**Commit and Push Workflow Guide**

**Introduction**

This guide outlines the process of committing and pushing changes to a Git repository. By following the steps described below, you will ensure that your changes are properly recorded, reviewed, and merged into the main branch.

**Prerequisites**

Before you begin, make sure you have:

1. Installed Git on your local machine.
2. Cloned the Git repository you will be working on.
3. Familiarized yourself with basic Git commands and concepts, such as:

* Committing:
  + [Git Handbook - Recording changes to the repository](https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository)
* Branching:
  + [Pro Git book – Git Branching](https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell)
* Merging:
  + [Pro Git book – Basic Branching and Merging](https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging)

**Step-by-Step Workflow**

Follow these steps to commit and push your changes to the Git repository:

1. **Create a new branch:** Always create a new branch for your work to keep the main branch clean and stable. Base your new branch on the latest version of the main branch.

git checkout main  
git pull  
git checkout -b /feature/US39XXXX74\_short\_description\_here

In case your automation task doesn’t have a designated user story, make sure you name the branch concisely.

git checkout -b /feature/Regression\_short\_description\_here

1. **Edit files:** Make the necessary changes to the files in your repository using your preferred code editor, such as VS Code.
2. **Check your changes:** Confirm that you have modified the intended files and that there are no unintended changes.

git status  
git diff

1. **Stage your changes:** Add the modified files to the staging area, preparing them for the next commit. You can stage all changes or specific files and folders.

git add .

If you only want to stage specific folders or files

git add tests/sampleTestFile.ts

1. **Review staged changes:** Double-check that you've staged the correct files.

git status  
git diff --staged

If you stage something you didn’t mean to, you can un-stage it.

git reset tests/sampleTestFile.ts

1. **Commit your changes:** Create a new commit with a descriptive message. Include a relevant ticket or issue ID in the commit message if applicable.

git commit -m “US39XXXX74: Descriptive commit message”

1. **Address any pre-commit hook issues:** If pre-commit hooks detect issues, the commit may be aborted. Review the changes, stage them, and retry the commit if necessary.
2. **Push your changes:** Push your branch and commits to the remote repository.

git push origin HEAD

1. **Create a pull request:**   
   On GitHub Enterprise, create a new pull request to request a review of your changes.

* Pull request titles must begin with the Rally ticket ID that you are working on.  
  Example: “US39XXX74: Description here”
* The “source branch” should be the one you’ve created and the “destination branch” should be main.
* The primary required reviewers for the following projects are:
  + Choice DTC - **`<tim\_cacayan@optum.com>`**
  + Redesign - **`<mark\_maras@optum.com>`**

In case the primary reviewer is not available then you can add another reviewer:

**`<webrell\_camposo@optum.com>`**.

1. **Address review feedback:** Make any necessary changes based on the reviewer's feedback, commit the changes to the same branch, and push the new commits to the remote repository. This will automatically update the existing pull request.
2. **Merge the pull request:** Once the pull request has been reviewed and approved, merge it into the main branch. Delete the feature branch if it's no longer needed.

**Conclusion**

By following this commit and push workflow, you'll ensure that your changes are effectively tracked, reviewed, and incorporated into the main branch of the Git repository. This process promotes collaboration, code quality, and project stability.