

GIT | BitBucket | SourceTree with Selenium

Table of content

- Version Control
- Git software
- Good practices for pushing to the remote repository BitBucket
- Create Repository on BitBucket for Free
- Clone Bitbucket Repository
- Commit and Push to Bitbucket with Git
- Pull the code from BitBucket
- SourceTree
- Clone repository using SourceTree :
- Basic Operations of SourceTree
- Selenium With SourceTree & BitBucket & Git
- How to revert/rollback the push

Version Control

Version Control is the management of changes to documents, computer programs, large websites, and other collection of information.

Version Controls keeps track of all the version of your project, every small change create a new version of your project

For example : You may have a Java code, and now you are saving the file as **xyz.java** , after a couple of days, you feel like some code has to be changed, so you performed that changes. After a week, you realize that you made changes to some other rather than the required file.

What happens now, your job and the code you developed both are gone.

To avoid such kind of scenarios, a tool takes a back up of your code, every time there is any change.

The tool which handles the above backups are called as Version control tools, I have listed few below:

GIT [we are going to explore]

SVN

Bazaar

Mercurial

Tools to Improve Java Code Quality

Git software

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.

Git is distributed control system

It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

Git is an open-source program to keep track of changes in your files. It just takes screenshots of your files when you save them, and then it compares the files for differences.

Install Git :

Navigate to <https://git-scm.com/downloads>

Choose the to download based on your operating system [I am using WIN 10]

Open the installer and follow just standard procedure, do not change any

If you want you can bring the git shortcut to Desktop

User Details in Git

Setting user details like user name and email is important for tracking purposes; follow below commands to set details.

Before setting any values in git, you need to open git bash terminal, --global is nothing but the scope of the user details.

Set the user name to the git

```
git config --global user.name "karthiQ"
```

Verify the user name that you had set

```
git config --global user.name
```

Set the user email address to the git [to follow this tutorial, you can use gmail account as well]

```
git config --global user.email "karthiq@chercher.tech"
```

Verify the user email address that you had set

```
git config --global user.email
```

```
user@DESKTOP-3U7SD8F MINGW64 ~  
$ git config --global user.name "karthiQ"  
  
user@DESKTOP-3U7SD8F MINGW64 ~  
$ git config --global user.name  
karthiQ
```

```
user@DESKTOP-3U7SD8F MINGW64 ~  
$ git config --global user.email "karthiq@chercher.tech"  
  
user@DESKTOP-3U7SD8F MINGW64 ~  
$ git config --global user.email  
karthiq@chercher.tech
```

Page Object Model Folder Structure

Basics of Git CMD :

It is important to learn command line/CMD commands in git, even though we may not use them along without modern work IDEs but they will be helpful

Get the current folder details:

```
pwd
```

Create a new folder,

```
mkdir folderToCreate
```

Move into a sub-folder:

```
cd folderName
```

```
user@DESKTOP-3U7SD8F MINGW64 ~  
$ pwd  
/c/Users/user  
  
user@DESKTOP-3U7SD8F MINGW64 ~  
$ mkdir gitTutz  
  
user@DESKTOP-3U7SD8F MINGW64 ~  
$ cd gitTutz  
  
user@DESKTOP-3U7SD8F MINGW64 ~/gitTutz  
$
```

Create any file and save it in the folder that we have created just before; in the below step, I am using the vim to create a file.

Create a file in git:

```
touch abc.php
```

List the files in the current folder

```
ls
```

Remove/delete a file:

```
rm abc.php
```

Step out of the current directory:

```
cd ..
```

How to remove a directory:

```
rmdir directoryName
```

```
user@DESKTOP-3U7SD8F MINGW64 ~/gitTutz
$ vim abc.php

user@DESKTOP-3U7SD8F MINGW64 ~/gitTutz
$ rm abc.php

user@DESKTOP-3U7SD8F MINGW64 ~/gitTutz
$ cd ..

user@DESKTOP-3U7SD8F MINGW64 ~
$ rmdir gitTutz

user@DESKTOP-3U7SD8F MINGW64 ~
$
```

Create a Git Repository :

Create a folder called SampleRepo and move inside the folder for creating the git repository

Convert the created folder into the git repository with the below command.

```
git init
```

if you see the highlighted line in the output, then your repository got created successfully.

```
user@DESKTOP-3U7SD8F MINGW64 ~
$ mkdir SampleRepo

user@DESKTOP-3U7SD8F MINGW64 ~
$ cd SampleRepo

user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo
$ git init
Initialized empty Git repository in C:/Users/user/SampleRepo/.git/
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$
```

Create two files inside the repository using the touch command

```
touch abc.php
index.html
```

git status :

git status command shows the difference between the working folder and the local repository, just because we have created above two files in the current directory does not mean they are present in the local repository. Those two files are just present in the working directory, even though the working directory and local repository are the same.

Files will be moved to the local repository only when we commit them to the local repository.

The untracked files are nothing but the files that you have created but not committed.

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
(use "git add ..." to include in what will be committed)
```

```
abc.php
```

```
index.html
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

AutoIt in Selenium

Stage the files :

The stage is nothing but the pre-step for committing the files, Committing files staging is important.

We can add files to the stage using add command in the git.

```
git add abc.php
```

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
```

```
$ git add abc.php
```

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
```

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached ..." to unstage)
```

```
new file:   abc.php
```

```
Untracked files:
```

```
(use "git add ..." to include in what will be committed)
```

```
index.html
```

You can differentiate the added and not added/untracked files

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git add abc.php

user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   abc.php

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    index.html
```

How to remove a file from the stage

```
git rm --cached abc.php
```

We can add all the untracked files into stage using .(dot) command

```
git add .
```

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git add .

user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   abc.php
        new file:   index.html
```

Commit the files to Local repository :

Committing is nothing but adding our staged files into local repository

Committing is important, and you should commit whenever you feel like you have completed a unit.

commit command is used to commit the staged files, we can also set the message for the commit, messages could be your simple description of the changes, -m represents that following is a message

```
git commit -m "added two basic files"
```

commit outcome

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git commit -m "added two basic files"
[master (root-commit) c9979a2] added two basic files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 abc.php
 create mode 100644 index.html
```

we can see the commit history by using log command, this will tell us what branch we are in and the author, date, commit message.

```
git log
```

The outcome of the log, HEAD represents, to which repository user committed.

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git log
commit c9979a285091d7a4f2a35382a071c84b98bdfae (HEAD -> master)
Author: karthiQ
Date:   Sat May 19 02:37:01 2018 +0530
    added two basic files
```

CSV Files with selenium

Branch

A branch is a mirror of your current repo. When you branch, your current repo (usually your main branch) goes untouched. This is great when you want to create a feature in your application because you do not have to worry about breaking a working application since

The branch is nothing but an independent line of development. For example : People create dev branches from the Master Branch. Now developers will do all the edits or features in the dev branch. After the edits or developments of a feature are completed, then one after testing the dev branch will be merged into the master branch.

We can create a check the branches present in the git using git branch command; if we provide a new name along with the git branch, it will create a new branch.

The branch that we are currently in will be in a different color with an * asterisk in the front.

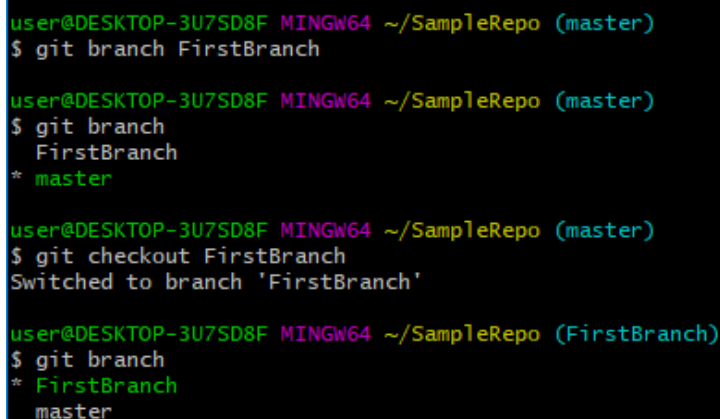
```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git branch
* master
```

Create your first Branch:

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git branch
    FirstBranch
* master
```

Switch to the newly created branch :

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git checkout FirstBranch
Switched to branch 'FirstBranch'
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (FirstBranch)
$ git branch
* FirstBranch
  master
```



```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git branch FirstBranch

user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git branch
    FirstBranch
* master

user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git checkout FirstBranch
Switched to branch 'FirstBranch'

user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (FirstBranch)
$ git branch
* FirstBranch
  master
```

Now, if you create any files, it will be created in a new branch only.

Git Ignore :

Sometimes you may have files that are related to the execution or some log files; these files are to be omitted when we push our source code.

Let's create files: report.log, error.log in our current working folder. With git status command you can see tracked and untracked files,

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (FirstBranch)
```

```
$ git status
On branch FirstBranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    error.log
    report.log
nothing added to commit but untracked files present (use "git add" to track)
```

Now in the above thing, we always get to see these useless log files in our untracked work, if we have two log files we do not have many issues but if we have hundreds of log files, then it will create bigger confusion, to avoid this kind of situation we can use git ignore.

When we want to ignore the files we have to make an entry of those files in .gitignore Below is content that I have created in .gitignore file, we can use # symbol for commenting.

```
#.gitignore file content
error.log
report.log
```

We can also ignore files based on regex, below file content ignores all the log files.

```
#.gitignore file content
*.log
```

If we try to see the tracked and untracked files, then we will not have those log files as part of untracked files.

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (FirstBranch)
$ git status
On branch FirstBranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
nothing added to commit but untracked files present (use "git add" to track)
```

Just creating a **.gitignore** file will not help us; we have to commit the **.gitignore** file so that it will become active

Merge

Merge is a definition within itself. If you were working on a branch in which you have your working feature built, and you want to merge that feature back to the master branch, you would need to do a merge. It combines them into one.

We use the git merge command to merge two branches.

*You should be in a branch, to which you are going to merge some new
branch [here master]*

If you are not in a target branch, then you have to checkout to that branch, and then only you should use the merge command.

You should make sure you do not have any uncommitted work in the secondary [FirstBranch here] before merging.

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (FirstBranch)
$ git commit -m "description of change"
```



```
[FirstBranch ab72069] description of change
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (FirstBranch)
$ git checkout master
Switched to branch 'master'
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git merge FirstBranch
Updating c9979a2..ab72069
Fast-forward
 .gitignore | 1 +
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
```

Clone

A clone is when you “clone” a remote repository and receive a local copy for your modification. Git will keep track of all your modifications locally, so you don’t have to depend on the reliability of the server.

Push

A push is how you get your local files to a remote repo so your team will have the option to pull your latest modifications.

Pull

A pull gets the latest code from a remote repository. When you do a pull, it also does a merge. A pull is basically a fetch and a merge.

Fetch

Fetch is a pull without the merge. This is helpful if you want to compare the code your team has put together before deciding to merge it to your local repository.

Fetch

Checkout lets you navigate between branches.

[Learn about Constructors in selenium and Java](#)

Good practices for pushing to the remote repository

Make sure to communicate what I am working on with teammates before I start any change or feature to avoid duplication of efforts.

Before pushing, make sure I have pulled all current files or changes from the master/remote.

Be sure there are no bugs in my current commit, and it is thoroughly tested.

Never commit anything that is not complete.

Use feature branches for anything but minor changes.

Commit often.

Use descriptive commit messages, something like “added new file”? is meaningless, “added new JS file for handling login modal behavior”, for example, is much better.

Headless browser in Selenium webdriver

BitBucket

BitBucket is a remote repository, which was later acquired by a company called Atlassian (yeah, the same company that owns JIRA). BitBucket enables the user to use Git and Mecerual software for integration from local repositories to remote repositories.

Features of BitBucket :

*Pull requests and code reviews
Unlimited private repos
Branch comparison and commit history
Bitbucket Mac and Windows client called SourceTree; Android app called BitBeaker
Bitbucket for Enterprises, called Stash
Integration with tools like Jira, Crucible, Bamboo, Jenkins*

Create Repository on BitBucket for Free

Bit bucket provides free repositories for the team, which has less than 5 members if your team is bigger, maybe you need to buy some premium stuff.

With free itself, you can create an n-number of repositories, which include private and public.

Steps to Create a Repository on BitBucket :

Navigate to <https://bitbucket.org>

Click Get Started and sign up for the bit bucket by providing your email id

Enter your username and password

Verify your email address

Create a unique user name for you

Click on Create repository

Here's where your work shines through

Set up a repository to get going with your code. After that, you'll find your relevant repositories and work right here.

Create repository

Import repository

Create a new repository

[Import repository](#)

Repository name*

Access level ☒ This is a private repository

Include a README? Yes, with a tutorial (for beginne...

Version control system ☒ Git ☐ Mercurial

[Advanced settings](#)

Create repository

Cancel

Apache POI with Selenium Integration

Clone Bitbucket Repository

Cloning repository is nothing but copying the remote repository into our local system, similar to download.

Steps to clone the Bitbucket repository :

Navigate to bitbucket repository and click + icon on the left.

Select Clone this Repository under get to work

It will open a Modal box, in that you can choose how you want to clone your repository either using https or ssh method.

I clone via https method

Copy the URL that starts with git clone

Clone this repository

HTTPS

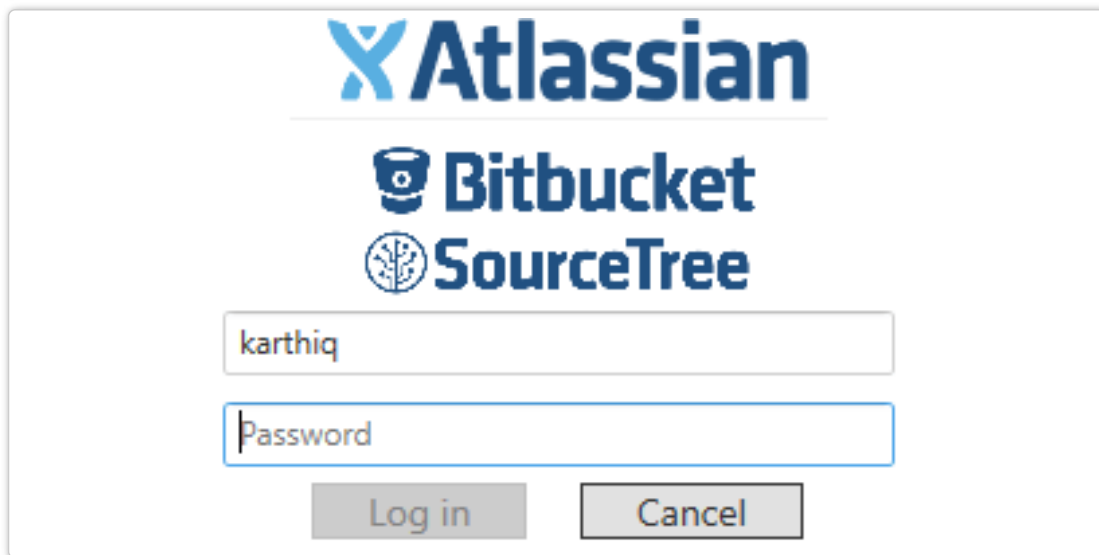
git clone https://karthiq@bitbucket.org/karthiq/demorepo.git

Paste the URL thing that copied in bitbucket and press enter key/return key

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git clone https://karthiq@bitbucket.org/karthiq/demorepo.git
Cloning into 'demorepo'...
```

Sometimes bitbucket asks to enter the password in git bash itself (password is masked to not appear, so when you enter you cannot see your password, so after entering your password press enter

Instead of the above step, sometimes bitbucket shows a pop up to enter password / with me this is the case/



Atlassian
Bitbucket
SourceTree

karthiq

Password

Log in Cancel

If everything successful then you will see below message

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ git clone https://karthiq@bitbucket.org/karthiq/demorepo.git
Cloning into 'demorepo'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
```

How to install Selenium Grid

Commit and Push to Bitbucket with Git

In the above-created clone folder, let's create few files inside the **demorepo**; you can create it using file explorer itself.

PC > Local Disk (C:) > Users > user > SampleRepo > demorepo

Name	Date modified	Type	Size
.git	5/19/2018 3:11 PM	File folder	
README	5/19/2018 3:11 PM	Markdown Source...	3 KB
testfile	5/19/2018 3:25 PM	Text Document	1 KB
urlfile	5/19/2018 3:25 PM	Text Document	1 KB

testfile - Notepad

File Edit Format View Help

this is my first file into remote repo
by the way I am karthiQ

urlfile - Notepad

File Edit Format View Help

my website url is http://chercher.tech

Add the files into stage using

add command in git and commit using the commit command.

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo (master)
$ cd demorepo/
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo/demorepo (master)
$ git add .
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo/demorepo (master)

$ git commit -m "first commit to remte repo"
[master 87ed4ae] first commit to remte repo
2 files changed, 3 insertions(+)
create mode 100644 testfile.txt
```

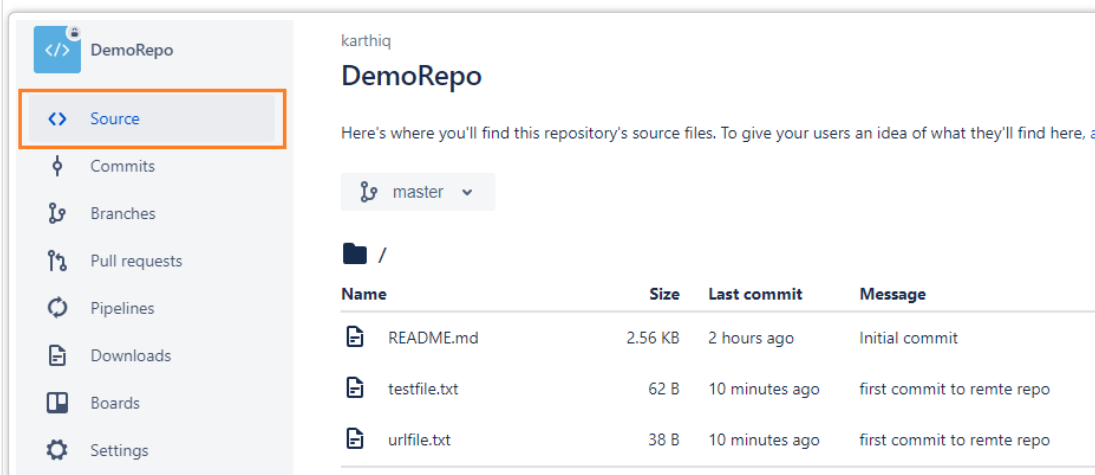
```
create mode 100644 urlfile.txt
```

Push the commit into the remote repository using push command.

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo/demorepo (master)
$ git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 415 bytes | 207.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/karthiq/demorepo.git
40bd670..87ed4ae master -> master
```

Note : In some tutorials, you may see git push origin master, that is also right, but it did not work.

You can verify the push by navigating into the bitbucket repository in your browser under the source tab.

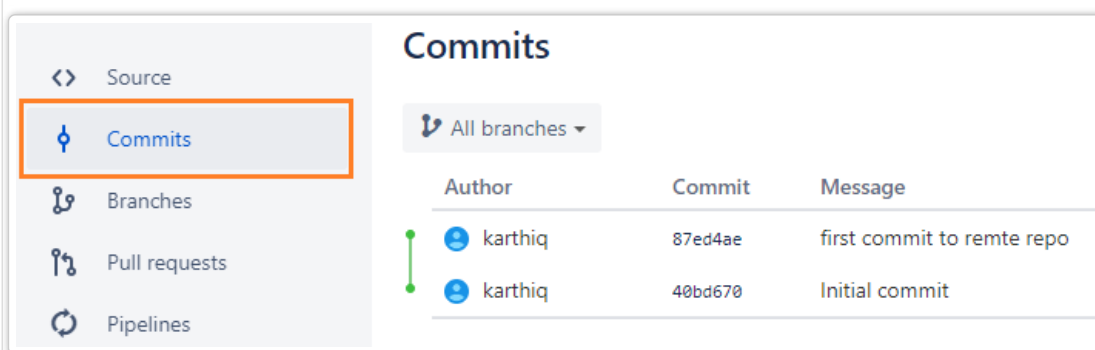


The screenshot shows the BitBucket interface for a repository named 'DemoRepo' by user 'karthiq'. The 'Source' tab is selected and highlighted with an orange box. The left sidebar contains links to Source, Commits, Branches, Pull requests, Pipelines, Downloads, Boards, and Settings. The main content area shows the 'master' branch selected. Below the branch selector, there is a table listing the files in the repository:

Name	Size	Last commit	Message
README.md	2.56 KB	2 hours ago	Initial commit
testfile.txt	62 B	10 minutes ago	first commit to remte repo
urlfile.txt	38 B	10 minutes ago	first commit to remte repo

You can also get the details of

the person who committed the change under the Commits tab.



The screenshot shows the BitBucket interface for the same repository, but with the 'Commits' tab selected and highlighted with an orange box. The left sidebar is the same, but the 'Commits' link is active. The main content area shows the 'All branches' dropdown. Below it, there is a table listing the commits:

Author	Commit	Message
karthiq	87ed4ae	first commit to remte repo
karthiq	40bd670	Initial commit

Data Provider in TestNG

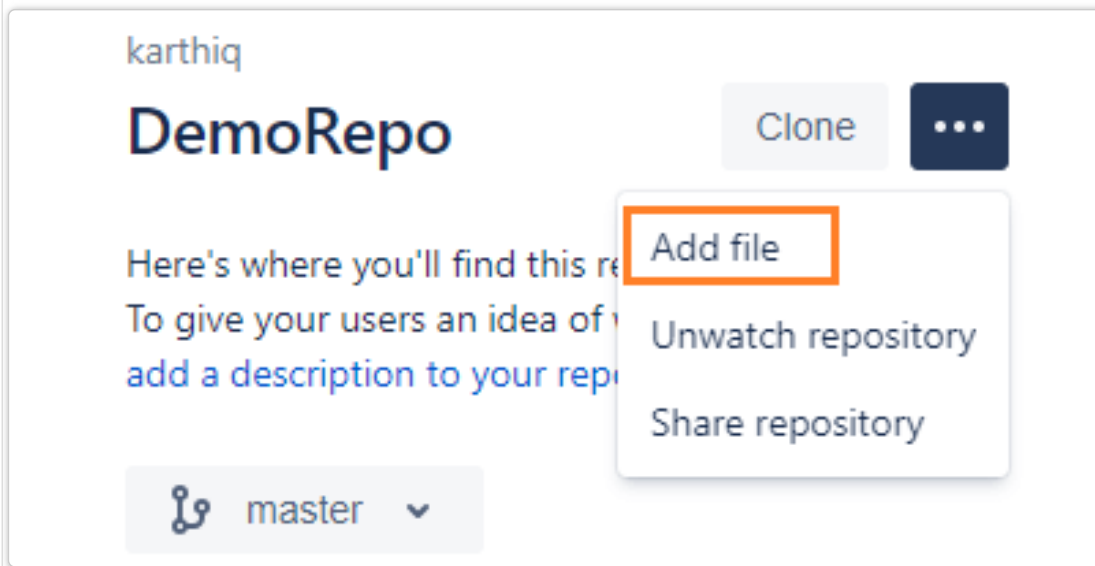
Pull the code from BitBucket

When there is more than one person working on a project, we may need to use the files created by other people.

To get the files from the bitbucket, we can use the git pull command. But make sure that other person first pushes the code into the bitbucket before you try for the pull.

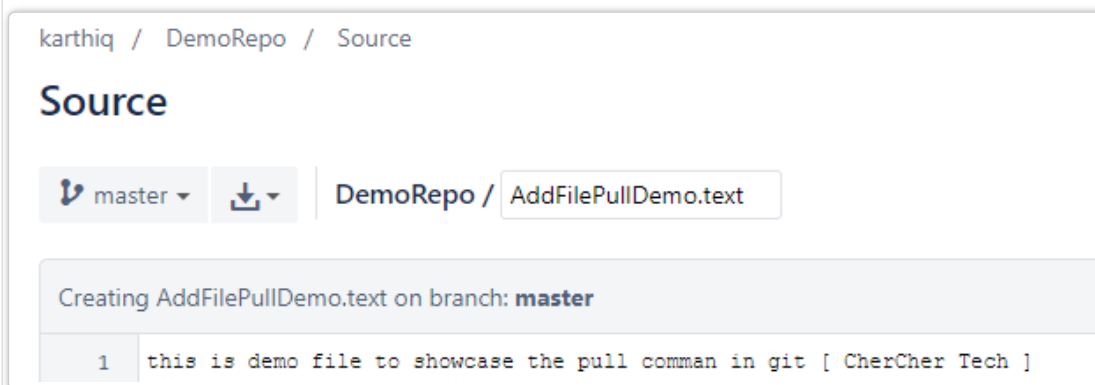
As I am the only person working on this, so I am going to create a file in the bitbucket repository for pulling to our local repository.

Navigate to our repository; click the three dots next to the clone button. Select the Add file option and provide name and content.



Add some name and content

to file



Now commit the file and

provide some message for the commit in bitbucket.

Once the creation is done, open your git and use the git pull command.

```
user@DESKTOP-3U7SD8F MINGW64 ~/SampleRepo/demorepo (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://bitbucket.org/karthiq/demorepo
 87ed4ae..f4b93c6 master    -> origin/master
Updating 87ed4ae..f4b93c6
Fast-forward
 AddFilePullDemo.text | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 AddFilePullDemo.text
```

At the end of the message, you can see AddFilePullDemo.txt got pulled. You can ensure by navigating to the respective folder in your local system.

Pull command performs two operations 1. fetch, 2. merge.

fetch brings remote changes to the local system.

merge merge the change from the fetch into your local files.

Next, we will learn how to use a UI interface to connect bitbucket and git rather than git bash console.

Event Listeners present in Selenium

SourceTree

SourceTree is freeware, which provides Graphical User Interface for version control software.

By default, git comes with Git Bash console, which looks similar to your terminal/cmd, with terminal handling changes becomes overhead.

To avoid this, we would be using the Graphical UI provided by SourceTree software.

Install SourceTree :

Navigate to <https://www.SourceTreeapp.com/>

Download the SourceTree software based on your operating system

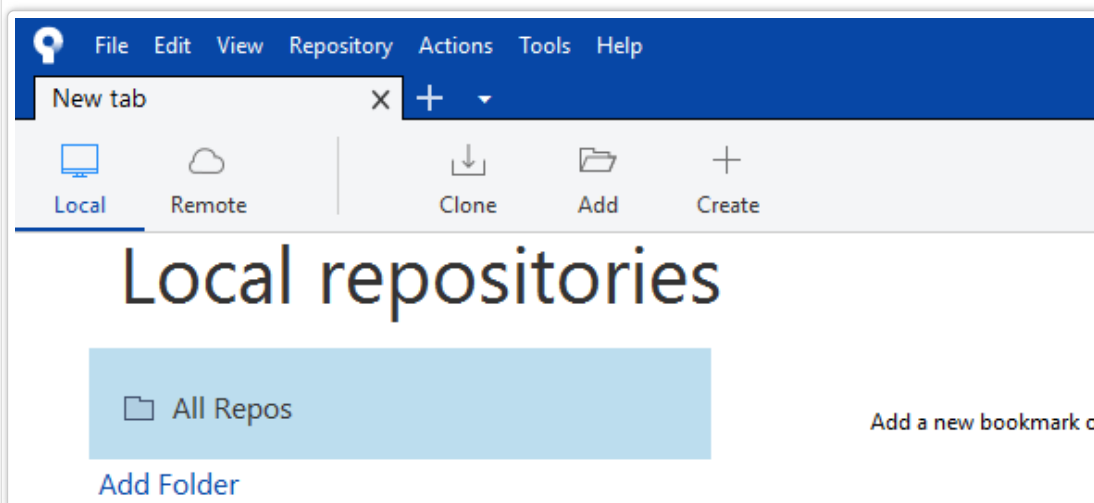
Agree to the Terms and select to Atlassian account

Enter your user name (we have created on bitbucket), and password for the same

If you have not installed or if you want to install some version controls, you can install [here](#)

Clone repository using SourceTree :

Once we complete the SourceTree installation, open the source tree software, you see below interface in win 10.



Now you can clone either

local repository or remote repository; I am cloning the same remote repository that we have created sometime back.

Click Clone icon in SourceTree, I hope You know how to take clone URL, please do provide without the git clone and choose a local folder to

which you want to download the remote repository.

Clone

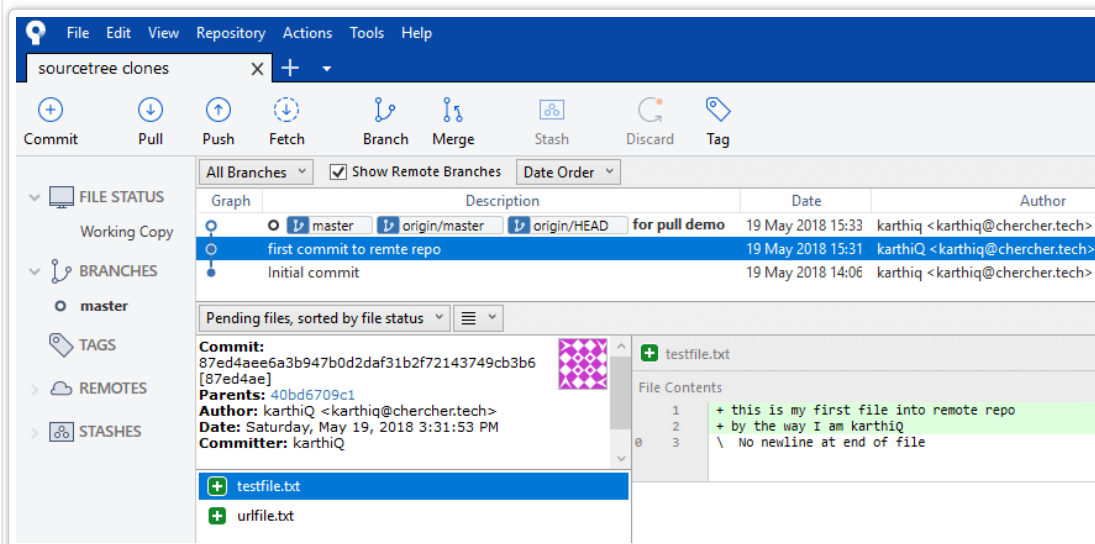
Cloning is even easier if you set up a [remote account](#)

Repository Type: This is a Git repository

Local Folder:

☒ Advanced Options

Once the clone is over, you should be able to see a window similar to below one.



Handle dropdowns in selenium

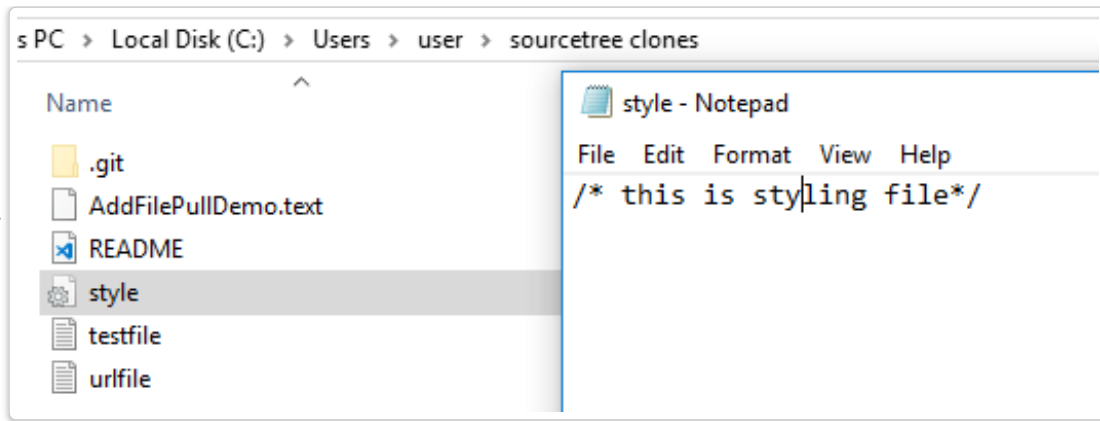
Basic Operations of SourceTree

We can perform all the operations SourceTree that we had performed using git bash; with SourceTree, we can perform more operations with simple clicks.

Stage Files in SourceTree :

Create a new file in the repository that we have cloned in the above step. I am creating a file called style.css and placing some text

inside the file.

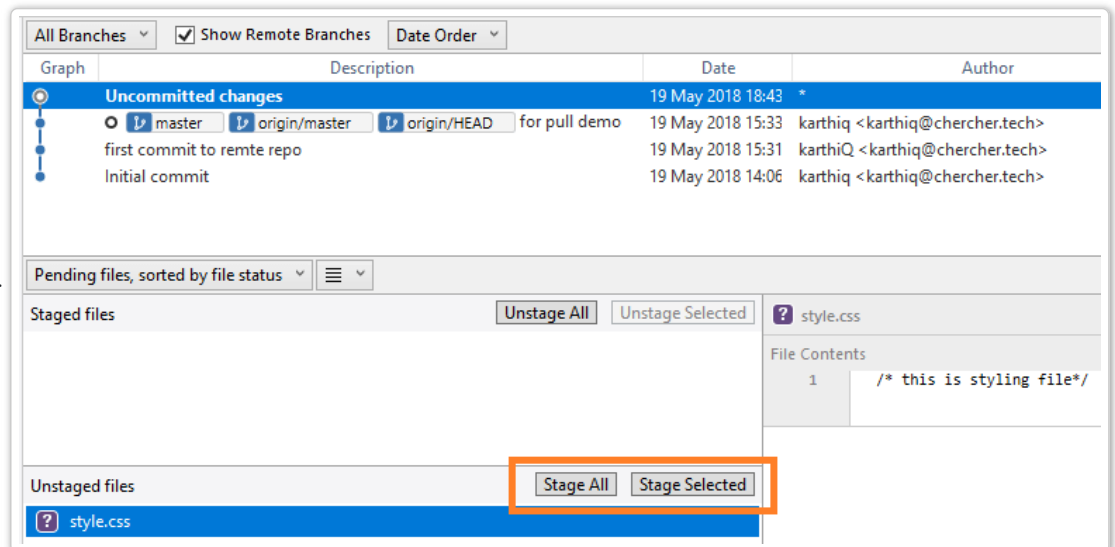


We have

created a file, and now we can perform the staging operation in SourceTree.

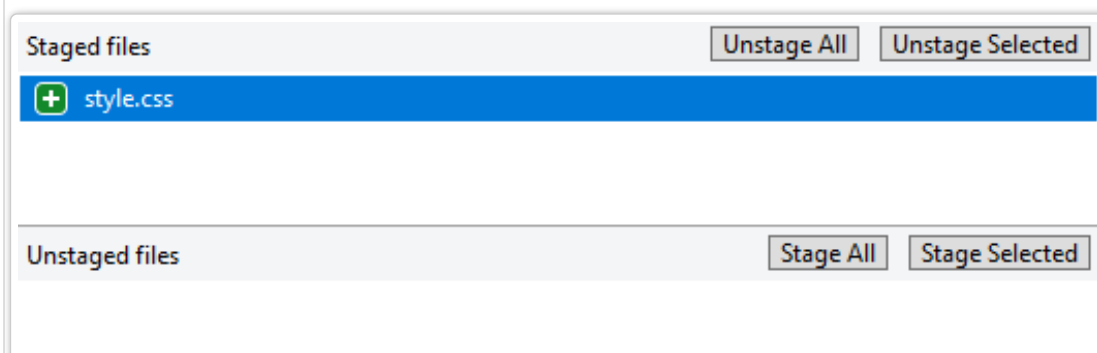
If you open SourceTree, you can see some un-committed changes in the top pane, and the file (style.css) we have created is in the bottom pane

under Unstaged files section.



The SourceTree automatically detects the changes in the repository.

To Stage any file, we need to select the file and click stage selected or stage all if you want to stage all the files.

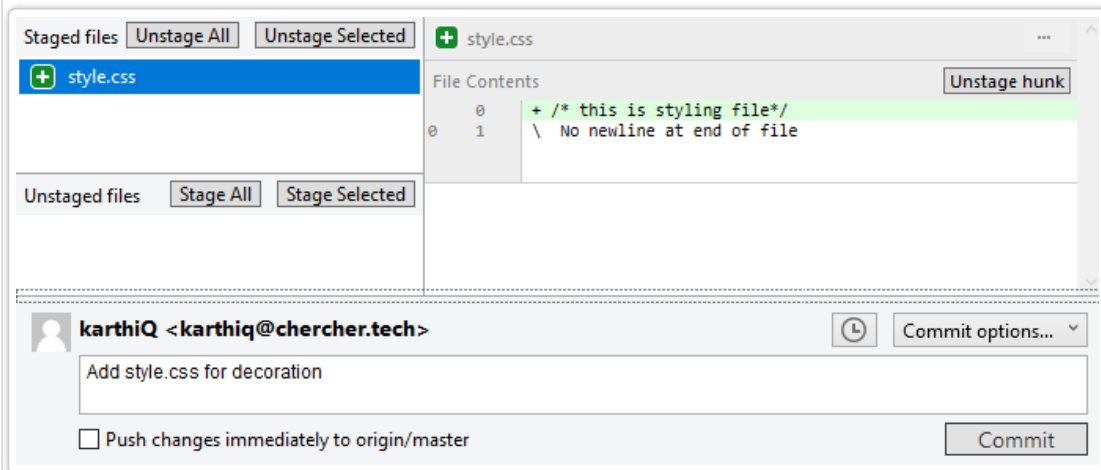


Similar to the stage, we can un-stage the files by selecting a file and then clicking unstage selected, unstage all buttons.

Commit files in SourceTree :

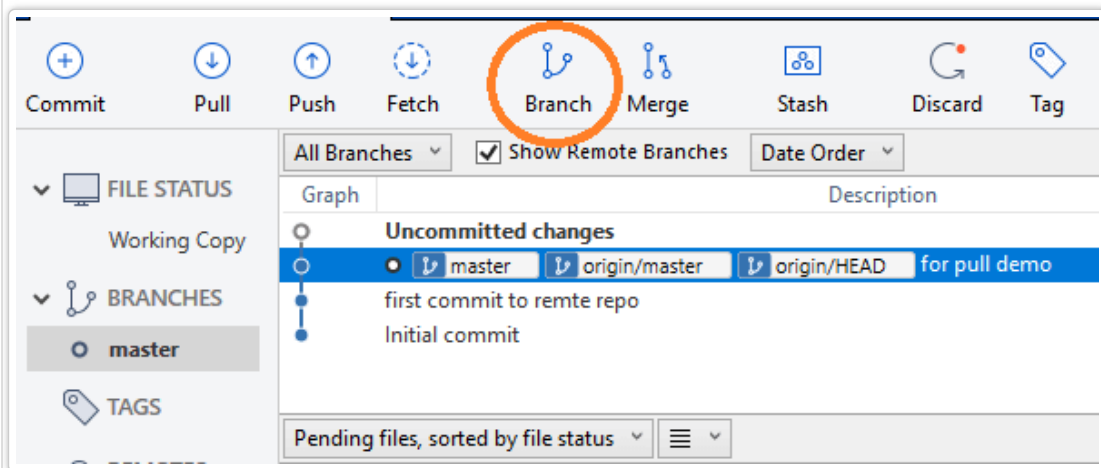
We can commit the files in SourceTree by clicking the commit icon at the top left corner in the SourceTree.

Once we click the commit button, then we have to select the staged files to commit, and provide a description for the changes and press the commit button at the bottom of SourceTree.

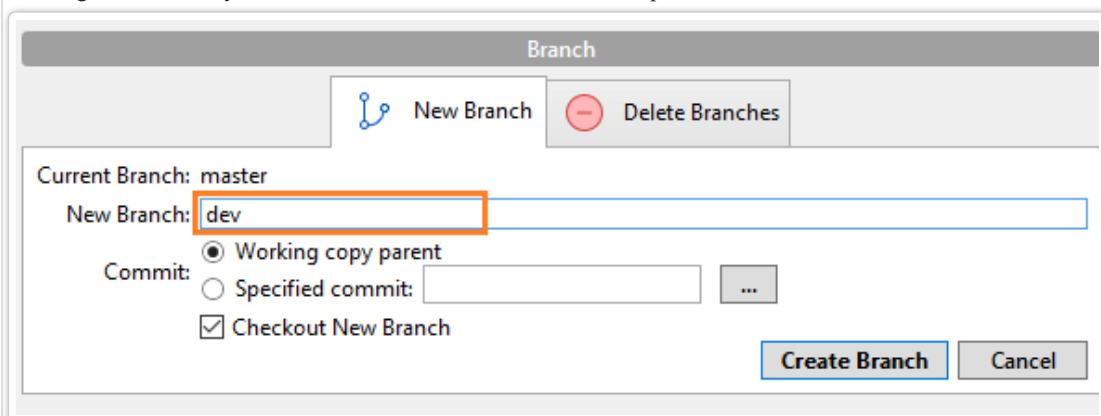


Branching in SourceTree :

We can create the dev branch in SourceTree by clicking the master branch and then clicking branch icon



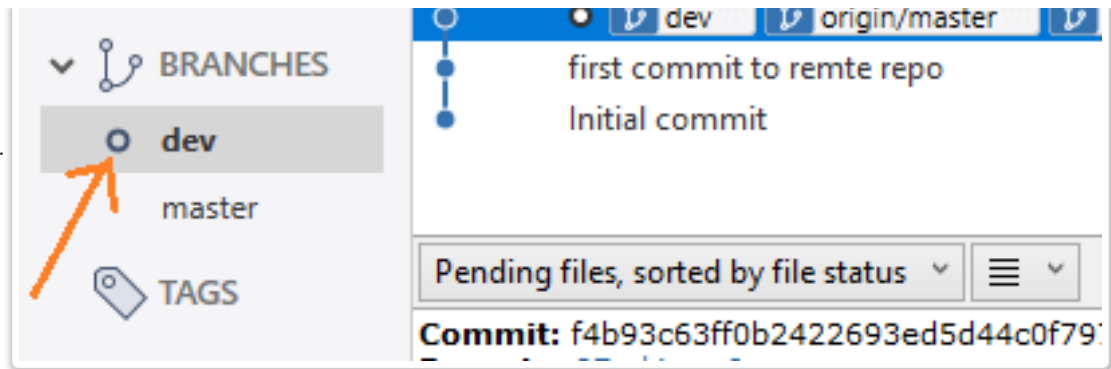
Provide the branch name, also select the working copy parent option to get all the details and code from the master. If you want to start working on new, then you have to select 'Checkout New branch' option



We can verify which branch is currently checked out by looking at the Branches section in SourceTree. The current branch will have a circle at



the front of the branch.



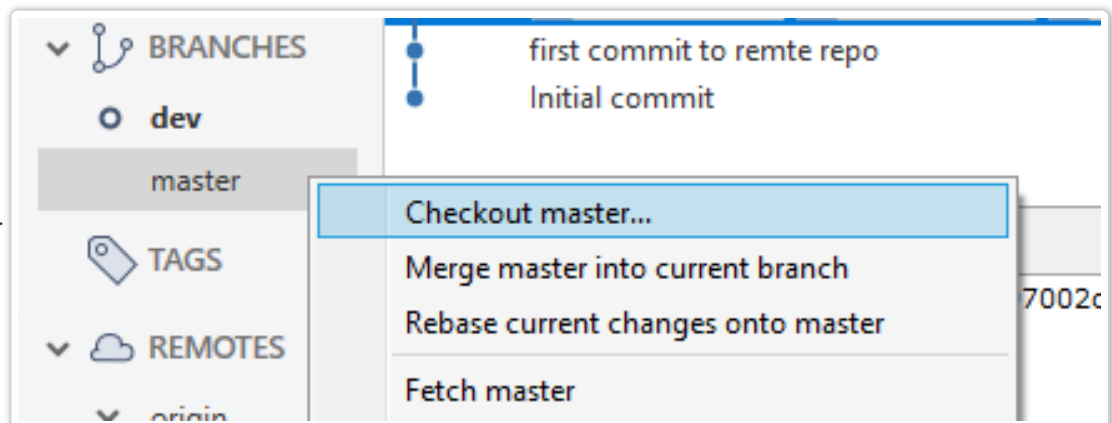
Merge in SourceTree :

Merge is a headache in any version control tool, because if we remove something that we have not developed, then it may create a scene in the team.

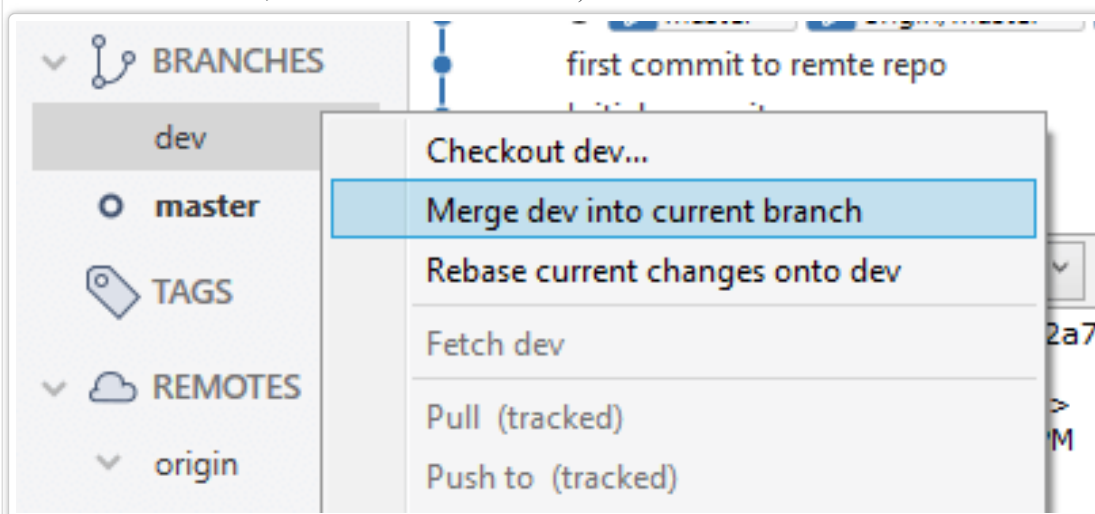
The below example shows the merge of two different branches without any conflicts: Let's add and commit a file in the repository, remember we are working in dev branch now. Do you remember, earlier I said we have to be in the main [master] branch to which we are going to merge some sub[dev] branch.

We are currently in the dev branch, so let's switch to the master branch. We can switch by right-clicking on the required Branch and selecting

the Checkout branch.

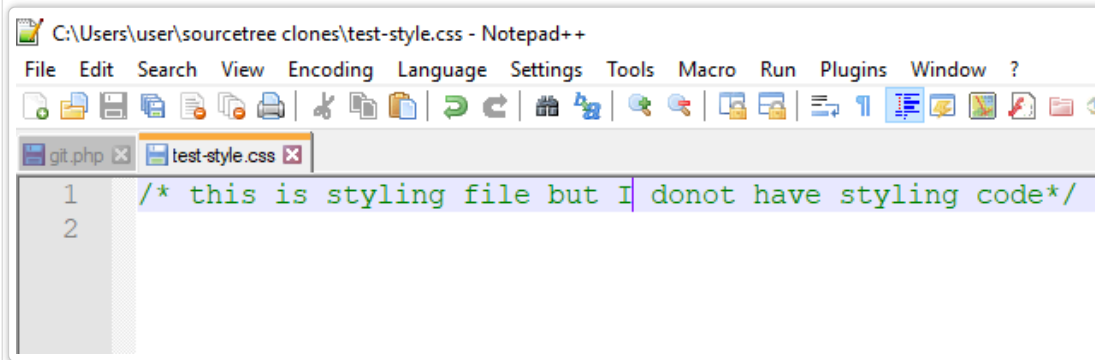


To Merge the sub-branch, we have to right-click on the sub-branch and select the Merge sub-branch with the current branch (we already switched to Master branch, so our current branch is master).

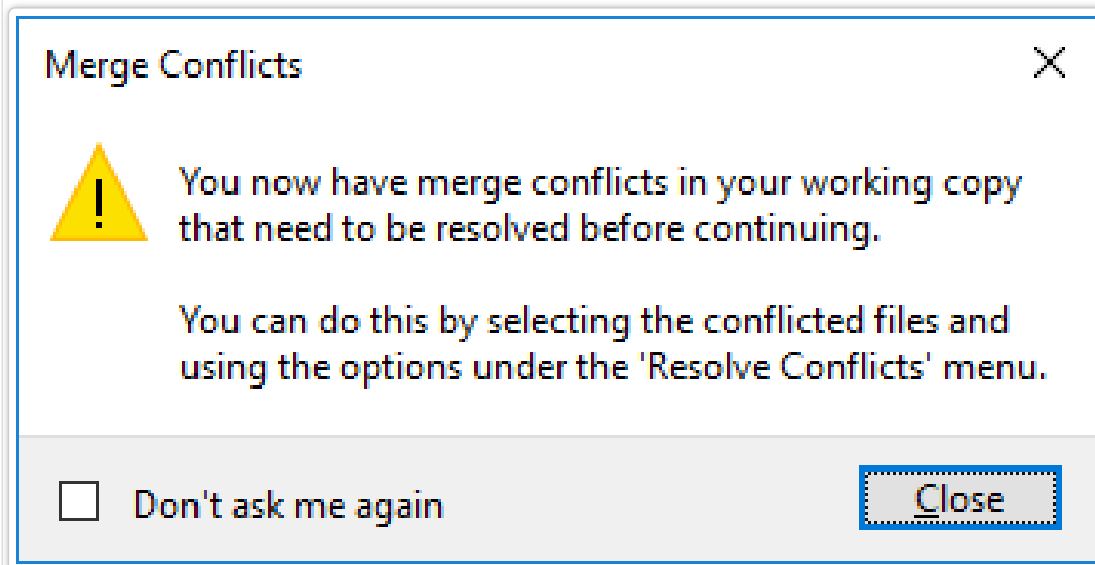


We have seen how to merge without any conflicts, but if we have any conflicts between files, then we have to resolve them, and then we have to merge.

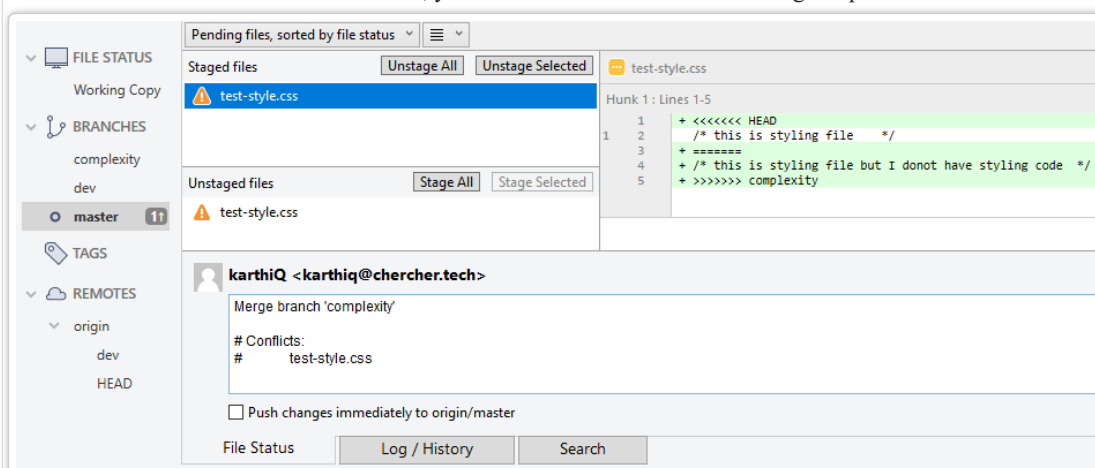
Let's create a branch called complexity from the master branch and edit the content of a file.



Switch to master and try to merge the sub-branch (complexity) with Master, now will see below pop up. Click Close button

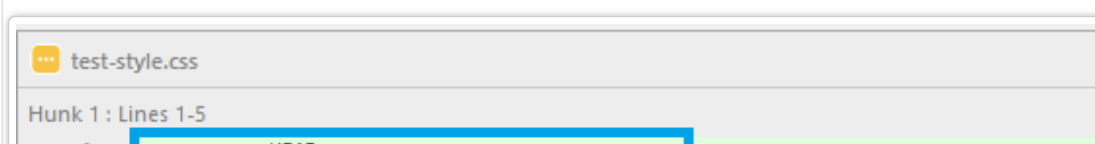


Now select the file status tab at the bottom, you can see the file conflicts at the right top corner.



In the below image, the first changes are with <<< which means that it is the Main branch value.

The second change is with >>> which means this is sub-branch value.



```

1 2  /* this is styling file  */
3
4  + /* this is styling file but I donot have styling code */
5  + >>>>>> complexity

```

We have to keep only the change that we want to keep, some editor may give you an option to choose/remove the values, but if you are using normal editor, then we have to remove lines manually.

test-style.css

Hunk 1 : Lines 0-0

```

0  - /* this is styling file but I donot have styling code */
0  + /* this is styling file  */

```

Once you resolve the conflicts, then you only need to push the code.

Push the code :

You just need to click the Push icon in the top, it will ask for the branches to push. Select the branches carefully and push it.

Push : sourcetree clones

Push to repository: origin https://karthiq@bitbucket.org/karthiq/demorepo.git

Push?	Local branch	Remote branch	Track?
<input checked="" type="checkbox"/>	complexity	complexity	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	dev	dev	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	master	master	<input type="checkbox"/>

☒ Select All

☒ Push all tags ☐ Force Push

Push Cancel

You can verify the pushed code in bitbucket.

karthiq

DemoRepo

Here's where you'll find this repository's source files. To g ▼ 📁 /

🔗 master ▼

Filter branches

Branches **Tags**

complexity

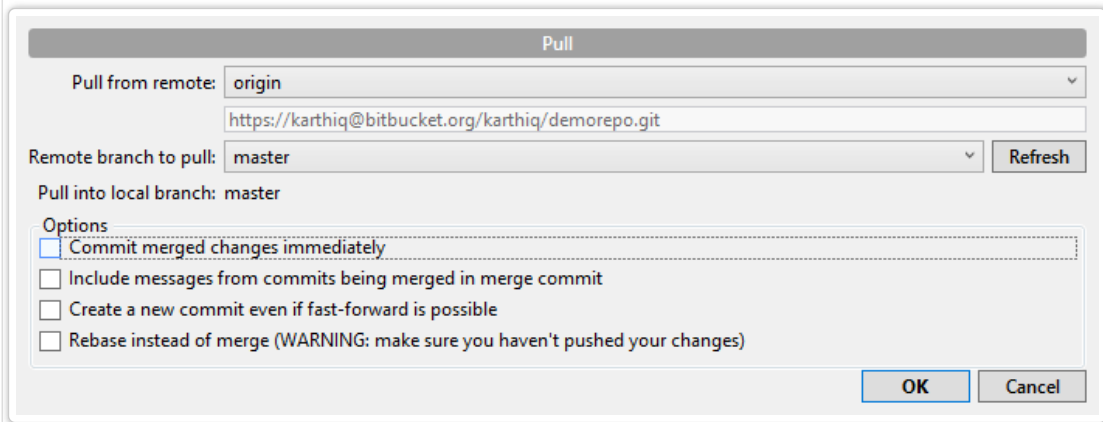
dev

Name	Size
AddFilePullDemo.text	70 B
README.md	2.56 KB
style.css	25 B
test-style.css	30 B
testfile.txt	62 B



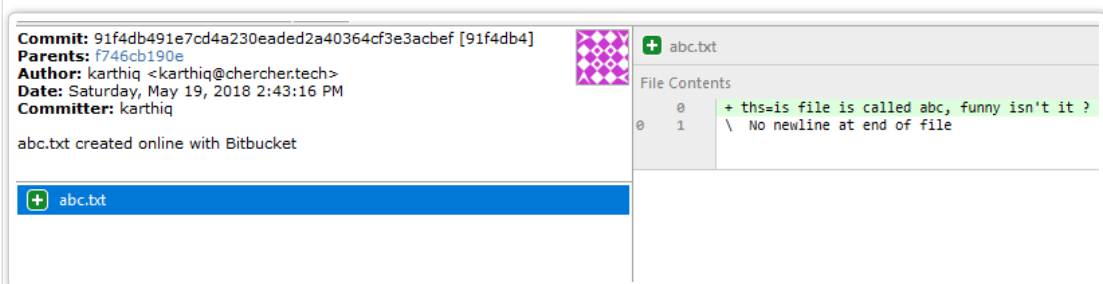
Pull in SourceTree :

Pull is nothing but making our local repository up to date with the remote repository, pull performs two operations 1. fetch, 2. merge



If you select rebase instead of merge, then it will remove all the save commits and uncommitted changes in the local repository.

Result of the pull request

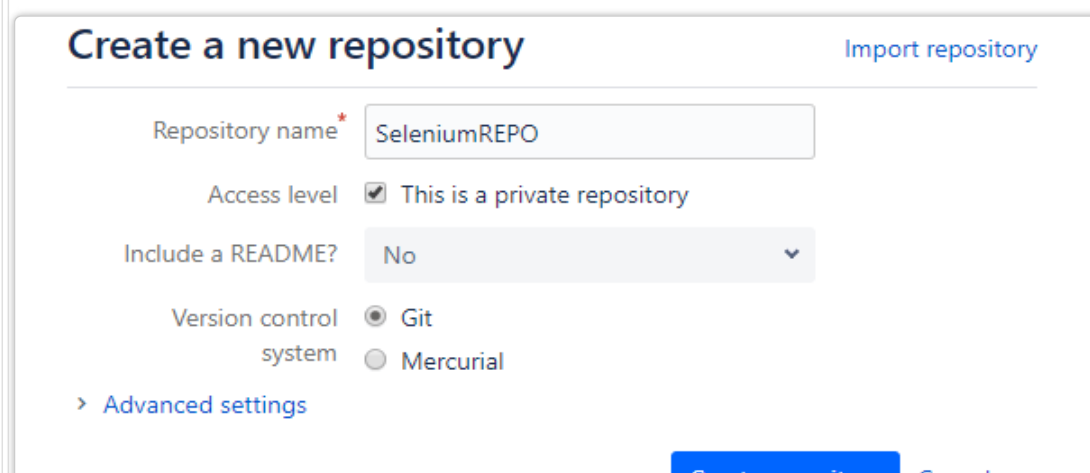


Selenium With SourceTree & BitBucket & Git

Now it is time for cooking, so far we have been learning what the raw materials required for cooking are, so let's start cooking.

Steps to push the Selenium Code to Bit Bucket :


Create a repository in the bitbucket repository.



Open the SourceTree and clone the bitbucket repository into your local

Clone

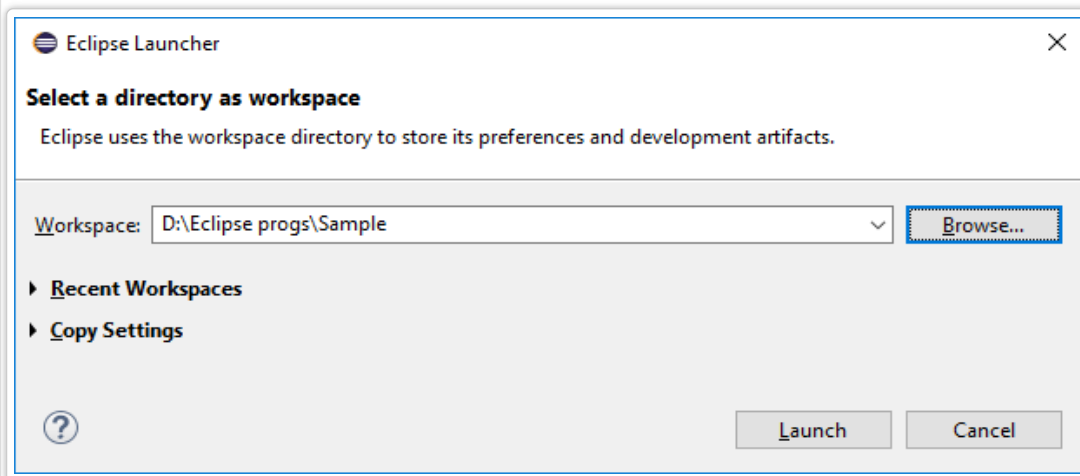
Cloning is even easier if you set up a [remote account](#)

Repository Type:  This is a Git repository

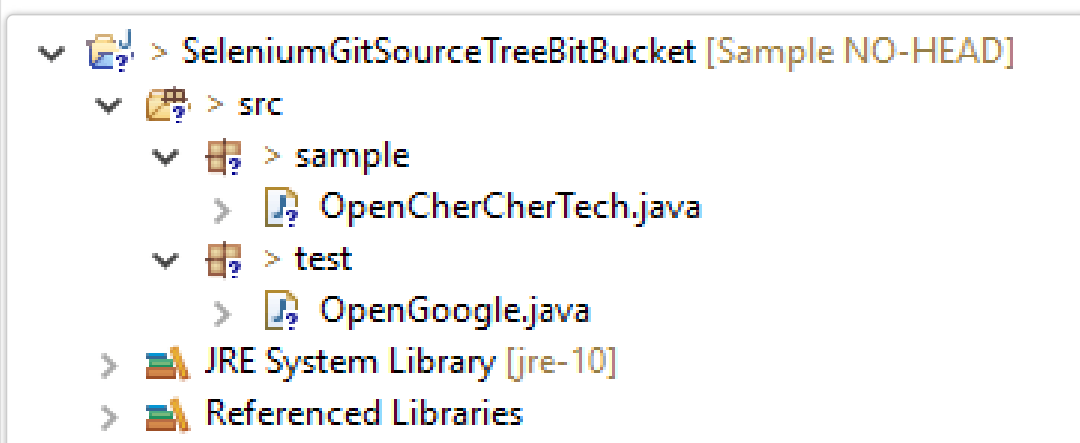
Local Folder:

☒ Advanced Options

Switch your workspace into the sample folder where we have cloned our remote repository



Create a sample java project in eclipse and create a few packages and Classes in it, you may see question marks on the file name.



Before pushing, you should make sure what are the files you want to push, more specifically, what you do not want in the remote repository. As per me, I want only src folder files, and I want to ignore every other folder.

```
# .gitignore file
/*
SeleniumGitSourceTreeBitBucket/jars
SeleniumGitSourceTreeBitBucket/bin
!/SeleniumGitSourceTreeBitBucket/
```

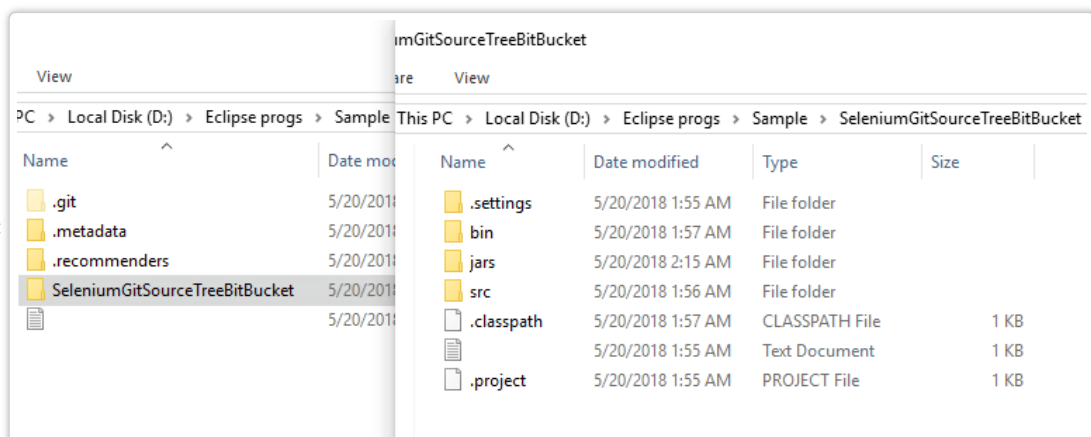
/* ignore everything -> in the current folder.

SeleniumGitSourceTreeBitBucket/jars -> ignore everything under SeleniumGitSourceTreeBitBucket/jars folder

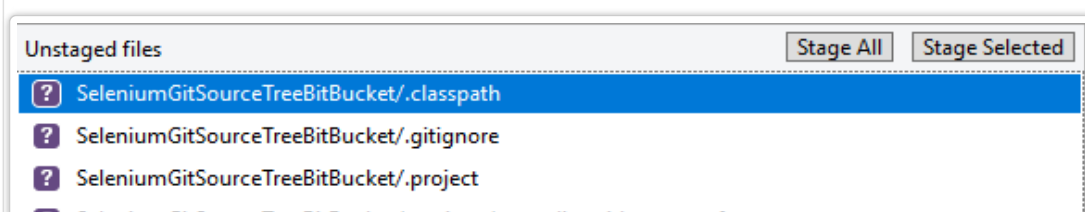
SeleniumGitSourceTreeBitBucket/bin -> ignore everything under SeleniumGitSourceTreeBitBucket/bin folder

!/SeleniumGitSourceTreeBitBucket/ -> donot ignore the files under this folder [basically consider this folder for pushing]

Folder structure:

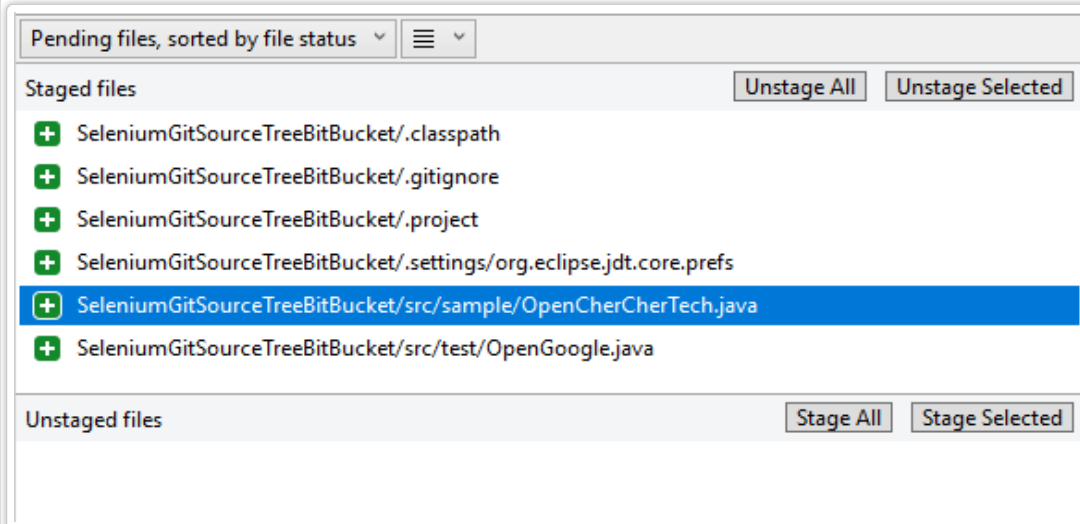


Open your sourceTree, and check the unstaged files, it should match with gitignore

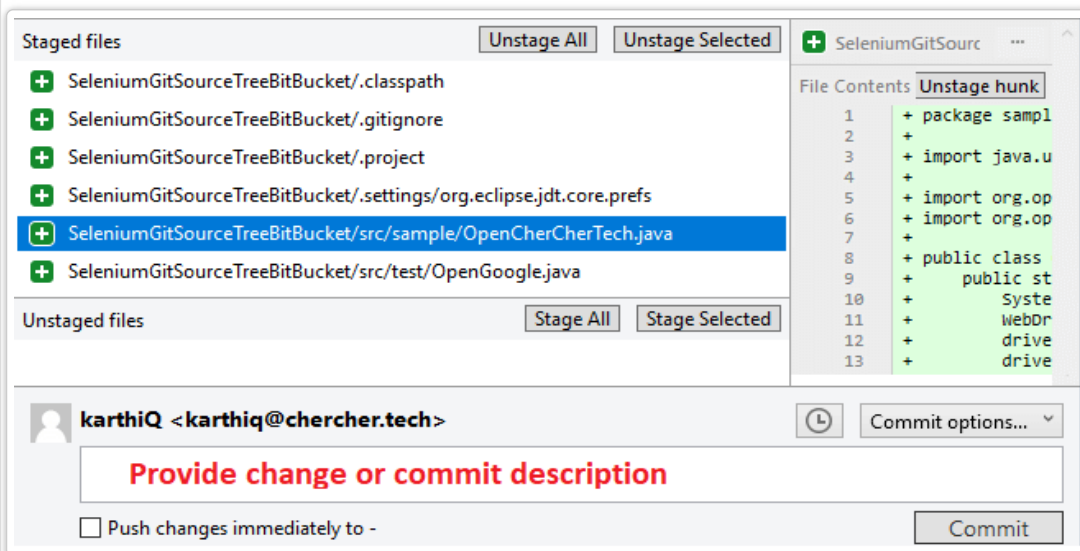


- ❌ SeleniumGitSourceTreeBitBucket/.settings/org.eclipse.jdt.core.prefs
- ? SeleniumGitSourceTreeBitBucket/src/sample/OpenCherCherTech.java
- ? SeleniumGitSourceTreeBitBucket/src/test/OpenGoogle.java

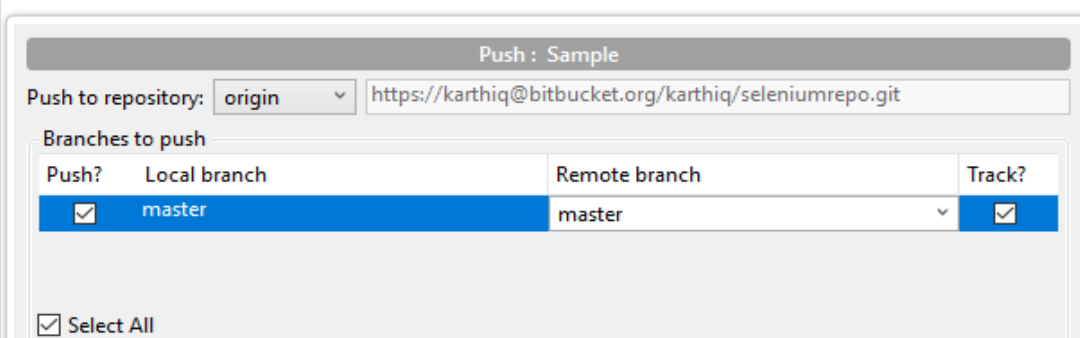
Stage the files present in the unstaged list using Stage All



Commit the files using the commit icon on the top



Before pushing, always take a pull request, now Push the files that we have committed in the last step using push icon at the top.



☒ Push all tags ☐ Force Push

Push

Cancel

Goto BitBucket repository and check whether push is successful or not, if you see the content you pushed, then push was a success.

seleniumrepo / SeleniumGitSourceTreeBitBucket			
Name	Size	Last commit	Message
..			
.settings		3 minutes ago	Committing selenium basic files
src		3 minutes ago	Committing selenium basic files
.classpath	415 B	3 minutes ago	Committing selenium basic files
.gitignore	6 B	3 minutes ago	Committing selenium basic files
.project	389 B	3 minutes ago	Committing selenium basic files

How to revert/rollback the push

Reverting is nothing, but you have pushed something into the remote repository, which was not necessary or wrong code or detail.

If it is a change of one 1- 10 lines in the same file means you can change them in your local and makes new push to override the wrong details, but if code affected was huge, then it is better to revert.

For reverting a particular push you need to get the commit id, you can get it from the bitbucket or the sourcetree.

```
git revert <commit id>
# I am reverting above push.
git revert 1f51b93
```

```

MINGW64; d/Eclipse progs/Sample
Revert "Committing selenium basic files"

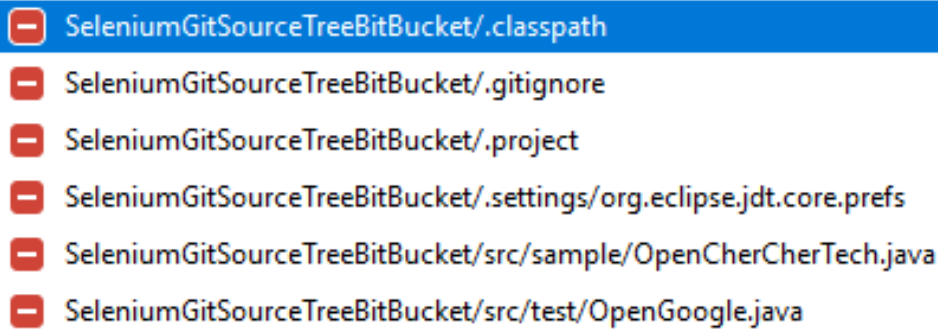
This reverts commit 1f51b93d8c080c659b31c0ba74f55009cff5265b.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is up to date with 'origin/master'.
#
# Changes to be committed:
#   deleted:    SeleniumGitSourceTreeBitBucket/.classpath
#   deleted:    SeleniumGitSourceTreeBitBucket/.gitignore
#   deleted:    SeleniumGitSourceTreeBitBucket/.project
#   deleted:    SeleniumGitSourceTreeBitBucket/.settings/org.eclipse.jdt.cor
e.prefs
#   deleted:    SeleniumGitSourceTreeBitBucket/src/sample/OpenCherCherTech.j
ava
#   deleted:    SeleniumGitSourceTreeBitBucket/src/test/OpenGoogle.java
~
~
~
<Eclipse progs/Sample/.git/COMMIT_EDITMSG[+] [unix] (02:45 20/05/2018)18,0-1 All
"D:/Eclipse progs/Sample/.git/COMMIT_EDITMSG" [unix] 18L, 737C

```

If you see bitbucket now you would be still seeing the files; the reason is we have not pushed the reverted files back

Once we execute the revert command, then we will have files unstaged (waiting as removal of change), now stage all files and commit and push them to clear the changes.


- 
- [-] SeleniumGitSourceTreeBitBucket/.classpath
 - [-] SeleniumGitSourceTreeBitBucket/.gitignore
 - [-] SeleniumGitSourceTreeBitBucket/.project
 - [-] SeleniumGitSourceTreeBitBucket/.settings/org.eclipse.jdt.core.prefs
 - [-] SeleniumGitSourceTreeBitBucket/src/sample/OpenCherCherTech.java
 - [-] SeleniumGitSourceTreeBitBucket/src/test/OpenGoogle.java

We have to verify the bitbucket that all the changes are removed.

karthiq

SeleniumREPO

Here's where you'll find this repository's source files. To give your users an idea of what they'll find

 master ▾



Name	Size	Last commit	Message
------	------	-------------	---------

Recommended Readings

[Maven with Selenium](#)

[Jenkins Integration with Selenium](#)

[Cucumber with Selenium | BDD](#)

[Parameterize Cucumber BDD with selenium](#)

[Featured Page Object Model in Selenium | Feature Framework](#)

[Page Object Model in Selenium | POM Framework](#)

[AutoIT in Selenium | Keyboard & Mouse](#)

[CSV Files in Selenium](#)