Git & GitHub Crash Course For Beginners

WHAT IS GIT?

Version Control System(VCS) for tracking changes in computer files（版本控制系统）

① Distributed version control (many developers can work on a single project without having to be on the same network)（分布式的，decentralized）
② Coordinates work between multiple developers
③ Track who made what changes and when in the system or project
④ Revert back at any time（可恢复）
⑤ Push your local repos to the remote one(something like github or bitbucket)
⑥ Need a internet connection if you want to push the local repos to the remote repos.

CONCEPTS OF GIT

① Keeps track of your code history with snapshots(快照) of your files, you make these snapshots by making a commit with a simple command
② you can always go back to see any-snapshot of previously written code and you can always revert back if you need to, code is extremely safe when you are working with git
③ you can put your code in a staging area(暂存区) before committing and writing to the snapshot, this is done with a simple add command.
④ The video will go over all the important commands
⑤ once you make a commit to a remote repository other developers can then pull that information onto their machines
⑥ you can also create branches

BASIC COMMADNS

$ git init // initialize a local git repository so you go into a folder via your command line or terminal, and it will create a dot git folder in that project and it's hidden by default, you almost never need to go into this folder, once you initialize that folder as a repository you can then start to run other commands
$ git add <file> //add the file or files that you specify（指定）to the staging area or the index and they'll then be ready for commit, you can run this command as many times as you need to before committing, and it'll just keep the files in that staging area
$ git status //if you want to see what you have in the staging area and ready for commit, you can just use git status and that will display paths that have display the differences between the working tree and the staging area or the index
$ git commit // you can go ahead and commit（提交） your files so you run git commit and it will take everything that's in the index of the staging area and put it into the local repository

the next three commands that I'm going to show you have to do with remote repositories so if you're working with github or bitbucket something like that
$ git push // you will have to add for instance github or another remote service, you'll have to add that, and you have to add your credentials（证书） so it'll ask for your password, and you can also create SSH keys with github so that you don't have to add passwords or anything like that

$ git pull // if you want to pull the latest changes from a remote repository you can just do git pull, so if someone makes a change and you want the latest version you can use pull
$ git clone //git clone will copy a remote repository into your current folder so if you find a program or a project or module that you like on github you can simply clone it and that'll download on your machine

INSTALLING GIT

Linux(Debian)
$ sudo apt-get install git
Linux(Fedora)
$ sudo yum install git
Mac
http://git-scm.com/download/mac
Windows
http://git-scm.com/download/win
some notes:
for Windows there's a few different programs you can use but I'd always suggest this, you can download the Installer and you can set it up really easily so this also comes with **a tool called git bash which is a command line tool and it gives you a more Linux the environment on Windows.** I almost always use that tool over the standard Windows command line. now that there is graphical tools that you can use where you don't need to use commands, but I would strongly recommend then they use the commands, because if you only use the GUI you're not really learning anything. you're not really learning how git works under the hood, and if you end up somewhere where you don't have that tool you're going to be lost, and it's not like it's really hard if it was if it was really difficult then maybe I'd say I'll use that but it's not hard at all. alright at least for the basics, it can get complicated to really know how it works internally but for things that you're going to need it for most likely you're not going to need to know everything.

this is the git bash tool that they-give you so I would definitely suggest-using this over the standard Windows command line, so now if we say get --version you'll see the version and then you can also use this from within the standard coming in line so if we go cmd and we open up the Windows command line and get --version it should also work. so you can use git from either one now. what I'm going to do is create a project, so let's see on my desktop I'm going to create a new folder and just going to call this my app, so when you have git bash you can also go ahead and right click and do git bash here and that will open up a window there as well. so I'll just use this one and you can **hold ctrl and scroll and you can make this bigger or smaller** which is nice okay so what we want to do first, let's open this up as well, kind of have both of these open at the same time. so what I want to do is just create a couple of files that our project would have, so let's say
$ touch index.html
or you could use your file manager, and let's create an

$ touch app.js
okay so we have these two files that this is our application and what we want to do is, let's open up atom which is the text editor that I'm using and I'm just-going to add that folder as a project. We add something into the index.html file. we'll save that and now what I want to do is to initialize this folder as a git repository. so to do that we just want to say
$ git init
now once we do that it's-going to create a .git folder in your directory. now we can't see it here because I'm on a new Windows installation and I don't have hidden files enabled so what you want to do is if you're not seeing it here is go to view options and then change folder search options go to view and then show hidden files. I'm also going to uncheck hide extensions for known file types, because I want the extension to be shown all the time, so let's say apply, and now you'll see there's a .git folder. now this stuff in here don't worry about the meaning of the content. so we're not going to look into that. **so now we have this initialized as a git repository so we can start to use git commands.**
now one thing you should do before you start anything is you want to **add your name and email address to git** so to do that use the config command
$ git config --global user.name 'cyf'
$ git config --global user.email 'chenyifanxjtu.phy@gmail.com'
So now that that's set up. let's go ahead and let's add the index.html file to our git repository, so to do that we can say
$ git add index.html
it doesn't tell us anything when we do that but if we want to check to see what's in the staging area we can do
$ git status
it's the command we're going to use quite a bit, and what this is telling us is that the index file has changed to be committed and that we've added it to the staging area and then app.js is a file that's untracked.
okay so it's telling us to add it so if you want to remove for instance index.html from the staging area you can do RM, so let's go ahead and do that
$ git rm --cached index.html
And now if we do
$ git status
you'll see that both files are now untracked.
Now there's different ways you can add files, for instance we could say
$ git add *.html
what that will do is it will add any HTML files to the staging area. so now if we say
$ git status
you'll see that it added the index file it also would have added any other HTML files that we had, so I'm going to go ahead and take the index out again with the RM command
$ git rm --cached index.html
and then I'm going to show you how to add everything. so if you want to add every

file you can say
$ git add .
and now if we say
$ git status
you'll see that both the index and the app file are in the staging area.
ok so we haven't committed them yet, and what I want to do is edit one of these-files, so we'll go over to index.html and I'm just going to put in an exclamation(感叹号) here and then save and then if we go and we say
$ git status
you can see we now have this modified: index.html, so this means that we've added some kind of change in the file while it was in the staging area. so it's telling us to go ahead and add it, so
$ git add .
and then
$ git status
And now you can see that it's back in the staging area.
so now we're-going to do a commit so for that we say
$ git commit
now there's an option that I would always suggest you use but I'm going to show you without it first, so if we say
$ git commit
it's going to go ahead and open up this **vim editor** which can be a little confusing because if you start to type it's not going to do anything. **what you need to do is click I to go into insert mode** and now I can type. okay so I'm just going to say initial commit. here these some number signs(#) are basically comments(注释). okay so anything that has the number sign in front of it is treated as a comment. so I'll just going to **remove it from this initial commit** and that gives us a command of initial commit.
okay now to get out of this, we want to do escape, which takes us out of insert mode. we should type in
:wq
and enter(输入完代码后按 ESC 进入命令行模式，再按 SHIFT+L 键右边的按键（就是输入冒号）进入底行模式再打 wq 就可以保存退出了)
and now you'll see that those files have been committed and it tells us how many files have changed and gives us the file names.
okay so that's how you do a commit, so if we were now to say
$ git status
It says nothing to commit because we've already committed all of our changes. If we were to go edit a file now and do git status, it'll show us again which files are in staging and which files are unchanged.
so let's go ahead and edit the app.js file. I'm just going to do
$ console.log('Hello');
I'm going to go ahead and add it with
$ git add .

now I want to do a commit, but I want to show you **how we can skip that whole edit stage**. so we can say

$ git commit -m   'Changed app.js'

So now it's made that commit and it's skipped that whole editing stage we just included the comment within that command. okay when we clear this out

$ clear

I want to show you how to use git ignore, so this is a file that we can include the files or folders that we don't want to include in our repository at all. So even if we do

$ git add .

where it adds everything, it's not going to add that. So let's create it

$ touch .gitignore

now if you try to create this file in the file manager, you probably won't be able to because it starts with a dot, so it's going to look at it as an empty name. so either do our touch command or you can create it inside an editor. so what we want to do is open up git ignore and **we're also going to create a file that we wouldn't want to include**. so let's say

$ touch log.txt

usually you don't want to include log files. so let's add something to this, we'll just say there are error logs and save that. and what's going to git ignore and we'll say log.txt and save that's. **all you have to do is add the file name**. now if we go over here and we say

$ git add .

and then git status you can see the only file that's in the staging area, that's change does git ignored, normally log text would be there, because we just changed it, but since we added it to git ignore it's not going to get added. **you can also add entire directories,** so let's say we have will say dir1, and let's do dir2, and inside these I'm just going to create new file app1.js. we'll create a file called app2.js, and we'll just add something to those. Okay I'll copy it and we'll also put that in app1.js. okay so if we go to git ignore and **we don't want dir2 included at all**. so we could just put in

/dir2

let's go over here and go

$ git add .

and then

$ git status

and notice that only dir1 is in there. even though there are two files that we change, it's not in there, because we added the whole folder to git ignore. you could even say

*.txt

and **it'll withdraw all text files**. there's other expressions you can do as well, I'm-not going to go through all of them. it's all in the documentation. but just know that **if you don't want something included, you want to use git ignore**. the last thing I want to look at before we look into remote repositories is branches. So let's say that you're a developer working on a project with a bunch of people, and your **assign the task of adding a login.** so you don't want to start making changes, and you know push to the repository and have the main codebase edited and changed without finishing the functionality. so what you would do is **you could create a branch called login or whatever, and then work in that branch and you can still commit it and all that, but it's not going to be committed to the main branch**, **which is called master**. you can see right here, when we say git status it tells us we're on the master branch. now to create a branch we can say

```
$ git branch login
```

we'll pretend we're adding some login functionality. just doing that doesn't change us to the branch, if I say git status it still tells us that I didn't commit these changes. so let me just commit these before we go ahead and **switch branches,** so we'll say

```
$ git commit -m ‘another change’
```

okay so you'll see that got committed to master, because we're not in that login branch so. to switch we can say

```
$ git checkout login
```

okay so now we're in the login branch, so I'm going to clear that out, and then I'm going to create a new file while we're in this branch

```
$ touch login.html
```

okay so over here let's going to login.html and we'll just say whatever. and it's also edit index.html, so I'm just going to add down here, I'm just going to say login Form and then let's say

```
$ git add . and then let's commit it
$ git commit – m ‘login form’
```

so now let's go ahead and switch back.

```
$ git checkout master
```

And now the log in HTML file is gone and if we look at index.html, that part of it that we edited is also gone. so the reason for that is because that's in the login branch. **now if we want to merge that** if we finish the functionality and we're ready to merge. then we can say while we're in the master.

```
$ git merge login
```

so it's going to open this editor up, so let's do I and let's see, please enter a commit message to explain why this merge is necessary. so we'll say

added login

and then we'll do escape

:wq(同上面操作)

so now even though we're in the master branch, we can now see the log in HTML. there's a lot more to branching and merging if you want to get more complicated, but I want to keep this simple for beginners. this is a good chance if you're working on your own projects that you won't even use branches. **so let's go ahead and now work with a remote repository. so I'm going to go to github.** so what we want to do is go up to create new repository and I'm just going to call this my Apps sample. you can call it whatever you like. put a description, say sample app for tutorial. you have the option of public or private, so public, anyone can see it. Private, you can choose who sees it and commits, and I believe private cost money. I'm not sure, I've never actually tried to create a private repository on my account, and you can also choose to initialize it with a readme file if you want. I'm just going to leave that unchecked. you can also choose to add a git ignore, which we already have, or a license. so let's go ahead and clicks create repository. and then **it will take you to that page which has nothing in it**, so it's just giving us this message here, and it tells you exactly what to do. okay so you want to initialize your directory, which we've already done. it's telling us to add a readme but you don't have to, but you probably should especially if it's a public repository, because you want to add some information on your application. and you want to use .md for it to-be able to display nicely on the page, and md is markdown. now telling us how to make a commit, but we don't actually have to do this stuff. **what we want to do is add this repository as a remote repository,** if we go over now we say

$ git remote

that will list the remote repositories which we have none. so let's go and copy this, and we'll paste that in here,

$ git remote add origin
https://github.com/yfchenkeepgoing/myappssample_firsttryof_GITHUB.git

run it and then let's say

$ git remote

and it'll say origin, so down here what we want to do is git push and then you

$ git push -u origin main(我实际运行的时候会报错，估计是防火墙的原因)

在国内采用 ssh 连接，具体的方法在这里：
https://blog.csdn.net/qq_33369475/article/details/95372904
已经尝试成功，新建了一个 my first ssh key

we'll paste that in and that should push it to that repository. so we do have to login to github. it has to validate your credentials and password. okay so now it's going to push to the repository, to the master branch, and let's reload this and I'll see **we have our files here**. all right and then down here it says help people interested in this repository understand your project, by creating a readme. so let's go ahead and do that.

$ touch README.md

so let's open up the readme and I'm not going to go through the syntax of markdown, it's pretty easy if you want to just look at the documentation. but we're going to put a heading by using a number sign here, and we'll just say

#My app
This is my app

and we'll save, it let's go ahead and add it and then we'll commit

$ git commit -m ‘Added readme’

so right now it's on our local repository, we haven't actually pushed it yet, if I reload you'll see it's not there. so now that we've added that as a remote repository all we have to do is to

$ git push

and it'll push it right to github. so now if I reload you'll see we have the readme file and it's actually displayed here. now **if someone wanted to get this they could just clone it**. if we go to clone or download right here they could choose to download it as a zip, or they could clone it using this. go ahead and copy that link, and then let's create another folder, and I'll just call it myapp2, and let's open up gitbash here, and then what we want to do is say

```
$ git clone https://github.com/yfchenkeepgoing/myappssample_firsttryof_GITHUB.git
```

（上述操作可以由 https 完成，不需要用 ssh，当然也可以用 ssh 完成上述操作）
and it's going to pull everything in. it pulls the entire folder in, and if we open that up you can see we have the entire application here. you can clone anybody's project as long as it's public. now if you had multiple developers working on it and someone else made a change, and we want to go over here, back to our original folder you could just do

```
$ git pull
```

And that would pull everything down, you can see everything is up to date. okay I think that's as far as I want to go into this, so we covered all the basics, all the basic commands you need to start using git. I would suggest if you're not using it, at least now start to create your repositories for your project, create a github account if you don't have one, create the repositories on github and then push from your local machine to the github repository. you don't even have to use branches. All you really need to do is add in, commit and push. so hopefully this just helped some of you guys, I'm sure some of you already know all this stuff and mhopefully you just used it as a refresher or whatever.