cid:image001.png@01D03F4F.E69DEC60

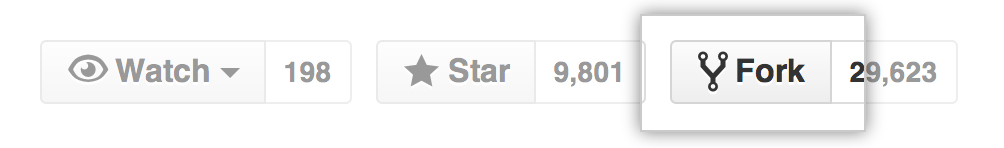
Asentech GIT Developers Guide



|  |  |
| --- | --- |
| **Prepared By:** **Asentech, LLC**  **Head Quarters:** 421 Executive Drive, Princeton, NJ 08540  **Office:** 908 751 4121 Ext: 156 | **Fax:** 866-809-2160  307, Kane Plaza, Mindspace, Off. Link Road, Malad (W), Mumbai 400064  **Office:** +91 226 118 8888 Ext: 30 | **Fax (India):** +91 22 2673 1843 | Innovate Energize Grow  www.asentechllc.com |
|  |  |

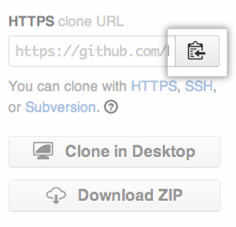
**Fork A Repo**

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original repository. On GitHub, navigate to the octocat/Spoon-Knife repository. In the top-right corner of the page, click Fork.

**Keep your fork synced**

You might fork a project in order to propose changes to the upstream, or original, repository. In this case, it's good practice to regularly sync your fork with the upstream repository. To do this, you'll need to use Git on the command line. You can practice setting the upstream repository using the same octocat/Spoon-Knife repository you just forked!

1. On GitHub, navigate to your fork of the Spoon-Knife repository.
2. In the right sidebar of your fork's repository page, click to copy the clone URL for your fork.



1. Open Terminal (for Mac users) or the command prompt (for Windows and Linux users).
2. Type git clone, and then paste the URL you copied in Step 2. It will look like this, with your GitHub username instead of YOUR-USERNAME:

*$ git clone* [*https://github.com/*YOUR-USERNAME*/Spoon-Knife*](https://github.com/YOUR-USERNAME/Spoon-Knife)

1. Press Enter. Your local clone will be created.

*$ git clone https://github.com/*YOUR-USERNAME*/Spoon-Knife*

*Cloning into `Spoon-Knife`...*

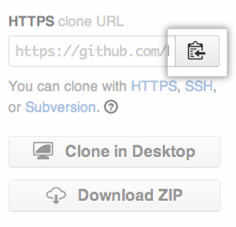
*remote: Counting objects: 10, done.*

*remote: Compressing objects: 100% (8/8), done.*

*remove: Total 10 (delta 1), reused 10 (delta 1)*

*Unpacking objects: 100% (10/10), done.*

1. On GitHub, navigate to the octocat/Spoon-Knife repository.
2. In the right sidebar of the repository page, click to copy the clone URL for the repository.



1. Open Terminal (for Mac users) or the command prompt (for Windows and Linux users).
2. Change directories to the location of the fork you cloned in Step 2: Create a local clone of your fork.
3. Type git remote -v and press Enter. You'll see the current configured remote repository for your fork.

*$ git remote -v*

*origin https://github.com/*YOUR\_USERNAME*/*YOUR\_FORK*.git (fetch)*

*origin https://github.com/*YOUR\_USERNAME*/*YOUR\_FORK*.git (push)*

1. Type git remote add upstream, and then paste the URL you copied in Step 2 and press Enter. It will look like this:

*$ git remote add upstream* [*https://github.com/octocat/Spoon-Knife.git*](https://github.com/octocat/Spoon-Knife.git)

1. To verify the new upstream repository you've specified for your fork, type git remote -vagain. You should see the URL for your fork as origin, and the URL for the original repository as upstream.

$ *git remote -v*

*origin https://github.com/*YOUR\_USERNAME*/*YOUR\_FORK*.git (fetch)*

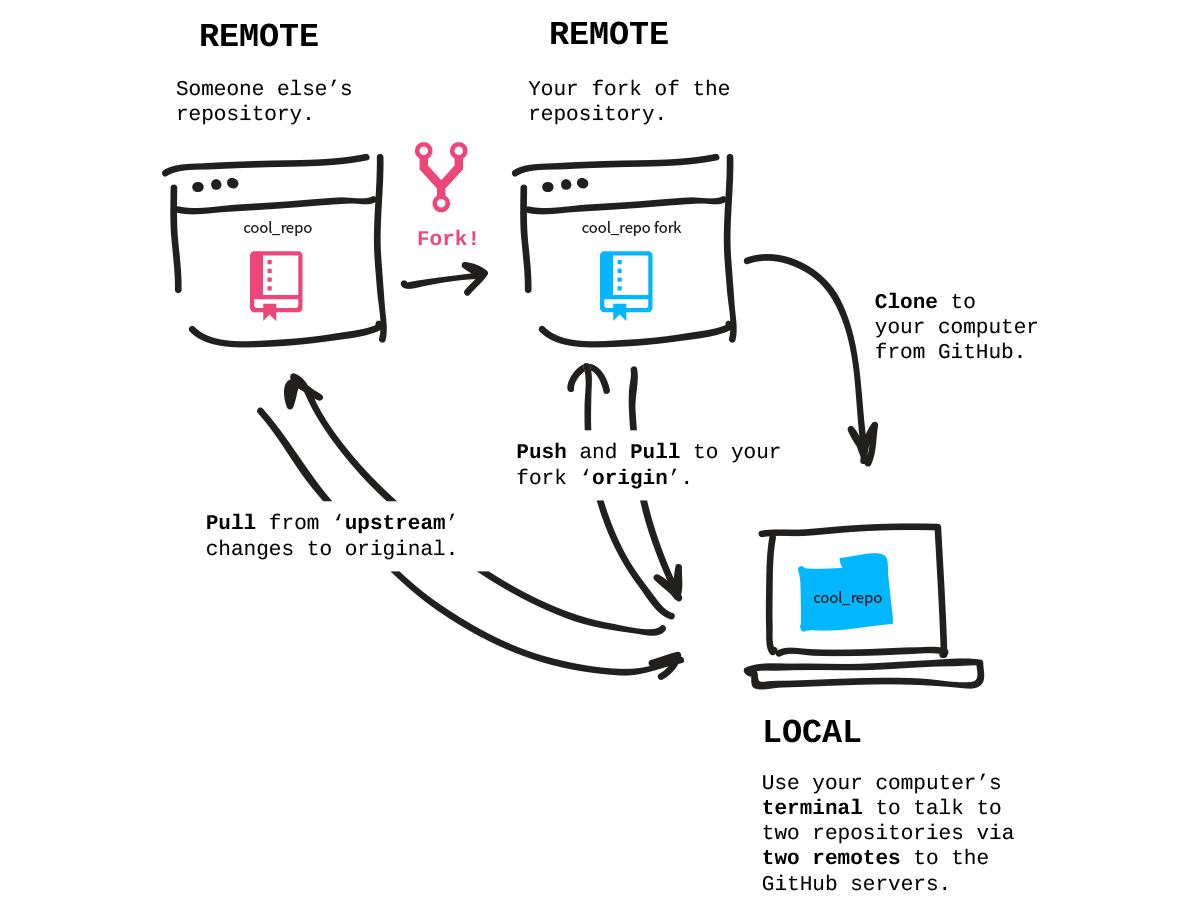
*origin https://github.com/*YOUR\_USERNAME*/*YOUR\_FORK*.git (push)*

*upstream https://github.com/*ORIGINAL\_OWNER*/*ORIGINAL\_REPOSITORY*.git (fetch)*

*upstream https://github.com/*ORIGINAL\_OWNER*/*ORIGINAL\_REPOSITORY*.git (push)*

1. Now sync your local repository with main repository use below command.

*$ git pull upstream [branch\_name]*

**

**Create a new branch with git**

In your github fork, you need to keep your master branch clean, by clean I mean without any changes, like that you can create at any time a branch from your master. Each time, that you want to commit a bug or a feature, you need to create a branch for it, which will be a copy of your master branch.

When you do a pull request on a branch, you can continue to work on another branch and make another pull request on this other branch.

Before creating a new branch pull the changes from upstream, your master needs to be up to date.

**Create the branch on your local machine and switch in this branch :**

*$ git branch [name\_of\_your\_new\_branch]*

*$ git checkout [name\_of\_your\_new\_branch]*

*OR*

*$ git checkout -b [name\_of\_your\_new\_branch]*

**Push the branch on github :**

*$ git push origin [name\_of\_your\_new\_branch]*

When you want to commit something in your branch, be sure to be in your branch.

**You can see all branches created by using :**

*$ git branch*

**Delete a branch on your local filesystem :**

*$ git branch -d [name\_of\_your\_new\_branch]*

**To force the deletion of local branch on your filesystem :**

*$ git branch -D [name\_of\_your\_new\_branch]*

**Delete the branch on github :**

*$ git push origin :[name\_of\_your\_new\_branch]*

**Git Commands Reference**

1. **git add [file\_name] or git add [directory\_name] :** In Git, you have to add file contents to your staging area before you can commit them. If the file is new, you can run git add to initially add the file to your staging area.

For example, we would now start adding our files to GIT and we would do that with git add. We can use git status to see what the state of our project is.

*$ git status*

*?? README*

*?? hello.rb*

So right now we have two untracked files. We can now add them.

*$ git add README hello.rb*

Now if we run git status again, we'll see that they've been added.

*$ git status -s*

*A README*

*A hello.rb*

1. **git status :** In order to see what the status of your staging area is compared to the code in your working directory, you can run the git status command.
2. **git diff :** A simple git diff will display in unified diff format (a patch) what code or content you've changed in your project since the last commit that are not yet staged for the next commit snapshot.

*$ git diff*

*diff --git a/hello.rb b/hello.rb*

*index d62ac43..8d15d50 100644*

*--- a/hello.rb*

*+++ b/hello.rb*

*@@ -1,7 +1,7 @@*

*class HelloWorld*

*def self.hello*

*- puts "hello world"*

*+ puts "hola mundo"*

*end*

*end*

1. **git commit :** Now that you have staged the content you want to snapshot with the git add command, you run git commit to actually record the snapshot. Git records your name and email address with every commit you make, so the first step is to tell Git what these are.

*$ git config --global user.name 'Your Name'*

*$ git config --global user.email you@somedomain.com*

Let's stage and commit all the changes to hello.rb file. In this first example, we'll use the -m option to provide the commit message on the command line.

*$ git add hello.rb*

*$ git status -s*

M hello.rb

*$ git commit -m 'my hola mundo changes'*

[master 68aa034] my hola mundo changes

1 files changed, 2 insertions(+), 1 deletions(-)

1. **git push :** To share the cool commits you've done with others, you need to push your changes to the remote repository. To do this, you run git push [alias] [branch] which will attempt to make your [branch] the new [branch] on the [alias] remote. Let's try it by initially pushing in 'master' branch to the origin.

*$ git push origin master*

*Counting objects: 25, done.*

*Delta compression using up to 2 threads.*

*Compressing objects: 100% (25/25), done.*

*Writing objects: 100% (25/25), 2.43 KiB, done.*

*Total 25 (delta 4), reused 0 (delta 0)*

*To git@github.com:schacon/hw.git*

*\* [new branch] master -> master*

1. **git pull :** This command pull data from origin that you do have locally.

*$ git pull*

*remote: Counting objects: 4006, done.*

*remote: Compressing objects: 100% (1322/1322), done.*

*remote: Total 2783 (delta 1526), reused 2587 (delta 1387)*

*Receiving objects: 100% (2783/2783), 1.23 MiB | 10 KiB/s, done.*

*Resolving deltas: 100% (1526/1526), completed with 387 local objects.*

*From github.com:schacon/hw*

*8e29b09..c7c5a10 master -> origin/master*

*0709fdc..d4ccf73 c-langs -> origin/c-langs*

*6684f82..ae06d2b java -> origin/java*

*\* [new branch] ada -> origin/ada*

*\* [new branch] lisp -> origin/lisp*