updates have occurred since the last pull. Unlike performing a pull, however, “Fetch” doesn’t apply any changes to your local

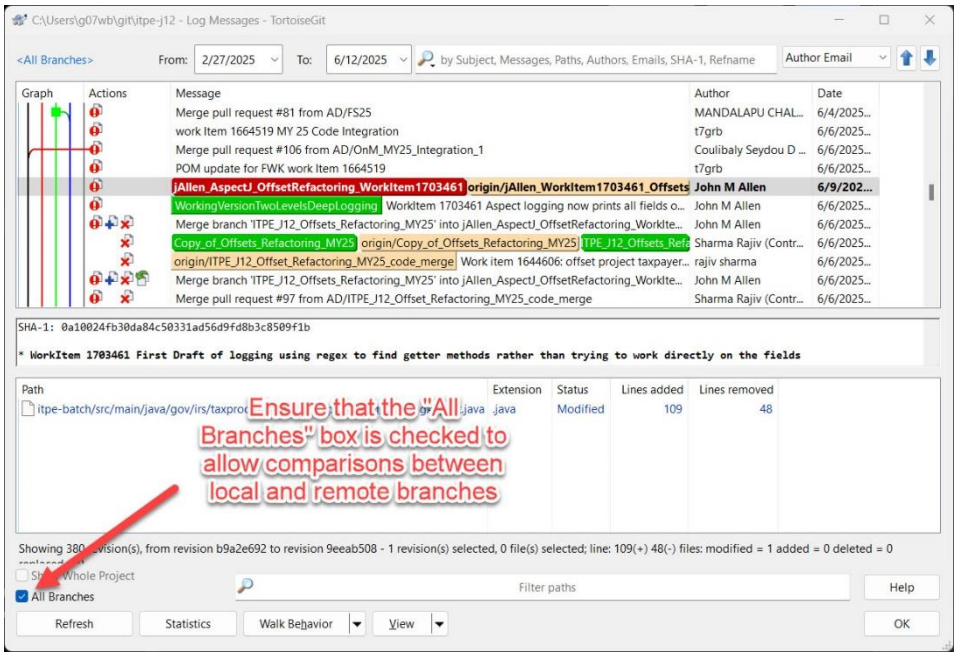
branches. If just gets the metadata so that everything displayed in the log is up to date



4. Right click on the top-level folder of your git repository. Hover over the “Tortoise Git” submenu and then select the “Show Log” option. If you’re using Windows 11, you might need to click “Show More Options” before the “Tortoise Git” submenu appears.



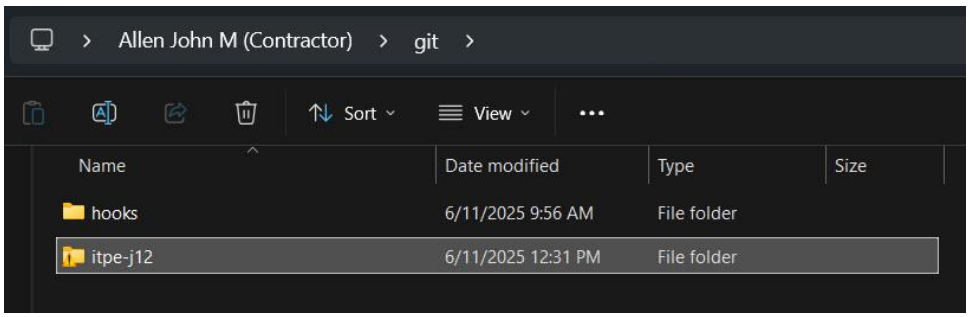
5. The Tortoise Log Dialogue should appear. Ensure that the “All Branches” check box is checked. This allows you to see if your local copy of the Team Branch is out of date with Github.



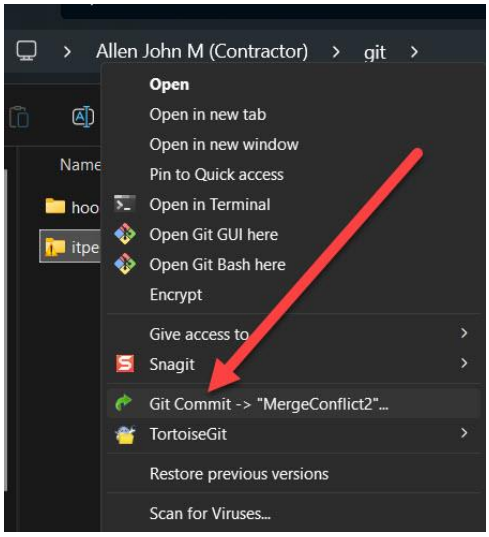
16. Resolving Conflicts with Tortoise

16.1. Steps for Resolving Conflicts

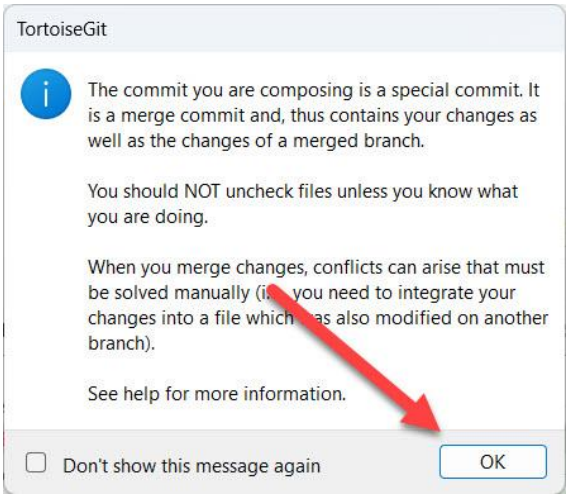
6. Open Windows Explorer and navigagte to directory above your git repository, such that the top level folder of your git repository is displayed. For most developers, this will be the directory C:\Users\<seid>\git



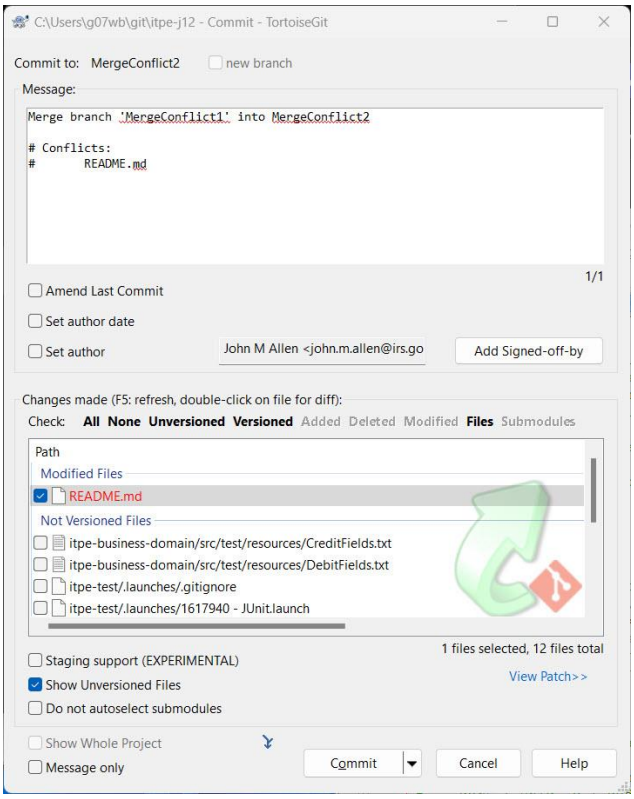
7. Right click on the top level folder containing your git repository and select the “Git commit”. If you are using Windows 11, you might need to click “Show More Options” before the Tortoise entries appear.**IMPORTANT: Refrain from actually committing anything during this process.** We are just using the commit dialogue as a convenient means of seeing all the conflicted files.



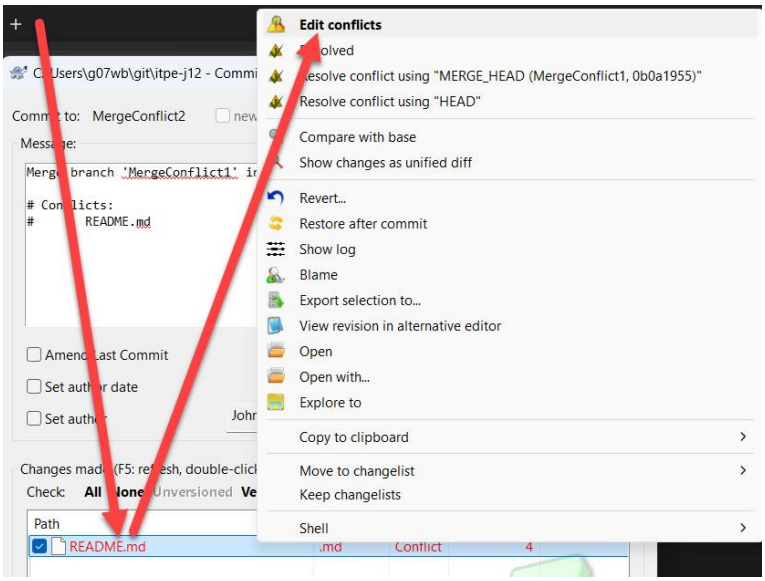
8. You may see a dialogue warning about the special nature of a merge commit, due to the conflicts. Click “OK” to close the dialogue



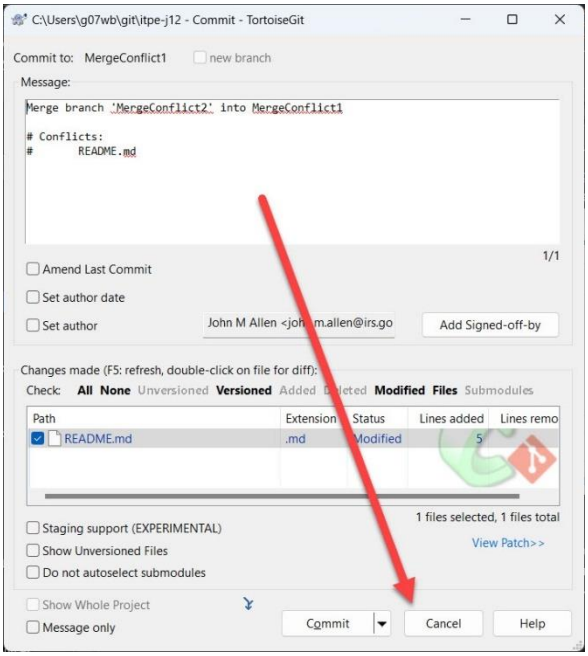
9. In the commit dialogue, you should see a list of all conflicted files in red font, and with the status “Conflict”



10. One by one, resolve the conflicts for each file in the “conflict” status. To do this, right click on the file and select “Edit Conflicts” See the Optionals Available for Resolving Conflicts sub-section for instructions on how to use the editor



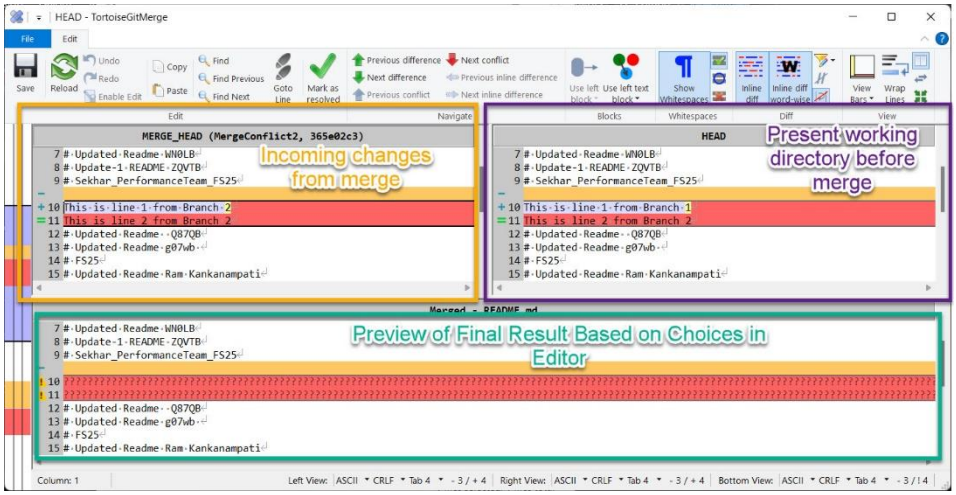
11. After all the files have changed from “conflict” to “modified” status (see the Saving File and Resolving Conflict Status on File sub-section for details on how to update the file status once all conflicts have been resolved). The merge is ready to be committed in Eclipse. Click “Cancel” on the Tortoise Commit dialogue.



12. Open Eclipse and commit the changes. See the Checking In Code Changes sub-section for instructions. Alternatively, if the conflicts were too difficult to resolve or you wish to create a branch to set a “backup” point before completing the merge, you can abort the merge. See the Abort Merge section for instructions

16.2. Overview of Conflict Editor

Just as with the tortoise log, Tortoise's conflict editor possesses much more functionality than Eclipse's conflict editor. Most noticeably, the conflict editor in Tortoise comes equipped with three frames instead of just two. The top two frames represent the incoming changes to be merged and the present working directory before applying changes respectively. NOTE: which side represents which may not remain consistent. Pay attention to the labels above each frame before making assumptions. The third frame on the bottom is a preview of the final result, based on the choices you've made in the editor.



Each line is color coded based on the type of change occurring. Lines with a white background represent lines in which no changes have occurred. Lines with an orange background represents some from of deletion between two different versions. Lines with yellow backgrounds represent additions between versions. Finally, red backgrounds on lines represents conflicts that haven't yet been resolved while green backgrounds represent resolved conflicts.

16.3. Options Available for Resolving Conflicts

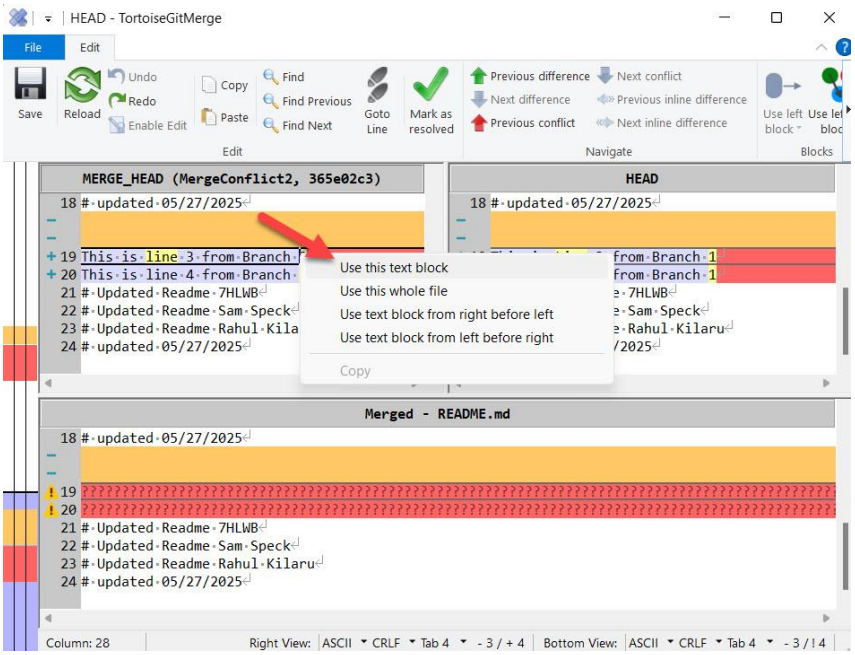
16.3.1. Overview

The editor allows you to perform changes in any of the three frames. You select a line by left clicking it. This will put a black outline around your selection. From there, you can tell it one of the following options:

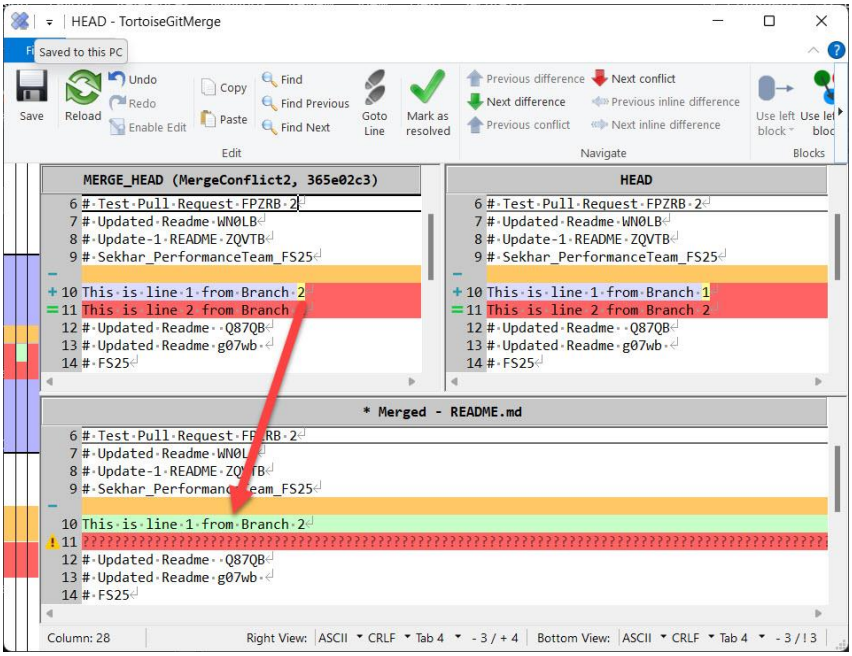
16.3.2. Selecting “Use this text block” from left or right frame

This option chooses the selected line, or block of lines, and chooses it over it’s counterpart from the left or right frame. See the following screenshots for examples of what this would look like:

Choosing Left Line Over Right Line

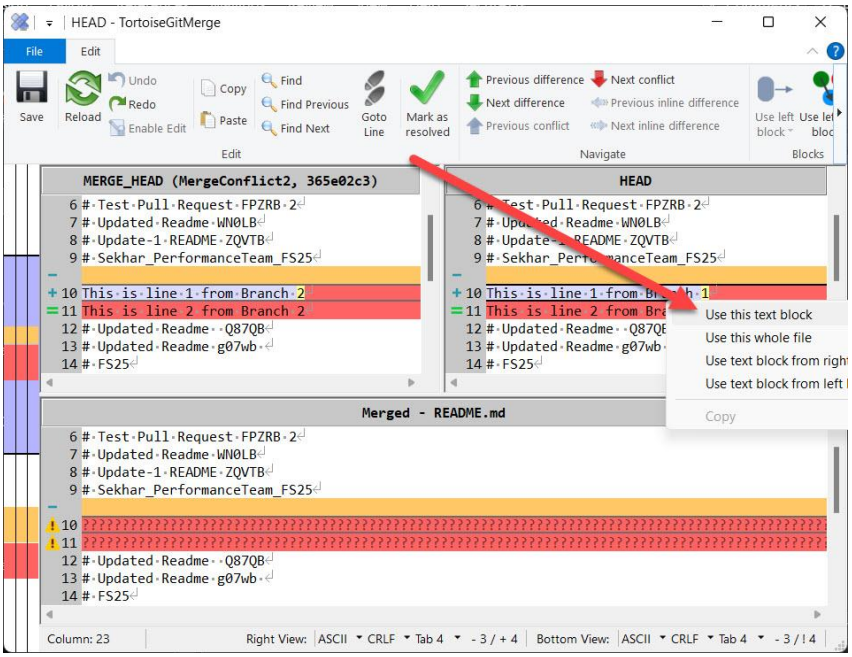


Select single line on the left and right click. Then click on “Use this text block” option

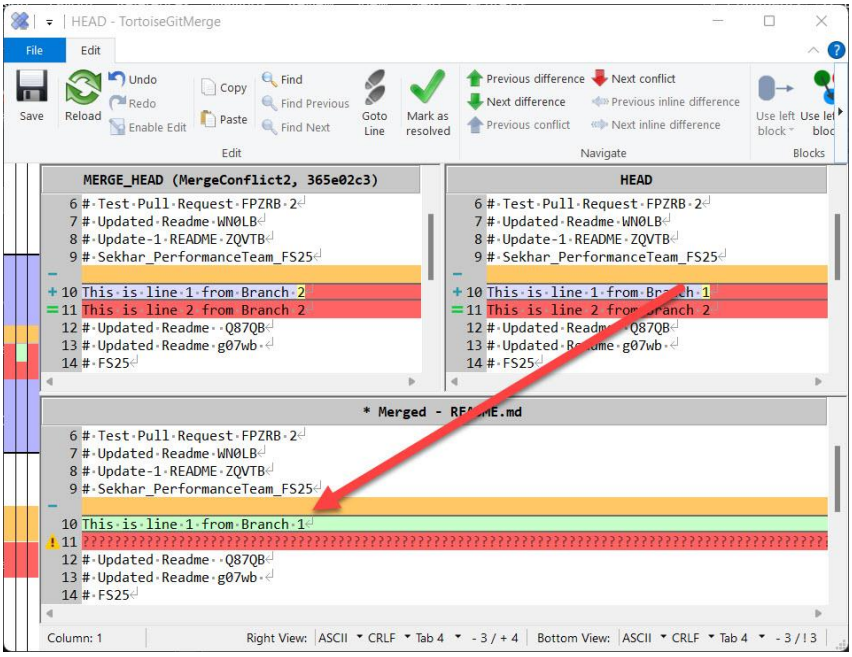


Result of Using the Line from the left frame

Choosing Right Line Over Left Line



Select single line on the right and right click. Then click on "Use this text block" option

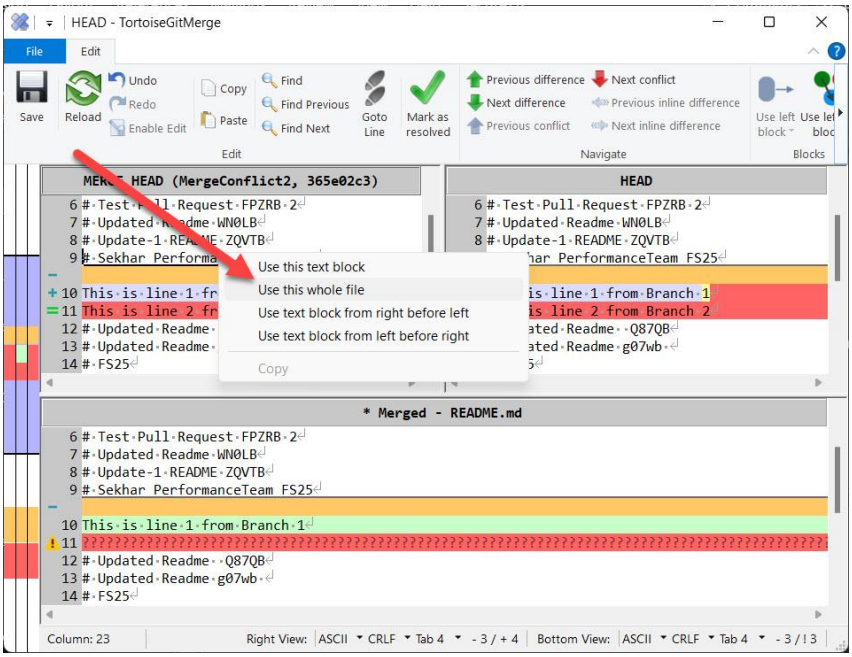


Result of Using the Line from the right frame

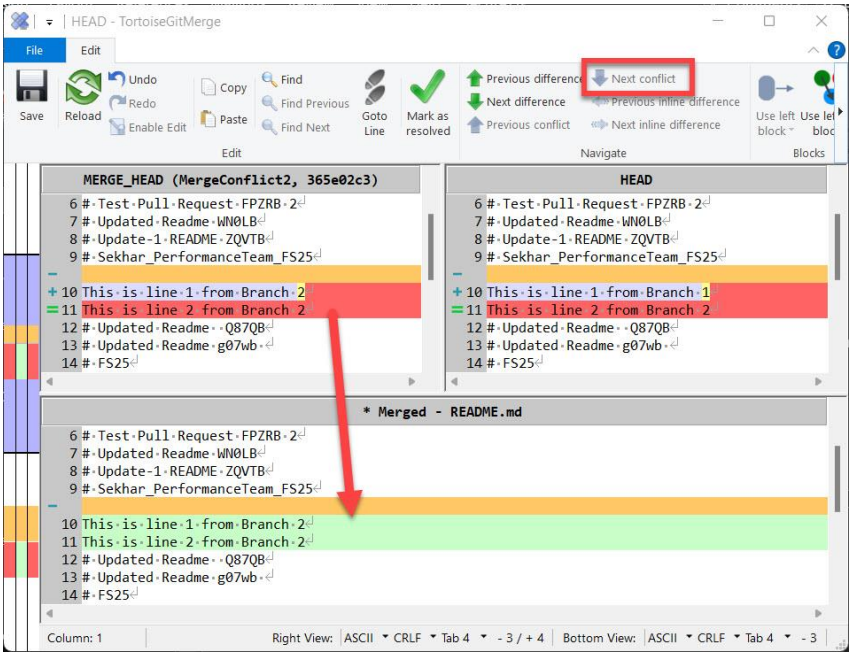
16.3.3. Selecting "Use this whole file" from left or right frame

This option is fairly self-explanatory – it resolves every conflict in the entire file by using the values in the left frame.

Choosing Left File Over Right File

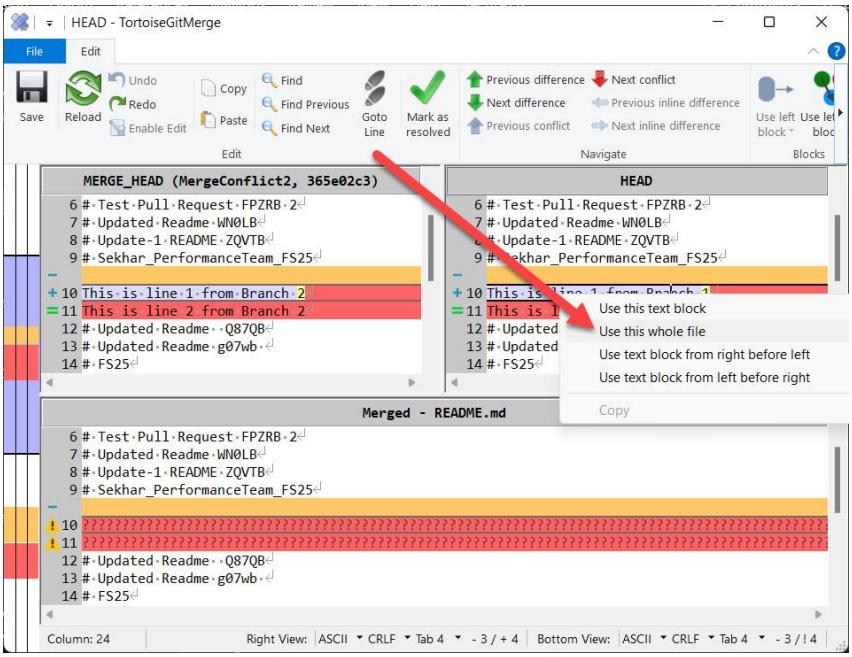


Selecting whole file from left frame

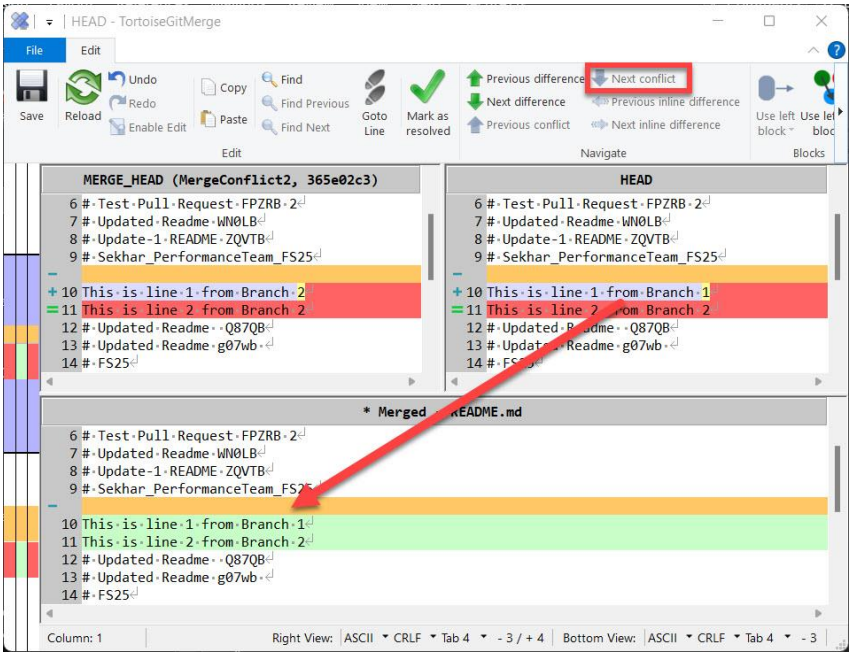


Result of selecting whole file from left frame

Choosing Right File over Left Life



Selecting whole file from right frame

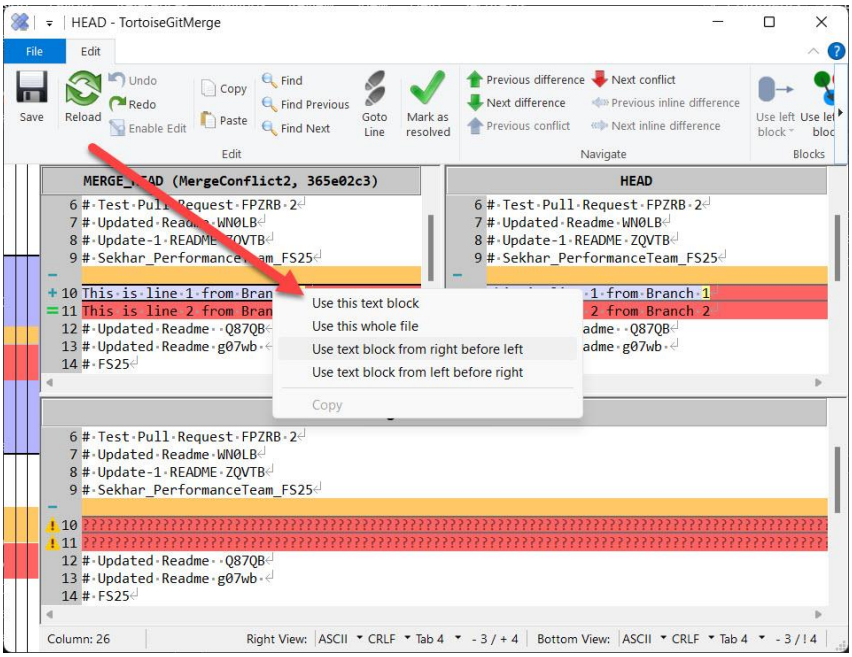


Result of selecting whole file from left frame

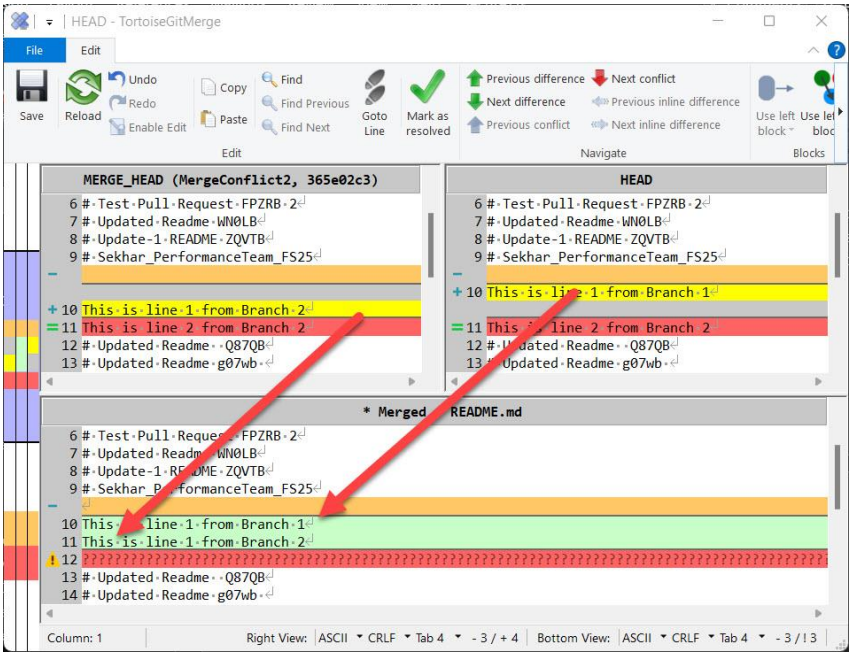
16.3.4. Selecting Use One Block Before the Other

Tortoise allows you the option to use both contending lines in the final result, you just need to specify whether to use the right line first or the left line first. Keep in mind, you can also select multiple lines at once. In that case, you would use all of the selected lines from one side before using all of the selected lines on the other side.

Selecting Right Line Before Left

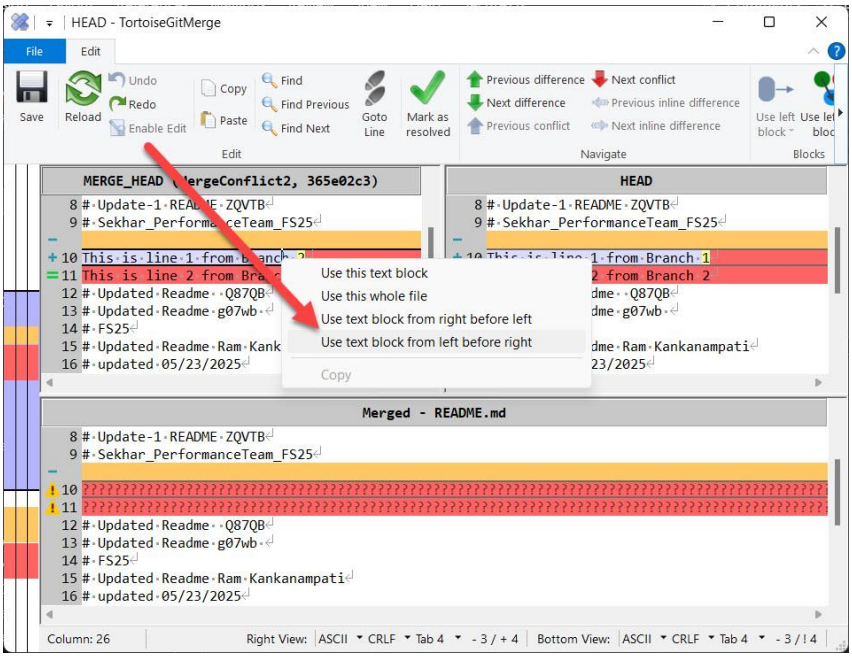


Selecting to use right line before using left line

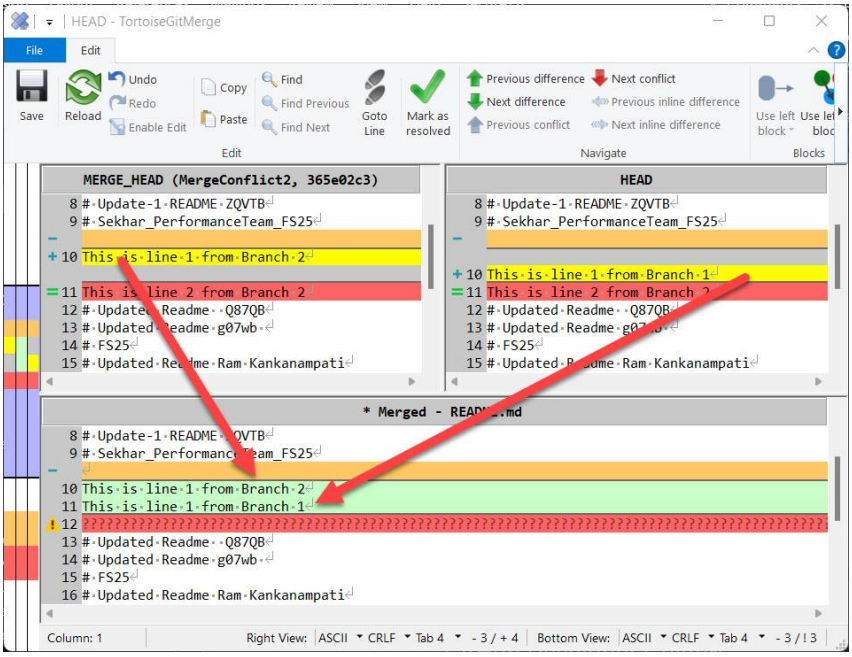


Result of using right line before using left line

Selecting Left Line Before Right



Selecting to use left line before using right line

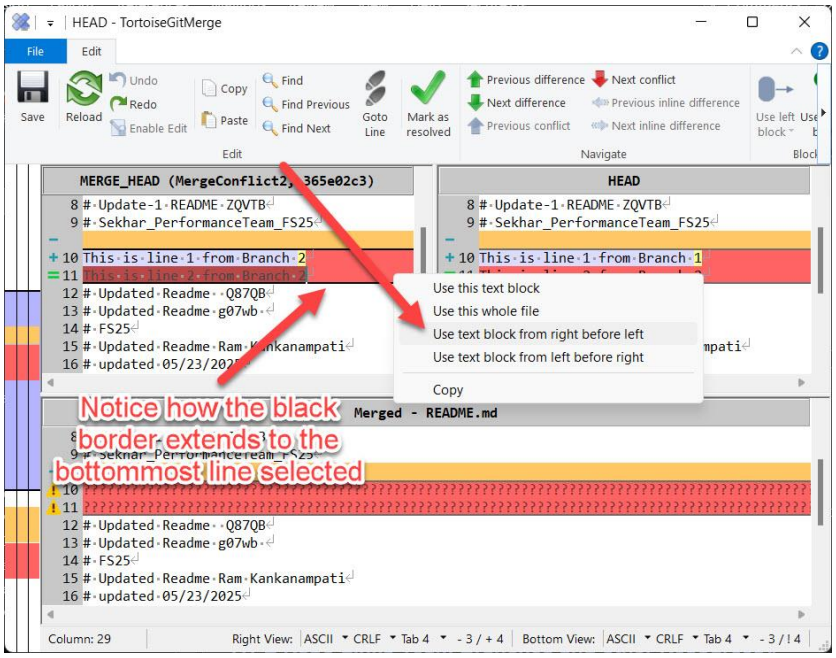


Result of using left line before using right line

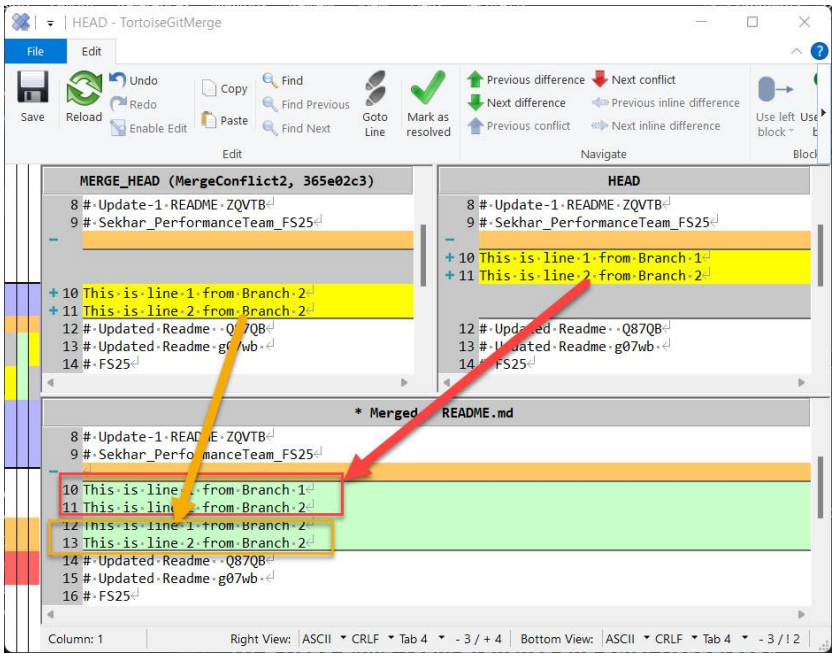
Selecting an Entire Block of Lines To Use One Before the Other

To select multiple lines, either click and drag across all desired lines or hold SHIFT while left clicking each desired line. After selecting an entire block, choosing

whether to use the left block first or the right works exactly as in the previous sub-section.



Selecting an entire block of lines to use right before left

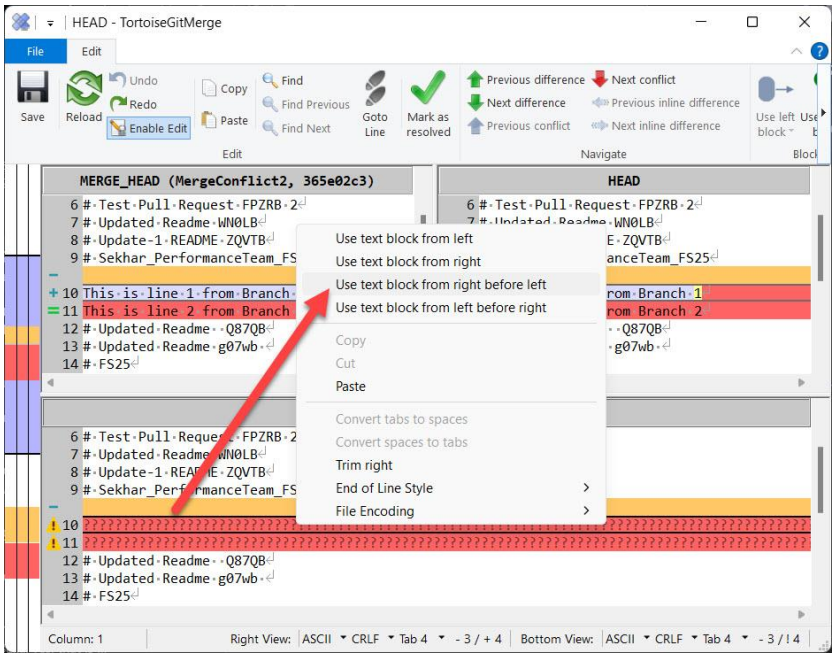


Result of using an entire block of lines – right before left

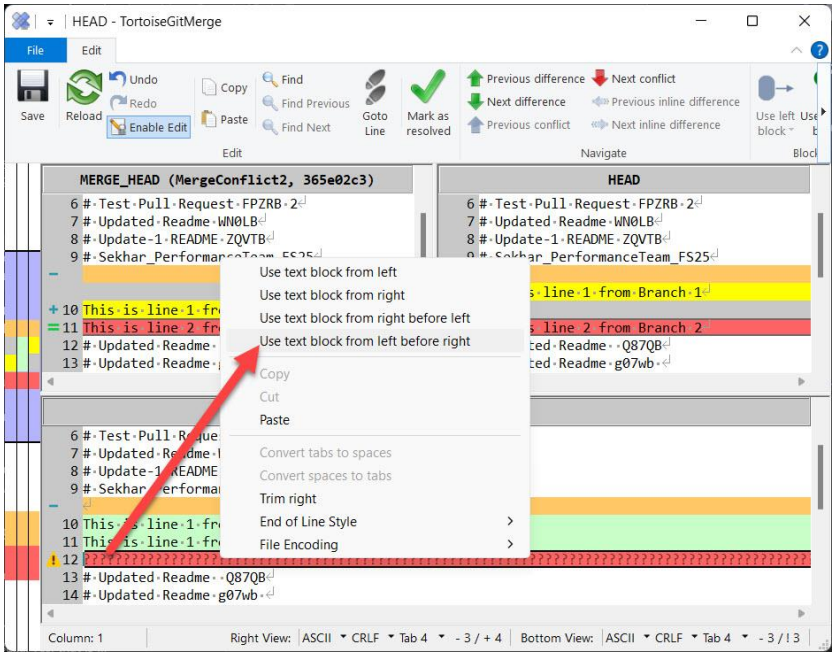
16.3.5. Using Preview Frame

The preview frame on the bottom offers the same options as in the top. Additionally, it allows you to type or copy and paste brand new changes directly into it, so long as “Enable Edit” is

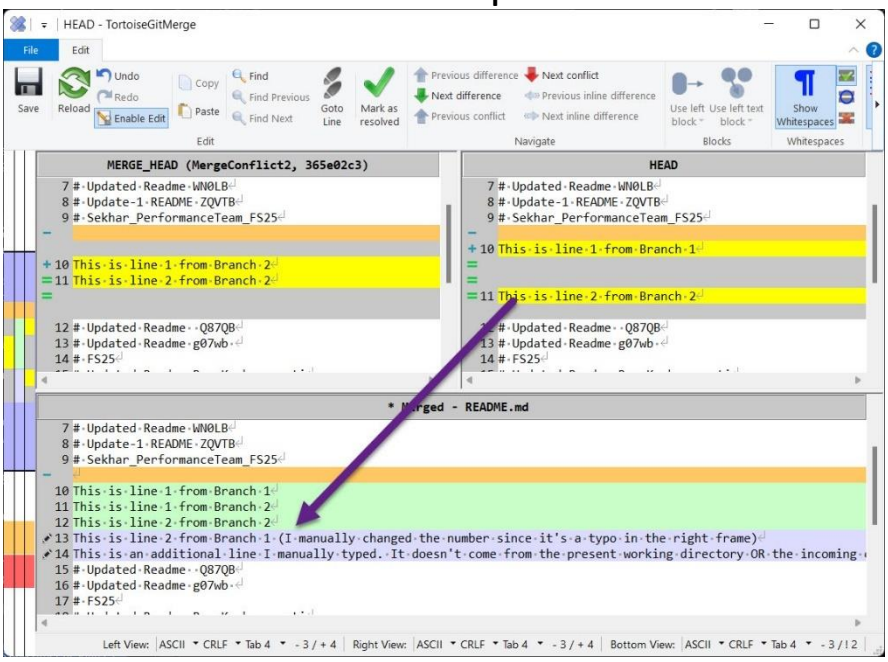
active. The following screenshots demonstrates executing various actions one after another, using the preview pane.



Using preview pane to chose first line from right before left



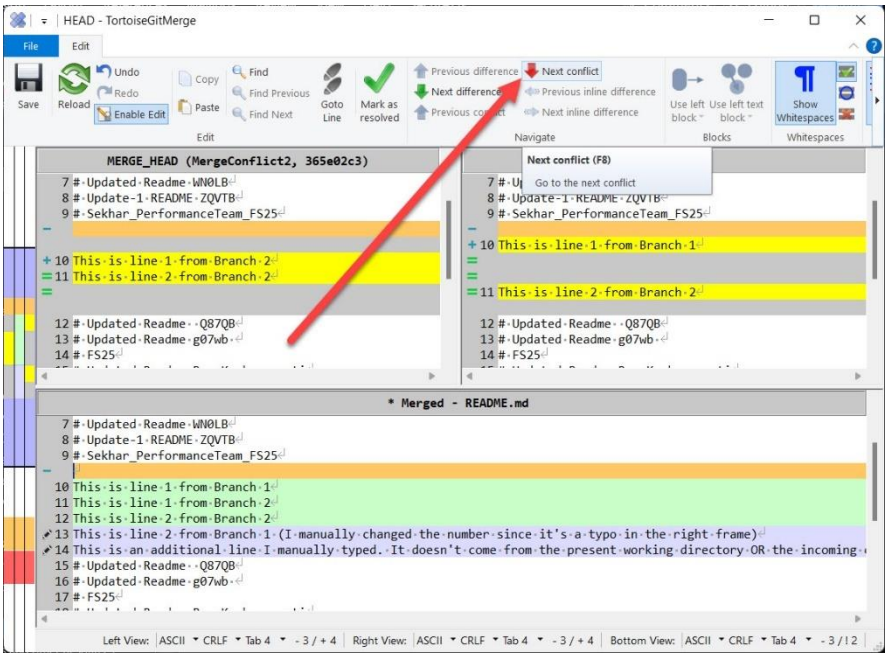
Using preview pane to choose second line from left before right



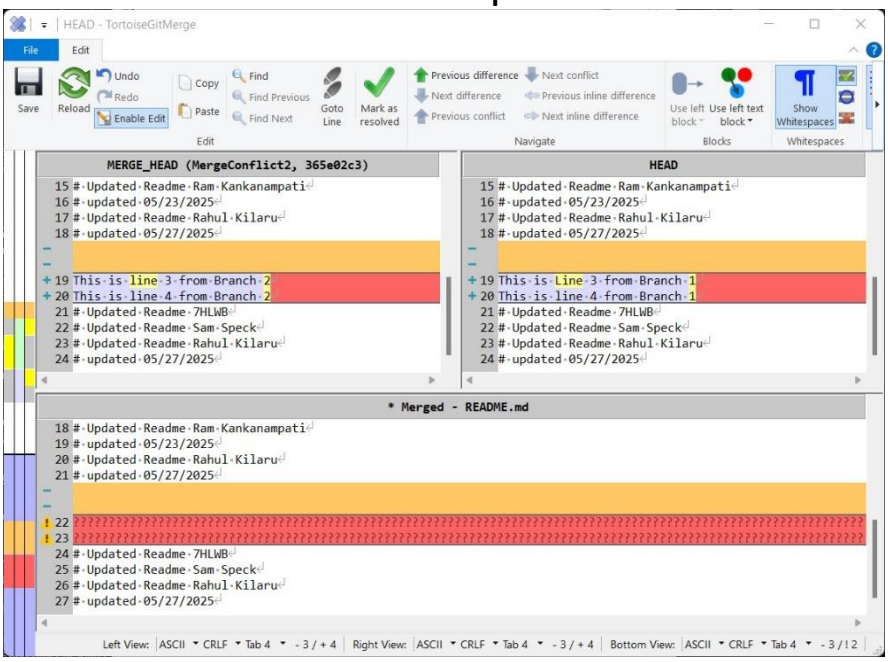
Using Preview Pane to manually enter new text

16.3.6. Go to Next Conflict

Tortoise offers a convenient button that will skip past all regular differences and immediately scroll to the next set of conflicting lines in the file.



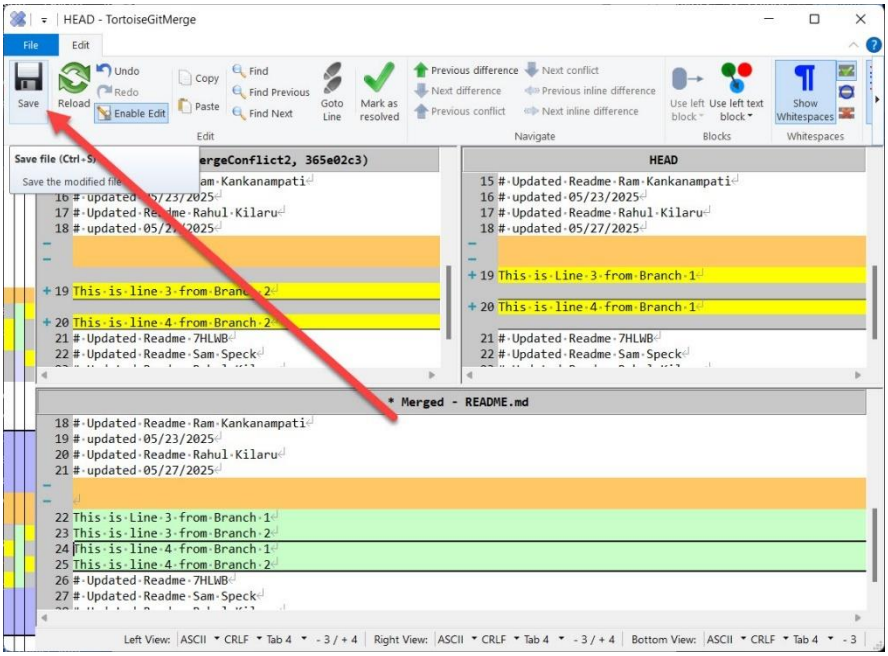
Clicking “Next Conflict”



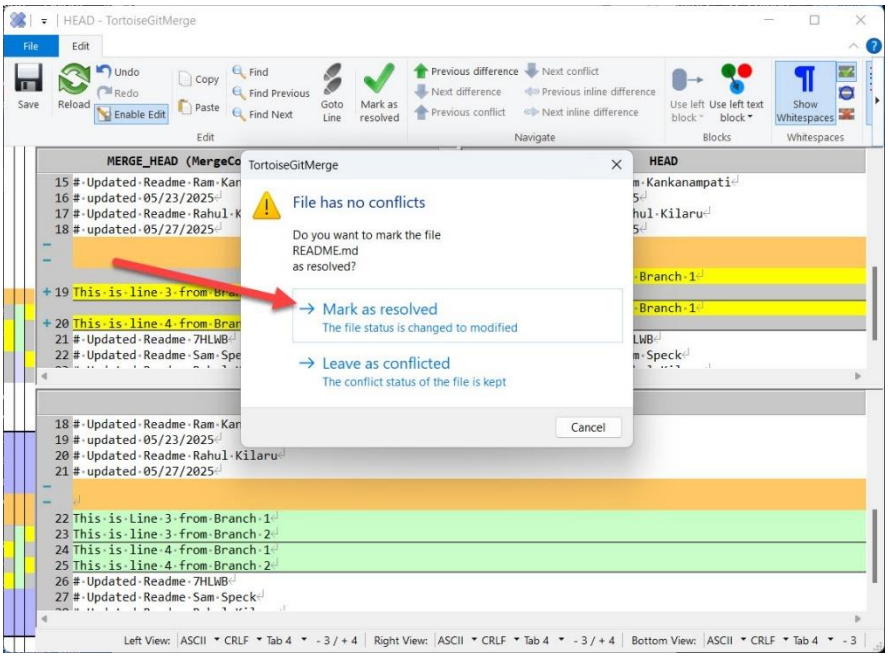
Result of clicking “Next Conflict”

16.3.7. Saving File and Resolving Conflict Status on File

Once you’ve resolved every conflict in the file, click on the save button in the upper left.

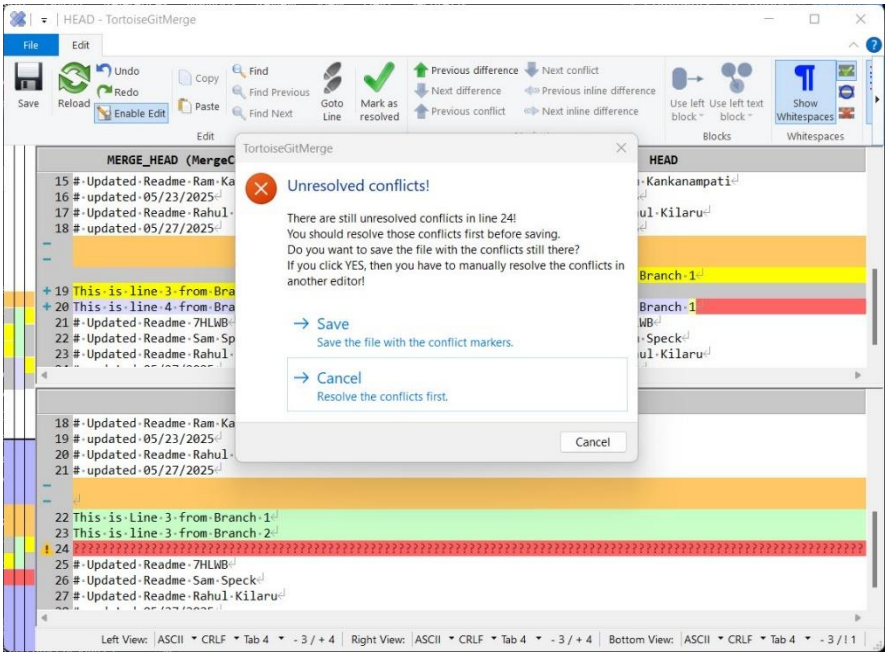


After clicking the save button, you will be presented with a pop up window. If you have successfully resolved all the conflicts in the file, you will be presented with a window prompting you to resolve the conflict status on the file. Select that option.



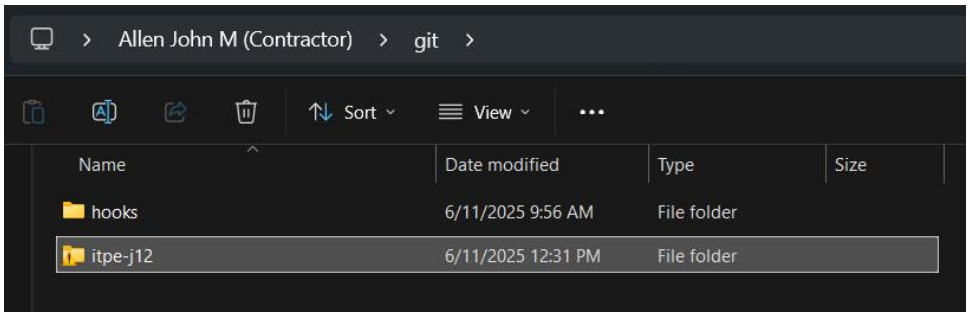
Saving and Marking File as Resolved

If you have missed any conflicts in the file, you will receive a window notifying that there are remaining lines that are unresolved. You can choose to save the file if you wish to preserve any choices you've already made. Keep in mind that this still won't resolve the conflict status on the file. You will need to return later to resolve the conflicts.

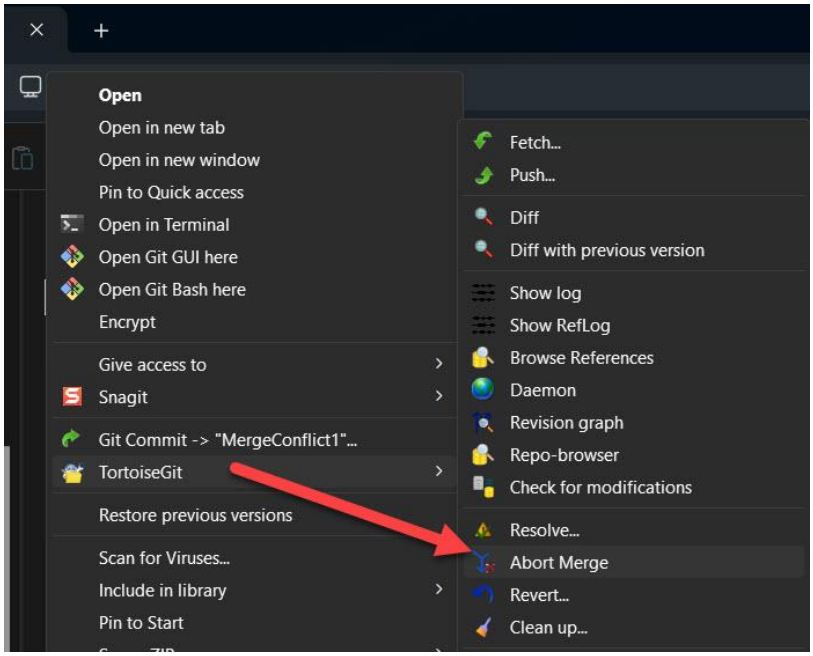


17. Aborting a Merge with Tortoise

1. Open Windows Explorer and navigagte to directory above your git repository, such that the top level folder of your git repository is displayed. For most developers, this will be the directory C:\Users\<seid>\git

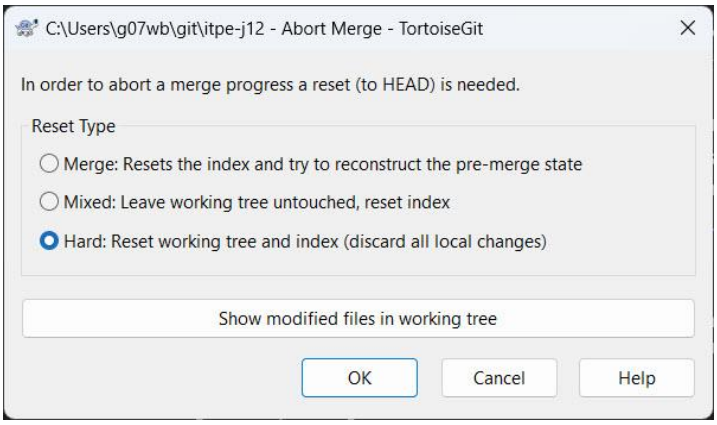


2. Right click on the top level folder of your git repository. Hover over the “Tortoise Git” submenu and then select “Abort Merge” For Windows 11 users, you may need to click “Show More Options” before being able to see Tortoise commands.

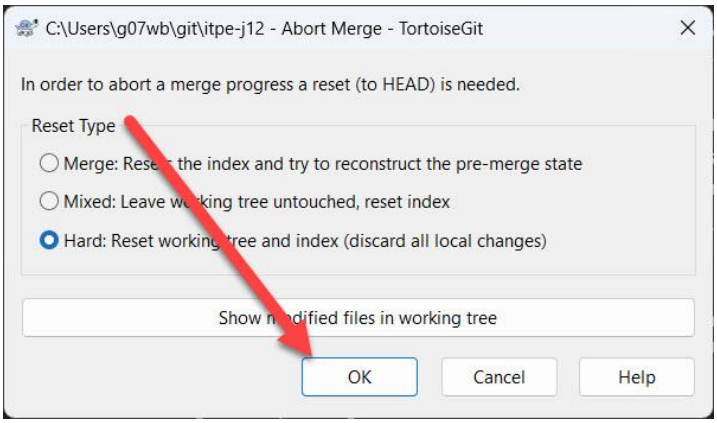


3. You will be presented a dialogue box with three options. Make a selection according to the following guidelines:
   - The cleanest option is to choose “Hard,” and thus it’s the recommended option. This option resets the present working directory to the state of your local checked out branch. You won’t have to worry about any of the incoming changes accidentally getting kept. On the other hand, you will also lose any uncommitted local changes that you forgot to stash.

- Select “Merge” if you have local changes that were not stashed prior to attempting your merge. Note that tortoise might not be able to untangle the merged change properly. In that case, you may need to revert all the changes and manually recreate them
- Avoid the “Mixed” option. If you successfully merged conflicts, then you should simply commit that merge and then add any new changes afterwards.



4. Click OK. The merge should be aborted and you can resume normal GIT operations



18. Special Instructions for the ALC Baselines Repo

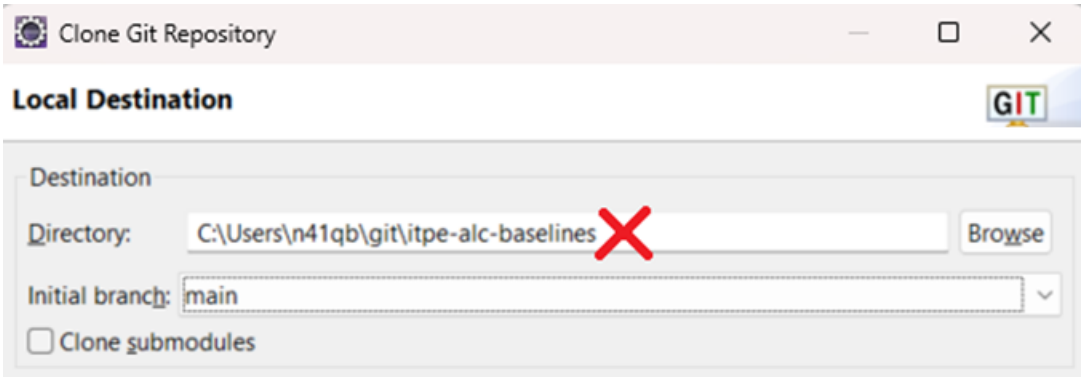
If you intend to utilize the macros in the MS Excel Listings, please make sure that you clone into one of the following folders (or a subfolder of them):

- C:\Users\SEID\UserApps SBU Data
- C:\Users\SEID\Documents\SBU Data
    - Note: If you choose this folder, please make sure that it is not your OneDrive folder (i.e. C:\Users\SEID\OneDrive - Internal Revenue Service\Documents)

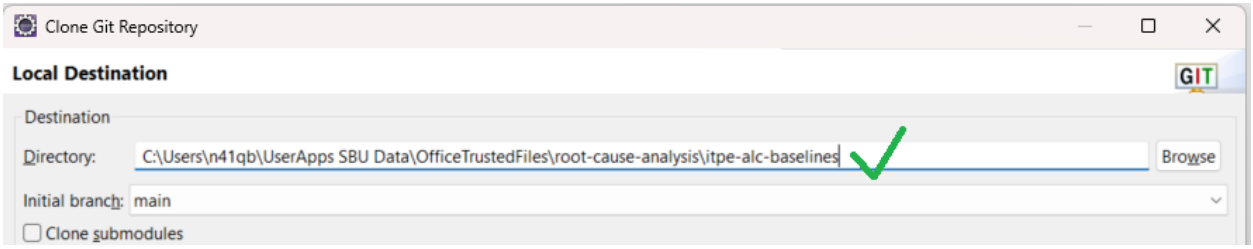
If you do not utilize MS Excel Listings (or just open them without using the macros) you can skip the following steps. If you do use them, please follow the instructions below as failure to do so will cause macros to run extremely slowly since the files will be stored outside of a Microsoft “Trusted Location”.

By default., git will attempt to clone into C:\Users\SEID\git folder. To avoid this, you can update the clone destination during the clone setup process via the “Local Destination” dialog box (which I believe is the 3rd dialog box). I.e.

When prompted with this:



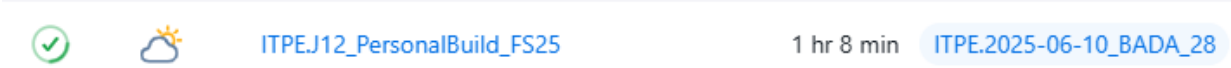
Update the directory to something along the lines of this. All that matters is that the clone is being done to the UserApps SBU Data/SBU Data folder or a subfolder.



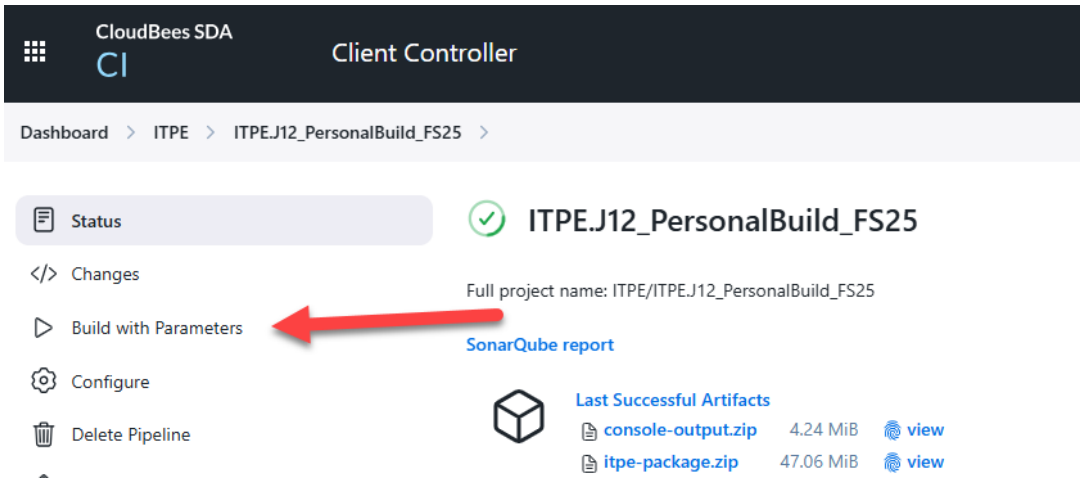
19. Running a Personal Build

The Jenkins pipelines for ITPE can be found HERE

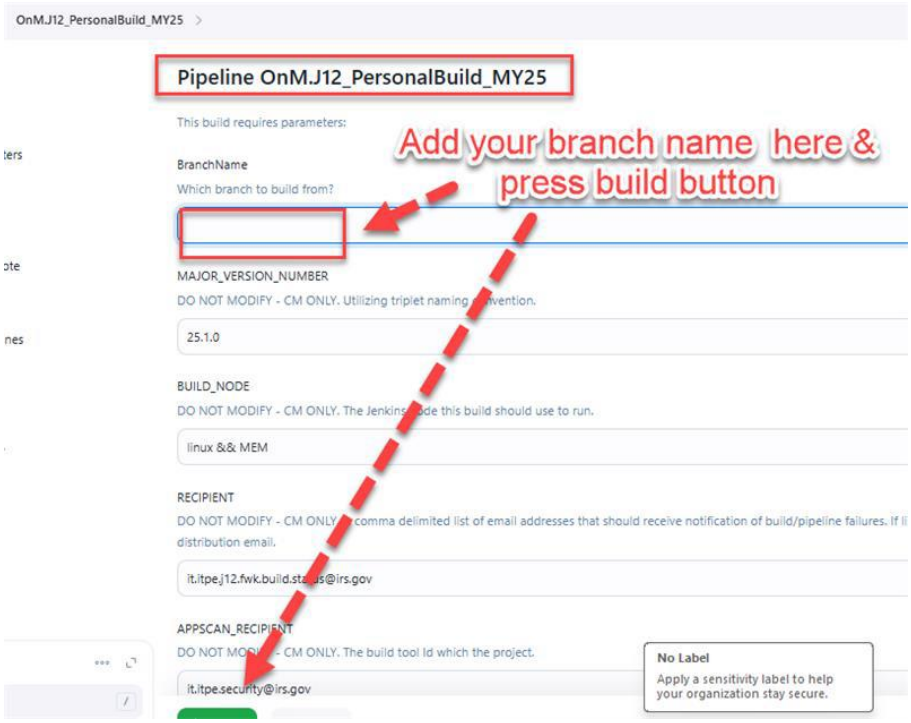
1. Click on the appropriate Personal Build Pipeline (e.g. ITPE.J12_Personal Build _FS25)



2. On the pipeline page select “Build with Parameters”



3. On the following screen type in your Branch name and click Build



20. ITPE Repo Links Team Branches and other Resources

FWK: git@github.enterprise.irs.gov:AD/itpe-fwk.git

C-Mod: git@github.enterprise.irs.gov:AD/itpe-c-module.git

CBP: git@github.enterprise.irs.gov:AD/itpe-cbp.git

J12: git@github.enterprise.irs.gov:AD/itpe-j12.git

J15: git@github.enterprise.irs.gov:AD/itpe-j15.git

RCAS: git@github.enterprise.irs.gov:AD/Root-Cause-Analysis-Service-By-Deloitte.git

ALC Baselines: git@github.enterprise.irs.gov:AD/itpe-alc-baselines.git

BB-Writeups:https://github.enterprise.irs.gov/AD/itpe-building-block-documentation.git

Team 1 FS25 Branch: Team1-FS25-Functional

Team 2 FS23 Branch: Team2-FS25-Functional

GitHub Enterprise Onboarding:

GitHub-Enterprise_ onboard.pdf

Git Cheat Sheet:

Adobe Acrobat Document