Git Commands and Best Practice

Table of Contents

[I. Git Setup 2](#_Toc37226659)

[II. Clone, Branch, Edit and Merge 2](#_Toc37226660)

[III. GitHub SSH Setup and Push Updates 4](#_Toc37226661)

[ Push updated master to remote origin (GitHub) 4](#_Toc37226662)

[ Push a branch to remote origin (GitHub) 5](#_Toc37226663)

[IV. Pull and Team Coordination 7](#_Toc37226664)

[ Pull changes from remote repository 7](#_Toc37226665)

[ Team Coordination 8](#_Toc37226666)

[V. Other Git Commands 8](#_Toc37226667)

# Git Setup

Git Installation information can be easily found online. Initiate a Git repository, and setup some global properties (we use ‘#’ sign as indication of a command line prompt).

1. Create a new repository

# git init

1. Configure global identity: email, user name, and (if Windows version) carriage return.

# git config --global user.email [hsun@convergeti.com](mailto:hsun@convergeti.com)

# git config --global user.name "hanningsun"

# git config --global core.autocrlf true

1. Configure store password (so that you don’t have to type it everytime) and editor (TextPad, etc), then list global config.

# git config --global credential.helper wincred

# git config --global core.editor "atom --wait" (or “vi”)

# git config –global –l

user.email=hsun@convergeti.com

user.name=hanningsun

core.autocrlf=true

core.editor=atom --wait

credential.helper=store

# Clone, Branch, Edit and Merge

To clone is to create a local repository from a ‘master’. Usually, you don’t directly work on the ‘master’ or origin branch, but create a new branch before working on it.

1. Clone a repository from GitHub. You can either a) clone master only, or b) clone a single branch
2. # git clone --depth 1 <https://github.com/ConvergeTI/eIV>
3. # git clone --single-branch --branch branch1 <https://github.com/ConvergeTI/eIV>

Note: there is no physical folder for a branch you’ve created. The ‘branch’ is a logic concept or reference point.

1. View the history log of the repository

# git log

commit 8bab7afa2b5fa1f7d8d4ea9744fb39f48bbc9c69 (grafted, HEAD -> master, origin/master, origin/HEAD)

Author: echoi-cti <echoi@convergeti.com>

Date: Fri Feb 21 05:29:03 2020 -0500

Convert svn:ignore properties to .gitignore.

1. Before add or make any changes, check the current branch and create a new one.

# git branch

\* master

# git checkout -b march15

Switched to a new branch 'march15'

# git branch

\* march15 (Note: the \* shows the current branch you’re in)

master

1. Create a new file, ‘README.md’, and add it to your branch (or remove a file from the branch after adding it).Then check the status.

# git add README.md (or git rm –f README.md)

# git status

On branch march15

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: README.md

1. Commit the changes to the branch with a comment to describe the changes.

# git commit -m "Added README file"

[march15 83bad7b] Added README file

1 file changed, 5 insertions(+)

create mode 100644 README.md

1. Merging a branch to master. You need to checkout master first (have the ‘Head’ pointing to it).

# git checkout master

Switched to branch 'master'

Your branch is up to date with 'origin/master'.

# git merge march15

Updating 8bab7af..83bad7b

Fast-forward

README.md | 5 +++++

1 file changed, 5 insertions(+)

create mode 100644 README.md

# GitHub SSH Setup and Push Updates

After you created a new branch, or updated a local master repository, you may want to push the changes to the ‘origin’, the remote master on GitHub. But first, you need to follow the instructions on <https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent> on how to setup the SSH key so that a remote push is allowed by GitHub.

The idea is that you generate a private key for the SSH agent locally, and copy the public key to the personal settings of the GitHub account. The above link provides details steps. From your terminal, do the following steps.

1. start ssh agent:

# eval $(ssh-agent -s)

Agent pid 1020

1. add your privatekey to the agent:

# ssh-add ~/.ssh/id\_rsa

Identity added: /c/Users/hsun/.ssh/id\_rsa (hsun@LAPTOP-ALDRFFGC)

1. copy public key to github (go to settings).

clip < ~/.ssh/id\_rsa.pub

## Push updated master to remote origin (GitHub)

1. Now we can push the changes we made on the local master to the master of origin at GitHub.

# git push

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 8 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 337 bytes | 337.00 KiB/s, done.

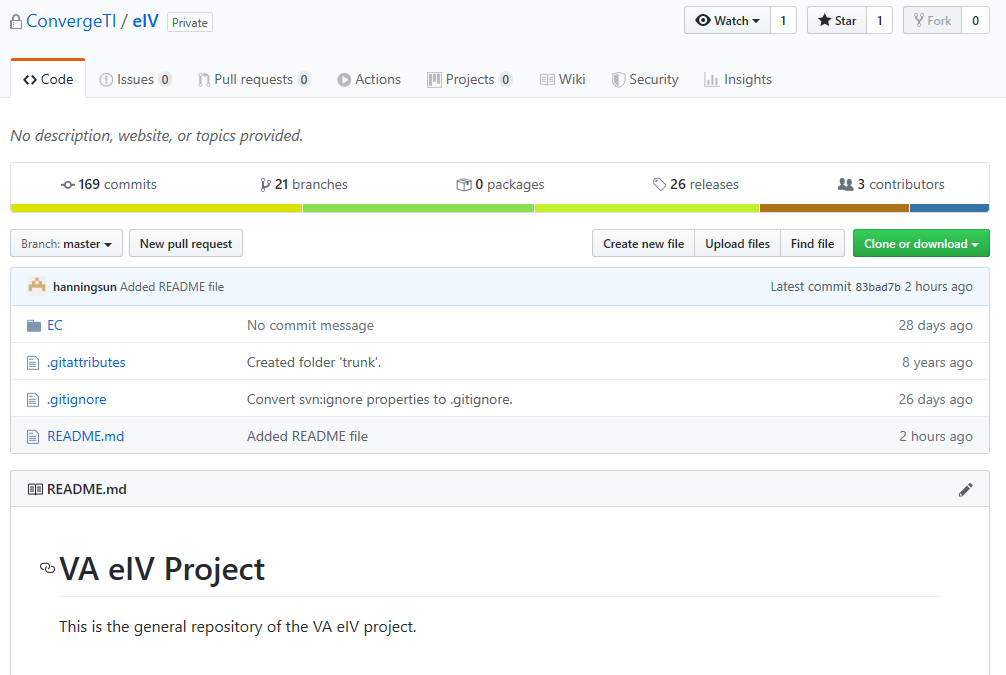
Total 3 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/ConvergeTI/eIV

8bab7af..83bad7b master -> master

On GitHub, you can see the newly added README.md file.



## Push a branch to remote origin (GitHub)

Previously we merged our local branch to the local master, and push local master directly to the remote (origin) master branch. You may also want to store your branch changes to the remote (GitHub) repository. In this case, we need to push it with the ‘set-upstream’ option.

1. Go back to the local branch, create a new file, and commit the change.

# git check out march15

# echo "Testing branch only -- ignore this branch" > test-note.txt

# git add test-note.txt

# git commit -m "Adding a test note file"

1. Now push the branch to the origin ( the ‘–set-upstream’ option is needed only the first time)

# git push --set-upstream origin march15

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 8 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 316 bytes | 316.00 KiB/s, done.

Total 3 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

remote:

remote: Create a pull request for 'march15' on GitHub by visiting:

remote: https://github.com/ConvergeTI/eIV/pull/new/march15

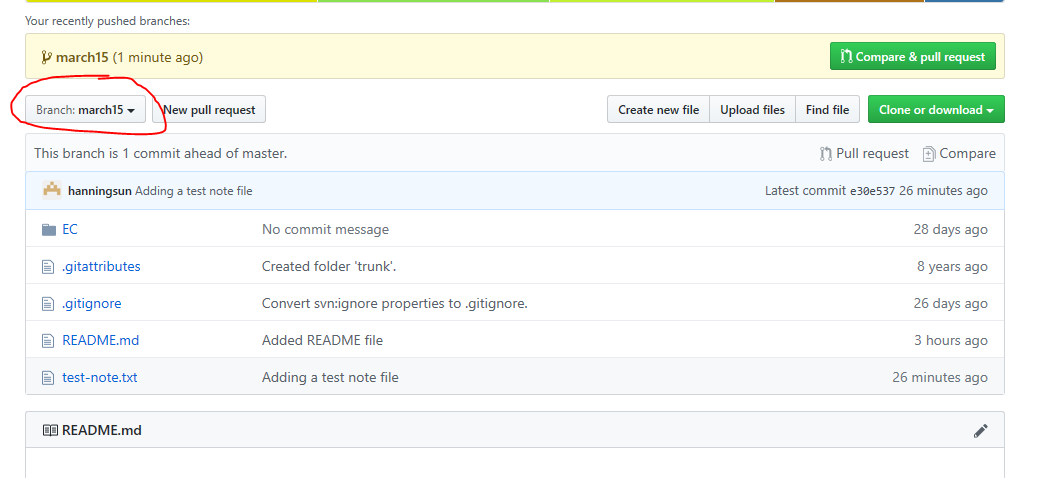
remote:

To https://github.com/ConvergeTI/eIV

\* [new branch] march15 -> march15

Branch 'march15' set up to track remote branch 'march15' from 'origin'.

Go to GitHub, we can see the new branch ‘march15’ has been added there.

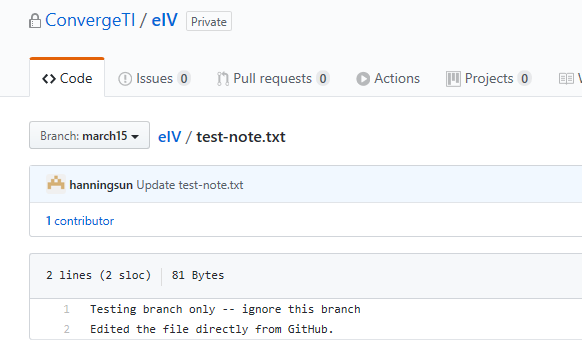


# Pull and Team Coordination

When somebody else has pushed their changes to the remote origin (master or any branch), or directly modifies files on GitHub, then your local repository is out of sync. You need to ‘pull’ the latest updates to your local directory.

## Pull changes from remote repository

To demonstrate, let’s update the test-note.txt file on the ‘mach15’ branch directly on GitHub and commit it to the same branch. Next, we will try to pull the changes locally.



1. Make sure you are on the correct branch before you pull.

# git checkout march15

# git pull

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), 706 bytes | 54.00 KiB/s, done.

From https://github.com/ConvergeTI/eIV

\* branch march15 -> FETCH\_HEAD

Updating e30e537..26ec3c9

Check and see the test-note.txt file indeed has been updated. You can pull changes to the master branch similarly by ‘git checkout master’ first.

## Team Coordination

Now we understand how to clone or pull from the master, create branches of your own, work on your branch, and then merge and push your change to the remote repository, but there is still question on how to coordinate with each other when you have a team working on the same project.

Basically, there are two approaches. 1) One person-in-charge for master branch, 2) Multiple people in charge. For 1), other developers will push their branches to GitHub, and let the person-in-charge review the changes, and merge them. For 2), anybody in the team can push to the master, but needs to communicate with others first.

Regardless of the approaches chosen, it is clear that communication is the most important step for team coordination.

# Other Git Commands

There are many other git commands, such as remove a committed file, delete a branch, undo a merge, rebase a branch, and tag a head or commit, etc. These are useful commands and can be learned by practicing them.

1. Tagging is like marking a milestone. It’s typically used to tag release numbers.

# git tag v0.1

# git log

commit 26ec3c9981190dd73862695e912a014fae22cb14 (HEAD -> march15, tag: v0.1)

…

The above command tag the latest commit to tag v0.1. To view the tags, use ‘git tag –l’. You can also use tag with annotations:

# git tag -a v1.1 -m “version 1.1 for foreign characters fix”

1. If a branch is merged with master and no long in use, you can delete it. First you need to checkout away from the branch that you want to delete (cannot delete the branch you’re currently checked out with). Use ‘-D’ to force a delete.

# git checkout master

# git branch -d march15

error: The branch 'march15' is not fully merged.

If you are sure you want to delete it, run 'git branch -D march15'.

1. Undo a merge is called ‘revert’. It could get messy when doing that. Alternatively, you can correct the mistake on the branch, and merge the branch again with the master. To revert the last merge, first find out the hash# at the beginning of each line of git log:

# git log --oneline

26ec3c9 (HEAD -> master, tag: v0.1, march15) Update test-note.txt

e30e537 Adding a test note file

83bad7b (origin/master, origin/HEAD) Added README file

8bab7af (grafted) Convert svn:ignore properties to .gitignore.

# git revert -m 1 26ec3c9

hint: Waiting for your editor to close the file...

[master b5dc22f] Revert "Update test-note.txt"

1 file changed, 1 deletion(-)

1. The ‘.gitignore’ file is where you put files that should not be committed to the repository, such a log files. Here is how to remove a committed file from the repository. We need to put the committed file we want to delete in the .gitignore file first before remove it with git. Let’s remove the ‘test-note.txt’ from the repository. On the checked out branch, do the following:

# echo test-note.txt >> .gitignore

# git rm --cached test-note.txt

# git commit -m "Start ignoring test-note.txt"

Using the --cached option with ‘git rm’ means that the file will be deleted from your repository, but will remain in your working directory as an ignored file. Otherwise it will be removed from the local filesystem.

1. Rebasing is an alternative way to merge with a branch. It’s like sync’ing with another branch (master) in one step. Merging and rebasing could be very complicated when you have multiple branches/sub-branches involved. You can look them up on <https://git-scm.com/book/en/v2/Git-Branching-Rebasing> for more information.