**GIT Documentation**

**Important commands**

1. Config
2. Clone
3. Status
4. Add
5. Commit
6. Push
7. Pull
8. Checkout
9. Merge
10. Stash

**Config:**

* Git comes with a tool called git config that lets you get and set configuration variables that control all aspects of how Git looks and operates.
* Before starting with any other git commands need to set few variables like user name, user email, proxy server as below

**Syntax:**

$ git config --global user.name "John Doe"

$ git config --global user.email [johndoe@example.com](mailto:johndoe@example.com)

$ git config –global http.proxy <http://userName:pwd@proxy> server:port

**Clone:**

Clone is used to create a copy of repository from remote to local. In order to clone we need GitHub repository URL.

**Syntax:**

$ clone <url>

Eg: $ git clone <https://github.com/Archana265/Test.git>

**Add:**

The git add command adds a change in the working directory to the staging area.

**Syntax:**

$ git add .

The above command adds all the changes in working directory to staging area

Or

$ git <fileName>

**Status:**

The git status command displays the state of the working directory and the staging area.

Syntax:

$ git status

**Commit:**

The git commit command commits the staged snapshot to the project history.

**Syntax:**

$ git commit

Or

$ git commit -m "Your comment"

**Push:**

Pushing is how you transfer commits from your local repository to a remote repo.

Once we commit the changes from staging to local repo need to perform push to move the changes from local to remote repo.

**Syntax:**

$ git push

**Checkout:**

The git checkout command lets you navigate between the branches created by git branch.

**Syntax:**

$ git checkout <branchName>

**Branch:**

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes. This makes sure that unstable code is never committed to the main code base, and it gives you the chance to clean up your feature’s history before merging it into the main branch.

Can create a branch using GitHub and then pull command is used to update the local repository.

Once we create a branch in GitHub need to pull

**Syntax to check the branch list:**

$ git branch -r

**Merge:**

The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

**Syntax:**

$ git merge <branchName>

**Stash:**

The git stash command takes your uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from your working copy.

**Syntax:**

$ git stash

At this point you're free to make changes, create new commits, switch branches, and perform any other Git operations; then come back and re-apply your stash when you're ready.

git diff stash@{0} master--- to compare branch with stash

**Re-applying your stashed changes**

**Syntax:**

**$ git stash apply**

$ git stash apply stash@{2} ----- to revert stash of particular index from the statck

By default, git stash pop will re-apply the most recently created stash: stash@{0}

$ git stash show --- You can view a summary of a stash

Push existing project to GitHub:

Step 1:

Create a new repository in GitHub.

Step 2:

Using Git Bash go to project folder in the working directory.

Step 3:

Execute below commands one after the other

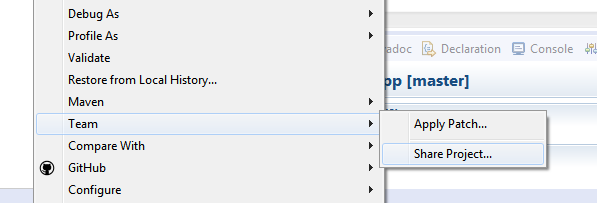
$ git init

$ git add .

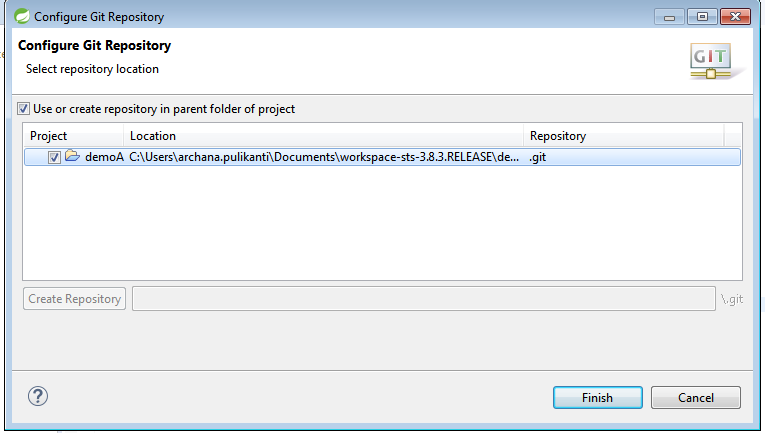
$ git commit –m “first commit”

Step 4:

In STS select project and go to project- Team- share Project

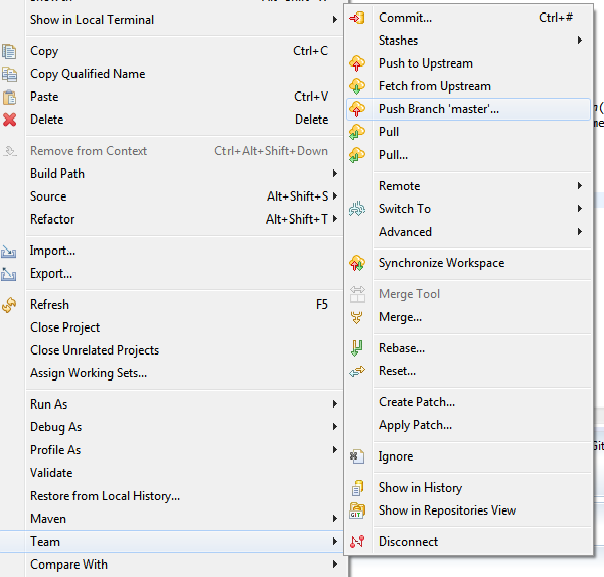


Step 5: Select the git repository as below and click finish

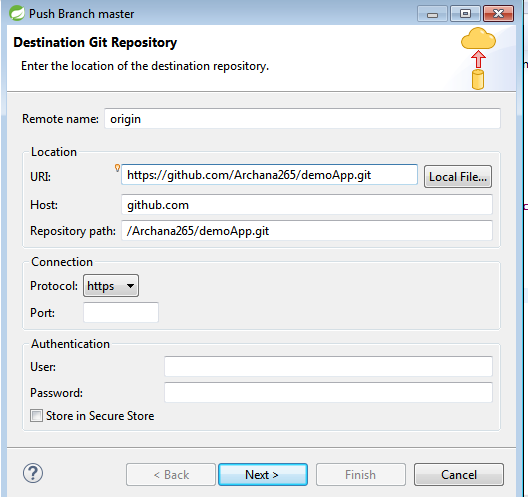


Step 6: Now to push the project to remote repository.

Select project- team- Push



Step 7: Provide the GitHub repository URI as below



Step 8:

Finish the push operation by providing GitHub credentials.

Terms to know:

* Staging: a cache of files that you want to commit