

Working guide for Git on windows environment

BASICS OF VCS

Version Control System (VCS); is a software that helps software developers to work together and maintain a complete history of their work.

Listed below are the functions of a VCS

- Allows developers to work simultaneously.
- Does not allow overwriting each other's changes.
- Maintains a history of every version.

Following are the types of VCS –

- Centralized version control system (CVCS).
- Distributed/Decentralized version control system (DVCS).

Example: Git and GitHub

ABOUT GIT

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance.

STEP 0: The first two things you'll want to do are install git and create a free GitHub account.

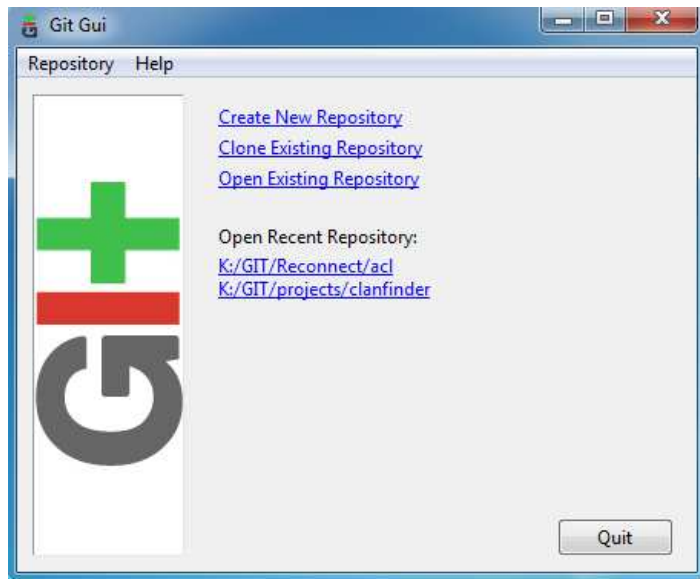
STEP 1: Install Git

First, you need to install on windows; to do so, follow the below steps.

1. Download and install the latest version of <https://github.com/git-for-windows/git/releases/download/v2.28.0.windows.1/Git-2.28.0-64-bit.exe>
2. Use the default options for each step in the installation
3. Remove Git Bash Desktop Icon
4. Go to Start → All Programs → Git → **Git GUI** and make a Desktop Shortcut

STEP 2: Setup SSH key

Open Git GUI.



Now click on **Show SSH Key** under the **Help** Menu.



It's possible that there is already a SSH key on your computer; it's best to remove or backup the key if you do not know where it came from. To do so, simply remove all files within: `C:\Users\<username>\.ssh`. Be sure to replace `<username>` with your Windows username.

You can generate a SSH key by **clicking on the Generate Key button**. When you do so, you will need to supply a passphrase for security purposes. Remember this passphrase; you will need to use it later.

STEP 3: Setup SSH Key with Hosted Git Repository

Github is not the only hosted Git repository available. It is, however, the most popular solution, and we'll use it as an example.

The SSH key you created allows you to push your changes to a hosted repository. So, in order to push changes from your computer, Github needs to know your public SSH key. That is easily accessible; simply **click the "Copy to Clipboard" button**.

Next, you need to provide your hosted repo service with your public SSH key. Similar to Github, most of these sites usually have a tab, called "SSH Keys". Click the tab and add your SSH key to the website.

The screenshot shows the GitHub interface for user 'Blaxus'. On the left is a sidebar with navigation links: Profile, Account Settings, Emails, Notification Center, Billing, Payment History, SSH Keys (highlighted), Security History, Applications, Repositories, and Organizations. Below these are sections for 'PRIVATE REPOS' (0 OF 0) and 'COLLABORATORS' (0 OF 0). The main content area is titled 'SSH Keys' and includes a link to a guide on setting up Git and SSH keys. It displays a table of existing SSH keys:

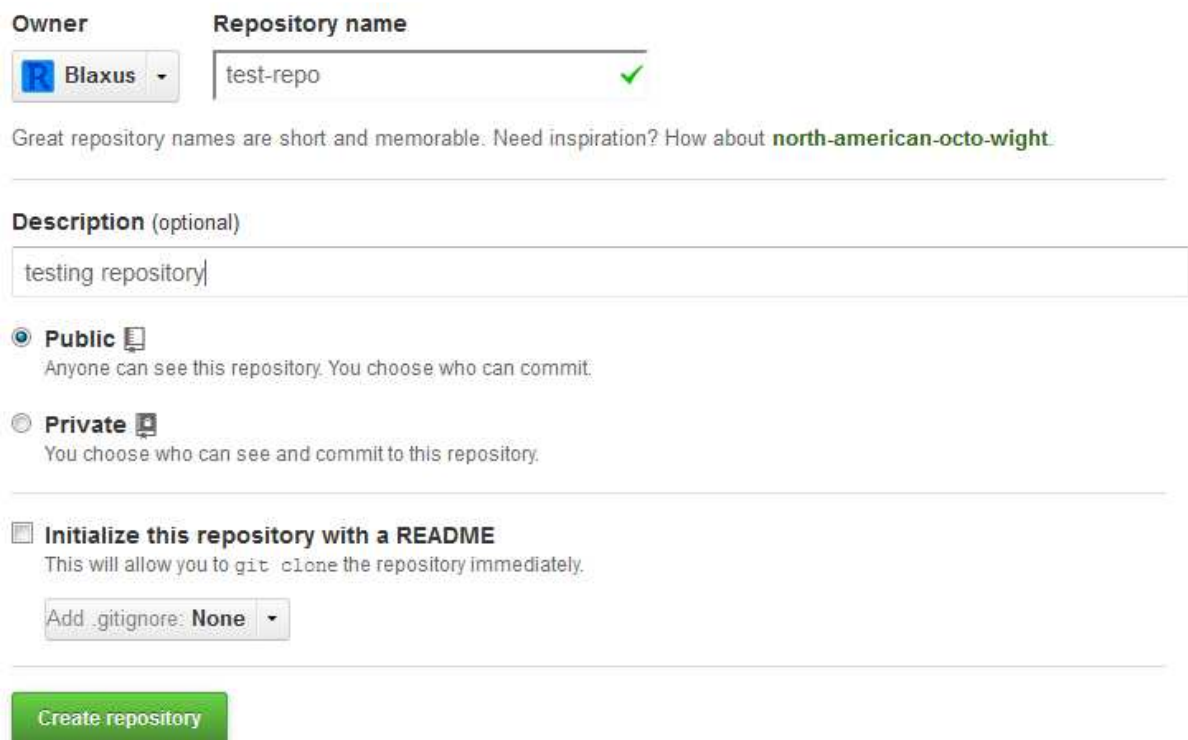
SSH Keys	Add SSH key
MacPC (30:45:76:3a:4d:85:77:86:4a:37:85:37:20:90:02:5a)	Delete
GamePC (a7:5d:5a:37:4d:5a:5a:20:02:c7:4d:5d:37:3a)	Delete
WindowsPc (20:8a:42:5a:10:8b:4a:77:4d:5d:37:3a:4d:5d)	Delete

Below the table is a form to 'Add an SSH Key'. It has a 'Title' field with the value 'HomePc'. The 'Key' field contains a public SSH key starting with 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQBAQ...'. At the bottom of the form is a green 'Add key' button.

The Title field is just a label to identify the SSH key; it is for your purposes only. For example, "WindowsPc"

STEP 4: Need to create new remote repository on Github

When creating a remote repo, Github offers to initialize the repository for you. This is a nice option, but for the purpose of learning how to setup for alternative websites, we will not check the initialize box.



The screenshot shows the GitHub 'Create new repository' form. At the top, there are two fields: 'Owner' with a dropdown menu showing 'Blaxus' and a 'Repository name' text box containing 'test-repo' with a green checkmark. Below these is a hint: 'Great repository names are short and memorable. Need inspiration? How about [north-american-octo-wight](#).' The 'Description (optional)' text box contains 'testing repository'. Under the 'Visibility' section, the 'Public' radio button is selected, with the text 'Anyone can see this repository. You choose who can commit.' The 'Private' radio button is unselected, with the text 'You choose who can see and commit to this repository.' Below this is a checkbox for 'Initialize this repository with a README' which is unchecked, with the text 'This will allow you to `git clone` the repository immediately.' Underneath is a dropdown for 'Add .gitignore' set to 'None'. At the bottom is a green 'Create repository' button.

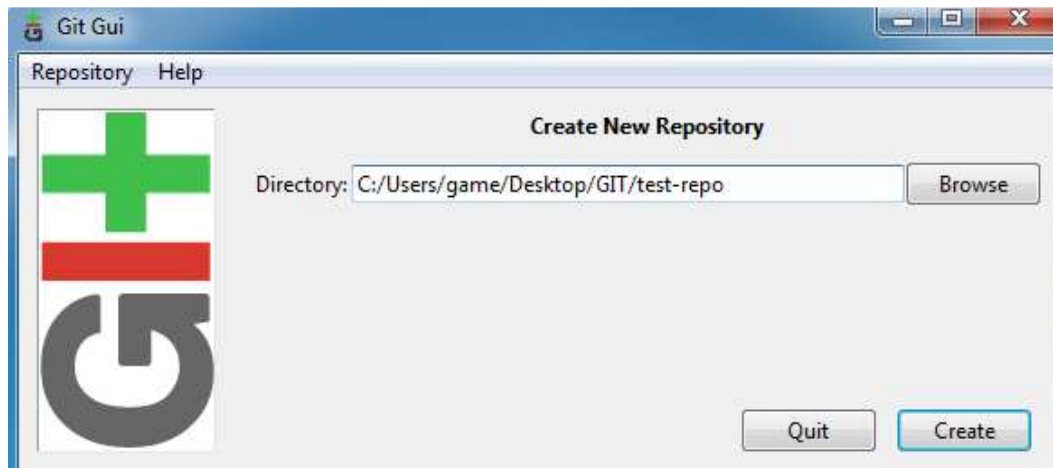
STEP 5: Create a Local Repository

In the Git GUI; clicking on "Create New Repository".



Select the location you wish to store your repository in. It is important to note that the selected repository location **MUST NOT** exist.

So select the location you wish, and append the name of the folder you want the repository to be in, like this:



In order for this new repository to be initialized, you must first create a file, any file, in your local repo. Then, you must Commit and Push to the remote Git repository location.

We'll review committing and pushing in **STEP 7**; I recommend you skip ahead if you do not wish to clone a repository.

Your remote Git location should look like similar to this: [git@github.com:Username/repository-name.git](https://github.com/Username/repository-name.git).

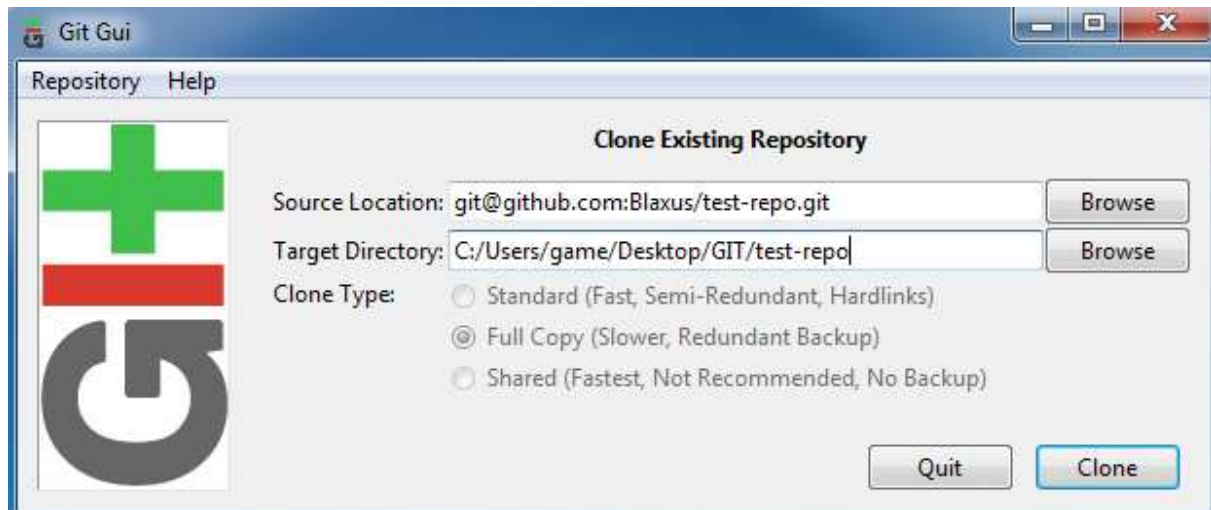
STEP 6: Clone a Remote Repository to a Local Repository

As I noted before, Github can provide you with an already initialized repository, and you can get started a lot faster than you normally would.

In order to clone a repository, click on the "Clone Existing Repository" link in the Git GUI window. An existing repository is one that is already initialized and/or has commits pushed to it.

In the Source Location field, fill in the Git remote repository location. The Target Directory field works much like how I showed you how to create a repository earlier.

Short version: select the location and append the folder you want the files to be in. Git will attempt to create it, and it will fail if it already exists.



There you go; now you should be all set to work locally.

STEP 7: Working with the GUI Client

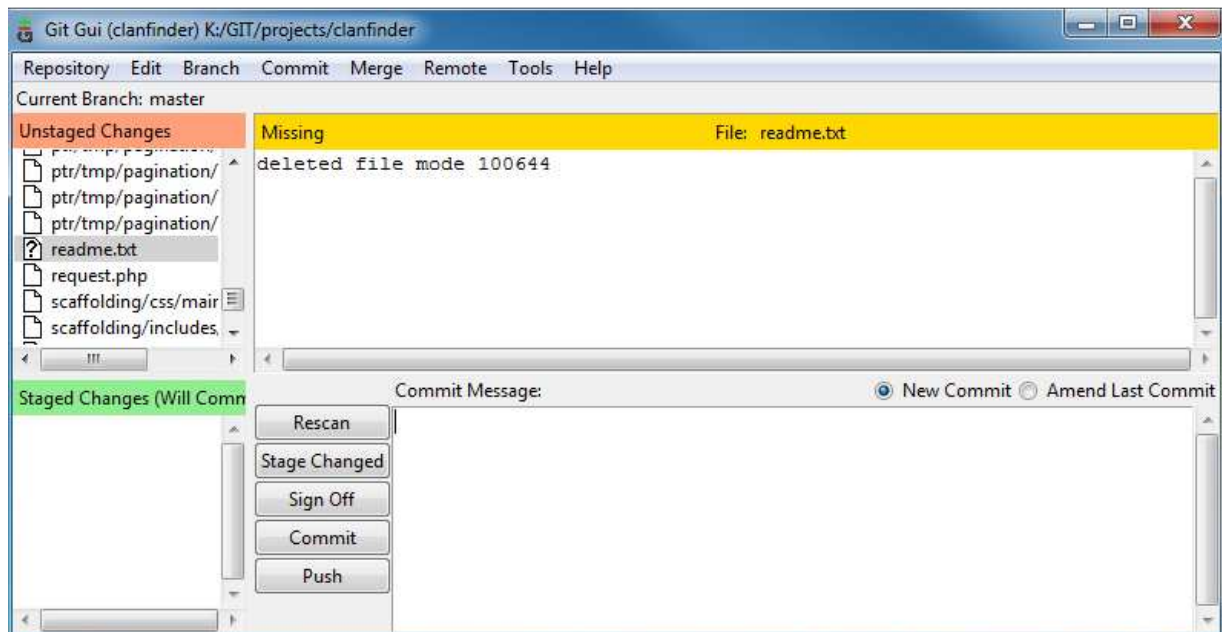
The Git GUI makes it easier to perform Git-related tasks, such as staging changes, commits, and pushes.

Stage Changed

When you move files to a Git directory, you will see all the files in the "Unstaged Changes" window.

This basically means that new files have been added, removed, updated, etc.

You can click on the "**Rescan**" button in order to see any new changes that may have occurred.

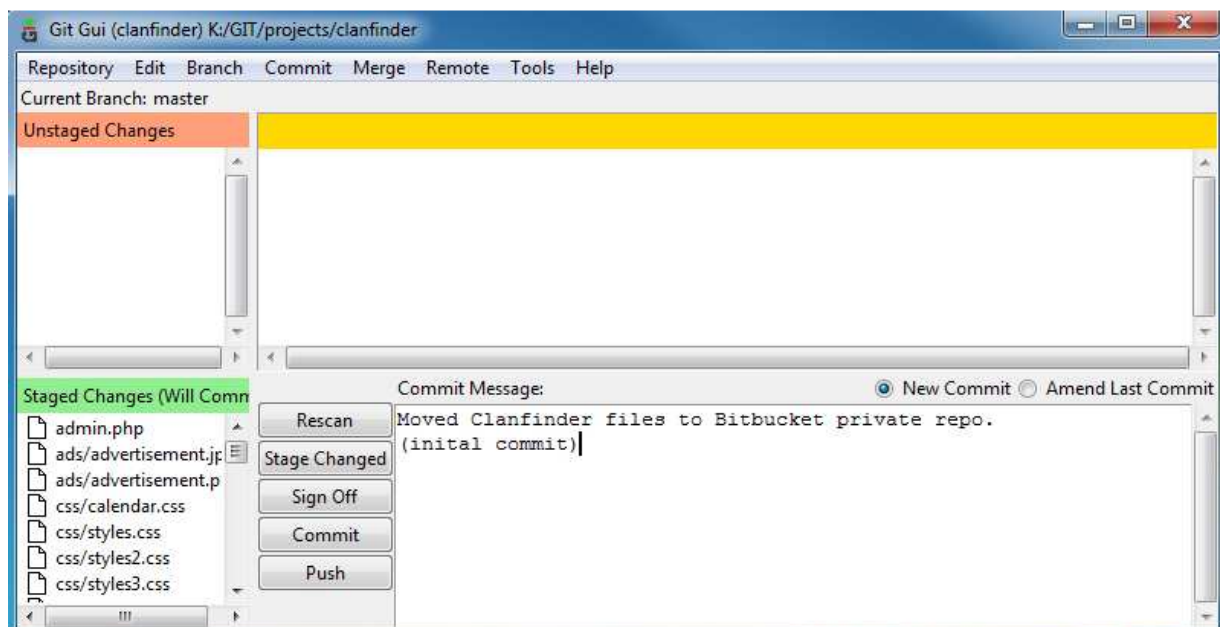


When you click the "Stage Changed" button, it will attempt to add all the new files to the Git index.



Commits:

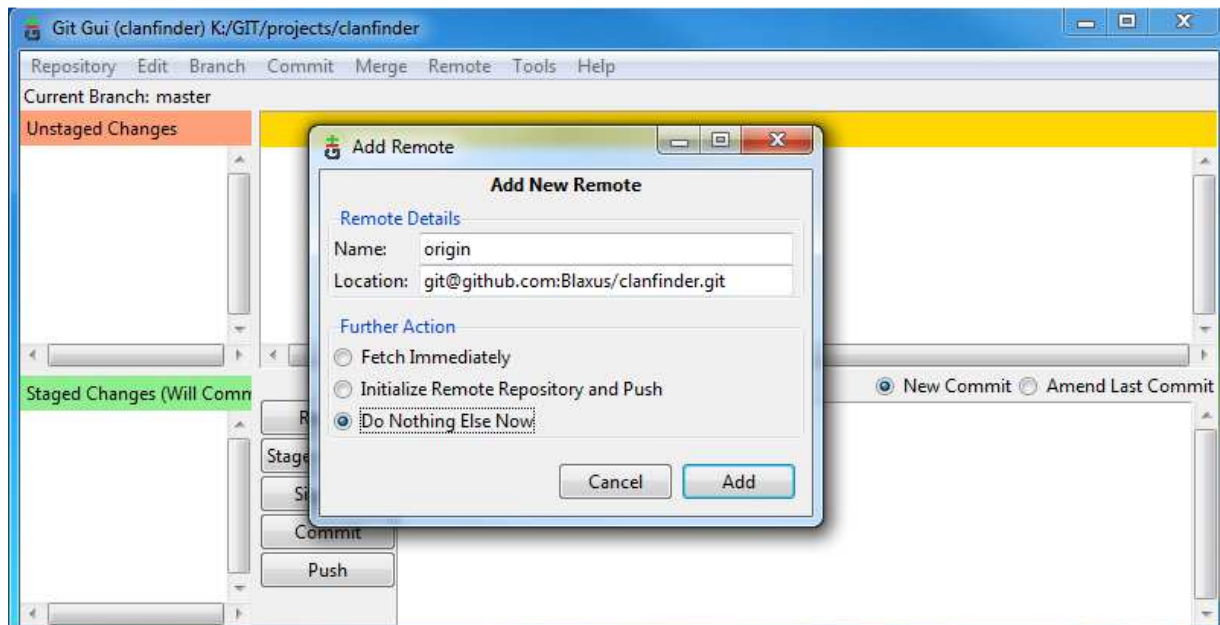
After you've staged your changes, you then need to commit them to your local repository. Type a Commit Message that makes sense to the changes that were made. When you are done, press the Commit button.



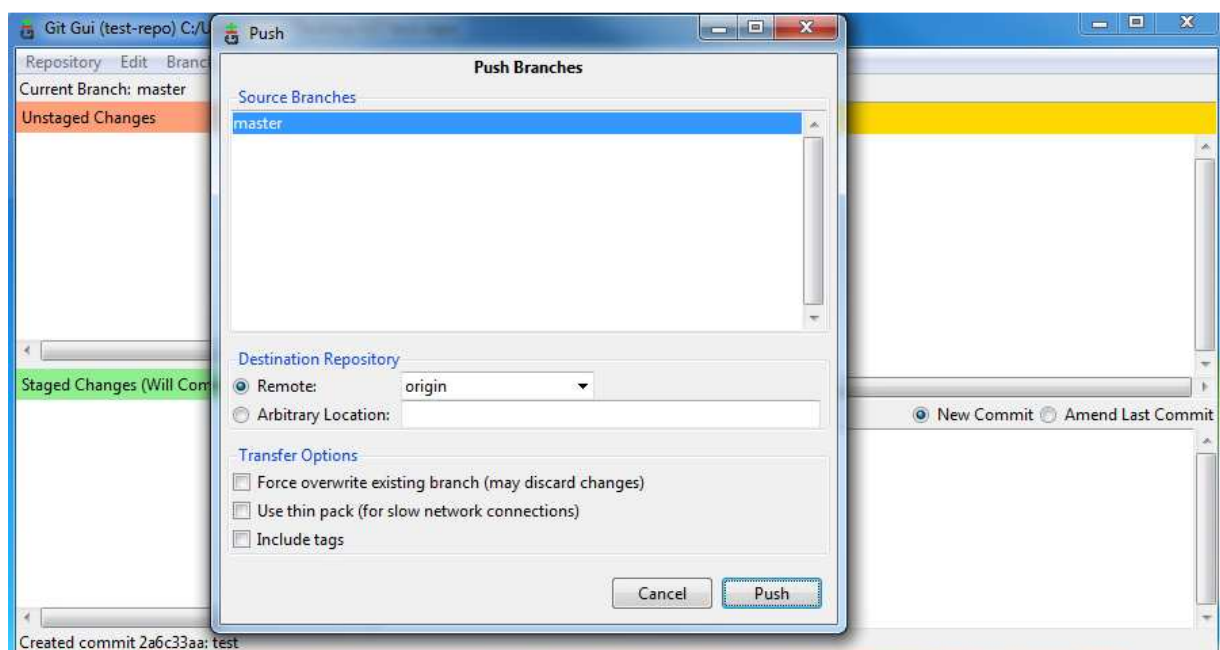
Pushing

Before others can access our new code, we need to push these changes to our hosted repository. Without pushing the changes, others would not be able to access the code.

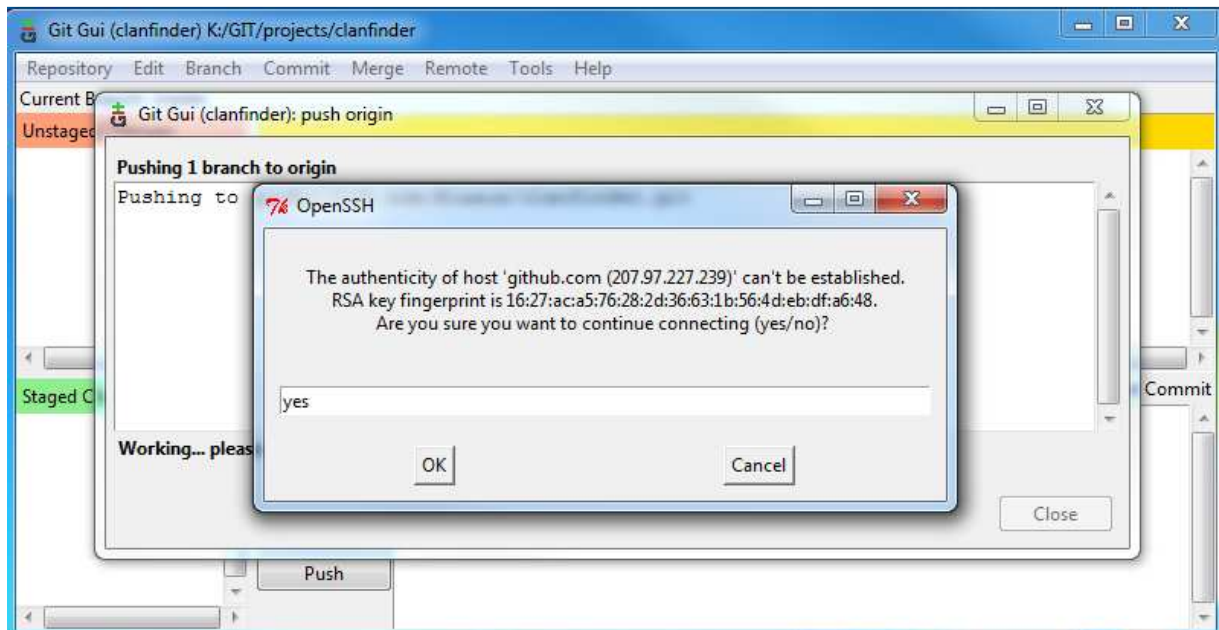
Before we can proceed to push, we need set up a location to push to. Most folks refer to this location as "origin". If you wish, you can select an option in the "Further Actions" area, but in my experience, doing nothing will benefit you the most. You can always clone or push later.



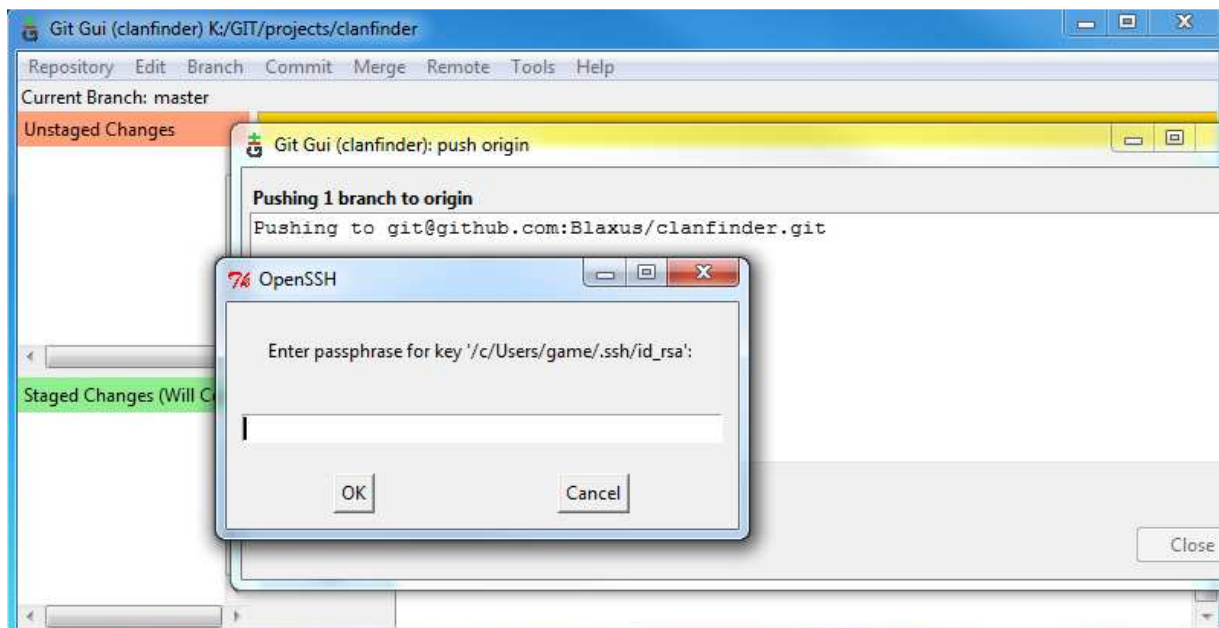
After adding the remote origin, you can simply press the Push button. It will ask you where you want to push to; most likely "origin" will be pre-selected (and it may be your only option). So simply click on the Push button again.



Next, you will be bombarded with window after window. But don't worry; this only happens the first time. Simply follow the instructions given to you.

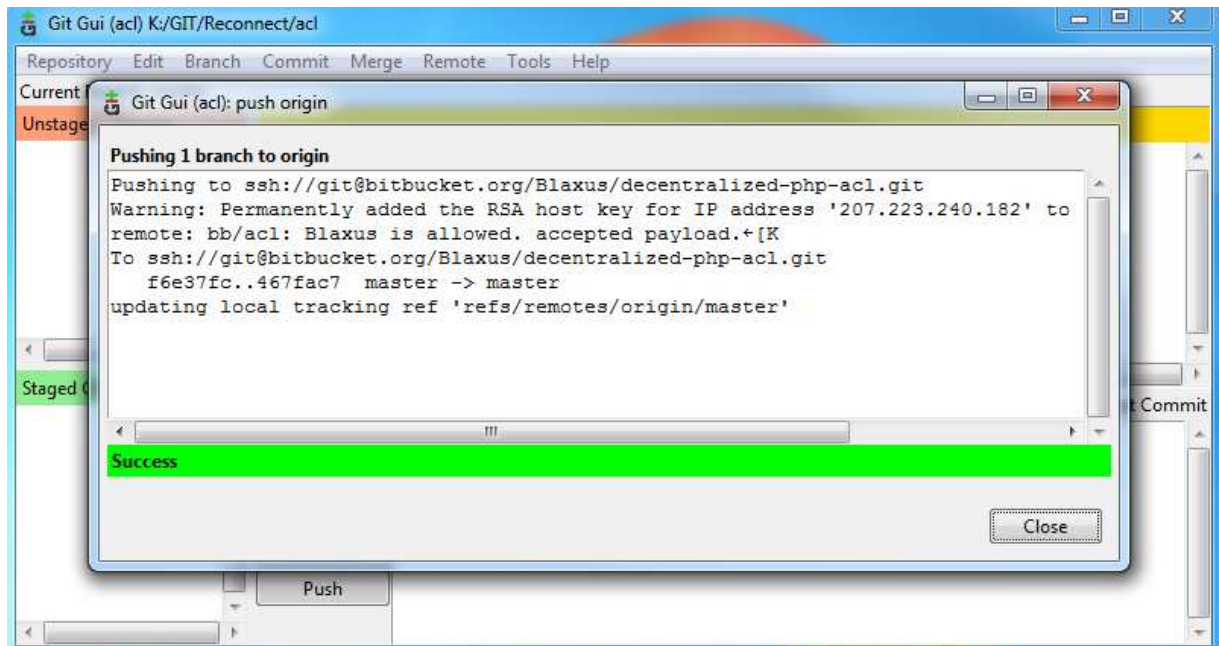


Git will ask you the passphrase of your SSH Key.



Don't panic if you see more then one request of your passphrase. It is completely normal! You don't have to worry as long as you are not told the passphrase was incorrect. Seeing multiple requests for your passphrase usually only occurs once per SSH key.

In the event that your push was complete, you should be greeted with a window similar to this:



END
