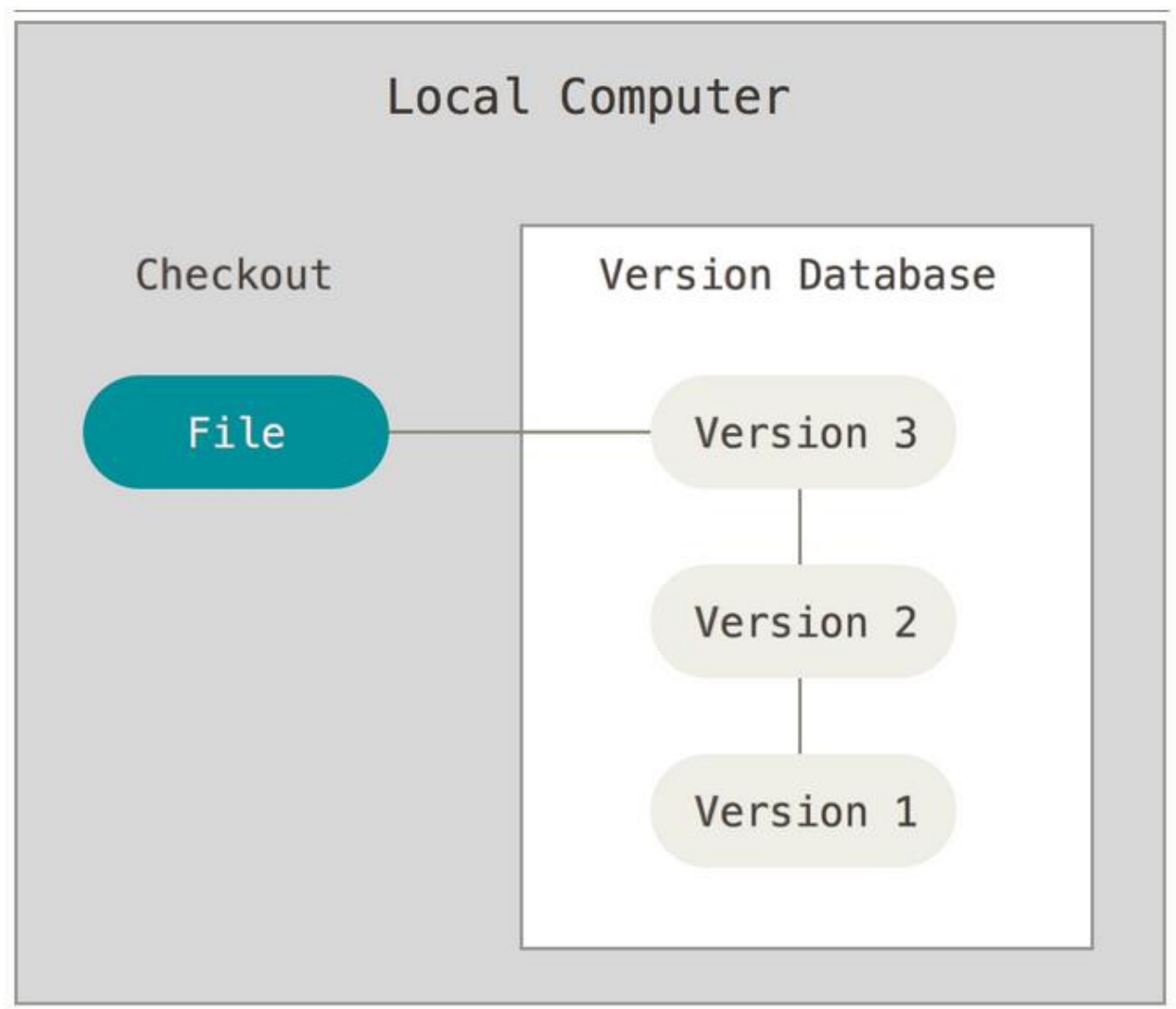


Configuration Management Sheets

Version 1.0. 16 April 2015

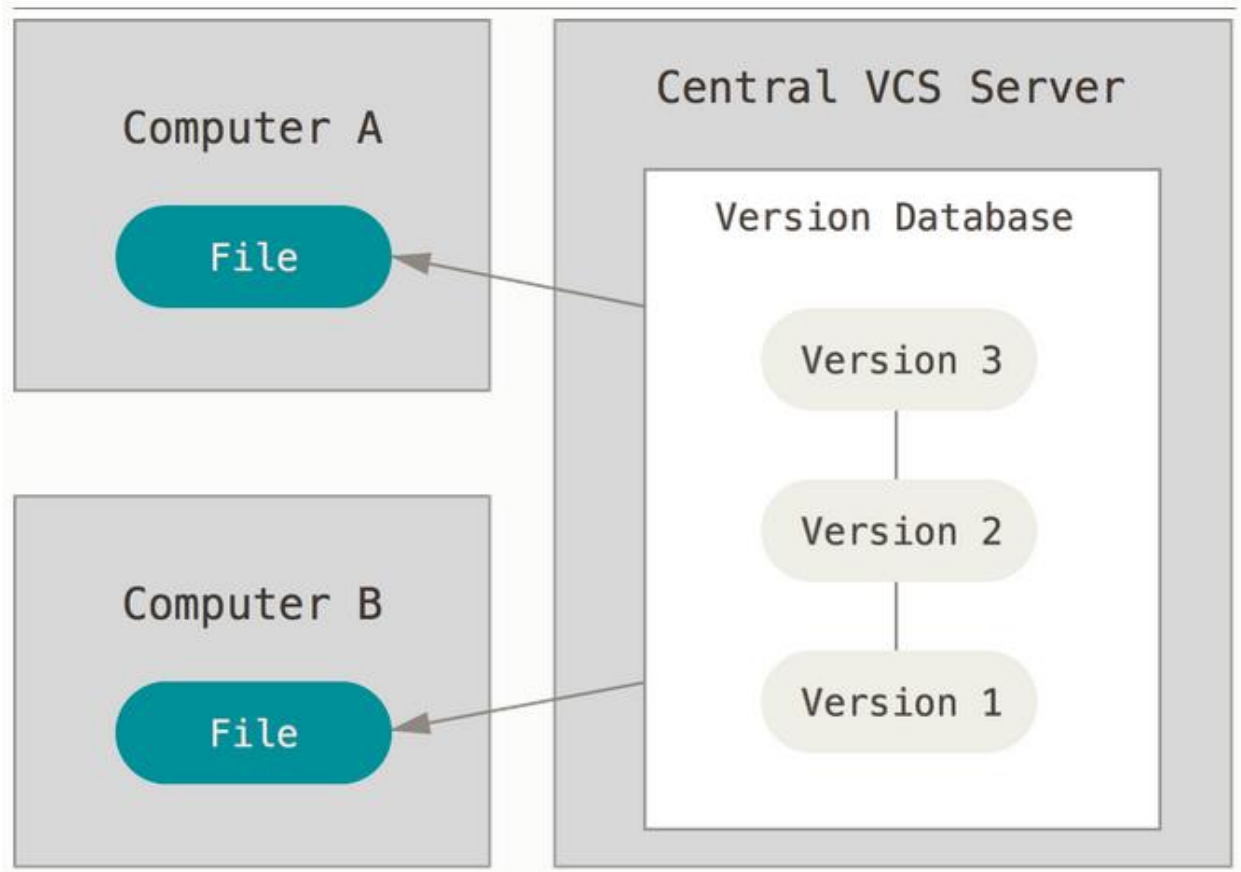
1. GitHub

- Bugs vs. changes vs. enhancements
- Great E-Book: <http://git-scm.com/book/en/v2>
- Local Version Control.



Local version control.

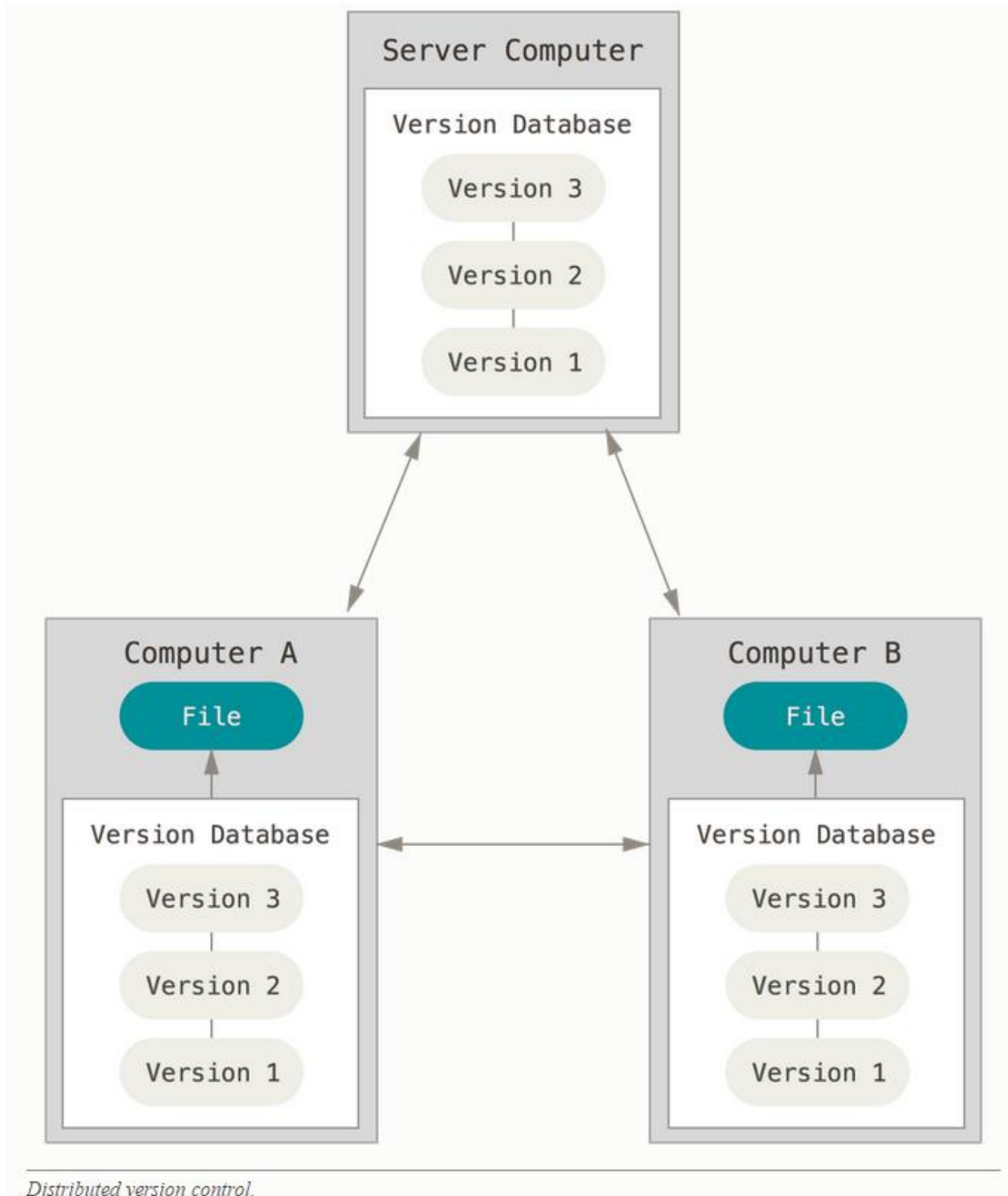
- RCS (popular local version control) works by keeping patch sets (that is, the differences between files) in a special format on disk; it can then re-create what any file looked like at any point in time by adding up all the patches.
- Centralized Version Control. Multiple users can check out a file.



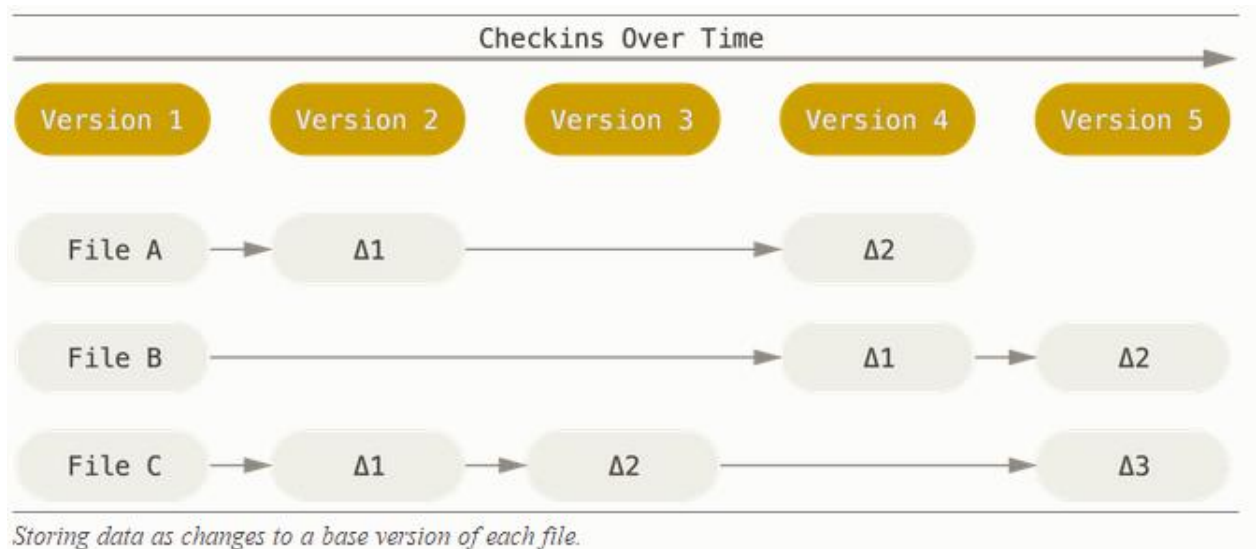
Centralized version control.

- The most obvious problem is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on. If the hard disk the central database is on becomes corrupted, and proper backups haven't been kept, you lose absolutely everything – the entire history of the project except whatever single snapshots people happen to have on their local machines. Local VCS systems suffer from this same problem – whenever you have the entire history of the project in a single place, you risk losing everything.
- In Distributed Version Control Systems such as Git, Mercurial, Bazaar or Darcs, clients don't just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these

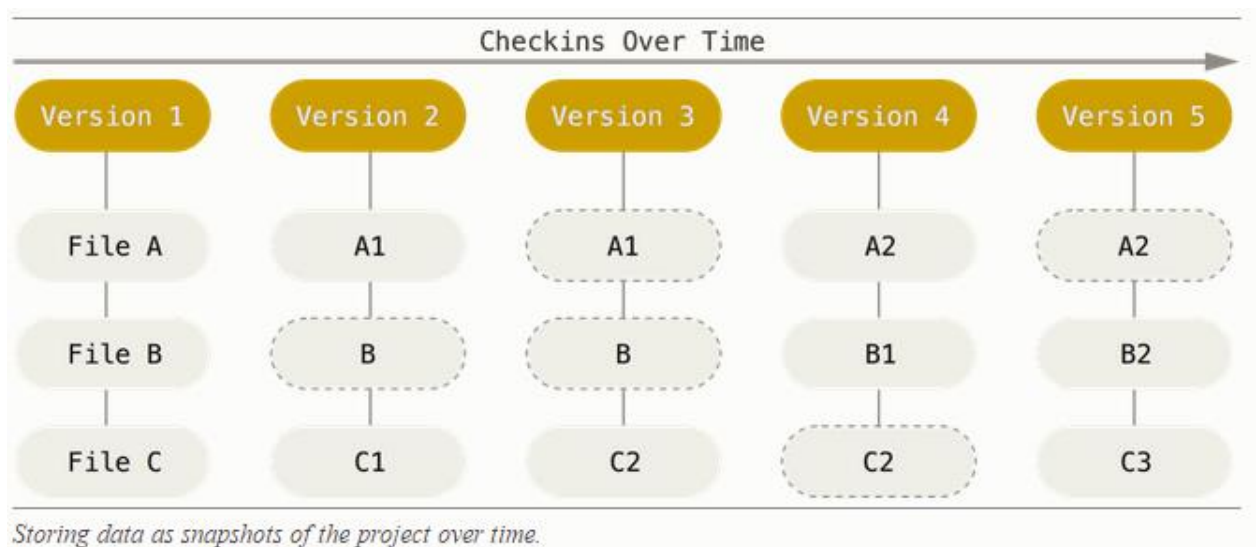
systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.



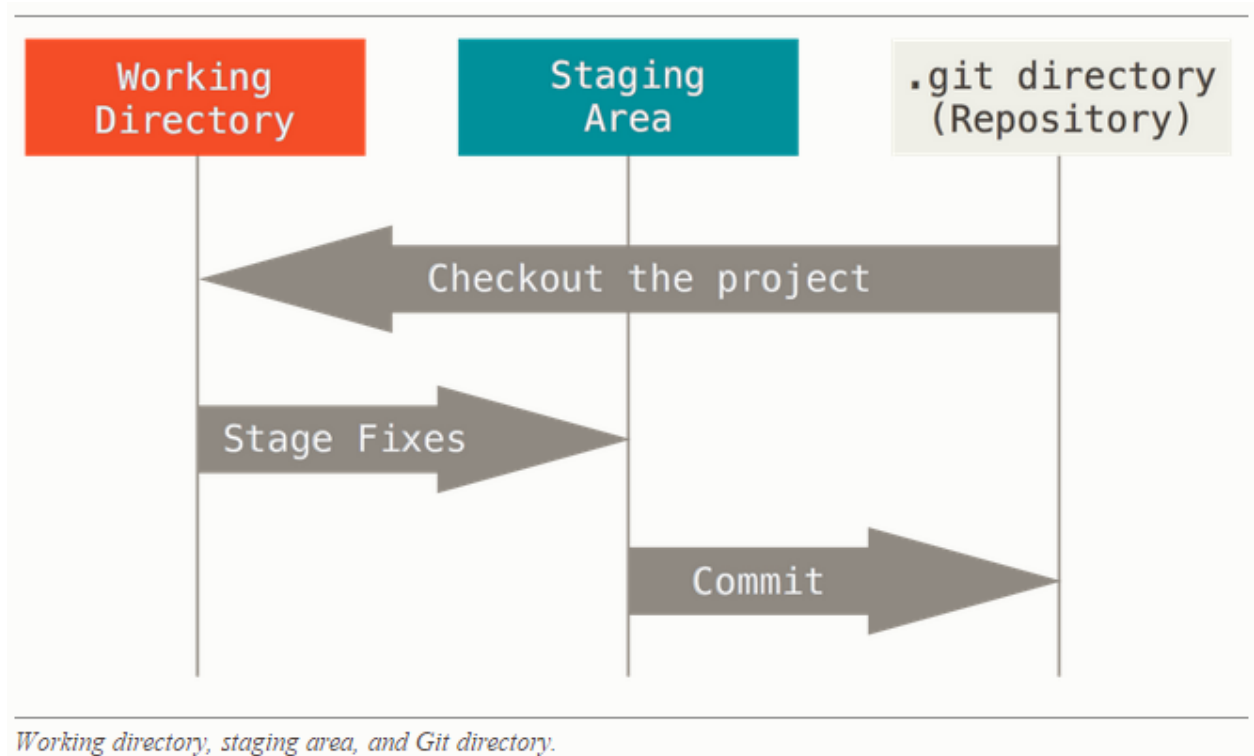
- Store versions as list of file-based changes (CVS, Subversion, Perforce, Bazaar, and so on):



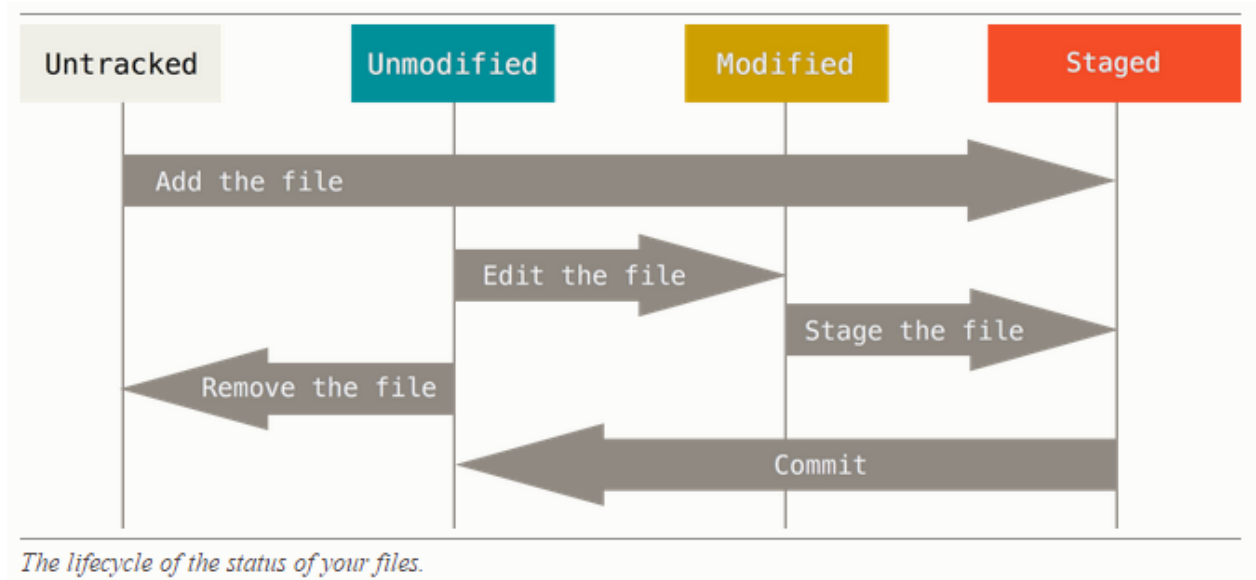
- Store versions as snapshots (Github):



- The 3 states of Git Files. The basic Git workflow goes something like this: 1) You modify files in your working directory. 2) You stage the files, adding snapshots of them to your staging area. – via “git add” 3) You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.



- Remember that each file in your working directory can be in one of two states: tracked or untracked. Tracked files are files that were in the last snapshot; they can be unmodified, modified, or staged. When you first clone a repository, all of your files will be tracked and unmodified because you just checked them out and haven't edited anything. As you edit files, Git sees them as modified, because you've changed them since your last commit. You stage these modified files and then commit all your staged changes (changes to be committed), and the cycle repeats.



1.1 What is GitHub

- Most importantly, GitHub is a collaborative and asynchronous workflow for building software better, together. GitHub is now the largest online storage space of collaborative works that exists in the world. Social network for code sharing.
- Below from [http://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1]
- GitHub itself isn't much more than a social network like Facebook or Flickr. You build a profile, upload projects to share and connect with other users by "following" their accounts. And while many users store programs and code projects, there's nothing preventing you from keeping text documents or other file types in your project folders to show off.
- Git – at the heart of GitHub. Thank famed software developer Linus Torvalds for Git, the software that runs at the heart of GitHub. Also creator of Linux OS. Git is *version control software*, which means it **manages changes to a project without overwriting any part of that project**. Git was used to help develop the core kernel for Linux.
- A version control application like Git keeps that from happening. You and your coworker can each upload your revisions to the same page, and Git will save two copies. Later, you can merge your changes together without losing any work along the way. You can even revert to an earlier version at any time, because Git keeps a "snapshot" of every change ever made.
- GitHub makes Git easier to use in two ways. First, if you download the GitHub software to your computer, it provides a visual interface to help you manage your version-controlled projects locally. Second, creating

an account on GitHub.com brings your version-controlled projects to the Web, and ties in social network features for good measure. So, GitHub is like a cloud on the web; whereas Git is your local copy on your PC or Mac.

1.2 Some Terminology

- Command Line. The computer program we use to input Git commands. On a Mac, it's called Terminal. On a PC, it's command prompt.
- Repository. A directory or storage space where your projects can live as files. It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host.
- Version Control. Basically, the purpose Git was designed to serve. When you have a Microsoft Word file, you either overwrite every saved file with a new save, or you save multiple versions. With Git, you don't have to. It keeps "snapshots" of every point in time in the project's history, so you can never lose or overwrite it.
- Commit. This is the command that gives Git its power. When you commit, you are taking a "snapshot" of your repository at that point in time, giving you a checkpoint to which you can reevaluate or restore your project to any previous state.
- Branch and Merge. How do multiple people work on a project at the same time without Git getting them confused? Usually, they "branch off" of the main project with their own versions full of changes they themselves have made. After they're done, it's time to "merge" that branch back with the "master," the main directory of the project.

1.3 Common Git Commands

- git init. Initializes a new Git repository. Until you run this command inside a repository or directory, it's just a regular folder. Only after you input this does it accept further Git commands.
- git config. Short for "configure," this is most useful when you're setting up Git for the first time. You have to configure your user name and your email. This is usually configured to be same as that on the "cloud" or GitHub.
- git help. Forgot a command? Type this into the command line to bring up the 21 most common git commands. You can also be more specific and type "git help init" or another term to figure out how to use and configure a specific git command.
- git status. Check the status of your repository. See which files are inside it, which changes still need to be committed, and which branch of the repository you're currently working on.

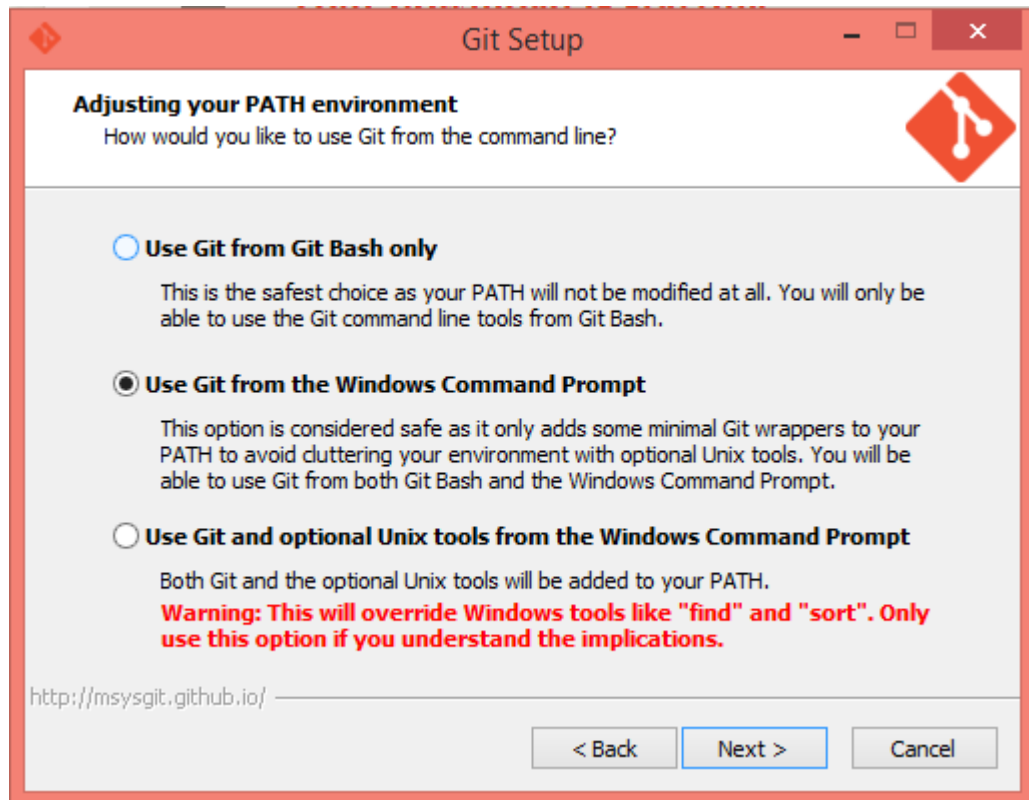


Genesys InfoCAD Company

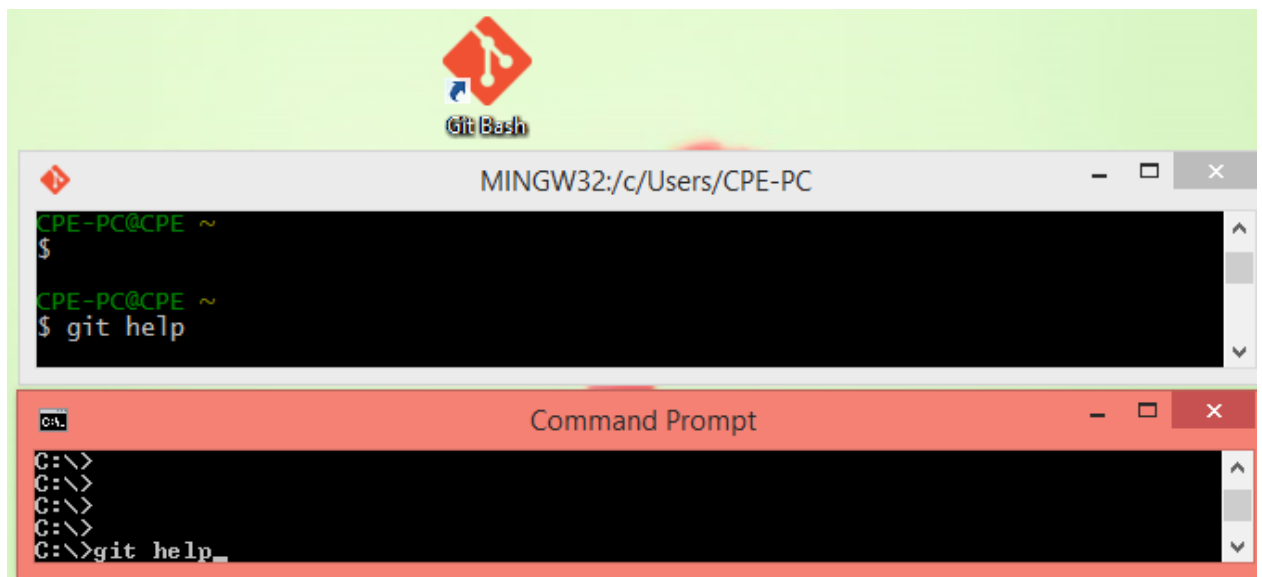
- `git add`. This does not add new files to your repository. Instead, it brings new files to Git's attention. After you add files, they're included in Git's "snapshots" of the repository.
- `git commit`. Git's most important command. After you make any sort of change, you input this in order to take a "snapshot" of the repository. Usually it goes `git commit -m "Message here."` The `-m` indicates that the following section of the command should be read as a message.
- `git branch`. Working with multiple collaborators and want to make changes on your own? This command will let you build a new branch, or timeline of commits, of changes and file additions that are completely your own. Your title goes after the command. If you wanted a new branch called "cats," you'd type `git branch cats`.
- `git checkout`. Literally allows you to "check out" a repository that you are not currently inside. This is a navigational command that lets you move to the repository you want to check. You can use this command as `git checkout master` to look at the master branch, or `git checkout cats` to look at another branch.
- `git merge`. When you're done working on a branch, you can merge your changes back to the master branch, which is visible to all collaborators. `git merge cats` would take all the changes you made to the "cats" branch and add them to the master.
- `git push`. If you're working on your local computer, and want your commits to be visible online on GitHub as well, you "push" the changes up to GitHub with this command.
- `git pull`. If you're working on your local computer and want the most up-to-date version of your repository to work with, you "pull" the changes down from GitHub with this command.

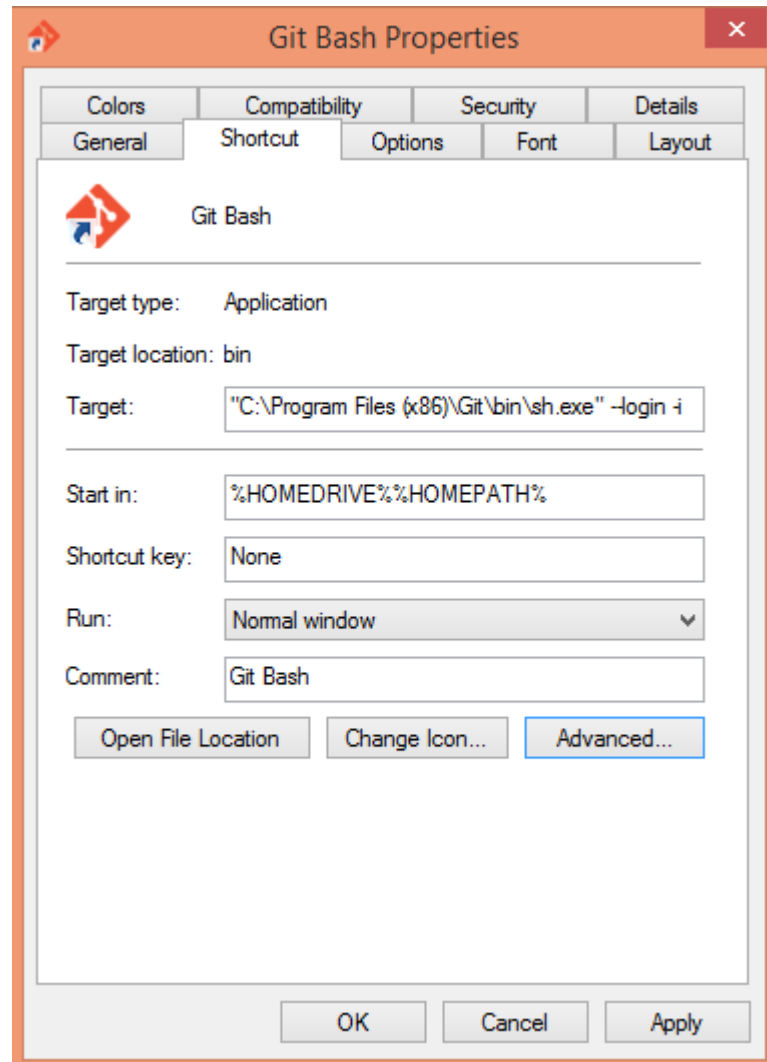
1.4 Setting up Git on PC or Mac

- But if you want to work on your project on your local computer, you need to have Git installed. In fact, GitHub won't work on your local computer if you don't install Git. Install Git for Windows, Mac or Linux. (see <http://git-scm.com/downloads>)

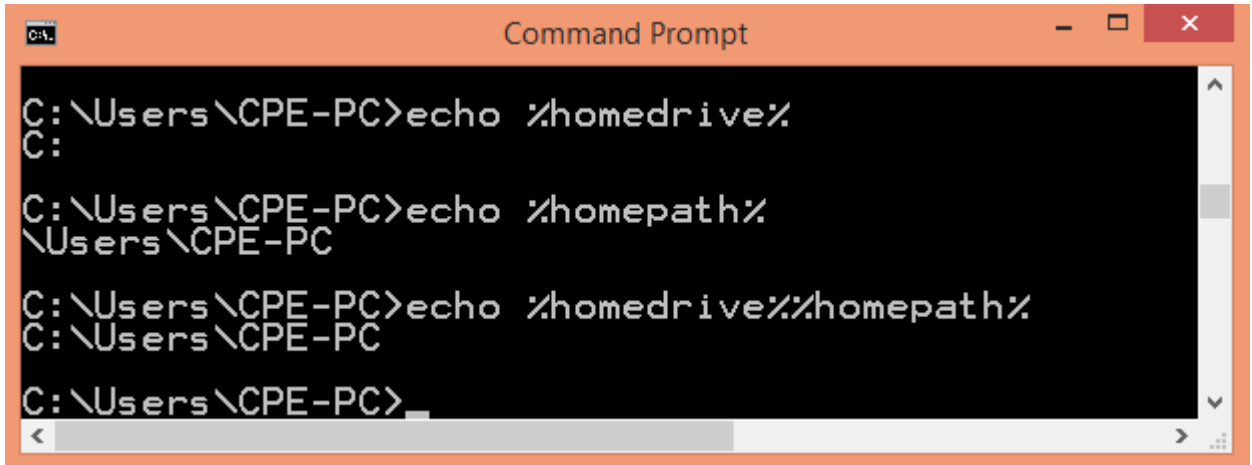


- Can run git commands from both the windows “Command Prompt” or “Git Bash”. In Mac it’s through Terminal.





- echo %HOMEPATH% - shows value of environment variable. Here, it starts in “C:\Users\CPE-PC” where CPE-PC is the name of my computer.



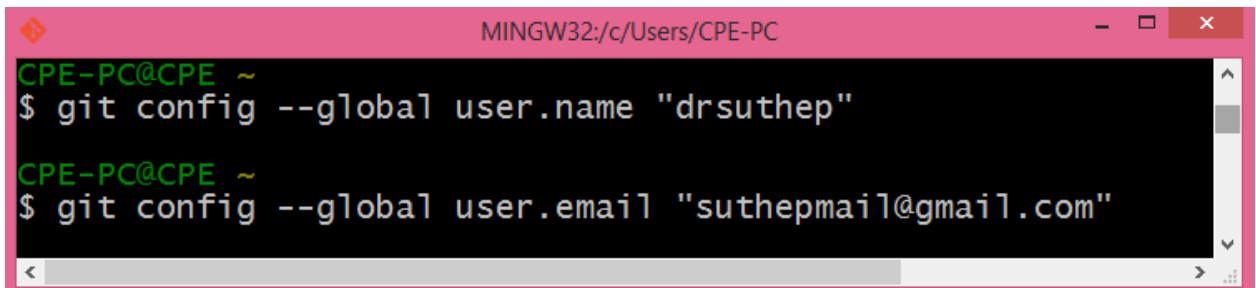
```

C:\Users\CPE-PC>echo %homedrive%
C:

C:\Users\CPE-PC>echo %homepath%
\Users\CPE-PC

C:\Users\CPE-PC>echo %homedrive%%homepath%
C:\Users\CPE-PC
  
```

- Link Git on PC (or Mac) to GitHub. Use 2 commands:
 - o `git config --global user.name "Your Name Here".`
 - o `git config --global user.email "your_email@youremail.com".`
- Your first Git Commands to setup (configure) your local Git account:

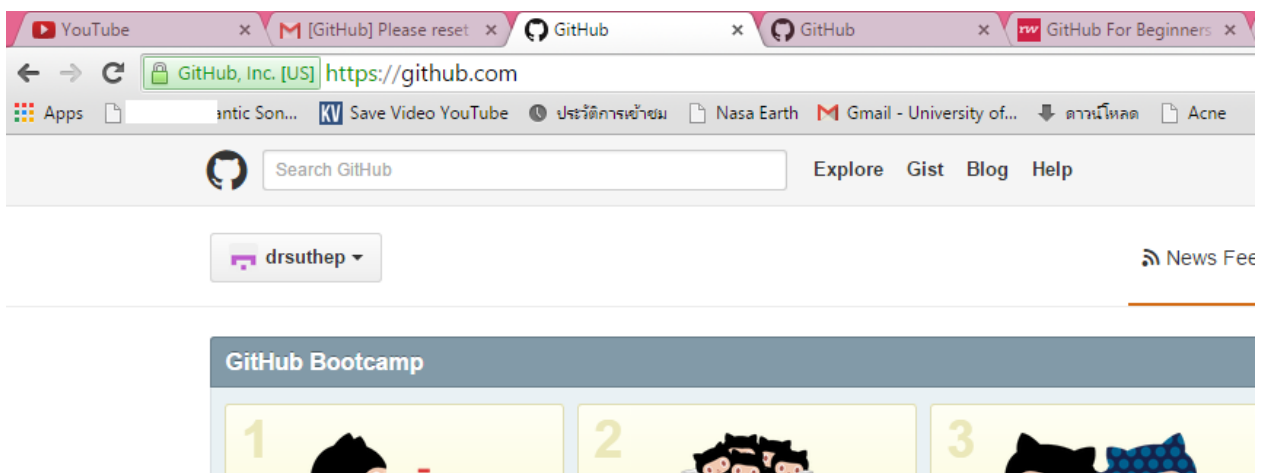


```

MINGW32:/c/Users/CPE-PC
CPE-PC@CPE ~
$ git config --global user.name "drsuthep"

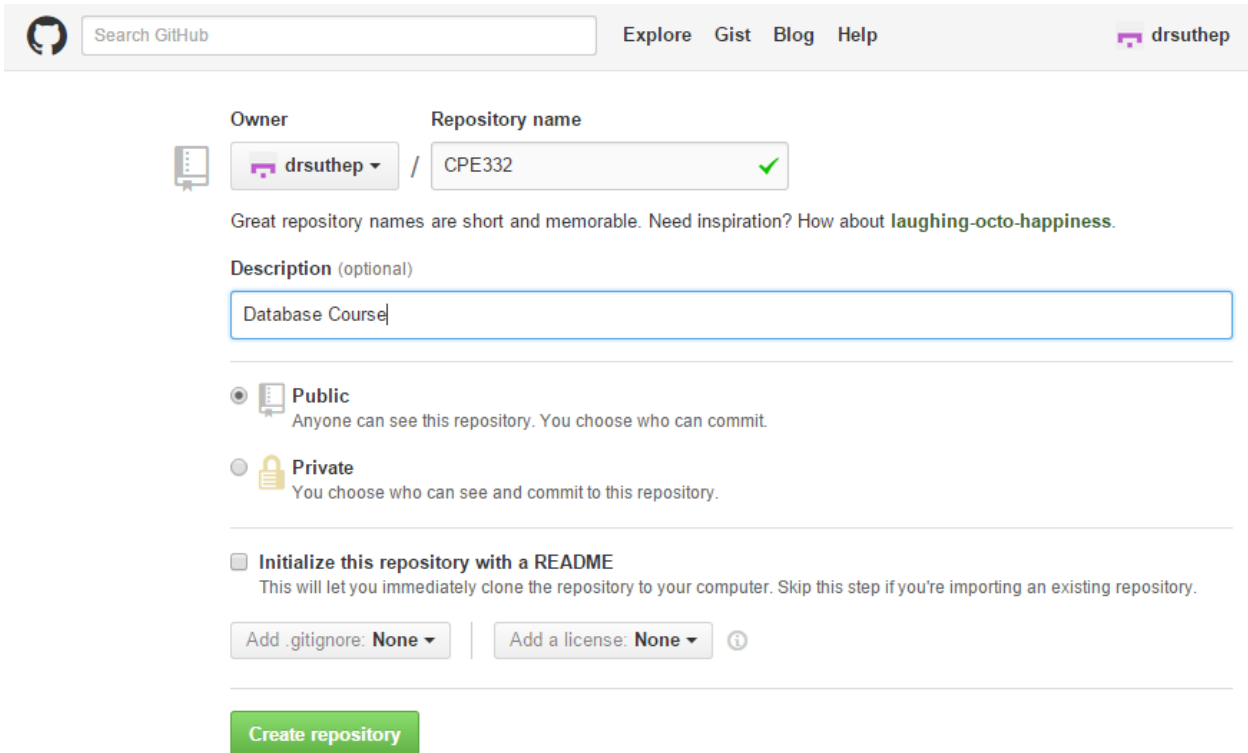
CPE-PC@CPE ~
$ git config --global user.email "suthepmail@gmail.com"
  
```

- Note I use the same user name and email as in GitHub:



1.5 Creating an Online Repository

- A project is a repository. A repository is a storage space to access your project, its files, and all the versions of its files that Git saves.



Search GitHub

Explore Gist Blog Help

drsuthep

Owner: drsuthep / Repository name: CPE332

Great repository names are short and memorable. Need inspiration? How about [laughing-octo-happiness](#).

Description (optional): Database Course

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

- Now our project has online space to live in.

1.6 Creating Local Repository

- Work is usually done on local machine, so we need to actually mirror that repository we just made as a local directory. This is similar to dropbox, but we need to be more explicit.

```

MINGW32:/c/Users/CPE-PC/CPE333
CPE-PC@CPE ~
$ mkdir ~/CPE333

CPE-PC@CPE ~
$ cd ~/CPE333

CPE-PC@CPE ~/CPE333
$ ls

CPE-PC@CPE ~/CPE333
$

```

- “git init” command will tell computer that this is a Git repository. Now you’ve got both an online and a local repository for your project to live in.

```

MINGW32:/c/Users/CPE-PC/CPE333
CPE-PC@CPE ~/CPE333
$ git init
Initialized empty Git repository in c:/Users/CPE-PC/CPE333/.git/

CPE-PC@CPE ~/CPE333 (master)
$ _

```

- Let’s create a “Readme.txt” file in notepad and save in this folder “C:\Users\CPE-PC\CPE333”

```

Readme.txt - Notepad
File Edit Format View Help
My readme file for Local Git
For course CPE 333

```

- This file is not tracked by Git until you tell it. Let’s check status of our folder using “git status”:

```

MINGW32:/c/Users/CPE-PC/CPE333

CPE-PC@CPE ~/CPE333 (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Readme.txt

nothing added to commit but untracked files present (use "git add" to track)
CPE-PC@CPE ~/CPE333 (master)
$

```

- The status says we are in master branch (not any personal branch of some user). Note that in above, the “Readme.txt” is just a file in the folder, but is not a tracked file. To start tracking this file must explicitly use “git add”. Now Git will track this file. “git add” is a multipurpose command – you use it to begin tracking new files, to stage files (ready for commit), and to do other things like marking merge-conflicted files as resolved. It may be helpful to think of it more as “add this content to the next commit” rather than “add this file to the project”

```

MINGW32:/c/Users/CPE-PC/CPE333

CPE-PC@CPE ~/CPE333 (master)
$ git add Readme.txt

CPE-PC@CPE ~/CPE333 (master)
$

```

- Time to commit the “Add Readme.txt”. This will take a snapshot of the project so far.

```

MINGW32:/c/Users/CPE-PC/CPE333

CPE-PC@CPE ~/CPE333 (master)
$ git commit -m "add Readme.txt"
[master (root-commit) 3e95f0f] add Readme.txt
1 file changed, 2 insertions(+)
create mode 100644 Readme.txt


CPE-PC@CPE ~/CPE333 (master)
$

```

- If you don't want to stage (git add) before committing each time, use so that all modified are staged before commit:
git commit -a -m "add Readme.txt" instead. The text is like a name of this snapshot version.
- Now it's time to push our first commit to GitHub (the web).

1.7 Connect Local Repository to Online Repository

- Each user can work alone on own computers, but upload or "push" your changes up to the GitHub repository when ready.
- The "CPE333" repository created on the web at GitHub for user "drsuthep" can be accessed by:
<https://github.com/drsuthep/CPE333.git>
- To make link to that remote GitHub use this command on your PC or Mac:
git remote add origin <https://github.com/drsuthep/CPE333.git>
To check that remote was successfully set use: "git remote -v". It shows all remote origins known to your local repository on your PC or Mac. It can be fetched from or pushed to in the process of data synch.



```
MINGW32:/c/Users/CPE-PC/CPE333

CPE-PC@CPE ~/CPE333 (master)
$ git remote add origin https://github.com/drsuthep/CPE333.git

CPE-PC@CPE ~/CPE333 (master)
$ git remote -v
origin https://github.com/drsuthep/CPE333.git (fetch)
origin https://github.com/drsuthep/CPE333.git (push)

CPE-PC@CPE ~/CPE333 (master)
$
```

- Use "git pull origin master" to get data from remote. Here "origin" is the remote site
<https://github.com/drsuthep/CPE333.git> and master is the "branch"

```

MINGW32:/c/Users/CPE-PC/CPE333

CPE-PC@CPE ~/CPE333 (master)
$ git pull origin master
From https://github.com/drsuthep/CPE333
 * branch                master      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 README.md | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 README.md

CPE-PC@CPE ~/CPE333 (master)
$ ls
README.md  Readme.txt

CPE-PC@CPE ~/CPE333 (master)
$

```

- Note you have 2 files now README.md and Readme.txt.
- Can use this command to pull as well if you do not want to use “origin” but the full remote path.

```


MINGW32:/c/Users/CPE-PC/CPE333

CPE-PC@CPE ~/CPE333 (master)
$ git pull https://github.com/drsuthep/cpe333 master
From https://github.com/drsuthep/cpe333
 * branch                master      -> FETCH_HEAD
Already up-to-date.

CPE-PC@CPE ~/CPE333 (master)
$

```

- At the repository in GitHub (on web) there is still only 1 file as the “Readme.txt” has not been pushed up yet.

 drsuthep / CPE333 Unwatch 1

Used to teach Software Engineering CPE 333. — Edit

2 commits 1 branch 0 releases 1 contributor

 branch: master CPE333 / + 

Update README.md

 drsuthep authored 2 hours ago latest commit 4f37320efd 

 README.md Update README.md 2 hours ago

 README.md

CPE333

Used to teach Software Engineering CPE 333. We will show some examples of configuration management here.

- Use “git push origin master” where master is the name of your branch to push the changes uploaded to GitHub on the web.

```

MINGW32:/c/Users/CPE-PC/CPE333

CPE-PC@CPE ~/CPE333 (master)
$ git push origin master
Username for 'https://github.com': drsuthep
Password for 'https://drsuthep@github.com':
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 550 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/drsuthep/CPE333.git
 4f37320..e3faa27  master -> master

CPE-PC@CPE ~/CPE333 (master)
$

```

- The repository on GitHub web is now updated 2 have 2 files:

drsuthep / CPE333 Unwatch 1

Used to teach Software Engineering CPE 333. — Edit

4 commits 1 branch 0 releases 1 contributor

branch: master CPE333 / +

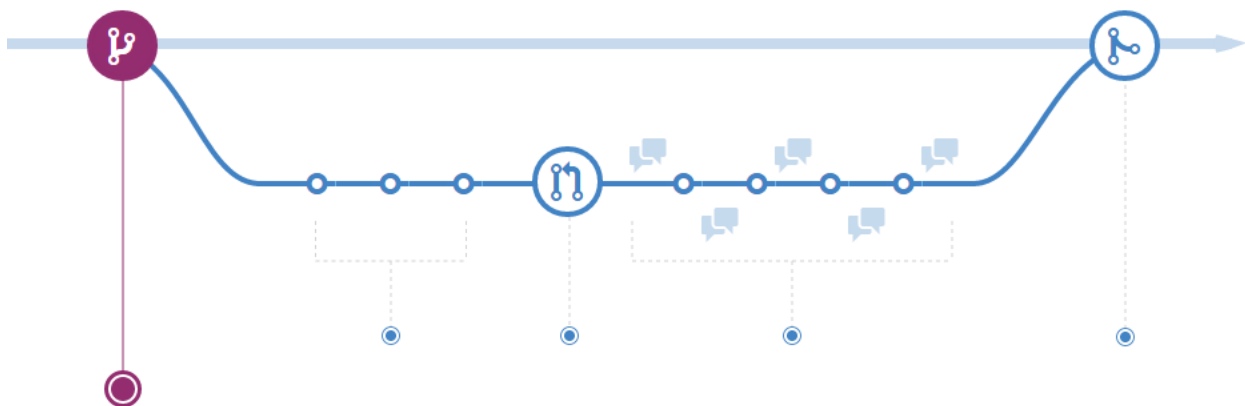
Merge branch 'master' of https://github.com/drsuthep/CPE333

drsuthep authored 8 minutes ago latest commit e3faa279f1

README.md	Update README.md	2 hours ago
Readme.txt	add Readme.txt	34 minutes ago

1.8 Branching

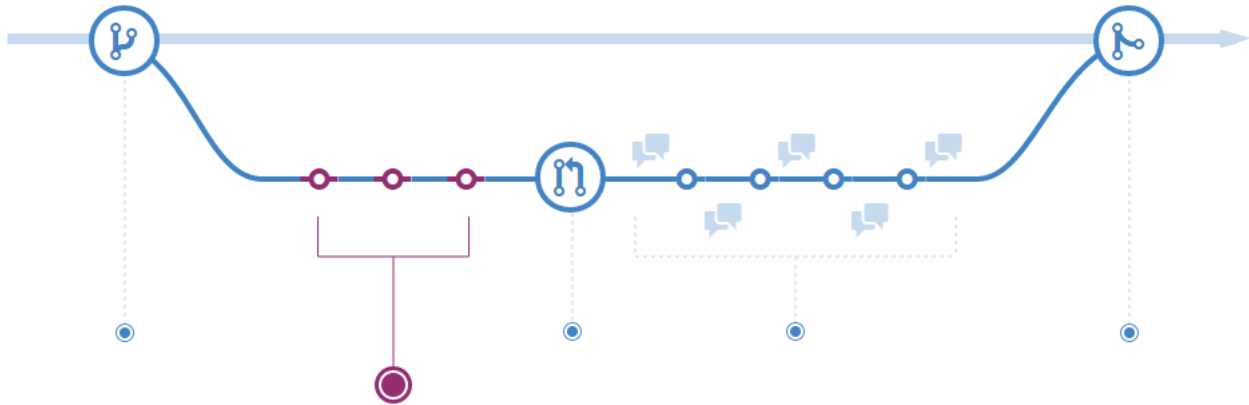
1. Create a branch



When you're working on a project, you're going to have a bunch of different features or ideas in progress at any given time – some of which are ready to go, and others which are not. Branching exists to help you manage this workflow.

When you create a branch in your project, you're creating an environment where you can try out new ideas. Changes you make on a branch don't affect the master branch, so you're free to experiment and commit changes, safe in the knowledge that your branch won't be merged until it's ready to be reviewed by someone you're collaborating with.

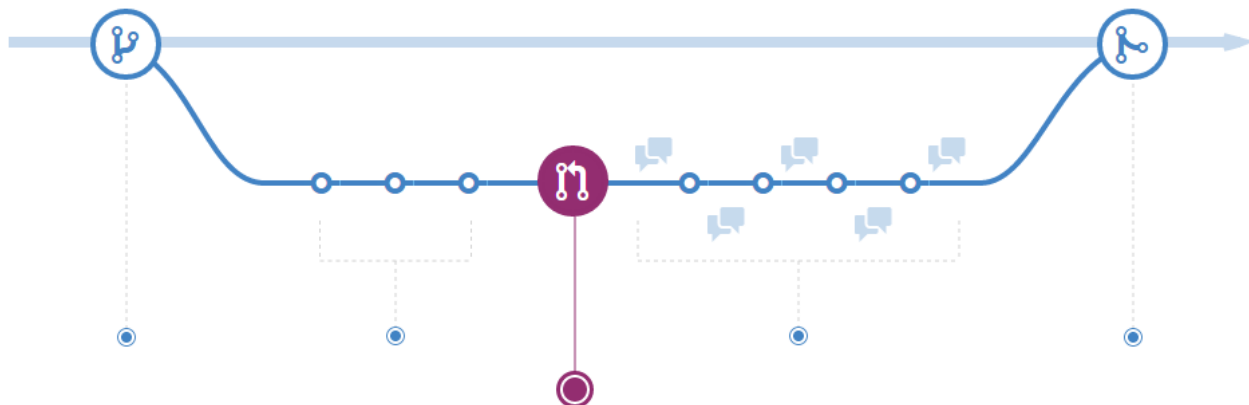
2. Add commits



Once your branch has been created, it's time to start making changes. Whenever you add, edit, or delete a file, you're making a commit, and adding them to your branch. This process of adding commits keeps track of your progress as you work on a feature branch.

Commits also create a transparent history of your work that others can follow to understand what you've done and why. Each commit has an associated commit message, which is a description explaining why a particular change was made. Furthermore, each commit is considered a separate unit of change. This lets you roll back changes if a bug is found, or if you decide to head in a different direction

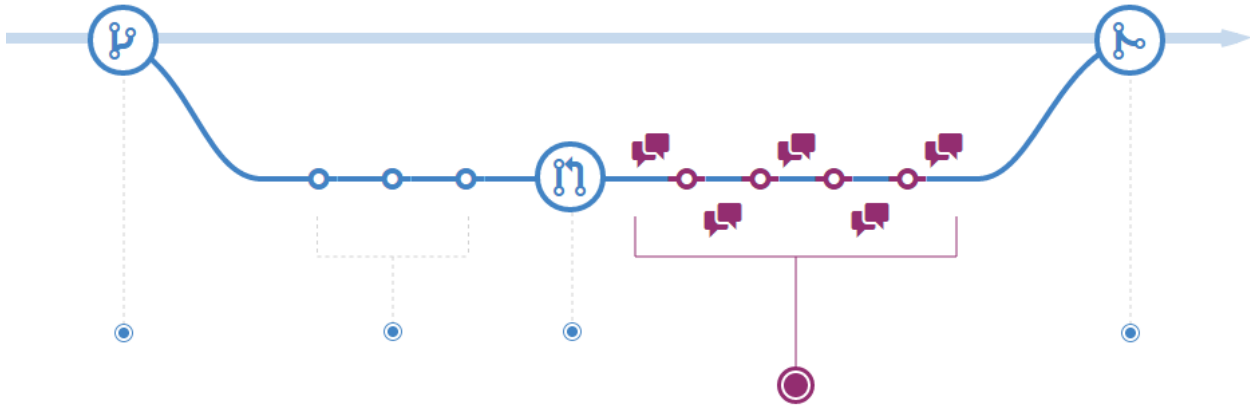
3. Open a Pull Request



Pull Requests initiate discussion about your commits. Because they're tightly integrated with the underlying Git repository, anyone can see exactly what changes would be merged if they accept your request.

You can open a Pull Request at any point during the development process: when you have little or no code but want to share some screenshots or general ideas, when you're stuck and need help or advice, or when you're ready for someone to review your work. By using GitHub's @mention system in your Pull Request message, you can ask for feedback from specific people or teams, whether they're down the hall or ten time zones away.

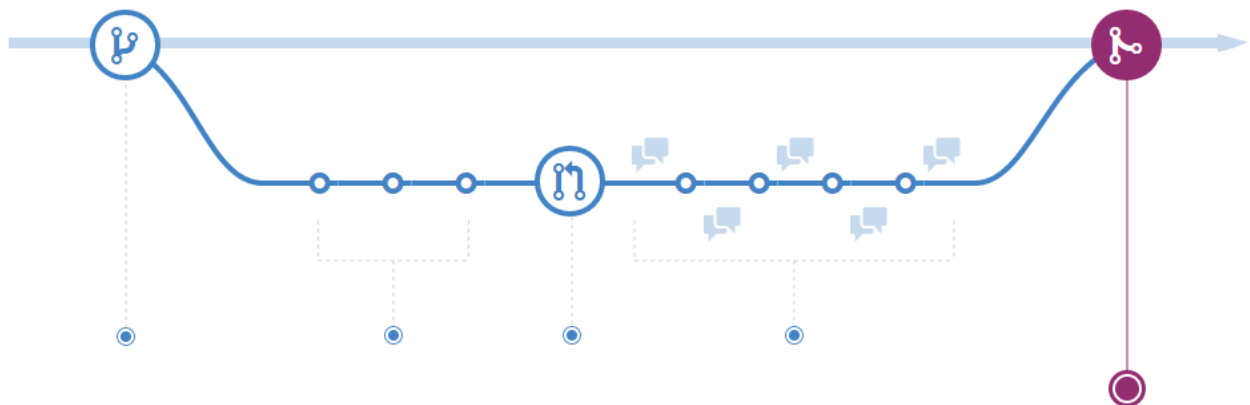
4. Discuss and review your code



Once a Pull Request has been opened, the person or team reviewing your changes may have questions or comments. Perhaps the coding style doesn't match project guidelines, the change is missing unit tests, or maybe everything looks great and props are in order. Pull Requests are designed to encourage and capture this type of conversation.


You can also continue to push to your branch in light of discussion and feedback about your commits. If someone comments that you forgot to do something or if there is a bug in the code, you can fix it in your branch and push up the change. GitHub will show your new commits and any additional feedback you may receive in the unified Pull Request view.

5. Merge and deploy








Once your Pull Request has been reviewed and the branch passes your tests, it's time to merge your code to the master branch for deployment. If you want to test things before merging in the repository on GitHub, you can perform the merge locally first. This is also handy if you don't have push access to the repository.


Once merged, Pull Requests preserve a record of the historical changes to your code. Because they're searchable, they let anyone go back in time to understand why and how a decision was made.




[Explore](#)
[Gist](#)
[Blog](#)
[Help](#)



[drsuthep](#)



[drsuthep / CMLab](#)


 Unwatch

2



 Star

0



 Fork

0

The readme file needs to be updated #1


 Open


drsuthep opened this issue 33 seconds ago · 0 comments




drsuthep commented just now

Owner


No description provided.



 suthep64 was assigned by drsuthep just now



Write

Preview


 Markdown supported


 Edit in fullscreen

Leave a comment

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Close issue

Comment

Labels

None yet

Milestone

No milestone

Assignee



suthep64

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

2 participants

Lock issue