* Using OpenStudio with Git and GitHub

https://github.com/NREL/OpenStudio

Note: OpenStudio is currently a private repository. If you cannot access to the above URL, please [contact us](mailto:openstudio@nrel.gov?subject=Request%20for%20OpenStudio%20GitHub%20Access) with the username of your github.com account.

Contents

[Interacting with Git 2](#_Toc361411172)

[GitHub for Windows 2](#_Toc361411173)

[Command Line 2](#_Toc361411174)

[TortoiseGit 2](#_Toc361411175)

[SmartGit/Hg 2](#_Toc361411176)

[Git Extensions 3](#_Toc361411177)

[Creating an SSH Key 3](#_Toc361411178)

[Configuring Git Settings 3](#_Toc361411179)

[Information for Third Party Collaborators 4](#_Toc361411180)

[OpenStudio’s Git Workflow 4](#_Toc361411181)

[Cloning the Repository to your Local Computer 5](#_Toc361411182)

[Listing Branches 6](#_Toc361411183)

[Creating a Branch 6](#_Toc361411184)

[Switching to an Existing Branch 7](#_Toc361411185)

[Committing Your Changes 7](#_Toc361411186)

[Checking For Modifications 8](#_Toc361411187)

[Updating Your Local Repository, and Stashing 8](#_Toc361411188)

[Reintegrating a Branch into Develop 9](#_Toc361411189)

[Pushing All of Your Local Changes to GitHub 10](#_Toc361411190)

[Deleting a Branch 10](#_Toc361411191)

[Other Useful Commands 10](#_Toc361411192)

[Getting the Latest Commit Hash 10](#_Toc361411193)

[Viewing the Log 10](#_Toc361411194)

[Reverting All Working Directory Changes 10](#_Toc361411195)

[File Operations 11](#_Toc361411196)

# Interacting with Git

Depending on your operating system and personal preferences, there are a variety of options for interacting with the repository on your computer:

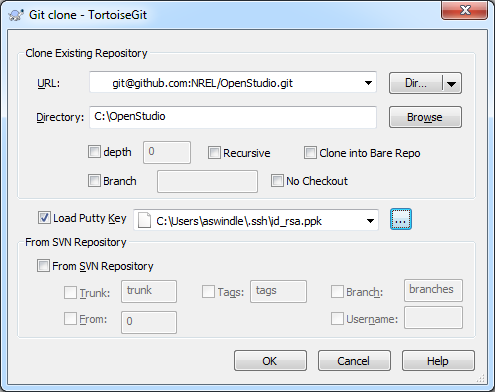
|  |  |
| --- | --- |
| GitHub for Windows A very simple, intuitive GUI for Windows users. This GUI will be sufficient for most third party contributors. Only works with github.com.  [Win](http://github-windows.s3.amazonaws.com/GitHubSetup.exe) |  |
| Command Line Ideal for power users and for running commands directly on the repository. It also includes a basic Git GUI, as well as gitk, a low-level Git GUI. This may be a prerequisite for some of the following tools, and also comes packaged with some of the following tools..  [Win](http://git-scm.com/download/win) – [Mac](http://git-scm.com/download/mac) – [Linux](http://git-scm.com/download/linux) |  |
| TortoiseGit A helpful GUI for normal users who are already familiar with TortoiseSVN and its Explorer integration. In some cases, the interface can be unintuitive.  [Win](https://code.google.com/p/tortoisegit/wiki/Download) |  |
| SmartGit/Hg The best graphical interface for Git – Understand that the non-commercial license can ONLY be used for open-source projects, such as OpenStudio.  [Win](http://www.syntevo.com/smartgithg/download?file=smartgithg/smartgithg-win32-setup-jre-4_6.zip) – [Mac](http://www.syntevo.com/smartgithg/download?file=smartgithg/smartgithg-macosx-4_6.dmg) – [Linux](http://www.syntevo.com/smartgithg/download?file=smartgithg/smartgithg-generic-4_6.tar.gz) |  |
| Git Extensions A useful GUI for visualizing tree changes, Git Extensions also includes context-menu integration in Explorer.  [Win](http://sourceforge.net/projects/gitextensions/files/latest/download?source=navbar) – [Mac](https://git-extensions-documentation.readthedocs.org/en/latest/getting_started.html#installation-mac) – [Linux](https://git-extensions-documentation.readthedocs.org/en/latest/getting_started.html#installation-linux) |  |
|  |  |

# Creating an SSH Key

Establishing a secure connection to GitHub.com can be achieved over HTTPS with a username and password, or with SSH keys and an optional passphrase. SSH keys are highly recommended for security and performance. GitHub has thoroughly documented the process: Carefully follow the instructions for your platform to create an SSH key and add it to GitHub.com

[Win](https://help.github.com/articles/generating-ssh-keys#platform-windows) – [Mac](https://help.github.com/articles/generating-ssh-keys#platform-mac) – [Linux](https://help.github.com/articles/generating-ssh-keys#platform-linux)

On Windows, some GUI tools like TortoiseGit require a Putty SSH key instead of an OpenSSH key. To convert the key you just generated to Putty format, launch Puttygen from the start menu, load your private key (~/.ssh/id\_rsa), and then save the new private key (~/.ssh/id\_rsa.ppk):



# Configuring Git Settings

To start, you should configure some of Git’s global settings (which are saved in ~/.gitconfig). If you leave out the --global flag, the settings will be applied only to your current repository, which is also fine.

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

git config --global user.email johndoe@example.com

git config --global push.default simple

The [user.name setting](https://help.github.com/articles/setting-your-username-in-git) can be anything, although we recommend using your real name. The user.email setting should match an email that you have associated with your GitHub account. Finally, the push.default setting is just to set the push behavior to the new default for Git 2.0 and avoid notices on the command line.

There are [many more configuration settings](http://git-scm.com/book/en/Customizing-Git-Git-Configuration) as well. To see all of your current settings:

git config --list

# Information for Third Party Collaborators

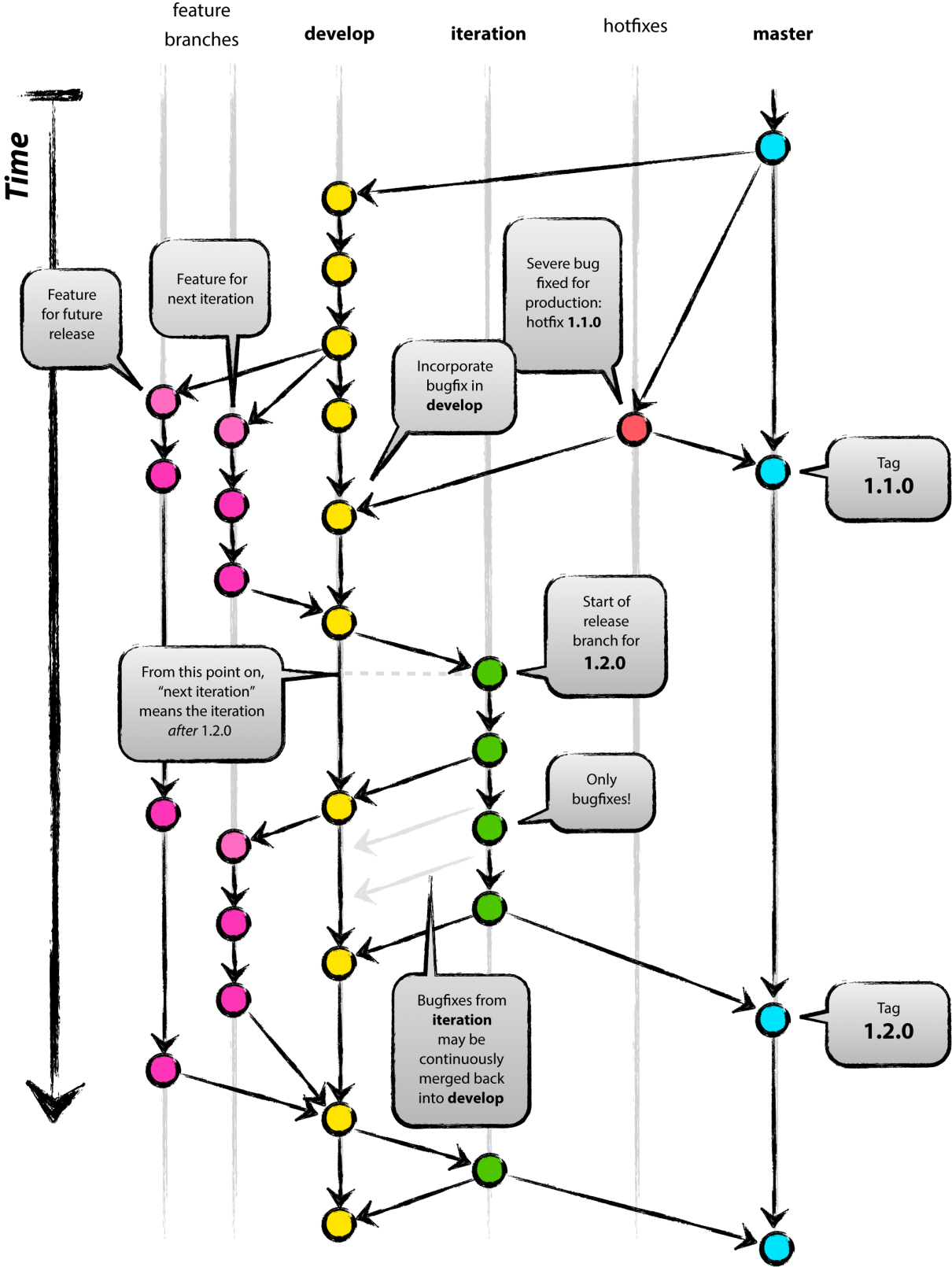
Since you have read-only access to the OpenStudio repository, you should [create a Fork](https://github.com/NREL/OpenStudio/fork) in order to make changes: 

Then, in the following clone step, replace *NREL* in the clone URL with your GitHub username.

Finally, when your changes are ready to be approved for inclusion in the main OpenStudio repository, click the  Compare button in your fork and follow [GitHub’s instructions](https://help.github.com/articles/creating-a-pull-request) for submitting a pull request.

# OpenStudio’s Git Workflow

All work should be completed in feature branches created from the *develop* branch. Biweekly iterations will be branched from *develop* to *iteration*, and releases will be branched from *iteration* to *master*. No commits or development work should be made to *iteration* or *master* unless you are authorized to modify that iteration or release.



# Cloning the Repository to your Local Computer

Now that you have your SSH key configured, you can create a local clone of the repository. If you want to download the latest stable release, select the *master* branch. Otherwise, if you want to work with the latest development code, use the *develop* branch:

git clone -b develop git@github.com:NREL/OpenStudio.git .

The final dot is required if you want to clone into your current directory. Without the dot, this command will create a directory called OpenStudio and clone into that. This clone operation will download ~87MB of files and reconstruct the full develop branch within that directory (which will then total ~337MB). This command also makes all branches available locally.

If you choose to use the HTTPS protocol instead of SSH, you may encounter a SSL certificate issue. This can be caused by enterprise network equipment. If you determine that the warning is a false positive, you can instruct Git to ignore the warning:

git config http.sslVerify "false"

# Listing Branches

When viewing branches, the current branch is marked with an asterisk. To view local branches:

git branch

To view remote branches that you have available:

git branch -r

To view all local and remote branches that you have available:

git branch -a

# Creating a Branch

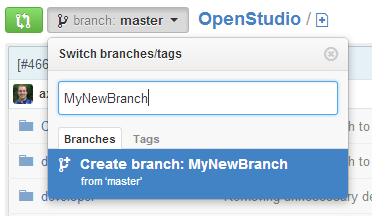
To create a new local branch and switch to it:

git checkout -b mynewbranch

To make this branch available to everyone, push the branch to GitHub:

git push origin mynewbranch

Alternatively, you can do both these steps by typing the new branch name into the branch filter on GitHub and click Create branch:



# Switching to an Existing Branch

To switch to a remote branch that you haven’t already downloaded (branches that were created after you cloned the respository or after your most recent fetch/pull), you should run the following command to get an updated list of remote branches and prune your branches. Pruning removes all branches that have been deleted from GitHub:

git fetch -p

Then, you can checkout and switch to any local or remote branch with the following command:

git checkout mybranch

or

git checkout develop

# Committing Your Changes

To track new files, *and* to stage modified files for commits:

git add mynewfile

or

git add mymodifiedfile

After creating your branch and making changes, commit all your staged changes and modified/deleted files. The first line of your commit message should be a very brief description of the commit, followed by more details:

git commit -m "#3 Bug number & brief commit summary go here (~65 chars)

> More details can go on the additional lines [delivers #12345678]"

Adding a “-a” flag to your commit command will automatically commit all modified files, even if you haven’t explicitly used “git add” on them.

# Checking For Modifications

To check for staged changes, unstaged changes, and untracked files:

git status

or

git status -sb

Adding the “s” flag produces a very concise status, and the “b” flag tells it to indicate the current branch.

# Updating Your Local Repository, and Stashing

To update your local repository and all branches, *and* update the files in your current branch:

git pull

or, if you understand the consequences of rebasing,

git pull --rebase

For more information about rebasing, please review the [online manual](http://git-scm.com/book/en/Git-Branching-Rebasing).

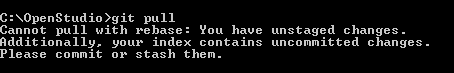
If you have uncommitted changes and you’re not ready to commit them, you will first need to stash your changes, then perform the pull, and then pop the changes:

git stash

git pull

git stash pop

Without stashing and popping, you may see a message like this:



A good example of when you might want to do this is if you’re not ready yet to commit the changes, but for one reason or another you need a clean working directory.

If the pop command triggered merge conflict resolution, the stashed changes won’t be removed from the stash stack. To explicitly remove the last item from the stash, use the following command:

git stash drop stash@{0}

To view the current contents of the stash stack:

git stash list

# Reintegrating a Branch into Develop

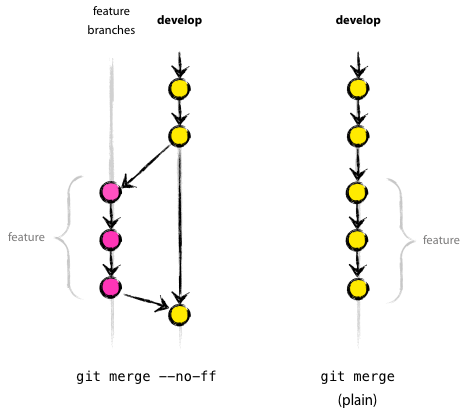
If a code review is necessary and your changes are complete, click the  Compare button in your branch and follow [GitHub’s instructions](https://help.github.com/articles/creating-a-pull-request) for submitting a pull request. After creating the pull request, you can assign it to the bug or ticket owner for review.

However, if your branch does not require a code review, then it can be merged immediately:

git checkout develop

git merge --no-ff mybranch

The no-fast-forward “--no-ff” flag is important for merging to maintain branch history, and it stays consistent with GitHub’s automatic merge settings:



# Pushing All of Your Local Changes to GitHub

When you’re ready to share your changes and commits from any branch with the rest of the team:

git push origin

# Deleting a Branch

To delete a local branch:

git branch -d mybranch

To delete a remote branch directly, use the  button on [GitHub](https://github.com/NREL/OpenStudio/branches). Alternatively, you can use the following command:

git push origin --delete mybranch

Note that if you delete an unmerged branch, the branch and all commits to it will be permanently deleted.

# Other Useful Commands

### Getting the Latest Commit Hash

To produce the SHA1 hashes of the latest commit, such as 932bca9f7dfab0d698dcdc04032762b6525237d5 or 932bca9, use the following commands respectively:

git rev-parse HEAD

or

git rev-parse --short HEAD

### Viewing the Log

To see the full git log or just that last 5 commits with concise output:

git log

or

git log --oneline --decorate -5

### Reverting All Working Directory Changes

To revert all working directory changes and files to the latest commit:

git reset --hard

### File Operations

To delete a file from your file system and the repository:

git rm myfile

To rename or move a file:

git mv myfile mynewfile