Apica AB
http://www.apicasystem.com

ZebraTester

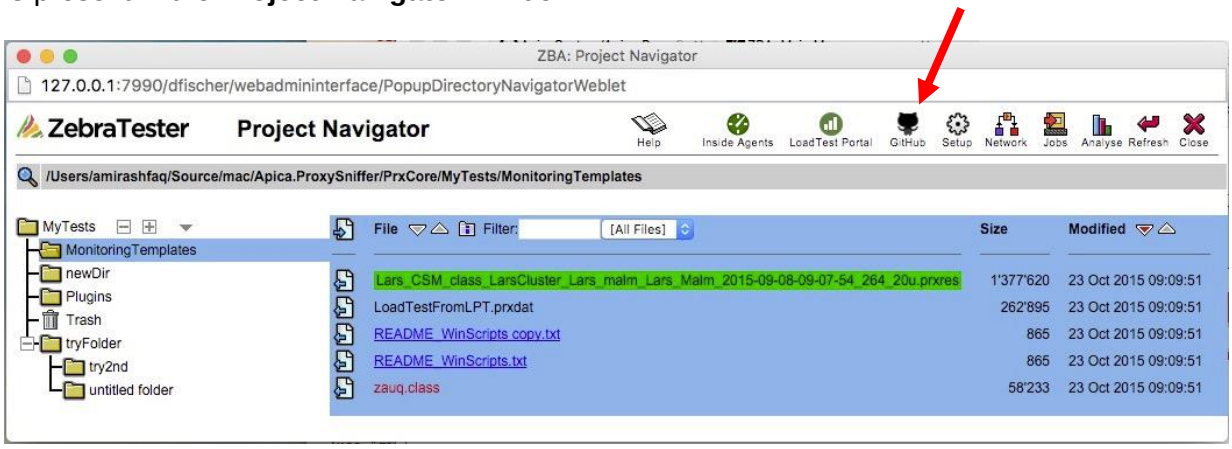
GitHub Integration Manual

Version 5.4
English Edition

Table of Contents

1. Overview, Installation and Configuration

The “Git Integration” allows the use of distributed version control system Git. The **GitHub** icon is present in the **Project Navigator** Window.



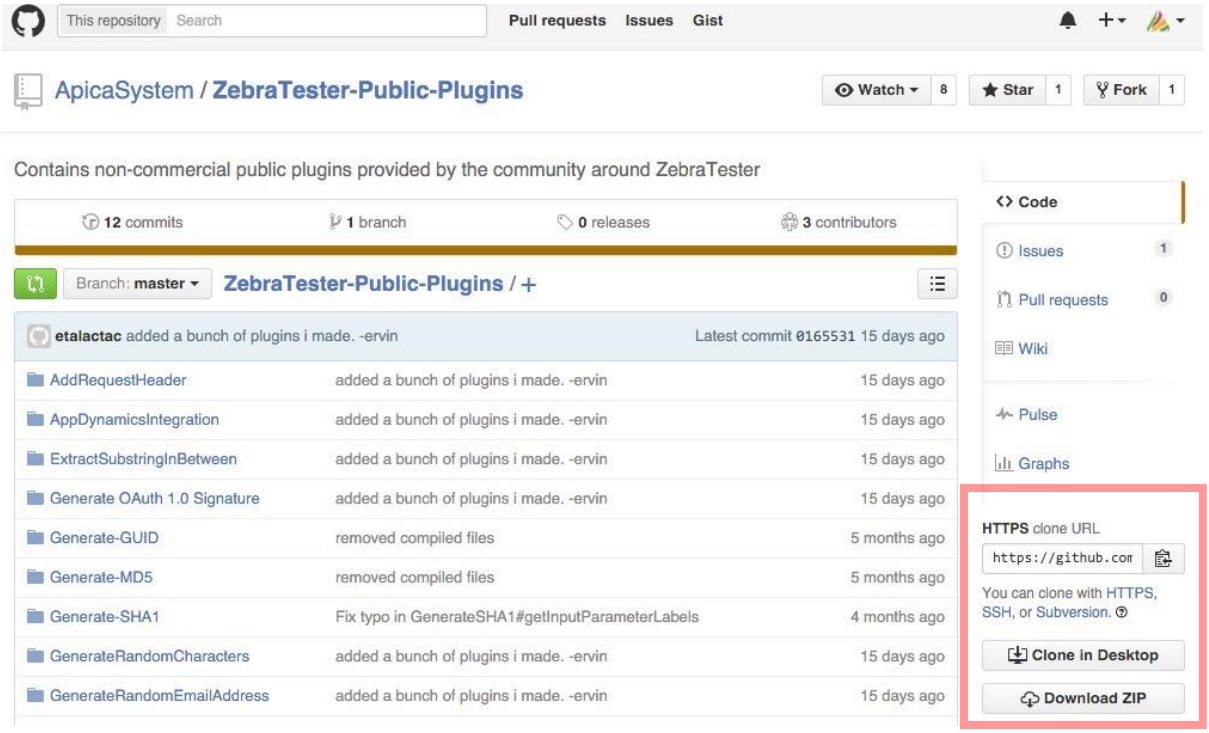
Supported ZebraTester Versions

The ‘GitHub integration’ requires ZebraTester version 5.4-E or later.

1.1. GitHub Server Configuration

There are several ways to clone repositories available but **ZebraTester GitHub-integration** supports **https** and **ssh** based connection, and both public and private repositories can be configured to be used with the GitHub integration.

For HTTPS type of connection, `https://` clone URL is needed and along with that you also need to provide username and password. For GitHub specifically, its very easy to acquire the clone URL.
On the right panel of GitHub repository main page there is a Clone tool available.



Copy the URL by clicking the button on to right side of the Url, and then fill out the Git integration form accordingly.

For SSH, which is a secure protocol, you need to clone the SSH url in a similar way, but other then that, you have to generate SSH key pair. Or you can use any exiting one if you already have, but for GitHub it's not allowed to use any key twice.

For SSH key generation and Adding it to your account it's recommended to follow the manual provided by GitHub.

https://help.github.com/articles/generating-ssh-keys/

Upon completion, press the Apply Configuration >>> Continue Button and the settings will be saved and you will be forwarded to the Step two of the configuration.

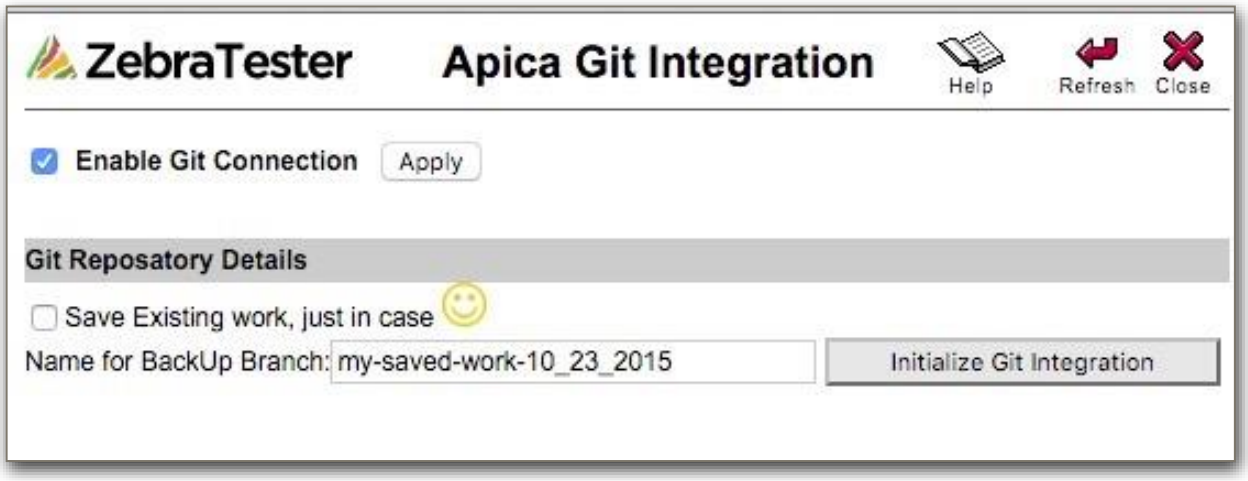
1.2. GitHub Initialization

To start with the GitHub integration, there are two scenarios,

Scenario 1:
Having fresh **ZebraTester** installation and there is nothing important in **My Tests** Folder, and it's not required to save any data from **My Tests** Folder.

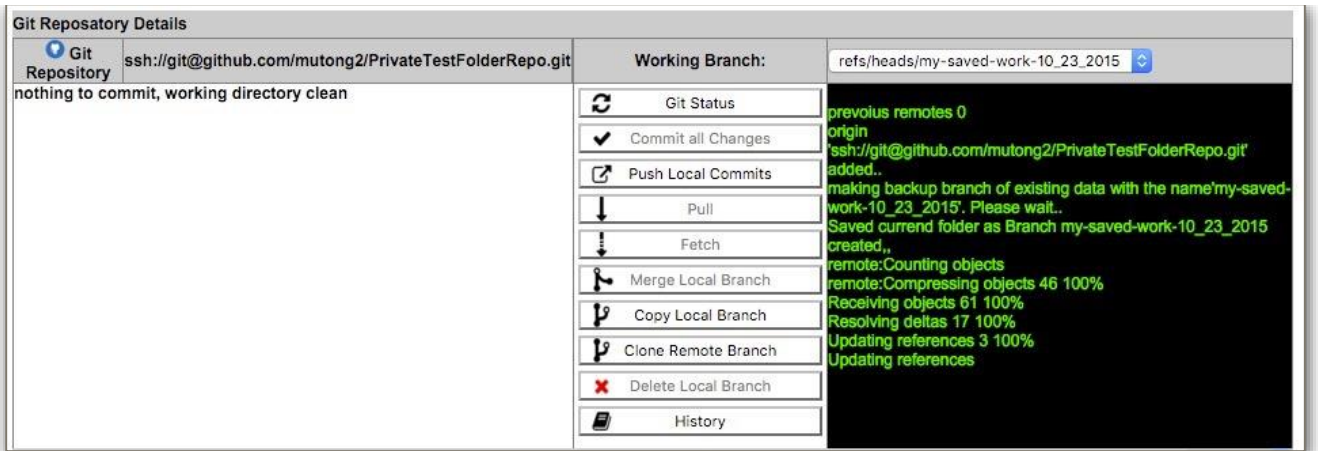
Scenario 2:
My Tests folder is old and has old work saved in it, and it's important to save that data.

When you are in GitHub Integration window for the first time, you will be asked if you are planing to save the existing data, please check the option **Save Existing Work**. this will save the existing Data as a local Branch, that you can later push to the GitHub Server.



And if you are not happy with the Name of Backup Branch you can give any name of your choice, And the push the button **Initialise Git Integration**.

If you chose to save the existing data you will get this as a branch in the next window, otherwise you need to clone remote branch for start working,



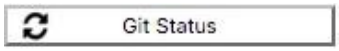
2. Git Operations

if you do not understand what Git is and the fundamentals of how it works, its recommended to have acquire some basic knowledge from following.

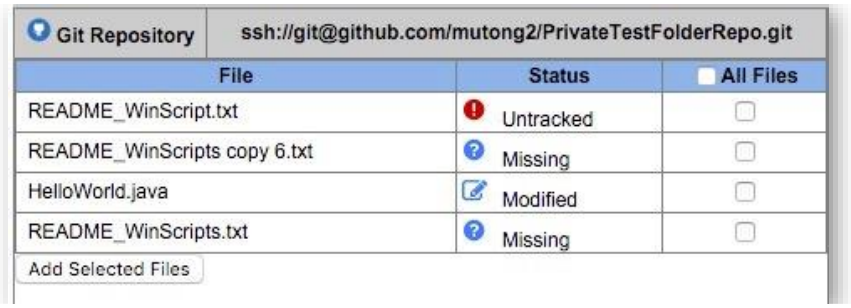
https://git-scm.com/doc

In the middle of the window, there are buttons to perform Git related Operations and following will be the explanation of them

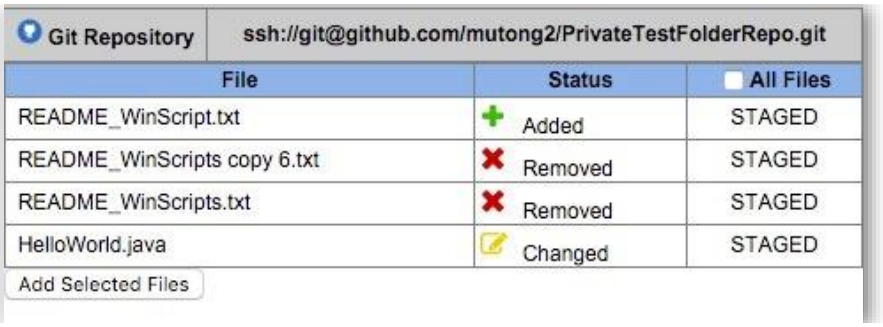
1.1. Git Status



This button is equivalent to Git command **git status** and if there will be any files changed since last commit will be displayed in the left side of the window. Else it will write "**nothing to commit, working directory clean**".

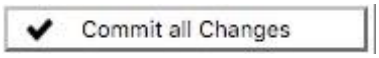


If there are changed files, you can add them by choosing few of them or **All Files**.



After Adding selected file, file will become Staged, and will be part of your next commit.

1.2. Commit

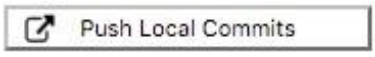


After file are staged, next step will to commit them and after pressing Commit Button , you will be asked to provide Commit Message.

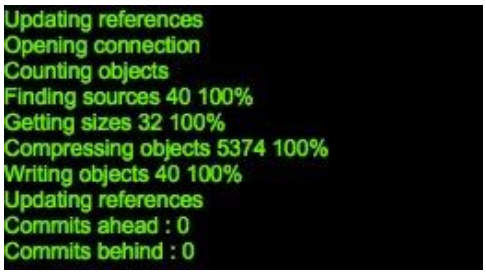
Commit message will be added to the commit. Its OK to leave the box empty, and in this case default msg will be added as a commit msg, that include the current time and the branch name.

e.g. „commit msg @ 12_36_53... in refs/heads/my-saved-work-10_23_2015 “.

1.3. Push



Next step will be to push the commits to the Git Server, and simply clicking the Push Button it will push the commits to Server, and if the local branch has no HEAD on server, new branch will be created.



The log screen will display the progress of every operation.

1.4. Pull

A `git pull` is what you would do to bring a local branch up-to-date with its remote version, and **automatically merges the commits without letting you review them first**. If you don't closely manage your branches, you may run into frequent conflicts.

1.5. Fetch

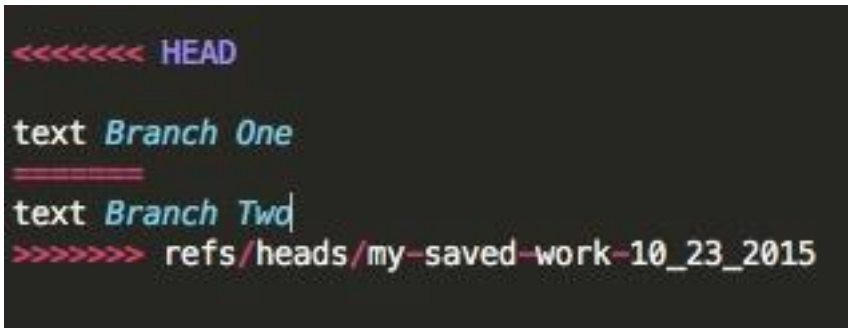
Fetch gathers any commits from the target branch, that do not exist in your current branch and **stores them in your local repository**. However, **it does not merge them with your current branch**. This is particularly useful if you need to keep your repository up to date, but are working on something that might break if you update your files. To integrate the commits into your master branch, you use merge.

1.6. Merge

After you have finished implementing a new feature on a branch, you want to bring that new feature into the main branch, so that everyone can use it. This is achieved by merging the branched. Upon clicking Merge Button you will be asked to chose the branch to merge and the branch to which it will be merged.

A conflict might arise after the merge, and the file that have conflict will be shown in the list of changed files

And when you Open that file using editor, you will see the conflict.



Solution is to change file as you actually want it to be, Add to staged and commit.

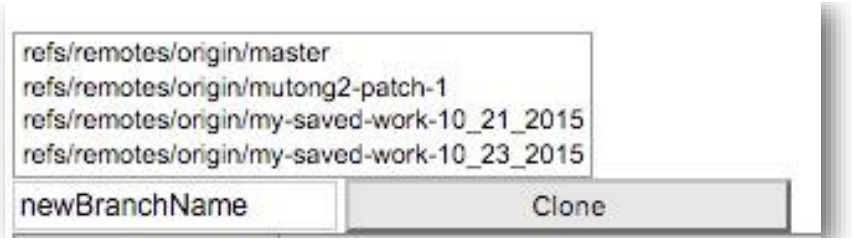
1.7. Copy

If you need exact copy of the branch you are working on, then use this option, it will ask for the name of the new Branch.

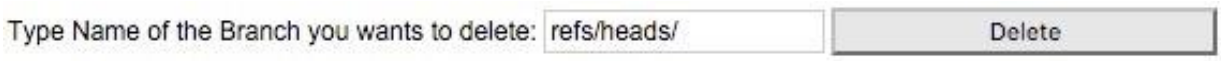
If the parent branch has remote Tracking on the server, child branch will also take it, and both will point to same HEAD branch.

1.8. Clone

Before you can start working locally on a remote branch, you need to have its copy as a local branch, After pressing this Button you will be provided with the list of all remote branches, you can chose, and give it same name, or if you already have a copy, and wants to have new copy with different name, you are allowed to give any name.



1.9. Delete



This will delete the current branch.
You have to type the branch name, and for that you need to have at least one more branch.

1.10. History

This is nice utility function to display the commit history of current branch.