Git

Clinton Walker

January 29, 2020

Overview

What is Configuration Management?

Software Configuration Management:

- Tracking changes
- Setting baselines/requirements
- Tracking configuration/versioning

Components of SCM:

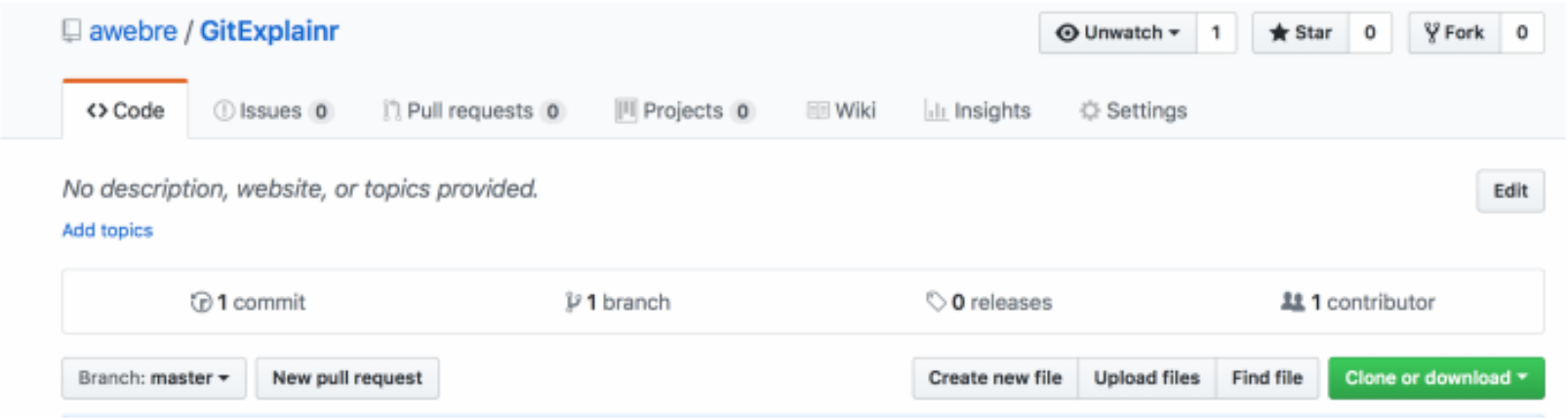
- Version Control
- Build Tools
- Issue Trackers

The ironic .gitignore

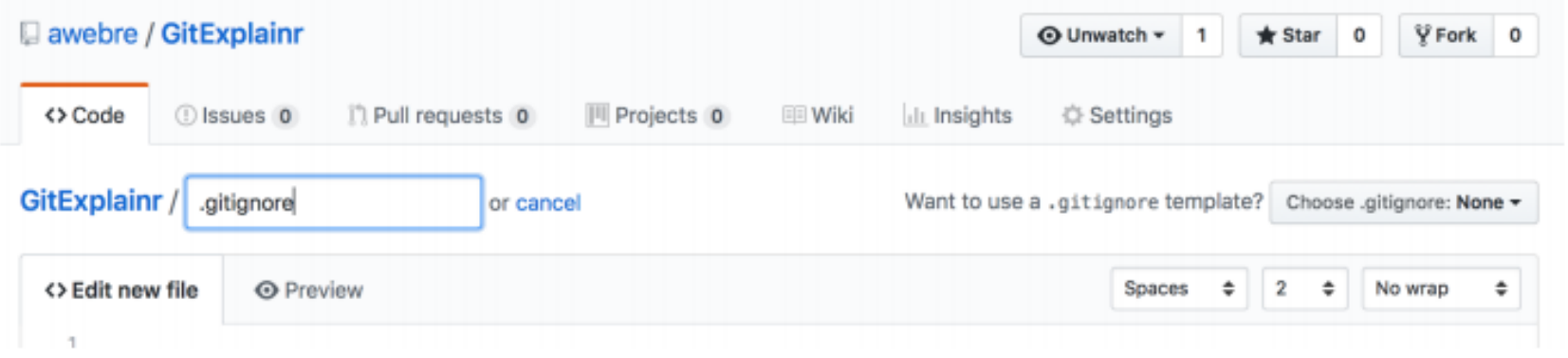
Despite having *ignore* in the name, the .gitignore is a very critical part of using git. This is what keeps track of what files and folders are not included in the repository. Typically these are things like machine specific settings that need to be generated. If a developer pulls down a project with machine specific files already included, there is a high chance that the project will not run. It is also used to exclude folders like the node_modules folder in a Node.js project. Folders like these should be generated on a machine after the developer clones the project. The next few slides will show the steps in creating a .gitignore.

Creating a .gitignore

The top of your Github page has a button called "Create new file" you can use to make a .gitignore. This is one of few cases where using Github directly is acceptable.



If you type ".gitignore" into the file name, it will ask if you want to generate an ignore file through Github's templates. Keep in mind it will only let you generate a template for one technology here.



Another option for generating a .gitignore is `https://www.gitignore.io/`. This will let you generate a template that includes multiple technologies.

Basic Operations

- Clone: Cloning a repository copies it from the remote source to a local machine. By default your local directory will checkout master if it exists.
- Checkout: Checking out a branch "moves" it to your local directory where you cloned the repository. Checking out a different branch essentially rotates the current branch you're on out of the directory and puts the new one in.
- Stage: Staging lets you select which modified files you want to put into a commit.

Basic Operations cont.

- Commit: Your staged changes will take effect in your local repository once you commit them. These changes are still local to your machine.
- Push: Pushing changes that you have committed will make them appear in the remote repository, where you team members can see them. This command pushes all changes that have been committed on all branches.
- Fetch: Fetch retrieves all changes made remotely to a branch. It is important to check for changes after checking out a branch and before merging.
- Pull: Pulling allows you to pull the changes for whatever branch you are currently on, or pull a new branch you have not checked out before.

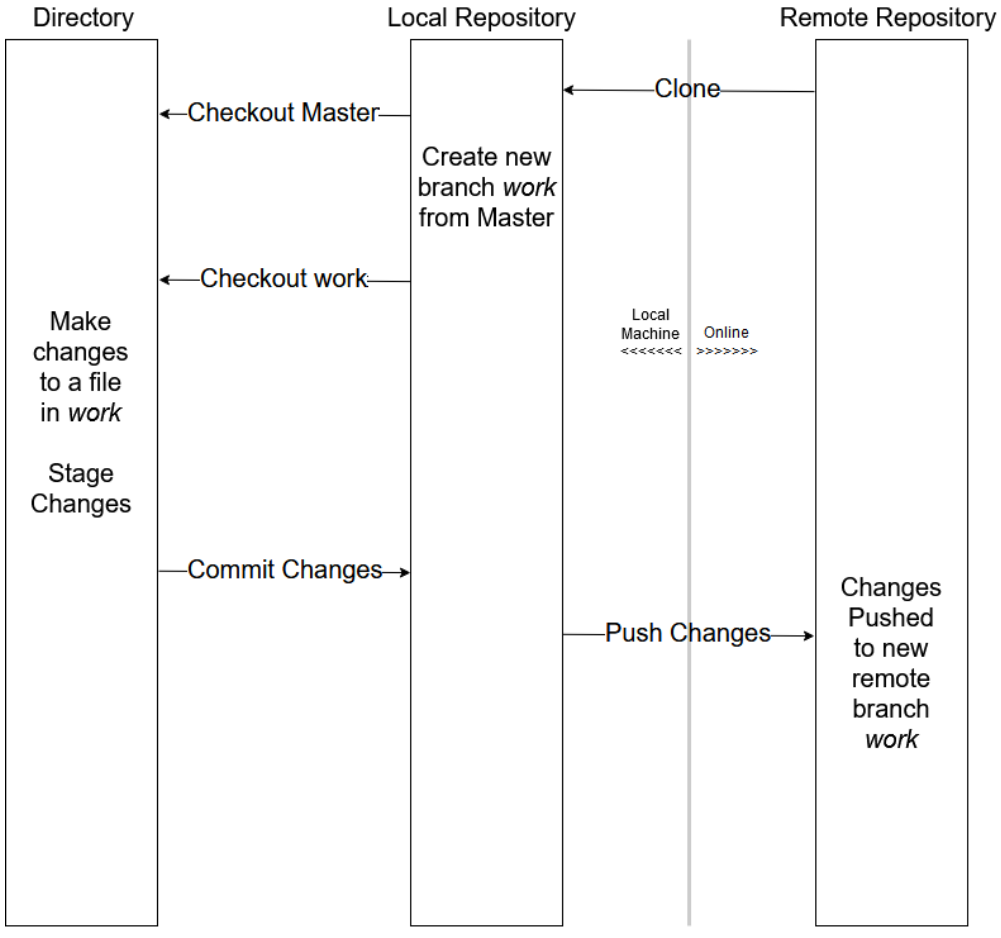


Figure: The Full Diagram

Cloning

On Github we can see the green "Clone or download" button. We will copy the HTTPS link here and use our git client to clone the repository.

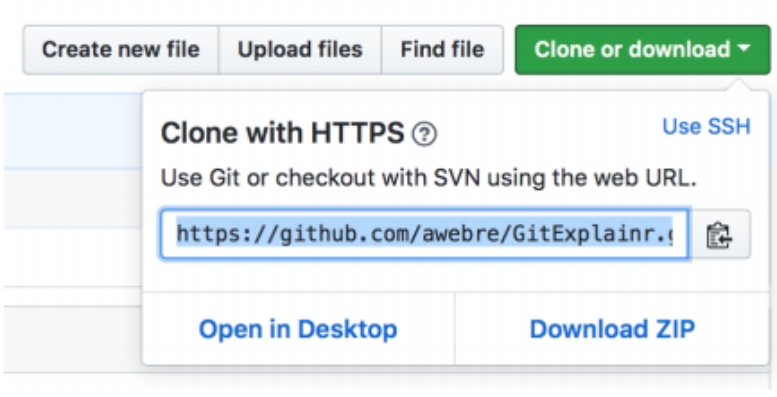
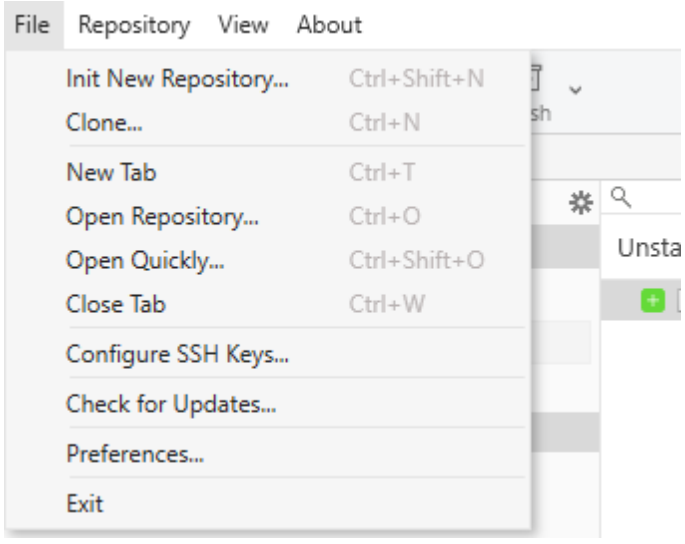


Figure: Github Clone Link

Now we head to our git client and paste this link into our clone popup.



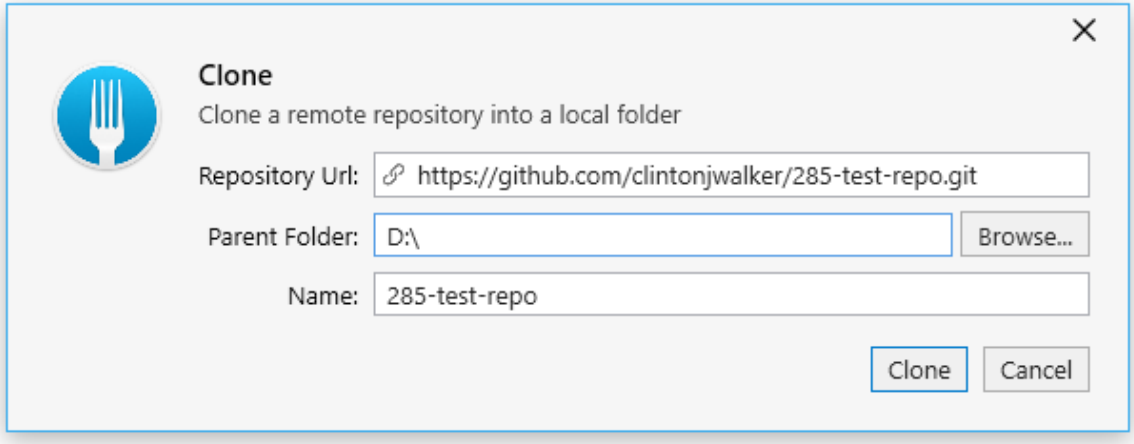


Figure: Fork Clone Dialog Box

With a new repository, we get something that looks like this.

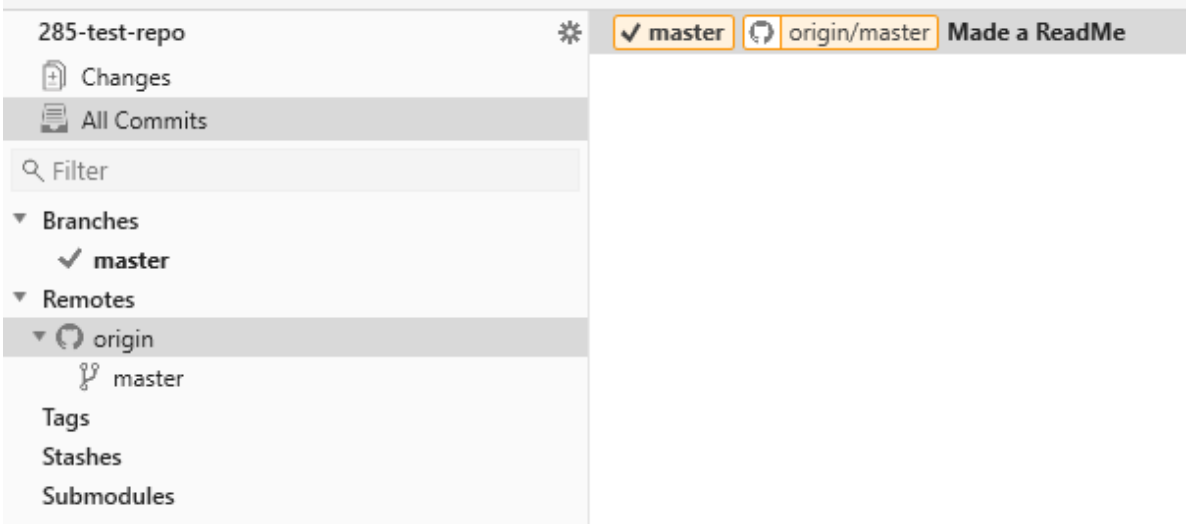


Figure: An Almost Empty Repository

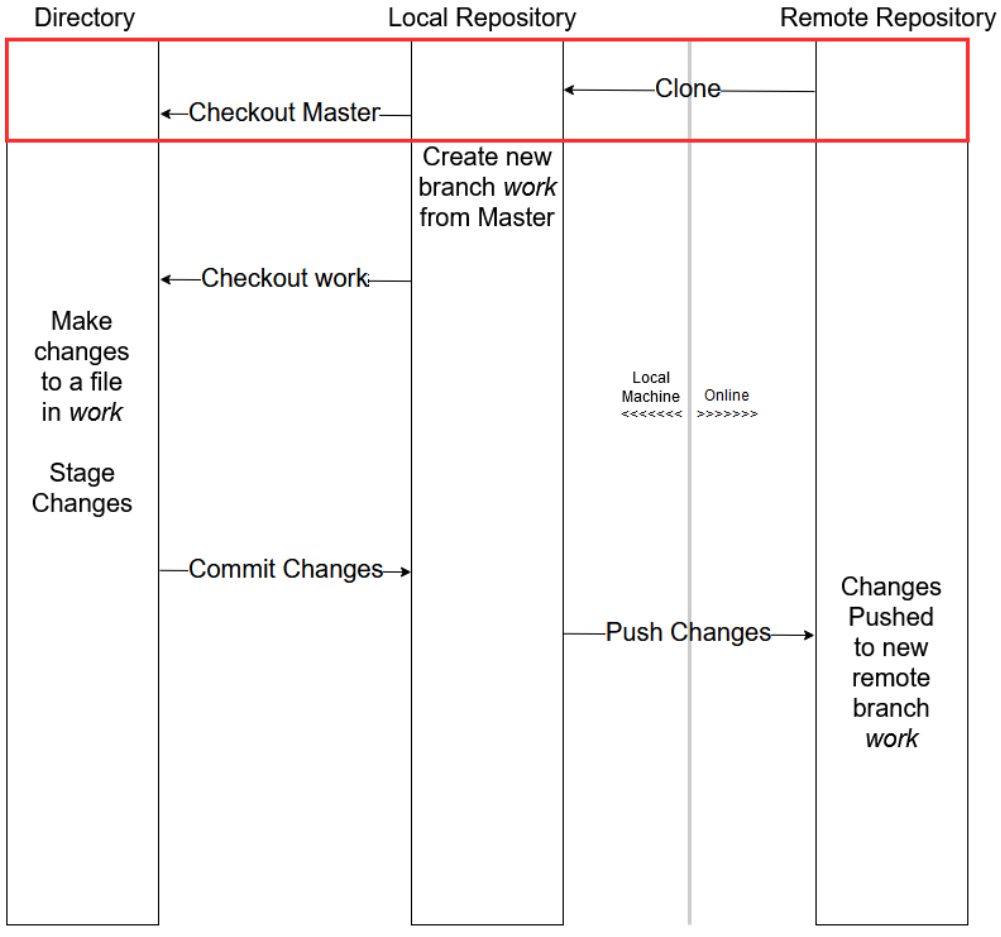


Figure: Cloning a repository

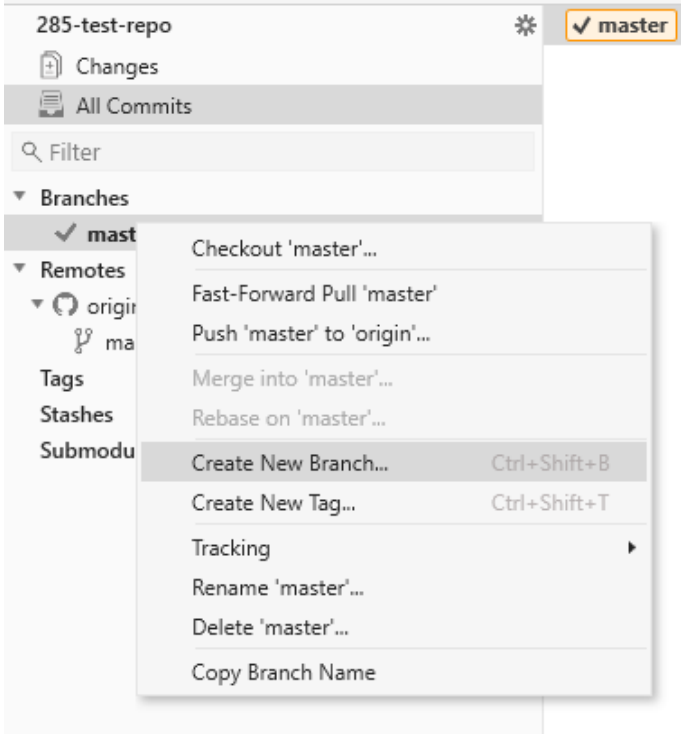


Figure: Making a branch

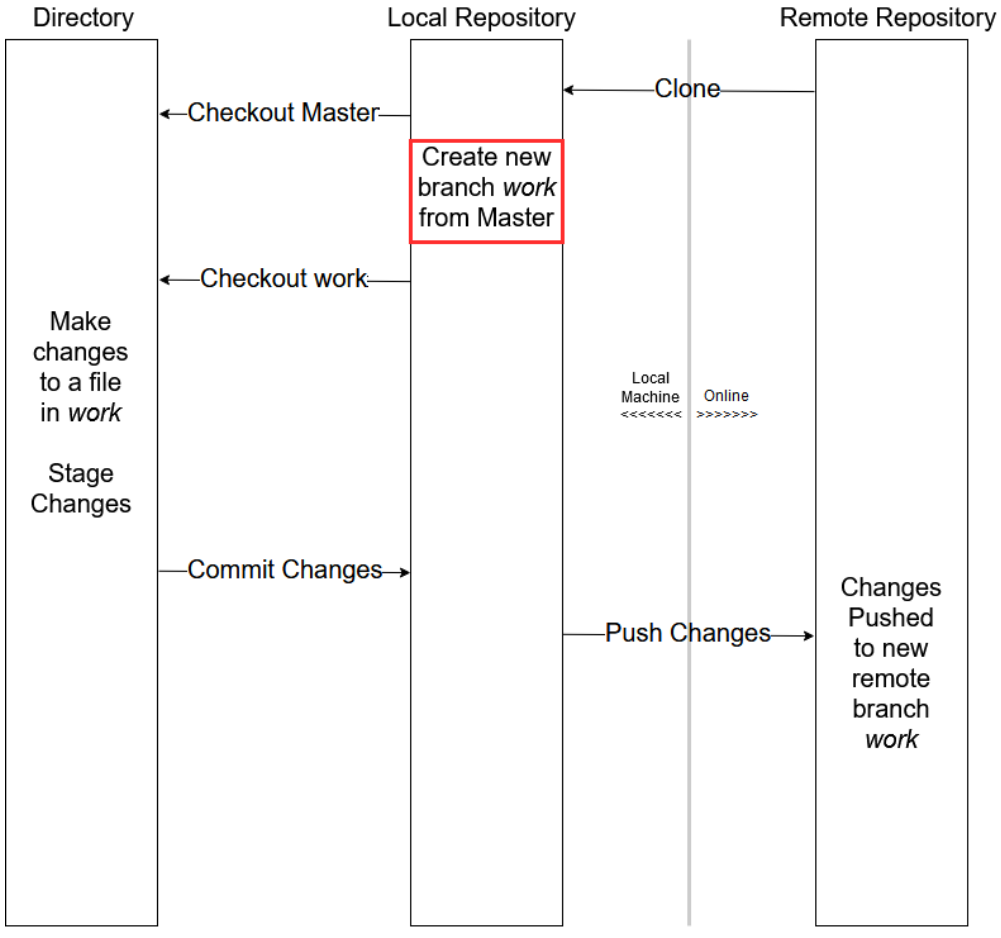


Figure: Making a branch

Note: The check mark next to master under Branches indicates that it is the branch currently checked out.

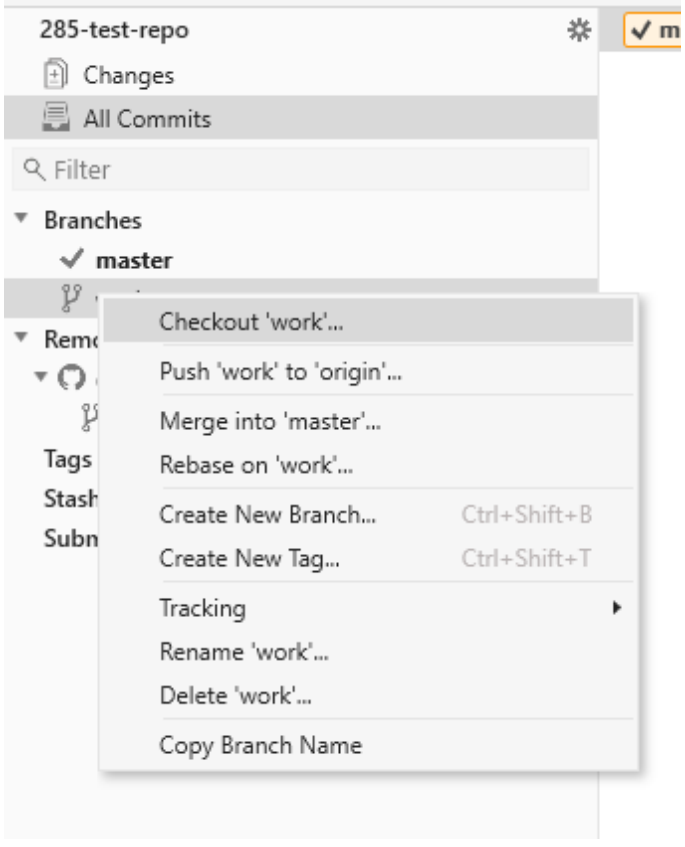


Figure: Checking out a branch

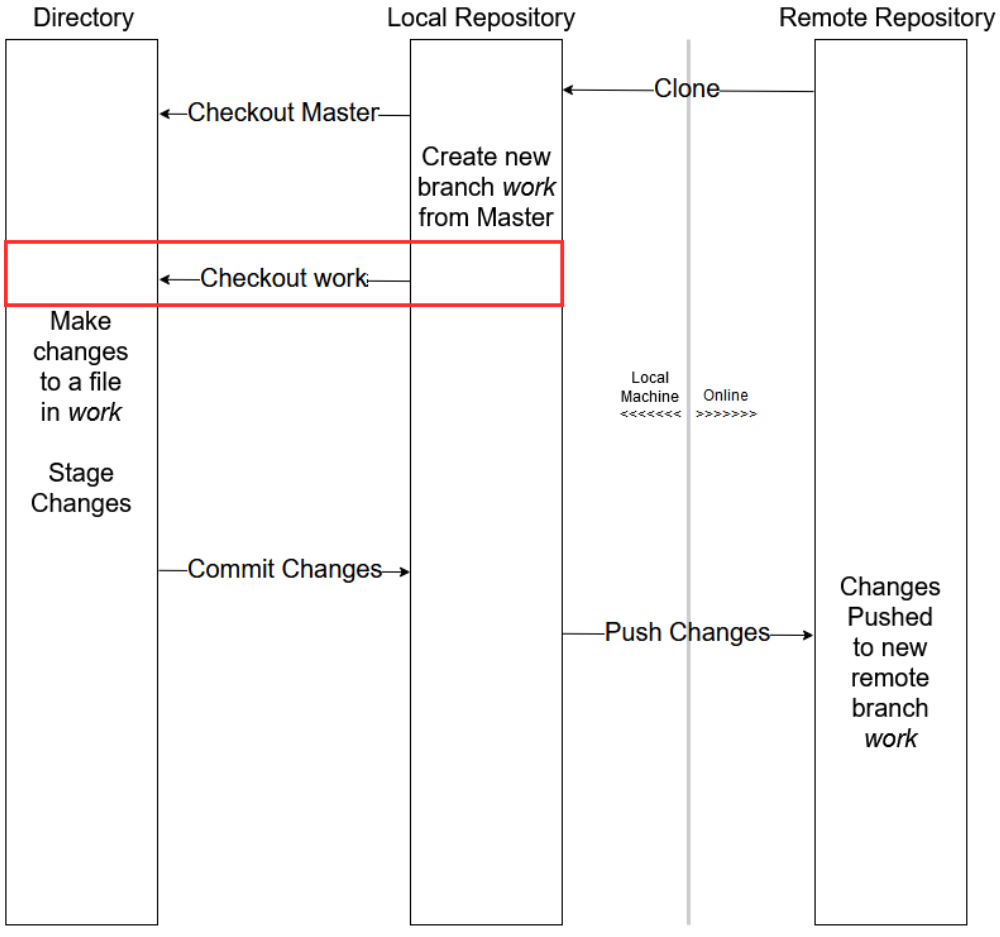


Figure: Checking out a branch

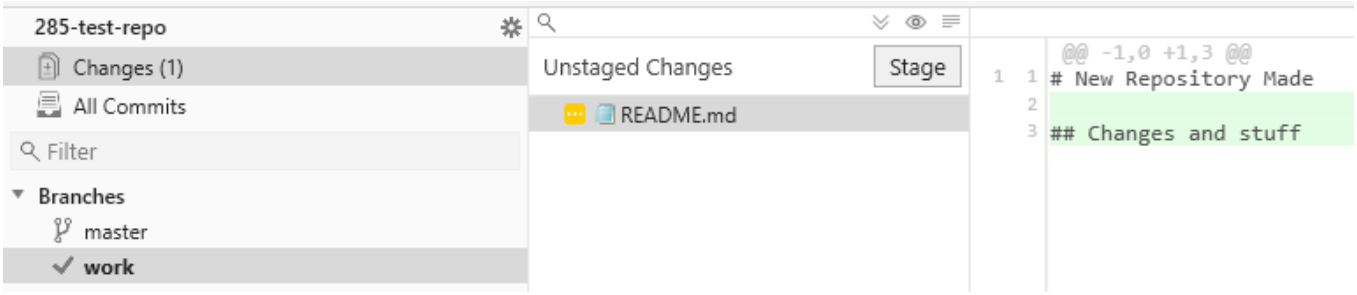


Figure: Making changes

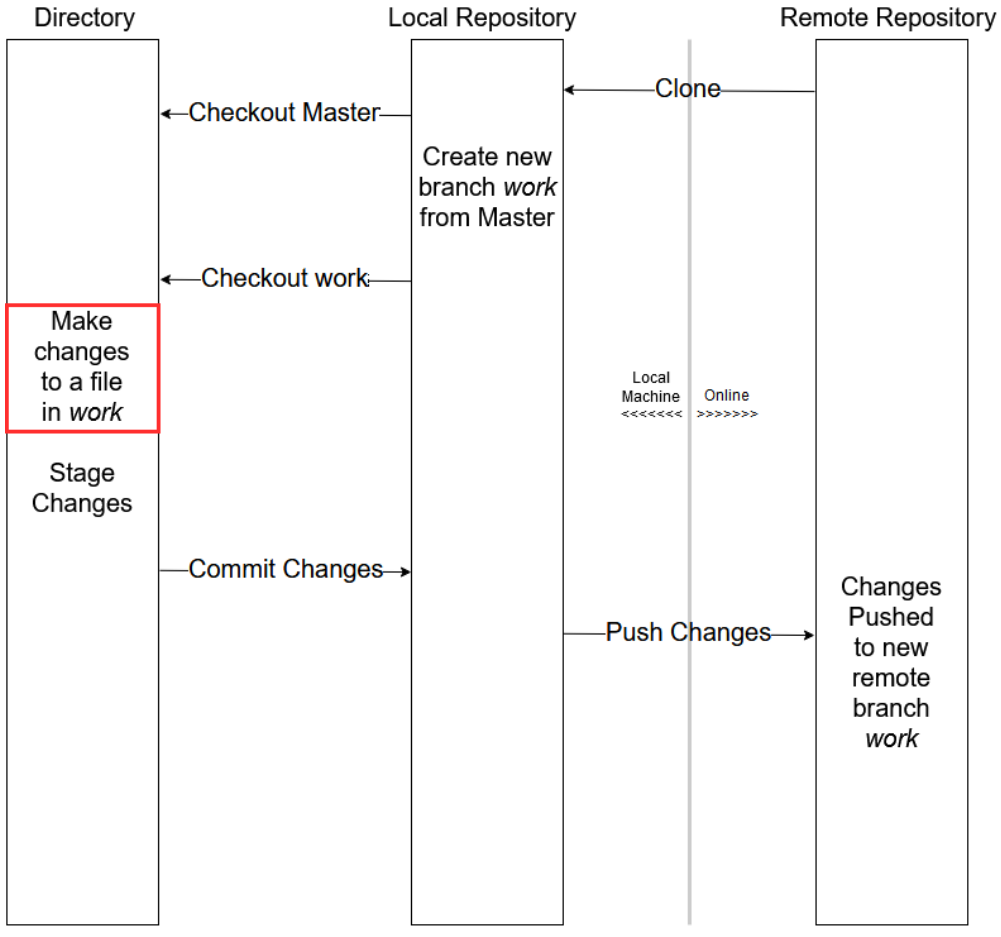


Figure: Making changes

Figure: Staging the changes

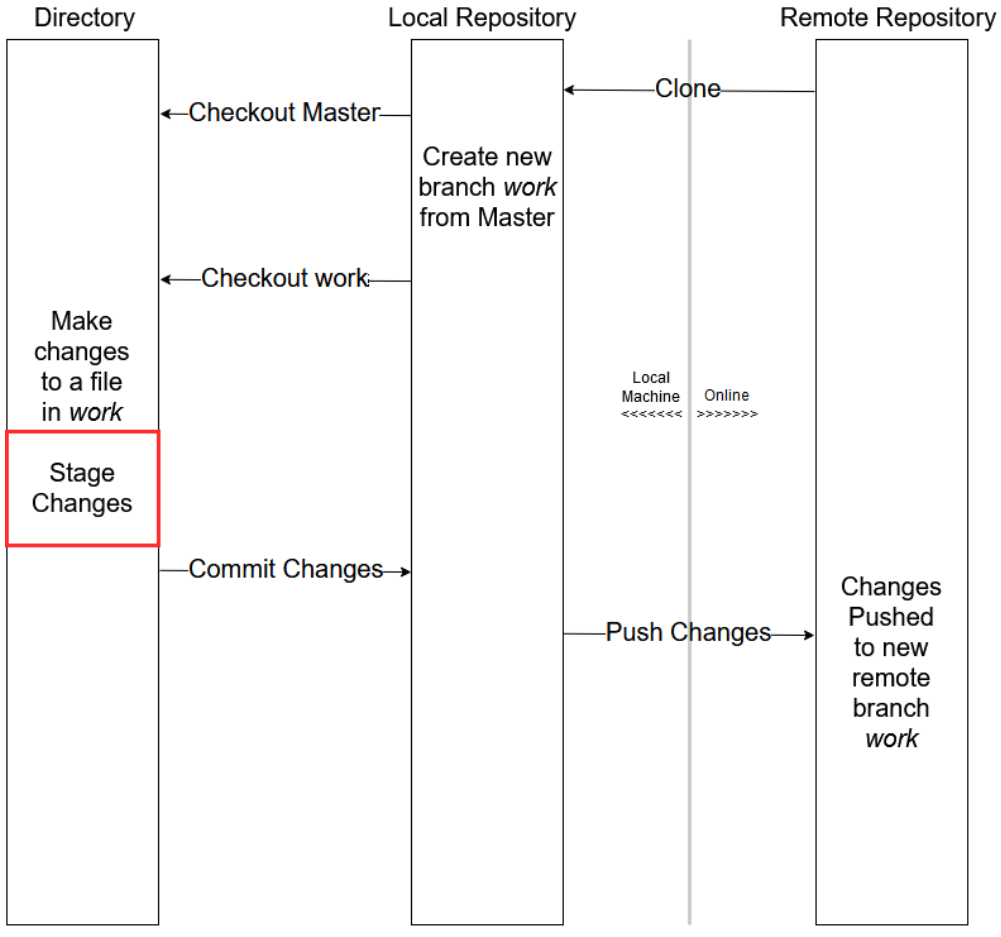


Figure: Staging the changes

Figure: Committing changes

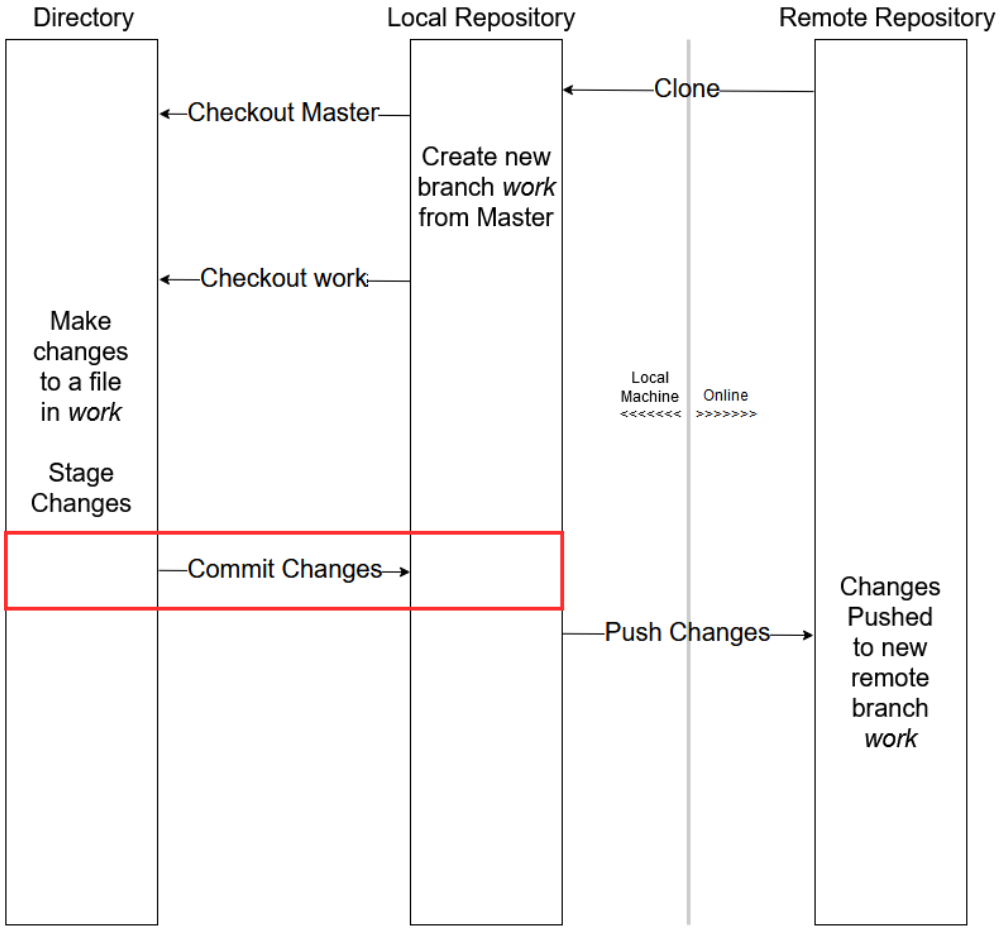


Figure: Committing changes

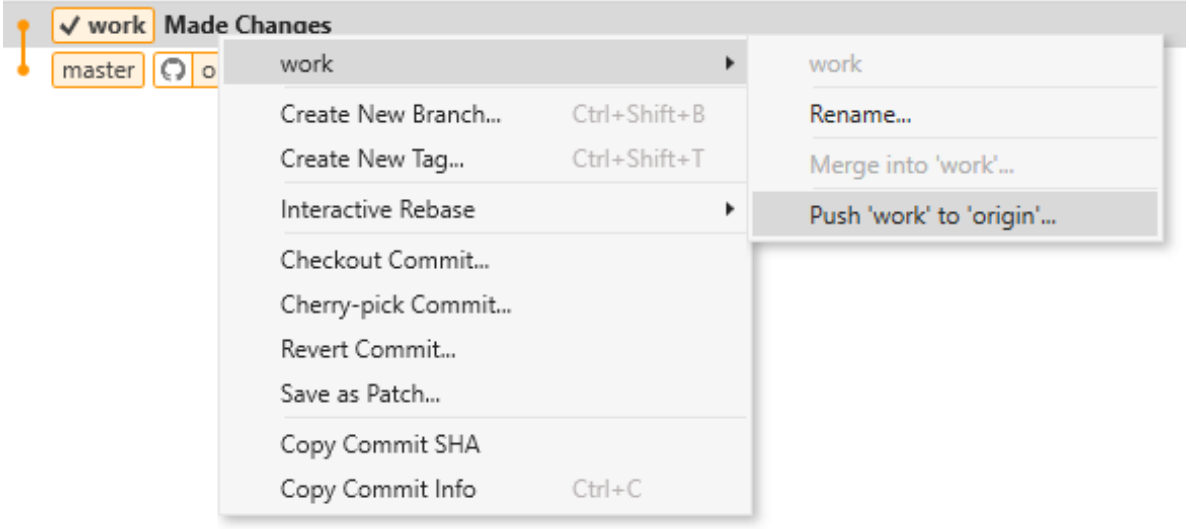


Figure: Pushing changes

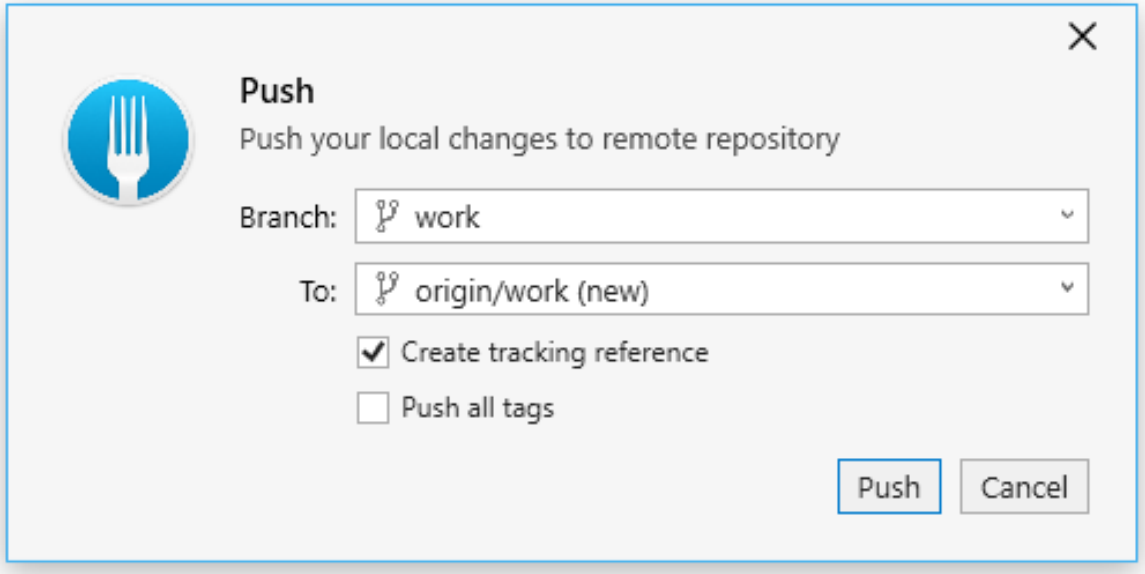


Figure: New Branch Dialog

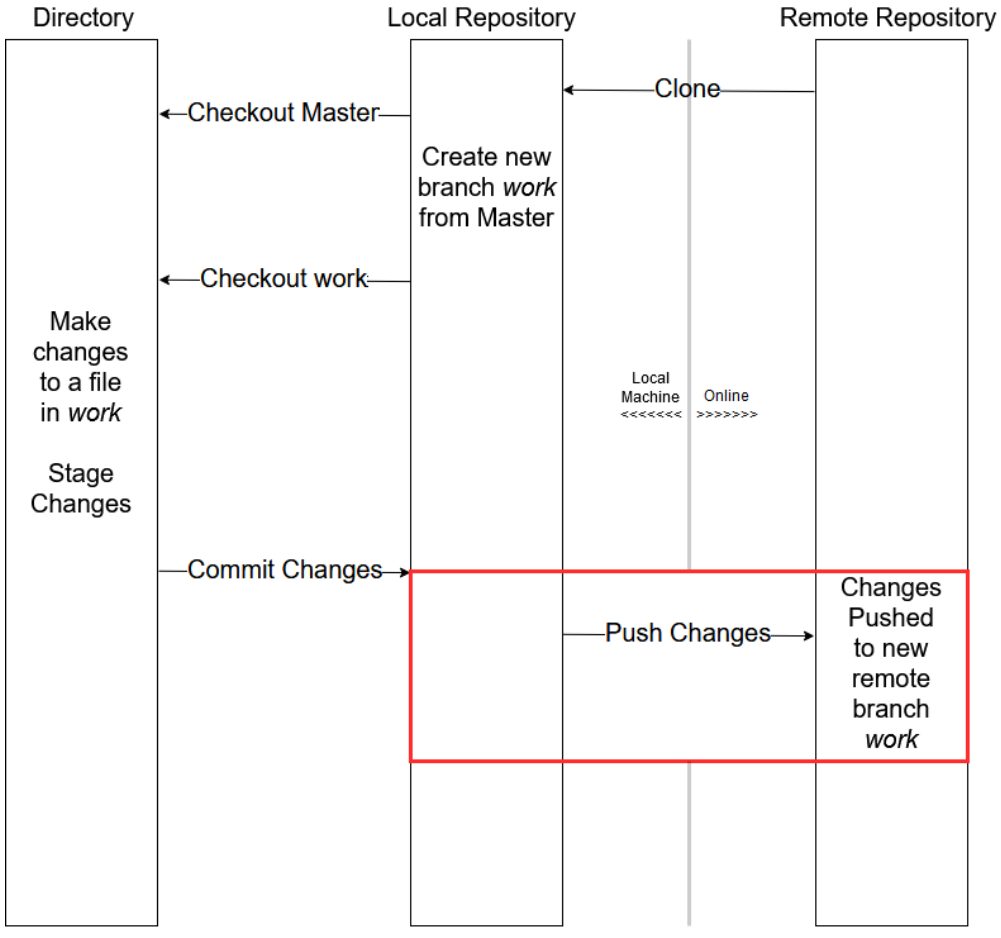


Figure: Pushing changes

You want to use the Fetch command often to check if any changes are made to master or possibly your own branch. If there are changes, you probably need to pull them.

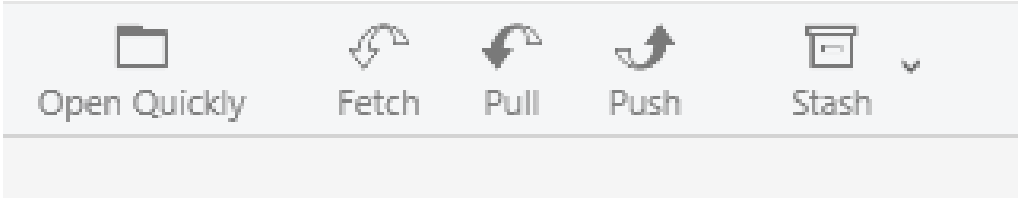


Figure: Fetch and Pull

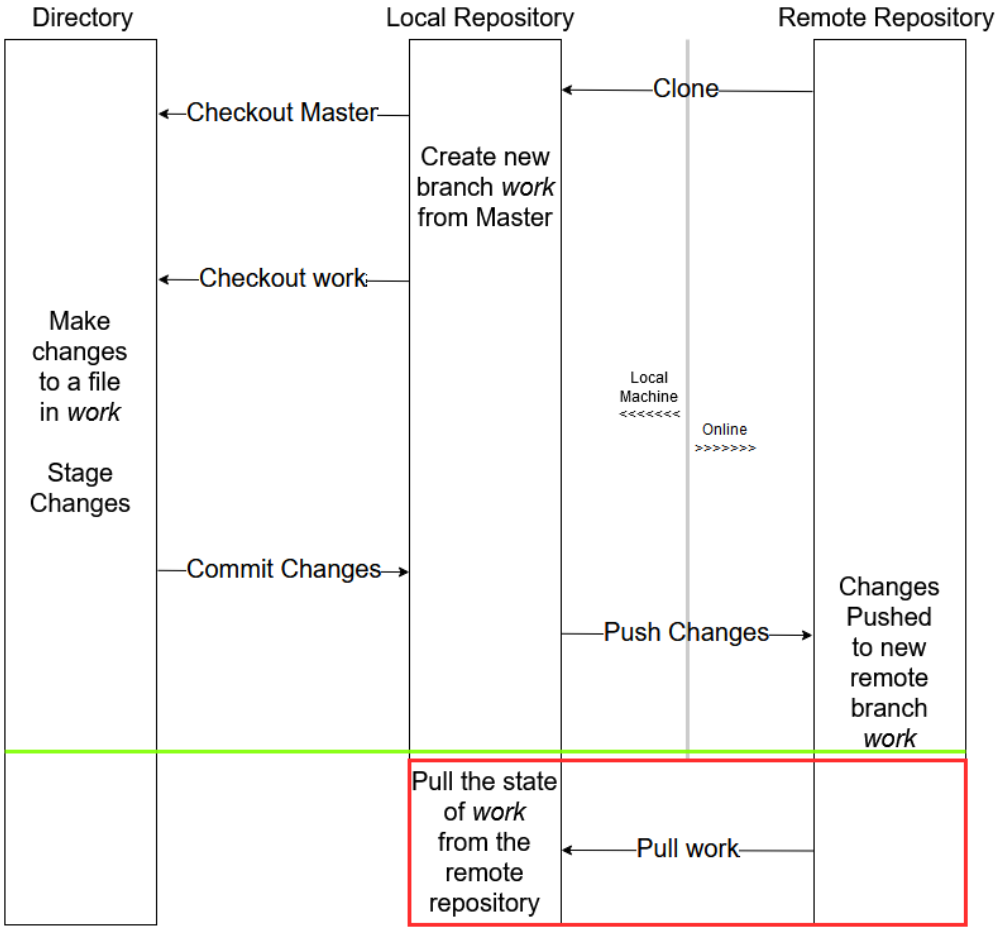


Figure: Pulling changes

Merging

Merging is something you will need to do to get your changes into master. If two branches merge without conflicts, then everything is fine. If they have merge conflicts, you will need to resolve the issues before merging. Communicate with your team members when there are merge conflicts.

Suggestions For Git Clients

- Fork: `https://git-fork.com/`
- Git Extensions: `https://gitextensions.github.io/`
- Github Desktop: `https://desktop.github.com/`
- Sourcetree: `https://www.sourcetreeapp.com/`
- Sublime Merge: `https://www.sublimemerge.com/`

- Write descriptive commit messages so you and your teammates can tell what was done in a commit.
- Each person should have their own branch they are working on. No one should be working directly on master or working on someone else's branch without talking to them about it. Make your branch name descriptive of who it belongs to and what you are working on, such as johnny-boi-google-login.
- Do not use Github to upload files or download your repository as a zip file. You should be using git properly so that your work is documented well and uploaded safely.
- Maintain a good file structure that makes things easy to find and conforms to your .gitignore. Changing where a directory is that is supposed to be ignored can include unwanted files in your project.
- Do not fork your repository.

A Few Resources on Using Git

- https://git-scm.com/book/en/v1/Getting-Started-Git-Basics
- https://confluence.lsstcorp.org/display/LDMDG/Basic+Git+Operations#/
- https://git-scm.com/book/en/v2

Questions?