Aerial and Underwater Robotics Society | JUIT

Introduction to Version Control System

Contributor: Akhilesh Kumar

Module 1

Introduction to Git VCS

Hello people! Let's get familiar with Version control system.

Git Version Control Git is a wildly-used version control system in for software development. It is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear work flows. Git was initially designed and developed in 2005 by Linux kernel developers for Linux kernel development. To simplify the above rather formal definition we can say Git allows groups of people to work on the same documents (often code, in our case we are going to use it for designing too) at the same time, and without stepping on each other's toes. It's a distributed version control system

Why git and github?

Why use git?

Version control is the only reasonable way to keep track of changes in code, manuscripts, presentations, and data analysis projects. I used to make numbered tar.gz files for a project. But exploring the differences is difficult, to say the least. And if you use git properly, you'll have annotated each small change.

Merging collaborators' changes made easy. Have you ever had to deal with a collaborator sending you modifications distributed across many files, or had to deal with two people having made changes to the same file at the same time? Painful. git merge is the answer.

Why use github?

Github is like facebook for programmers. Everyone's on there. You can look at what they're working on and easily peruse their code and make suggestions or changes.

It's really open source. "Open source" is not so open if you can't easily study it. With github, all of the code is easily inspected, as is its entire history. Well, someone who applied to GSoC will know the importance.

Github lowers the barriers to collaboration. It's easy to offer suggested changes to others' code through github. I fixed some problems in some very useful code developed by some organisation which I don't know, because it's hosted on github. So, I worked on a small contribution to an organization sympy because it was hosted on github.

You don't have to set up a git server. It's surprisingly easy to get things set up.

Let's have some fun

If you're still not sure about git and github, spend a bit of time looking through the repositories for packages of interest to you. For example:

- Rudar Daman Singla's
- Prashasy's
- My own akhilesh-k

Poke around in the files, and also the commits: Click the "Commits" tab and then click on a given commit to see what changes were made.

If you feel lazy, you may follow me! :D

Module 2

Git Commands

- Installing Git

  $ sudo apt-get install git

- Cloning Repository

  $ git clone https://github.com/akhilesh-k/XYZBBLABLA.git

- Creating New Repository

  $ git init

- Setting Local Credentials

  $ git config --local user.name “akhilesh-k”

  $ git config --local user.email “akhilesh_k@outlook.com”

- Setting Global Credentials

  $ git config --global user.name “akhilesh-k”

  $ git config --global user.email “akhilesh_k@outlook.com”

- Log (Current Branch)

  $ git log

- Log (Remote Branch)

  $ git log origin/develop

- Creating Branch (Off of the origin)

  $ git branch demo origin/develop
  $ git checkout demo

- Show current status of modified files

  $ git status

- Editing files

  $ vim filename.py

- Staging Changes

  $ git add filename.py

- Commit changes

  $ git commit

- Adding Commit message ;-)

  $ git commit -m “May the thrust be with you! Hail AURS!"

- Pushing to Origin

  $ git push origin HEAD:develop

  $ git push origin demo:develop

Module 3

Hands on Experience

Your first time with git and github

If you've never used git or github before, there are a bunch of things that you need to do. It's very well explained on github, but repeated here for completeness.

- Get a github account.
- Install git.
- Set up git with your user name and email.
  - Open a terminal/shell and type:
  - $ git config --global user.name "Your name here"
  - $ git config --global user.email "your_email@example.com"

  (Don't type the $; that just indicates that you're doing this at the command line.)

  I also do:

  ```
  $ git config --global color.ui true
  $ git config --global core.editor emacs
  ```

  The first of these will enable colored output in the terminal; the second tells git that you want to use emacs.

  (Emacs is an editor)
- Set up ssh on your computer. See github's guide to generating SSH keys.
  - Look to see if you have files ~/.ssh/id_rsa and ~/.ssh/id_rsa.pub.
  - If not, create such public/private keys: Open a terminal/shell and type:
  - $ ssh-keygen -t rsa -C "your_email@example.com"
  - Copy your public key (the contents of the newly-created id_rsa.pub file) into your clipboard. **On a Mac**, in the terminal/shell, type: (This part came from internet. I really don't have a mac! Btw, I am a linux fanboy. But I envy people carrying that sexy machine around.)

- $ pbcopy < ~/.ssh/id_rsa.pub

- Paste your ssh public key into your github account settings.
  - Go to your github Account Settings
  - Click "SSH Keys" on the left.
  - Click "Add SSH Key" on the right.
  - Add a label (like "My laptop") and paste the public key into the big text box.
  - In a terminal/shell, type the following to test it:
  - $ ssh -T git@github.com
  - If it says something like the following, it worked:
  - Hi username! You've successfully authenticated, but Github does not provide shell access.

Start a new git repository

Your first instinct, when you start to do something new, should be git init. You're starting to write a new paper, you're writing a bit of code to do a computer simulation, you're mucking around with some new data … *anything*: think git init.

A new repo from scratch

Say you've just got some data from a collaborator and are about to start exploring it.

- Create a directory to contain the project.
- Go into the new directory.
- Type git init.
- Write some code.
- Type git add to add the files
- Type git commit.

The first file to create (and add and commit) is probably a ReadMe file, either as plain text or with Markdown, describing the project.

Markdown allows you to add a bit of text markup, like hyperlinks, **bold**/*italics*, or to indicate code with a monospace font. Markdown is easily converted to

html for viewing in a web browser, and GitHub will do this for you automatically.

A new repo from an existing project

Say you've got an existing project that you want to start tracking with git.

- Go into the directory containing the project.
- Type git init.
- Type git add to add all of the relevant files.
- You'll probably want to create a .gitignore file right away, to indicate all of the files you don't want to track. Use git add. gitignore, too.
- Type git commit.

Connect it to GitHub

You've now got a local git repository. You can use git locally, like that, if you want. But if you want the thing to have a home on github, do the following.

- Go to GitHub.
- Log in to your account.
- Click the new repository button in the top-right. You'll have an option there to initialize the repository with a README file, but I don't.
- Click the "Create repository" button.

Now, follow the second set of instructions, "Push an existing repository…"

```
$ git remote add origin git@github.com:username/new_repo
$ git push -u origin master
```

Actually, the first line of the instructions will say

```
$ git remote add origin https://github.com/username/new_repo
```

But I use git@github.com:username/new_repo rather than https://github.com/username/new_repo, as the former is for use with ssh (if you set up ssh as I mentioned in "Your first time", then you won't have to type your password every time you push things to github). If you use the latter construction, you'll have to type your github password every time you push to github.

Contribute to someone's repository

Say you want to contribute changes to someone else's repository (eg, this one).

- Go to the repository on github. (Say it's by myfriend, and is called the_repo, then you'll find it at http://github.com/myfriend/the_repo.)

  (** I have a friend Prashasy, do whatever shit you like with his codes! And don't forget to send a pull request with lovely comments! :D)
- Click the "Fork" button at the top right.
- You'll now have your own copy of that repository in your github account.
- Open a terminal/shell.
- Type
- $ git clone git@github.com:username/the_repo

  where username is *your* username.
- You'll now have a local copy of *your version* of that repository.
- Change into that project directory (the_repo):
- $ cd the_repo
- Add a connection to the original owner's repository.
- $ git remote add myfriend git://github.com/myfriend/the_repo
- Note the distinction between git@github.com: in the first case and git://github.com/ in the second case. I'm not sure why these need to be the way they are, but that's what works for me.
- Also note the first myfriend does not need to be the same as the username of myfriend. You could very well choose:
- $ git remote add repo_nickname git://github.com/myfriend/the_repo
- To check this remote add set up:
- $ git remote -v
- Make changes to files.
- git add and git commit those changes
- git push them back to github. These will go to *your version* of the repository.

- Note: if you get an error like:

- error: src refspec master does not match any.
- error: failed to push some refs to 'git@github.com:username/the_repo'

Then try git push origin HEAD:gh-pages (see stackoverflow.). Typing git show-ref can show what reference to put after HEAD.

- Go to *your version* of the repository on github.
- Click the “Pull Request” button at the top.
- Note that your friend’s repository will be on the left and *your repository* will be on the right.
- Click the green button “Create pull request”. Give a succinct and informative title, in the comment field give a short explanation of the changes and click the green button “Create pull request” again.

Pulling others’ changes

Before you make further changes to the repository, you should check that your version is up to date relative to your friend’s version.

Go into the directory for the project and type:

```
$ git pull myfriend master
```

This will pull down and merge all of the changes that your friend has made.

Now push them back to your github repository.

```
$ git push
```

Handling pull requests

Say your friend has suggested some changes to your code.

Ask them to get a github account and follow the instructions above: fork your repository, make the changes, and submit a pull request.

Once they do that, you’ll get an email about it. How to handle it?

Using the github website:

- Go to your version of the repository.
- Click on “Pull Requests” at the top.
- Click on the particular request.

- You'll see their comments on the pull request, and can click to see the exact changes.
- If you want them to make further changes before you merge the changes into your repository, add a comment.
- If you hate the whole idea, just click the "Close" button.
- If you want to merge the changes into your repository, click the "Merge pull request" button.
- Your github repository will now be fixed, but you'll want to get them into your local repository, too.
- Open a terminal/shell, and type
- $ git pull

Using the command line

You don't have to use the github website for this.

- Open a terminal/shell.
- Go into the directory for your project.
- Add a connection to your friend's version of the github repository, if you haven't already.
- $ git remote add myfriend git://github.com/myfriend/the_repo
- Pull his/her changes.
- $ git pull myfriend master
- Push them back to your github repository.
- $ git push
- The pull request on github will be automatically closed.

Dhanyawad!

Questions/Suggestions