CS 50 Web Design

APCSP Module 2: Internet

Unit 2: CSS (Chapter 20)

LECTURE 6C: MODERN WEB DEVELOPMENT TOOLS

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Introduction to the command line
- CSS preprocessors and postprocessors
- Build tools and task runners
- Git version control

Visual Studio Code Editor

SECTION 1

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Search Docs Download

Version 1.70 is now available! Read about the new features and fixes from July.

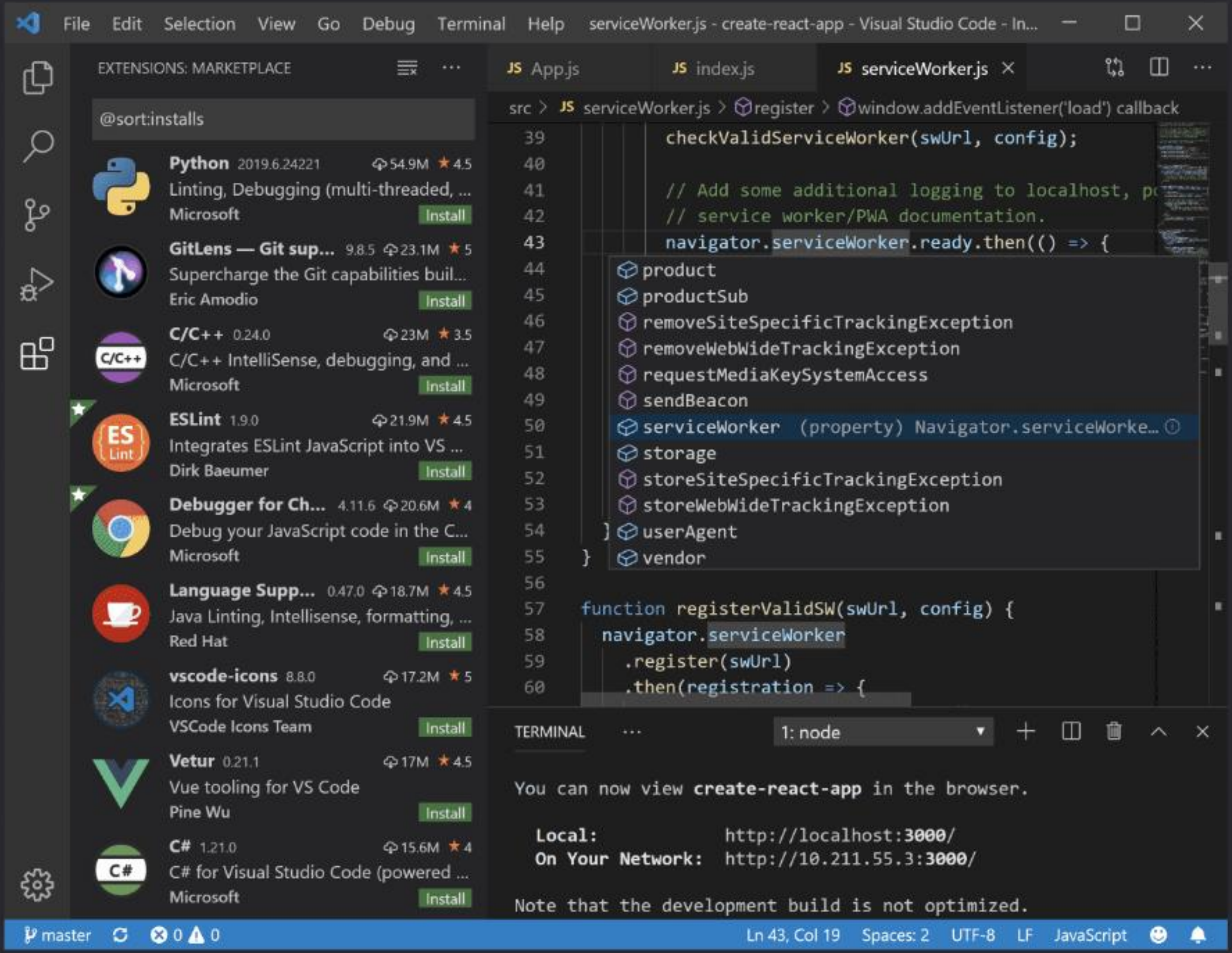
Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.



@sort:installs

Python 2019.6.24221 54.9M ★4.5
Linting, Debugging (multi-threaded, ...
Microsoft Install

GitLens — Git sup... 9.8.5 23.1M ★5
Supercharge the Git capabilities buil...
Eric Amodio Install

C/C++ 0.24.0 23M ★3.5
C/C++ IntelliSense, debugging, and ...
Microsoft Install

ESLint 1.9.0 21.9M ★4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer Install

Debugger for Ch... 4.11.6 20.6M ★4
Debug your JavaScript code in the C...
Microsoft Install

Language Supp... 0.47.0 18.7M ★4.5
Java Linting, Intellisense, formatting, ...
Red Hat Install

vscode-icons 8.8.0 17.2M ★5
Icons for Visual Studio Code
VSCode Icons Team Install

Vetur 0.21.1 17M ★4.5
Vue tooling for VS Code
Pine Wu Install

C# 1.21.0 15.6M ★4
C# for Visual Studio Code (powered ...
Microsoft Install

```
      checkValidServiceWorker(swUrl, config);

      // Add some additional logging to localhost, p
      // service worker/PWA documentation.
      navigator.serviceWorker.ready.then(() => {
        product
        productSub
        removeSiteSpecificTrackingException
        removeWebWideTrackingException
        requestMediaKeySystemAccess
        sendBeacon
        serviceWorker  (property) Navigator.serviceWorke...
        storage
        storeSiteSpecificTrackingException
        storeWebWideTrackingException
    } userAgent
  }   vendor

function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
```

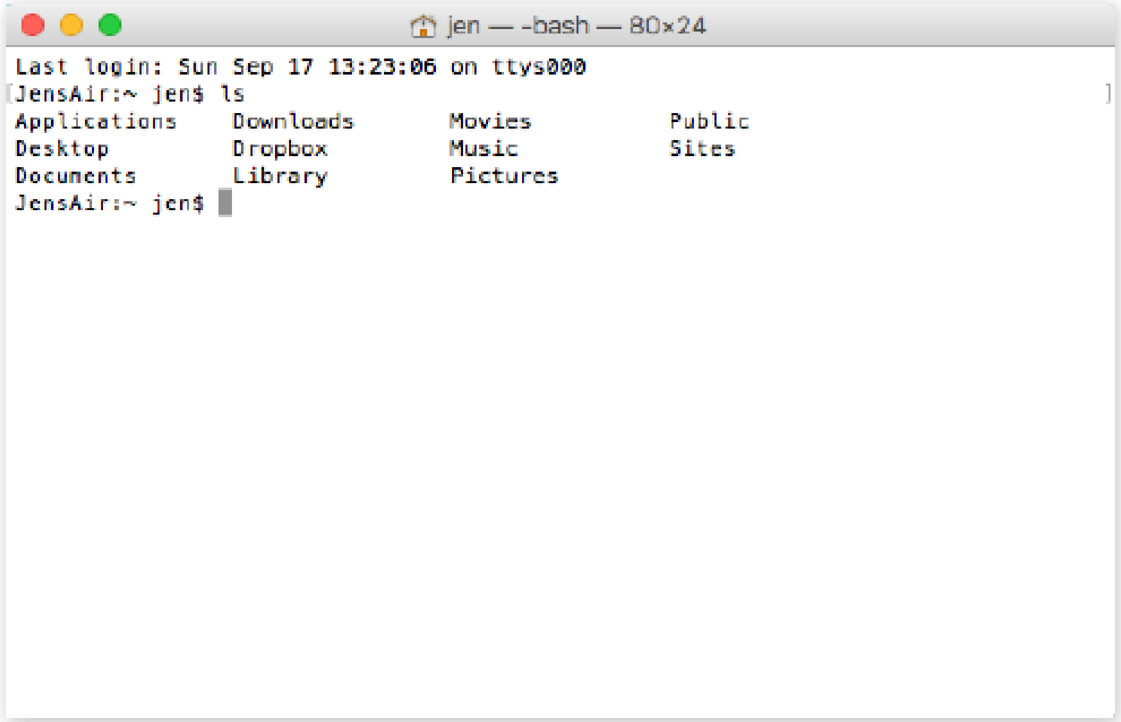
```
You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.211.55.3:3000/

Note that the development build is not optimized.
```

Command Line

SECTION 2

The Command-Line Terminal

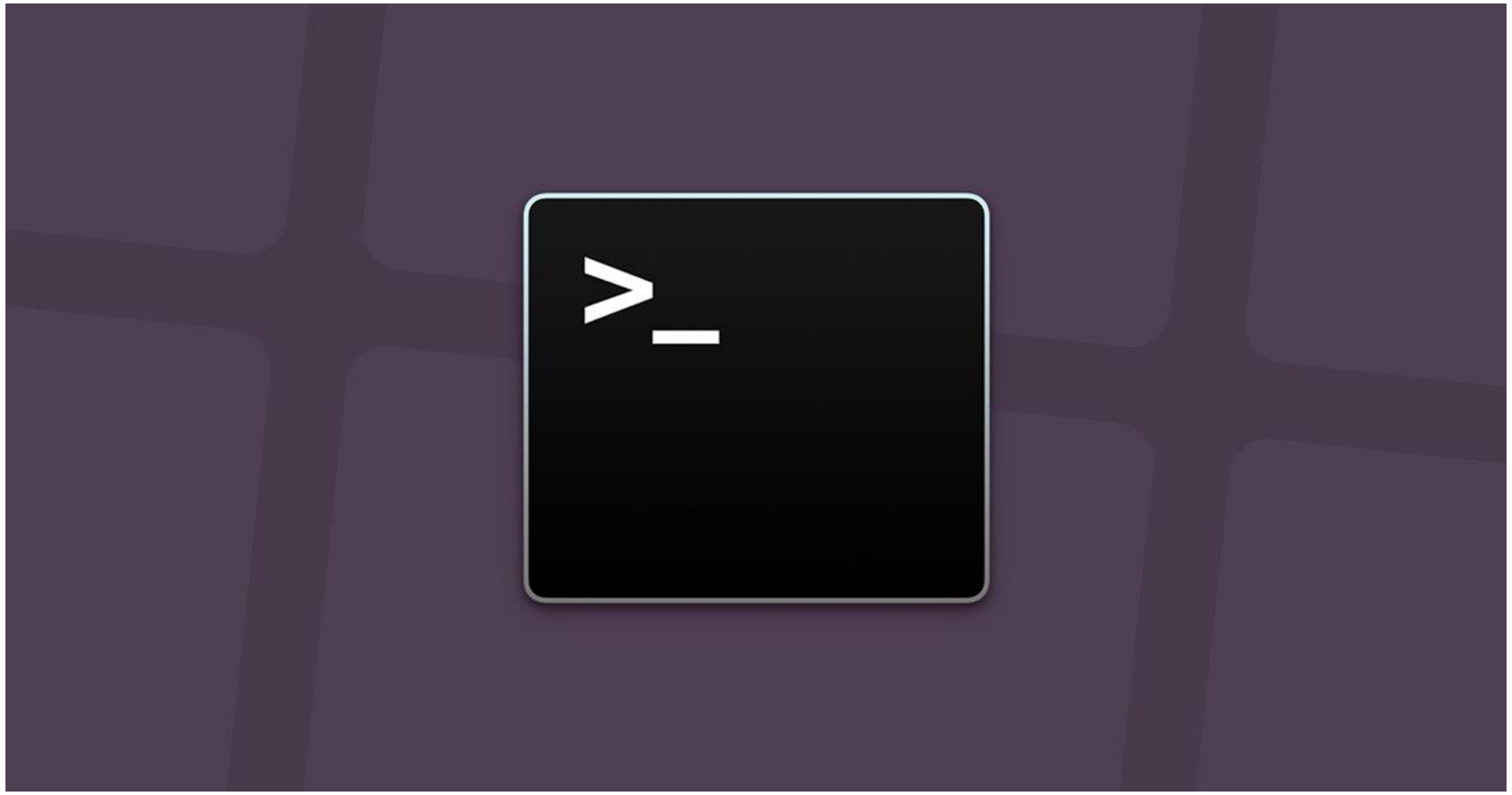
The program that interprets the commands you type is called a shell (visual interfaces are also technically a shell; they're just fancier). Every Mac and Linux machine comes installed with Terminal, which uses a shell program called bash. On macOS, you will find the Terminal program in Applications → Utilities (FIGURE 20-1).



```
Last login: Sun Sep 17 13:23:06 on ttys000
JensAir:~ jen$ ls
Applications    Downloads       Movies          Public
Desktop         Dropbox         Music           Sites
Documents       Library         Pictures
JensAir:~ jen$ 
```

FIGURE 20-1. The Terminal window in macOS.

Getting Started with Commands



25 Terminal Commands for MacOS
https://youtu.be/oStNbXzv7mE

A-Z Listing for Terminal Commands for MacOS
https://ss64.com/osx/

CSS Pre-Processor and Post-Processor

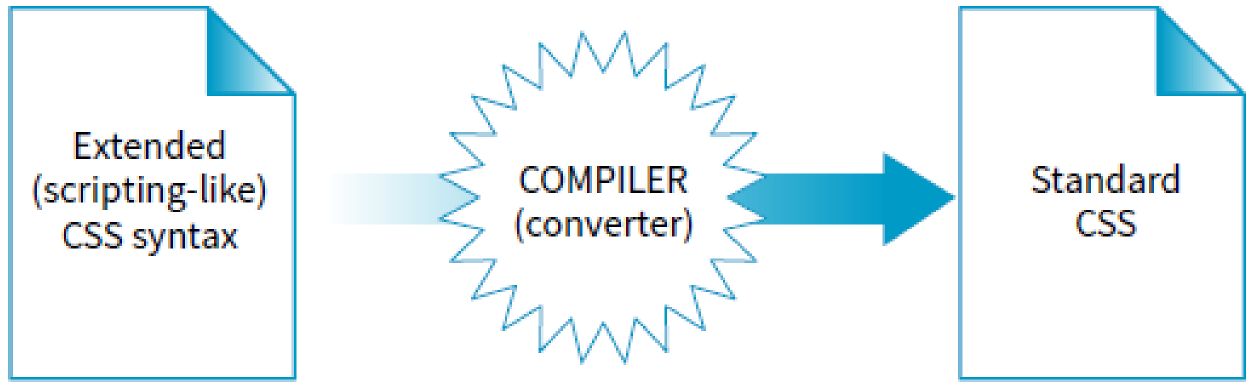
SECTION 3

CSS Pre-Processor and Post-Processor

I know that you are just getting used to writing CSS, but I would be negligent if I didn’t introduce you to some advanced CSS power tools that have become central to the professional web developer workflow. They fall into two general categories:

- Languages built on top of CSS that employ time-saving syntax characteristics of traditional programming languages. These are traditionally known as preprocessors. The most popular preprocessors as of this writing are Sass, LESS, and Stylus. When you write your styles in one of these languages, you have to use a program or script to convert the resulting file into a standard CSS document that browsers can understand.
- CSS optimization tools take your clean, standard CSS and make it even better by improving cross-browser consistency, reducing file size for better performance, and enhancing many other tasks. Tools that optimizebrowser-ready CSS are commonly known as postprocessors.

Introduction to Preprocessors (Especially Sass)



.scss	Sass Ruby gem or Node Sass (JS)	
.less	less.js	*.css*
.styl	stylus.js	

FIGURE 20-3. A simplified view of the role of a preprocessor.

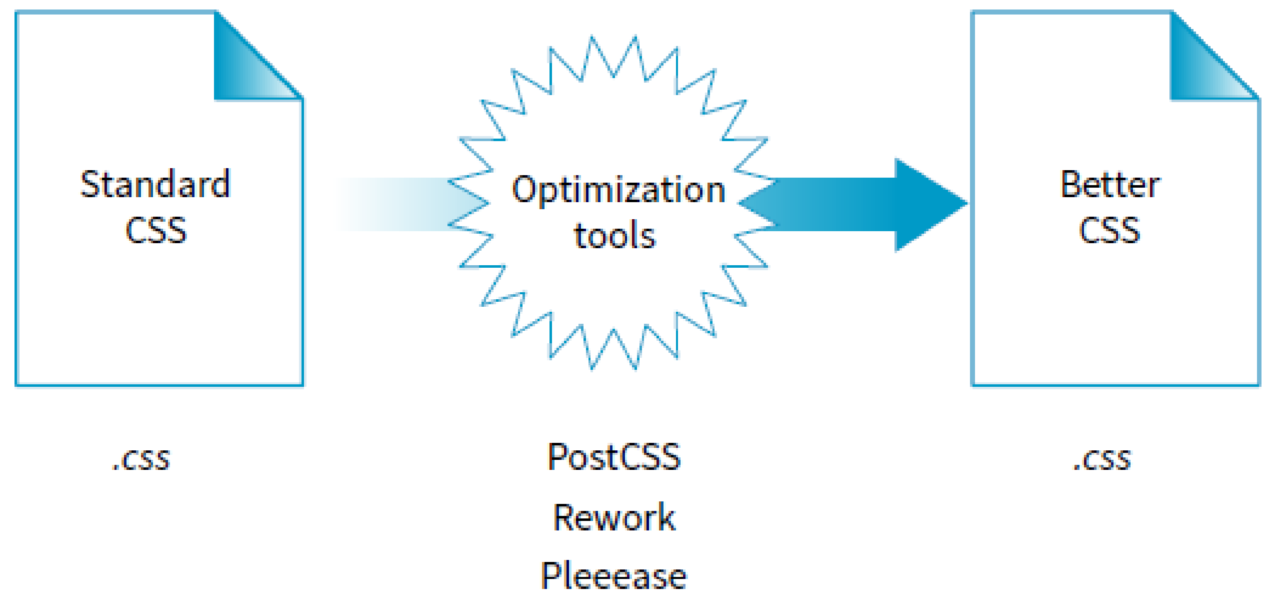


FIGURE 20-4. Postprocessors optimize existing, standard CSS files.

Introduction to Postprocessors (Mostly PostCSS)

Build Tools and Task Runner

SECTION 4

Build Tools (GRUNT AND GULP)

- In the world of software, a build process is required to test source code and compile it into a piece of executable software. As websites evolved from a collection of static HTML files to complex JavaScript-reliant applications, often generated from templates, build tools have become integral to the web development workflow as well.
- Some web build tools like Grunt and Gulp are commonly referred to as task runners. You use them to define and run various tasks (anything you might do manually from the command line) on your working HTML, JavaScript, CSS, and image files to get them ready to publish.

	Browserify	Grunt	Gulp	Webpack
Description	Browser-side require() the node way	The Java Script Task Runner	The streaming build system	Module bundler
Created	7 years ago (Feb, 2011)	7 years ago (Jan, 2012)	5 years ago (Jul, 2013)	6 years ago (Mar, 2012)
Version Average	every 6 days	every a month	every a month	every 4 days
Maintainers	40	4	64	520
Dependencies	48	17	13	25
Monthly downloads	2,261,042	2,038,947	3,387,902	14,960,064
Open issues	322	135	20	491
Open pull requests	30	20	0	30
GitHub stars	12,115	11,837	29,982	42,790
Subscribers	318	570	1,239	1,542
Forks	1,059	1,544	4,353	5,407

Automation

- You can automate your tasks, too, so they happen in the background without your needing to type commands. To do this, you tell the build tool to “watch” your files and folders for changes. When a change is detected, it triggers the relevant tasks to run automatically as you’ve configured them.
- Once you have the task runner configured and set to watch your files, you can go about your business writing CSS, and all that command-line stuff happens for you without ever touching a terminal application. Here’s how that might look. Imagine making a change to your Sass file and saving it. Grunt instantly sees that the *.scss* file has changed, automatically converts it to *.css* (see Note), and then reloads the browser to reflect your change.

Some Common Tasks

- The previous section on CSS processors should have given you an idea of some things that would be nice to automate. Allow me to list several more to give you a solid view of the ways task runners make your job easier.
  - Concatenation. It is common for web teams to divide style sheets and scripts into small, specialized chunks of *.css* and *.js*. When it's time to publish, however, you want as few calls to the server as possible for performance purposes, so those little chunks get concatenated (put together) into master files.
  - Compression and "minification." Another way to improve performance is to make your files as small as possible by removing unnecessary spaces and line returns. Build tools can compress your CSS and minify JavaScript.

Some Common Tasks

- Checking your HTML, CSS, and JavaScript for errors (linting).
- Optimizing images with tools that squeeze down the file size of all the images in a directory.
- Help committing or pushing changes to a version control repository (Git).
- Refreshing your browser to reflect whatever changes you just made to a file (LiveReload plug-in).
- Building final HTML files from templates and content data (see the sidebar “Building Sites with Data and Templates”).
- Running CSS pre- and postprocessors.

Grunt

- The first and most established web build tool is Grunt (gruntjs.com), presumably named for handling all of the “grunt work” for you. It is a JavaScript tool built on the open source Node.js framework, and you operate it using the command line.
- The compelling thing about Grunt is that the **development community** has created literally thousands of plug-ins that perform just about any task you can think of. Just download one, configure it, and start using it. You do not have to be a JavaScript master to get started.

Gulp

- Another popular option is Gulp (gulpjs.com), which has the advantage of running a little faster but also requires more technical knowledge than Grunt because you configure it with actual JS code. Other contenders as of this writing are Webpack (quite popular!), Brunch, Browserify, and Broccoli. New tools with amusing names pop up on a regular basis. Some developers simply use Node.js-based scripts without using a task-runner program as a go-between. The point is, there are plenty of options.

- You will find many online tutorials for learning how to download and configure the build tool of your choice when you are ready to automate your workflow. I hope that I have made you aware of the possibilities, and when a job interviewer mentions Grunt and Gulp, you'll know they aren't just suffering from indigestion.

Git Version Control

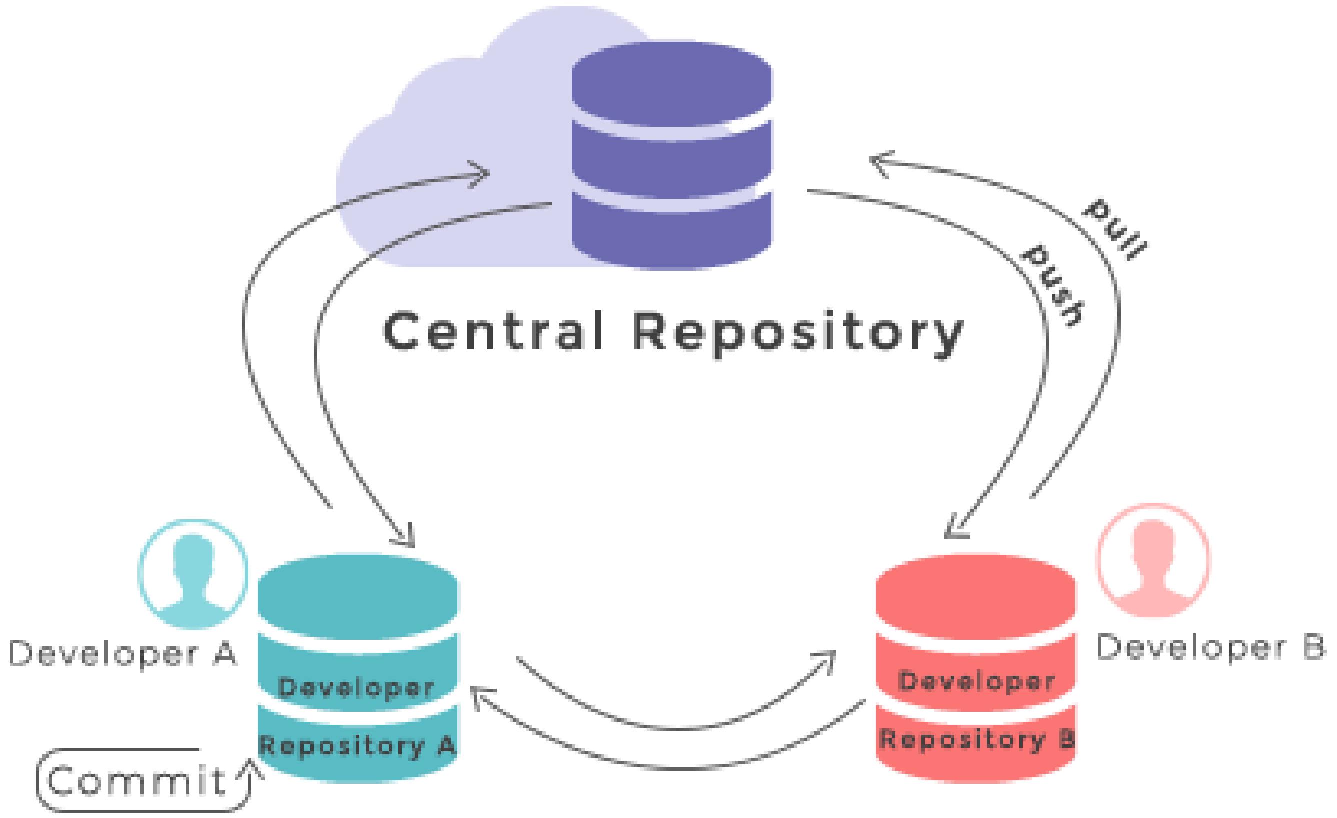
SECTION 5

Version Control with Git and Github

- We'll begin with a basic distinction: **Git is the version control program** that you run on your computer; GitHub (*github.com*) is a service that hosts Git projects, either free or for a fee. You interact with GitHub by using Git, either from the command line, with the user interface on the GitHub website, or using a standalone application that offers a GUI interface for Git commands. This was not obvious to me at first, and I want it to be clear to you from the get-go.
- **GitHub and services like it (see Note) are mainly web-based wrappers around Git**, offering features like issue tracking, a code review tool, and a web UI for browsing files and history. They are convenient, but keep in mind that you can also set up Git on your own server and share it with your team members with no third-party service like GitHub involved at all.

What is a Git Repository?

- A Git Repository is a storage of your project files, which makes it possible to save code versions and have access to them.



What is a Git Repository?

https://www.w3docs.com/learn-git/introduction4.html

- Git Init for Initializing a new Repository
- First of all, you need git init command to create a new repository. This command is only used once, while initialising a new repository.

```
git init
```

- Consequently, a new subdirectory and a new master branch will be created:

```
git init <directory>
```

- This creates an empty Git Repository in the specified directory. Running this command will create a new folder which contains the .git subdirectory.

Git clone for Cloning an Existing Repository

- The git clone command is used for creating a local clone of an already existing repository. It is a one-time operation, too.

```
git clone <repo url>
```

Git add and Git Commit for Saving Changes to the Repository

- Using git add and git commit commands, you can save the file version changes to your repository. Here is an example of the usage of these two commands:
    - change directories to /path/of/project
    - create a new file GitCommit.txt with contents "commit example for git repo"
    - git add GitCommit.txt to the repository staging area
    - create a new commit message describing the work done

```
cd /path/of/project
echo "commit example for git repo"
>> GitCommit.txt
git add GitCommit.txt
git commit -m "added GitCommit.txt
to the repo"
```

Git Push for Repo-to-Repo Interaction

- If you executed `git clone` command, it means that you already have a remote repository, so you can run git push command to push your changes to that repository.
- But if you used `git init`, you have no remote repository. In this case, you can use a hosted Git service, like Github or Bitbucket, and create your repo there, which will give a URL that you can add to your local repository and git push to the hosted repository.

Git Config for Configuration and Set up

- A remote repository is added to the local git config with the help of git remote command:

```
git remote add <remote_name> <remote_repo_url>
```

- After adding the remote repo, you can push local branches to it:

```
git push -u <remote_name> <local_branch_name>
```

Git Config for Configuration and Set up

- You may also need to set global Git configuration options such as username, or email. With the help of the git config command, you can configure your Git installation from the command line. This command defines everything (user info, preferences, the behavior of a repository). Below you will find some configuration options.
- Use the `--global` flag to set configuration options for the current user. Define the name of the author of all the commits in the current repository.

```
git config --global user.name <name>
```

Git Config for Configuration and Set up

- Adding the --local option or not using a configuration option at all, will define the user.name for the current local repository. Define the e-mail of the author of all the commits by the current user.

```
git config --local user.email <email>
```

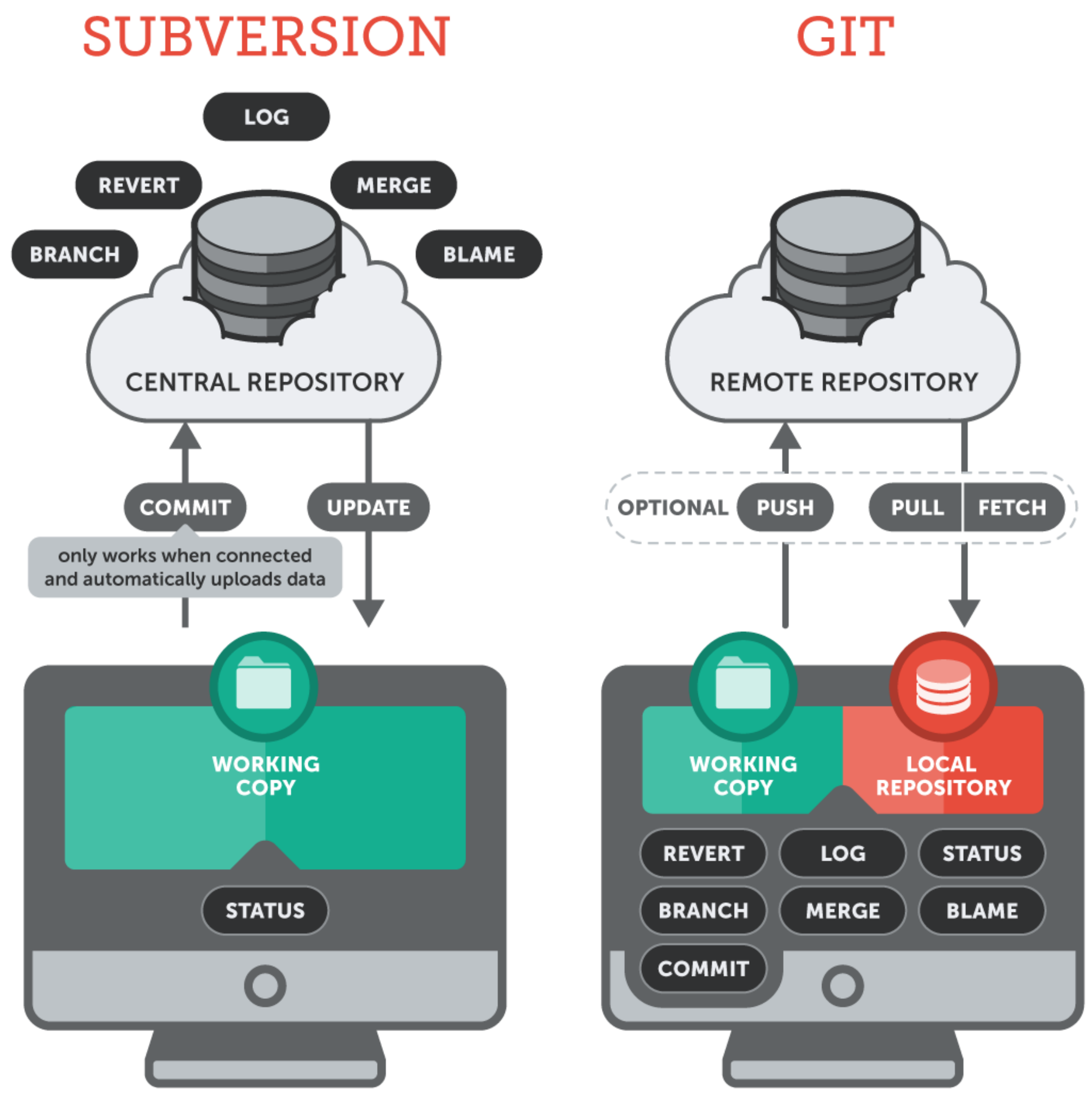
- Use the --system option to set the configuration for the whole system, that is to say all users and repos on a machine.

```
git config --system core.editor <editor>
```

Git Hosting Services

https://www.c-sharpcorner.com/article/git-for-absolute-beginners-with-command-line-interface/

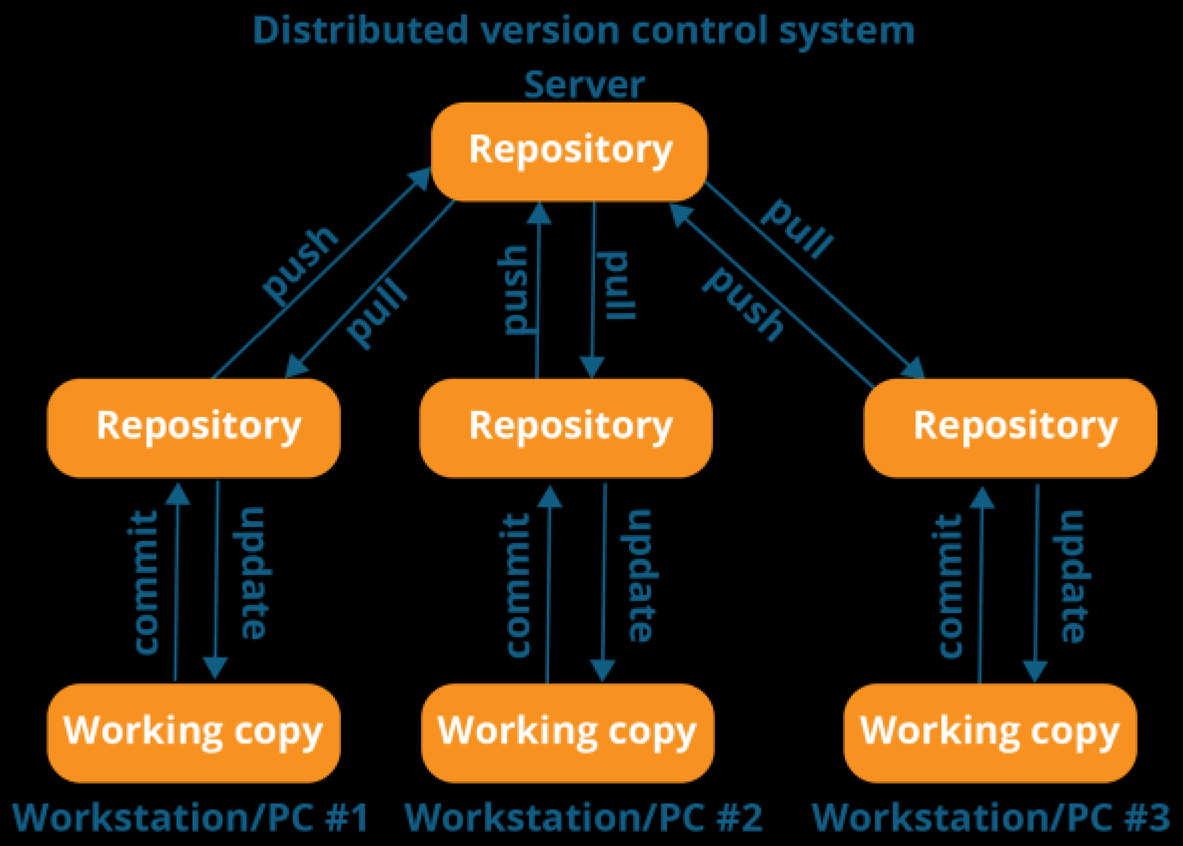
- Git is a popular **version control system** for your software development. It's focuses on being high powered, offering data integrity and being able to support non-linear workflows. Like you'll find with most version control solutions, Git tracks complete history with full version tracking functionality. Because Git is a distributed version control solution, it is able to be used right out of the box. Git stores each file revision of a file as a unique blob. Blob relationships are discovered by investigating tree and commit objects. Each newly added object is stored by using zlib compression.
- Git was actually first designed and developed by Linus Torvalds, the creator of the Linux kernel. Torvalds jokingly named "Git" after himself as the word is British slang for an "unpleasant person".



Git Hosting Services

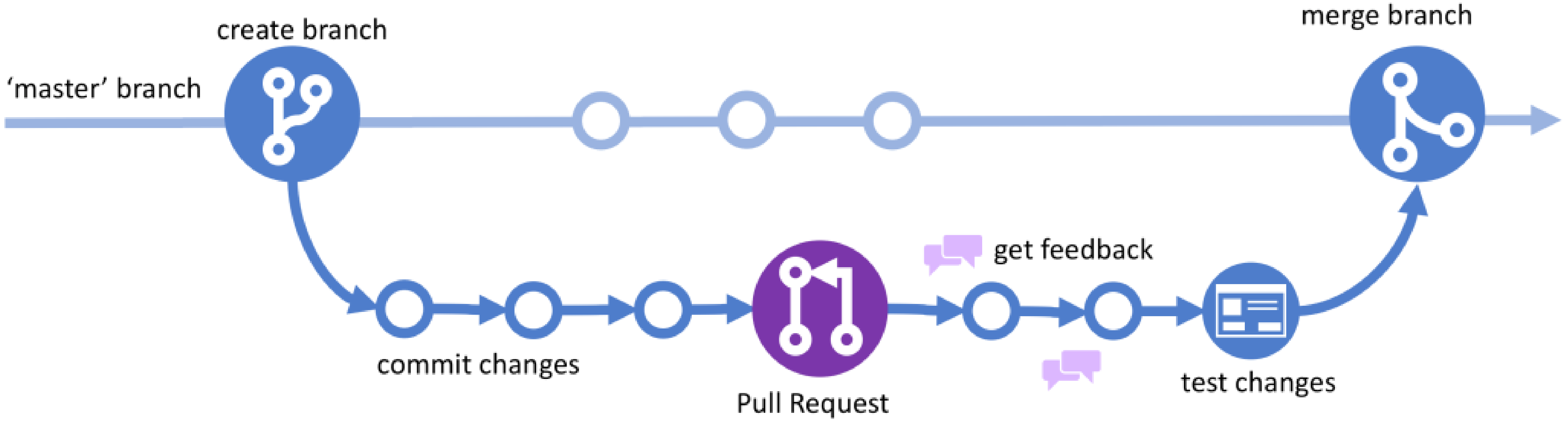
Git Hosting Services

Git is not the only version control system; there are many, you can see a comparison on Wikipedia, There are many online services which provide git services. Popular ones are GitHub, GitLab, BitBucket & Visual Studio Team Services (Yes, they do provide git, along with VSTS).



Distributed Version Control System

GitHub Flow



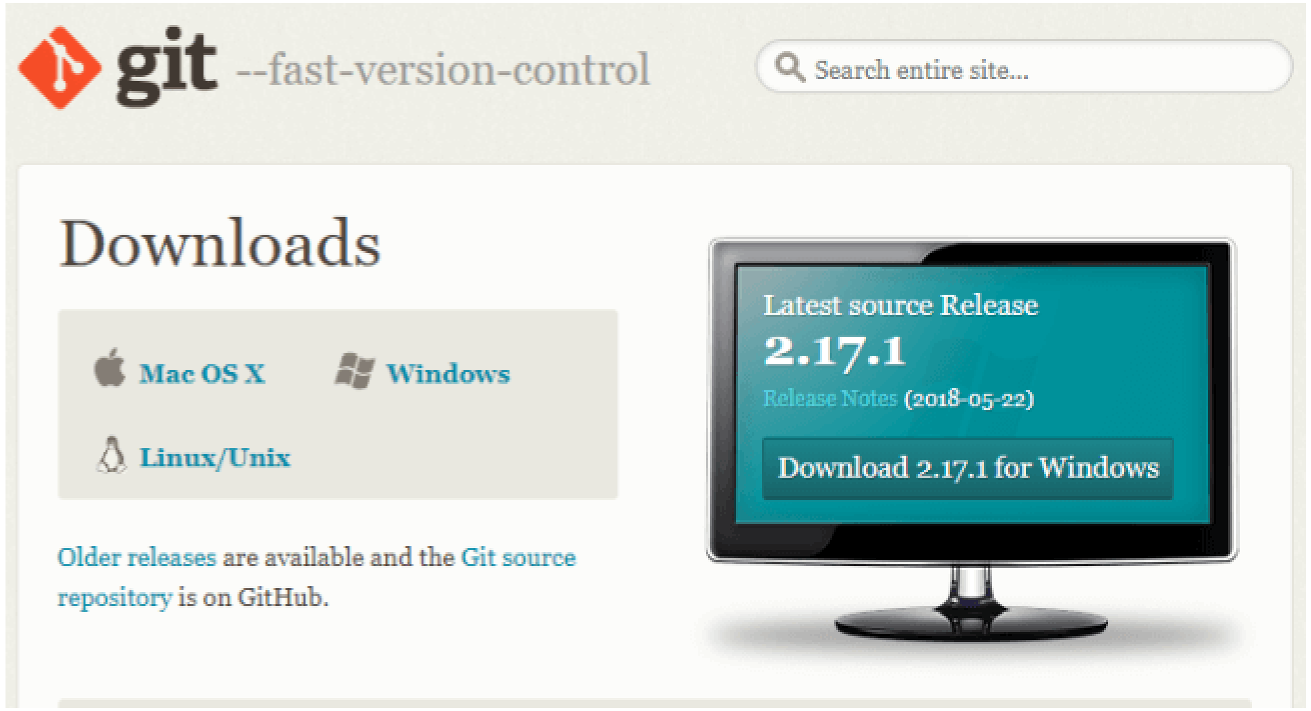
http://buildazure.com

Example Git Project

SECTION 6

Installing Git

Download appropriate version of git according to your platform from https://git-scm.com/ and install.


git --fast-version-control

Search entire site...

Downloads

Mac OS X Windows

Linux/Unix

Older releases are available and the Git source repository is on GitHub.

Latest source Release

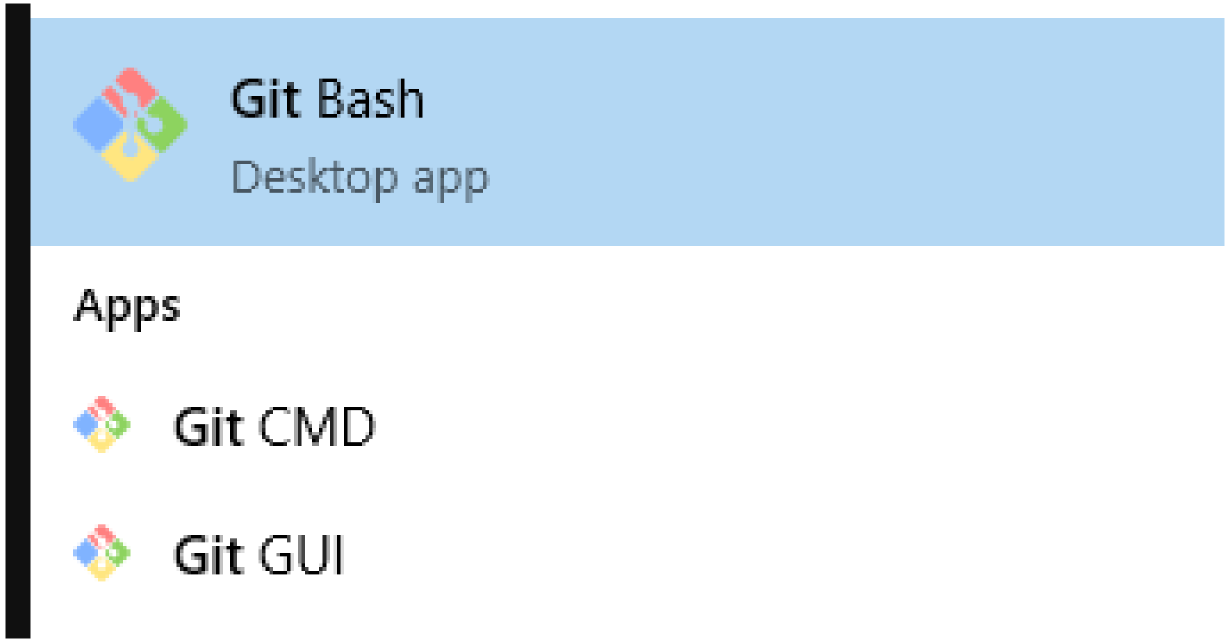
2.17.1

Release Notes (2018-05-22)

Download 2.17.1 for Windows

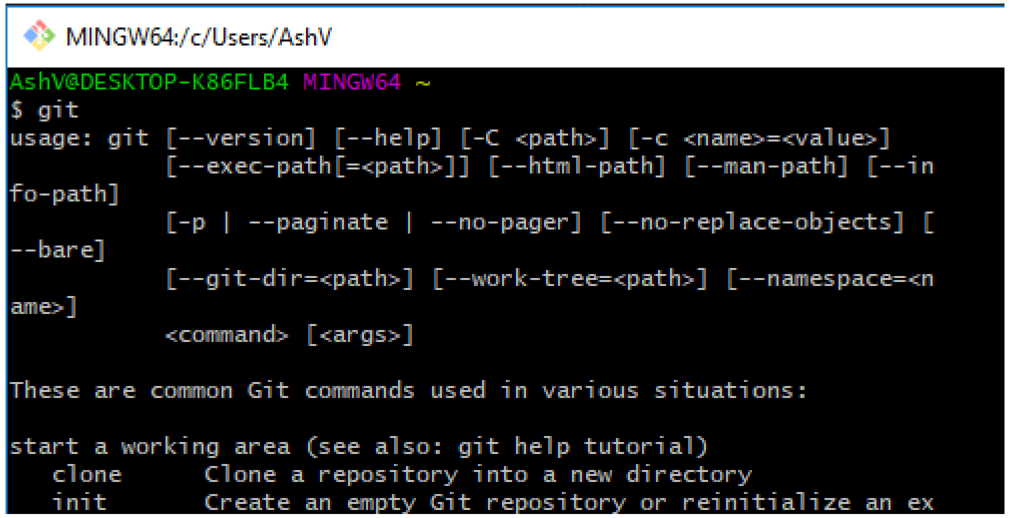
Verifying Installation

- You will get Git Bash, Git CMD, & Git GUI after successful installation of git. You can use a command line in either Git Bash or Git CMD according to your preference. If Windows is your preferred OS, then Git CMD is a better choice and if you love Linux, go for Git Bash. You can also run git commands from Windows command prompt if the environment variable is set.



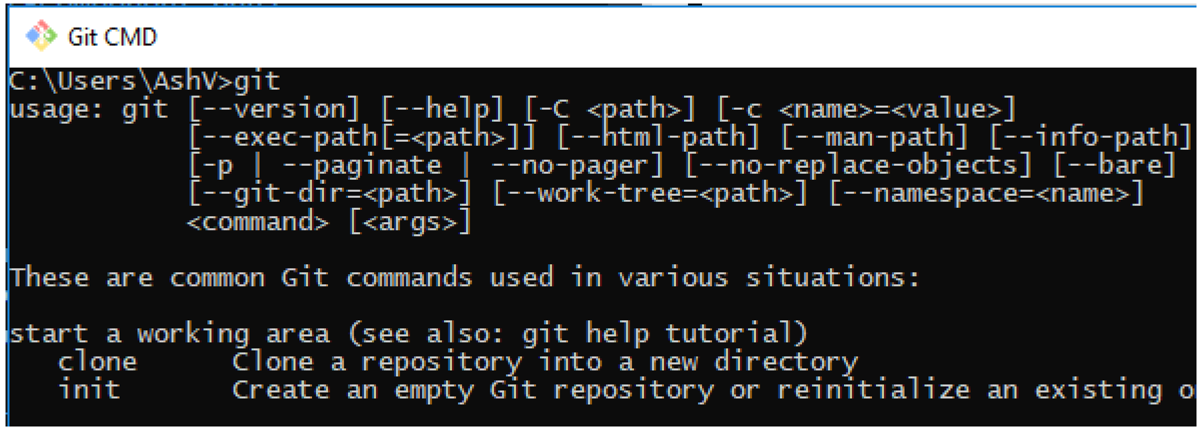
Verifying Installation

- Run git command to verify successful installation. All git commands should be written in small letters compulsorily.



MINGW64:/c/Users/AshV

```
AshV@DESKTOP-K86FLB4 MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--in
fo-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [
--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<n
ame>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an ex
```



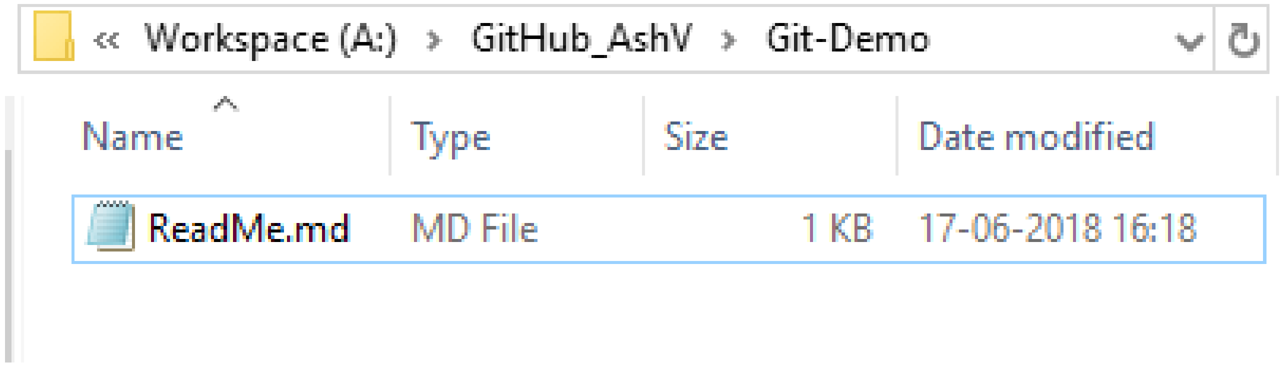
Git CMD

```
C:\Users\AshV>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing o
```

Creating New Repository

- As explained above, the repository is a place where source code is stored physically. Git init command does this job for us. You need to go to the folder which you want to make as a repository and run this command. No matter if the folder is empty or has files, you can make that a repository.

```
$ git init
```



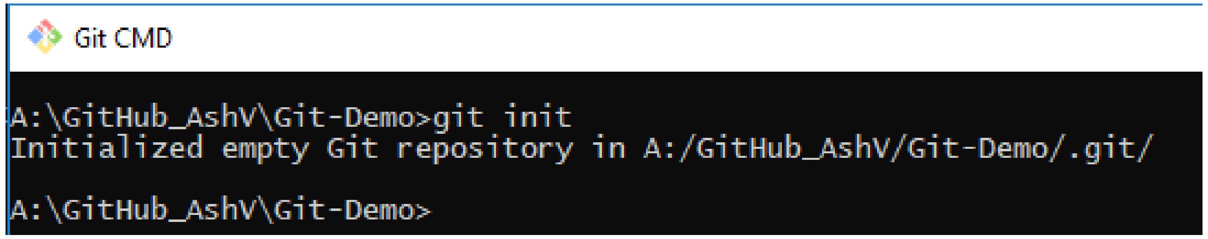
« Workspace (A:) › GitHub_AshV › Git-Demo

Name	Type	Size	Date modified
ReadMe.md	MD File	1 KB	17-06-2018 16:18

Creating New Repository

- As explained above, the repository is a place where source code is stored physically. Git init command does this job for us. You need to go to the folder which you want to make as a repository and run this command. No matter if the folder is empty or has files, you can make that a repository.

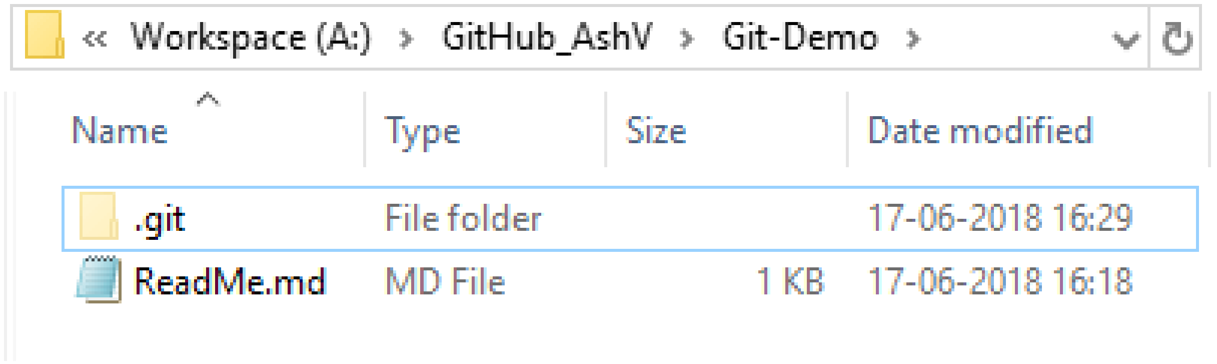
```
$ git init
```



```
A:\GitHub_AshV\Git-Demo>git init
Initialized empty Git repository in A:/GitHub_AshV/Git-Demo/.git/

A:\GitHub_AshV\Git-Demo>
```

Creating New Repository

- I created one folder in my file system with name Git-Demo & ran git init in this folder using the command line. You can see the message Initialized empty Git repository in A:/GitHub_AshV/Git-Demo/.git/. Did you observe .git in the message. Let's verify in the file system.



« Workspace (A:) › GitHub_AshV › Git-Demo ›

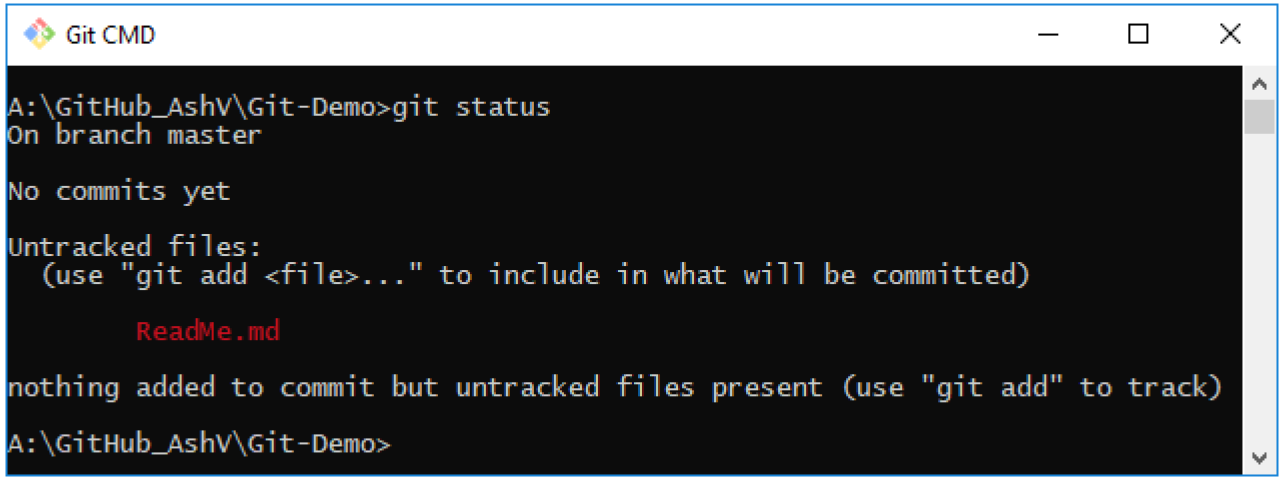
Name	Type	Size	Date modified
.git	File folder		17-06-2018 16:29
ReadMe.md	MD File	1 KB	17-06-2018 16:18

- There is one hidden folder created with name .git, this folder is responsible for maintaining versions and other metadata related to the repository. (Hidden files & folder should be visible in your OS settings to see this folder).

Status of Repository

- To get the current status of repository git status command is used. This command is very frequently used, let's see what is the status of our repository after git init.

```
$ git status
```



```
A:\GitHub_AshV\Git-Demo>git status
On branch master

No commits yet

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

        ReadMe.md

nothing added to commit but untracked files present (use "git add" to track)

A:\GitHub_AshV\Git-Demo>
```

- Hmmm, ReadMe.md is untracked 🤔, Untracked file means it will be there in the file system but git will not be aware of it. Let's add it to the repository.

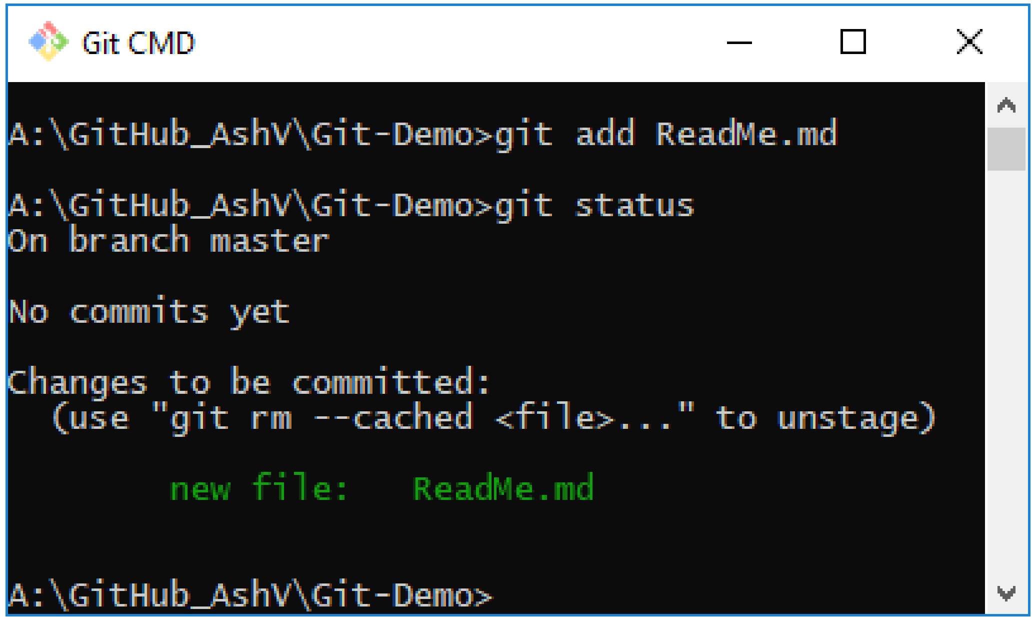
Adding/Staging Files to Repository

If you have observed in the message of git init, it says "Initialized empty Git repository", git init initializes the repository with no files even if you have files present in that folder. To add files in repository 'git add' command is used.

```
$ git add
```

Let's add ReadMe.md file to git and check the status.

```
$ git add ReadMe.md
```



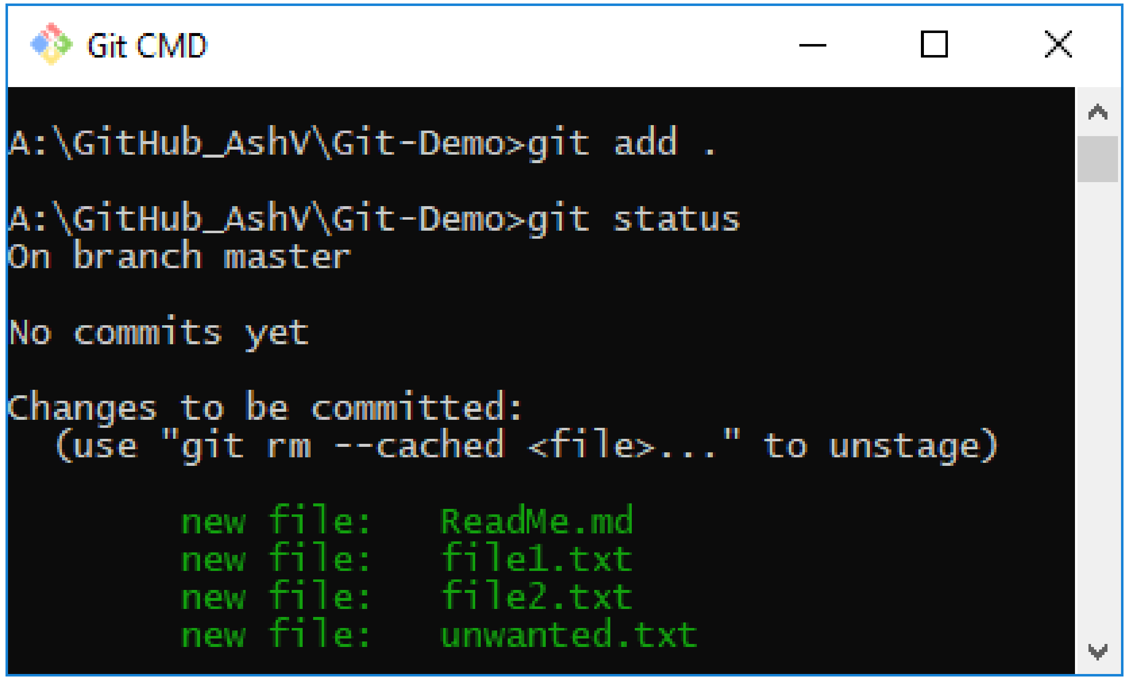
```
A:\GitHub_AshV\Git-Demo>git add ReadMe.md

A:\GitHub_AshV\Git-Demo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   ReadMe.md


A:\GitHub_AshV\Git-Demo>
```

Adding/Staging Files to Repository

If we have multiple files, then we have to run this command for each file, or we can add all files with git add . and remove/unstage unwanted files later.

I created 3 more files in folder "file1.txt, file2.txt & unwanted.txt" and adding them to the repository with git add ..

```
$ git add .
```



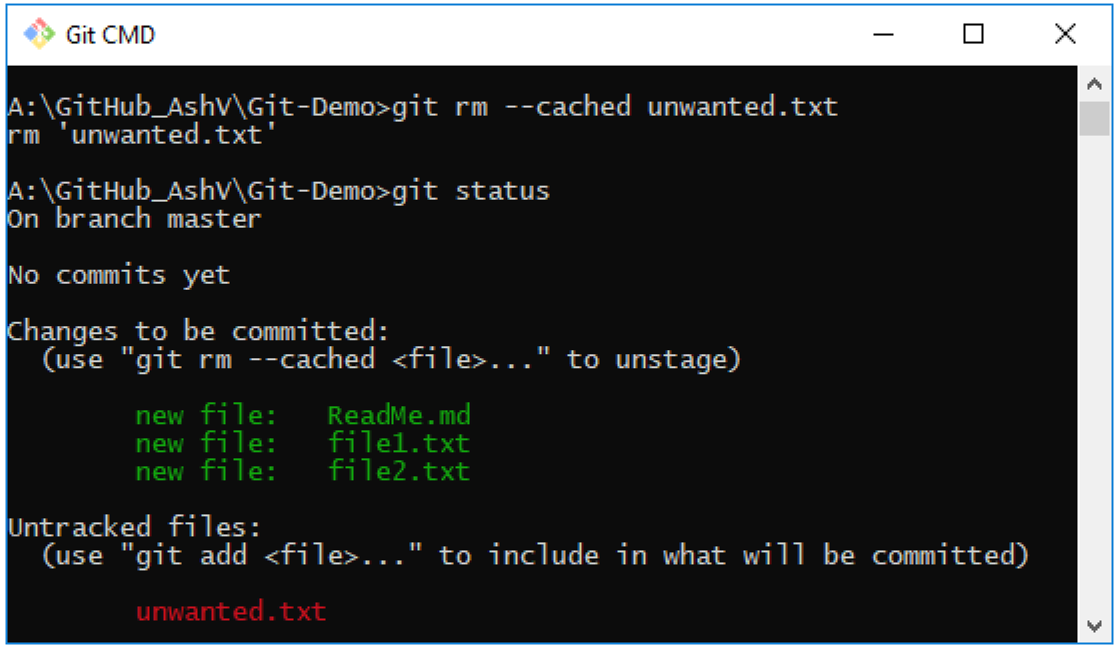
Git CMD

```
A:\GitHub_AshV\Git-Demo>git add .

A:\GitHub_AshV\Git-Demo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   ReadMe.md
        new file:   file1.txt
        new file:   file2.txt
        new file:   unwanted.txt
```

Removing/Unstaging Files in Repository

If you want to remove some staged file from repository 'git rm' command is used. If you wish to just unstage file from the repository but want to keep in the file system you need to use --cached in git rm command.

$ git rm --cached unwanted.txt

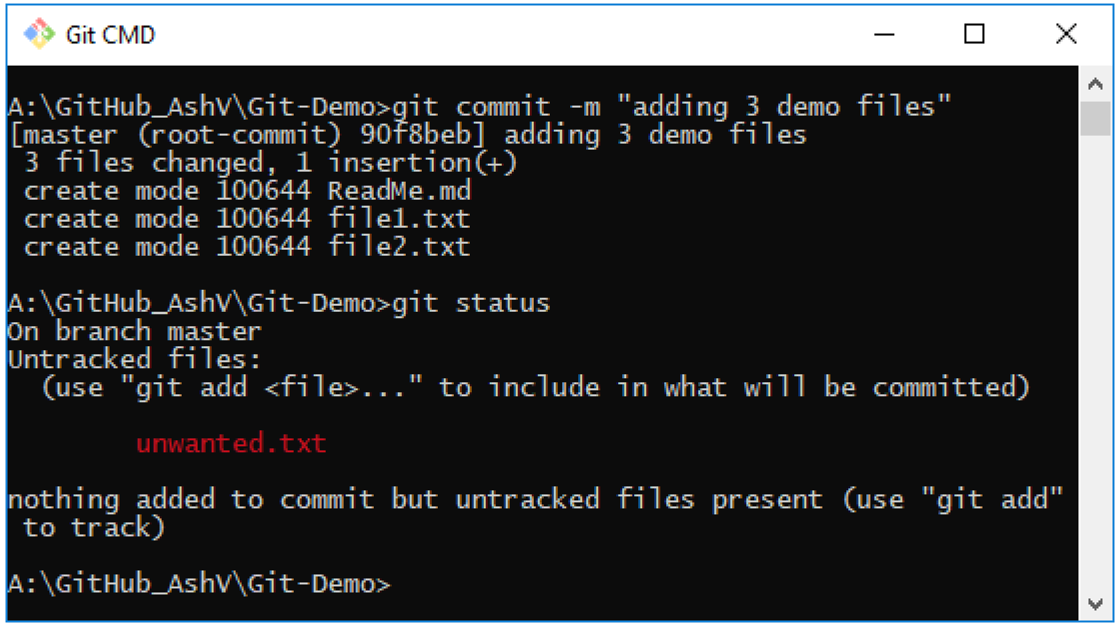


```
A:\GitHub_AshV\Git-Demo>git rm --cached unwanted.txt
rm 'unwanted.txt'

A:\GitHub_AshV\Git-Demo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   ReadMe.md
        new file:   file1.txt
        new file:   file2.txt

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

        unwanted.txt
```

Committing the Changes

Add git command: Just add files to repository, but to log your changes/make versions you need to commit your changes using git commit command. So far we have created 3 files, let's commit them. While committing changes we should provide proper commit message which describes what change are made in this commit. -m is used to provide commit message.

$ git commit -m "adding 3 demo files"



```
A:\GitHub_AshV\Git-Demo>git commit -m "adding 3 demo files"
[master (root-commit) 90f8beb] adding 3 demo files
 3 files changed, 1 insertion(+)
 create mode 100644 ReadMe.md
 create mode 100644 file1.txt
 create mode 100644 file2.txt

A:\GitHub_AshV\Git-Demo>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        unwanted.txt

nothing added to commit but untracked files present (use "git add"
 to track)

A:\GitHub_AshV\Git-Demo>
```

You can observe here before commit those 3 files were visible in green under "Changes to be committed" section, now they are committed, that's why they're not visible here.

Making Further Changes in Code & Committing Them

After commit if you make changes to existing files or new files, you need to again run git add to stage them & git commit to commit the changes. So far everything we did in repository is local in our machine. Let's see how to Push changes to server/remote repository.

Adding Remote Repository

Making Further Changes in Code & Committing Them

- After commit if you make changes to existing files or new files, you need to again run git add to stage them & git commit to commit the changes. So far everything we did in repository is local in our machine. Let's see how to Push changes to server/remote repository.

Adding Remote Repository

- Remote repository is the one which is located in the server, and different collaborators push changes to it, which can be taken by other collaborators.
- To have a remote repository we can use any of the services including GitHub, GitLab, BitBucket & Visual Studio Team Services. Basically they all work in the same way, so if you are able work with one, you can wok with others too. We will take GitHub as example here.

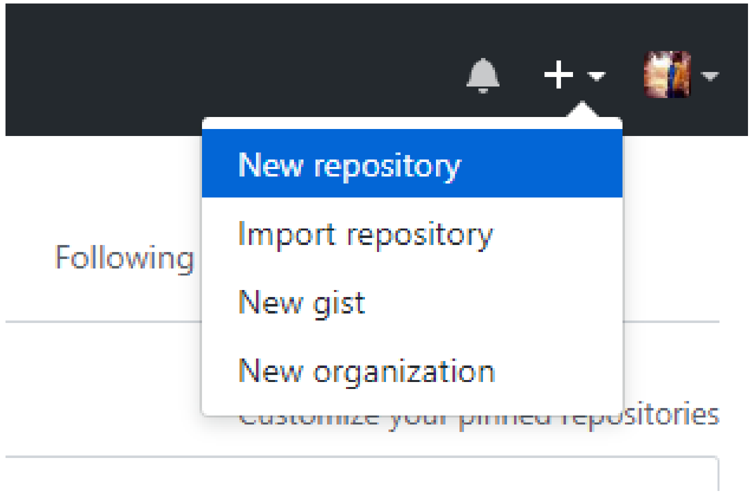
Adding Remote Repository

Private vs Public Repository

- Public repos are used by Open Source projects mostly, anyone can view their code and commit history e.g Selenium, .Net Core etc. Private repos are used when you don't want to make your code public e.g your client's code. Except GitHub, the above mentioned other services are providing private repositories for free. By the way you won't feel any difference between them while working with them.

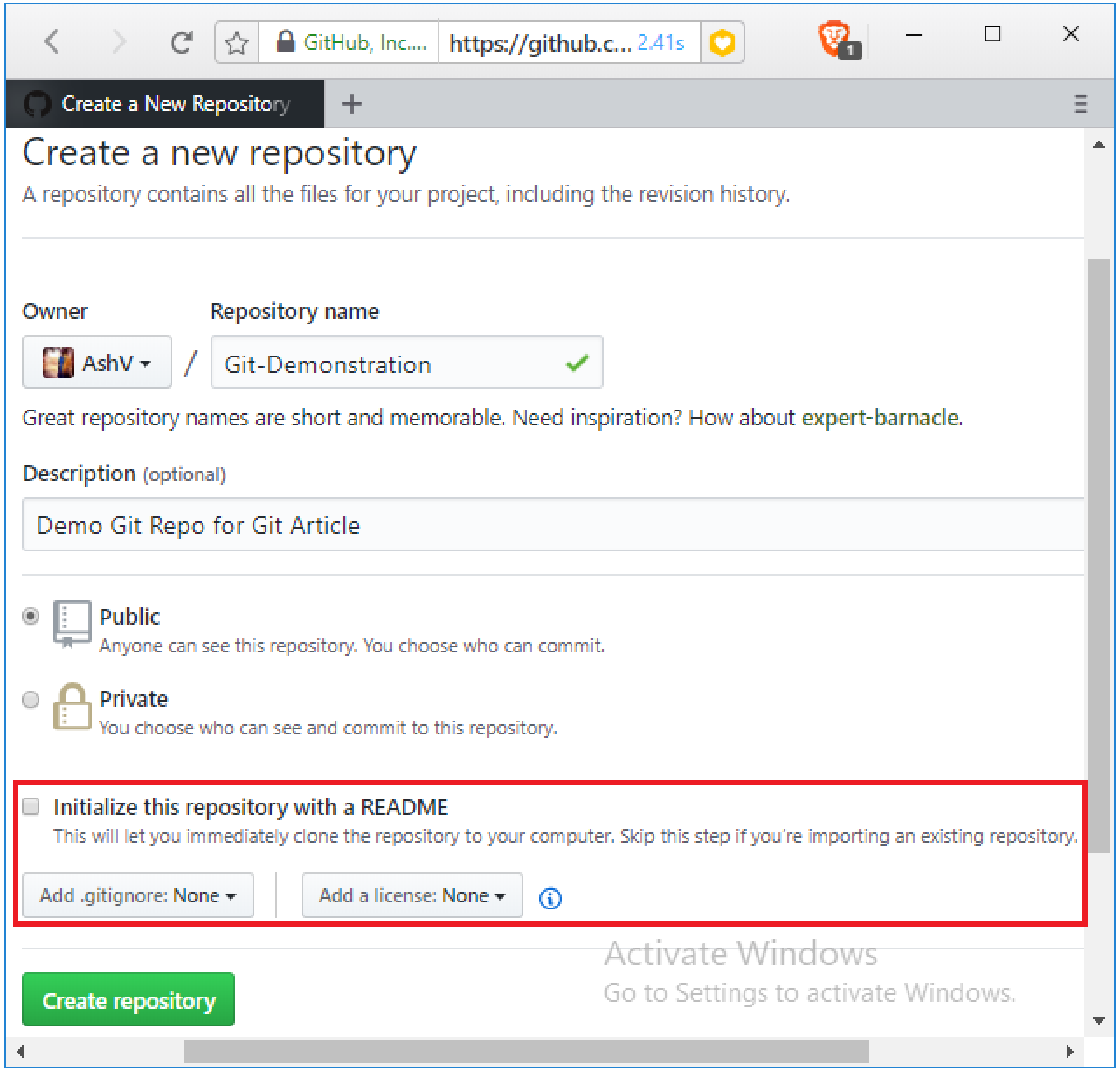
Create Repository in GitHub

- Navigate to (https://github.com)[https://github.com] login into your account or create one if don't have already. And click on create new repository.



- You can give any name here, it could be different than your folder name in local. I have given different name Git-Demonstration. Don't select anything in highlighted section those are for completely new repositories but here we have already created a repository locally.

Create Repository in GitHub



Create a New Repository

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: AshV / Repository name: Git-Demonstration ✓

Great repository names are short and memorable. Need inspiration? How about expert-barnacle.

Description (optional): Demo Git Repo for Git Article

[✓ 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

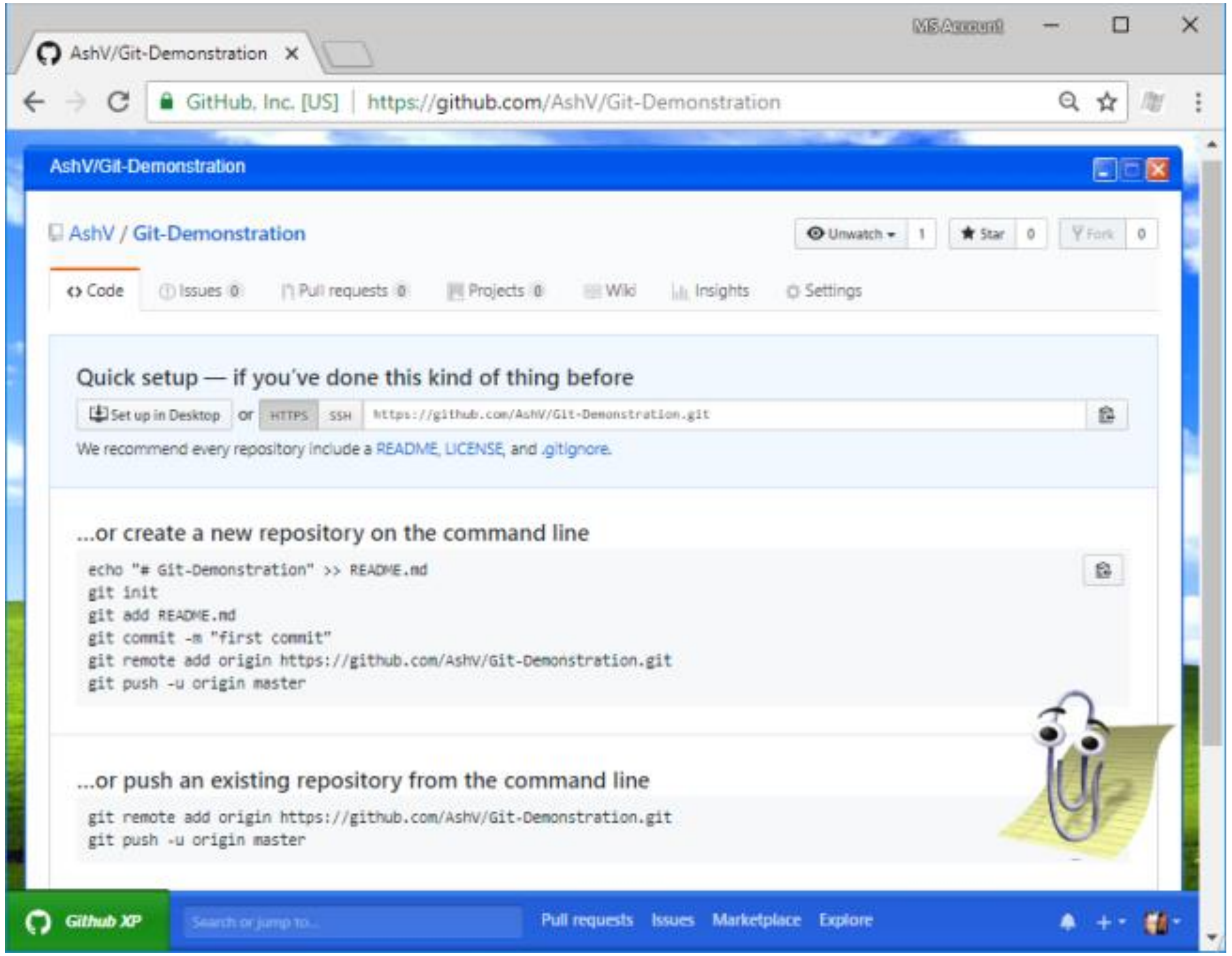
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

https://github.c... 2.41s
GitHub, Inc....
Create repository
Activate Windows
Go to Settings to activate Windows.

Create Repository in GitHub

After click on "Create Repository" you will see a similar screen with command given.



Quick setup — if you've done this kind of thing before

```
https://github.com/AshV/Git-Demonstration.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# Git-Demonstration" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/AshV/Git-Demonstration.git
git push -u origin master
```

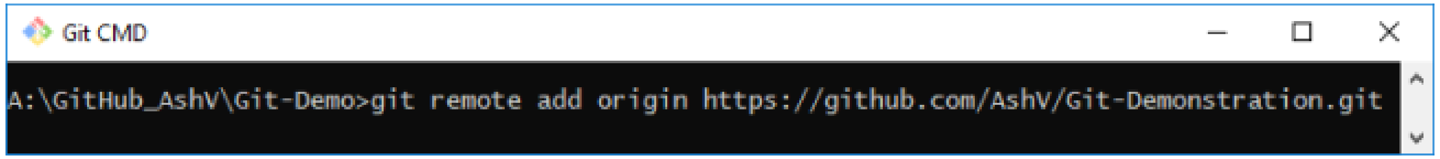
...or push an existing repository from the command line

```
git remote add origin https://github.com/AshV/Git-Demonstration.git
git push -u origin master
```

Link Remote Repository

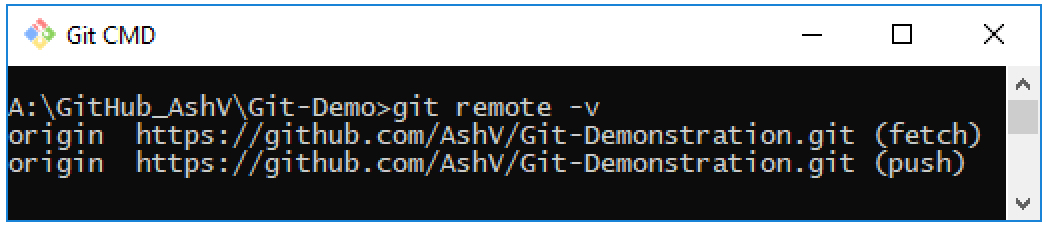
You can observe git remote command in above image, copy and run it in your machine to add remote repository in your local repository.

$ git remote $ git remote add origin https://github.com/AshV/Git-Demonstration.git



```
A:\GitHub_AshV\Git-Demo>git remote add origin https://github.com/AshV/Git-Demonstration.git
```

To verify remote URL you can run this command with -v
$ git remote -v



```
A:\GitHub_AshV\Git-Demo>git remote -v
origin  https://github.com/AshV/Git-Demonstration.git (fetch)
origin  https://github.com/AshV/Git-Demonstration.git (push)
```

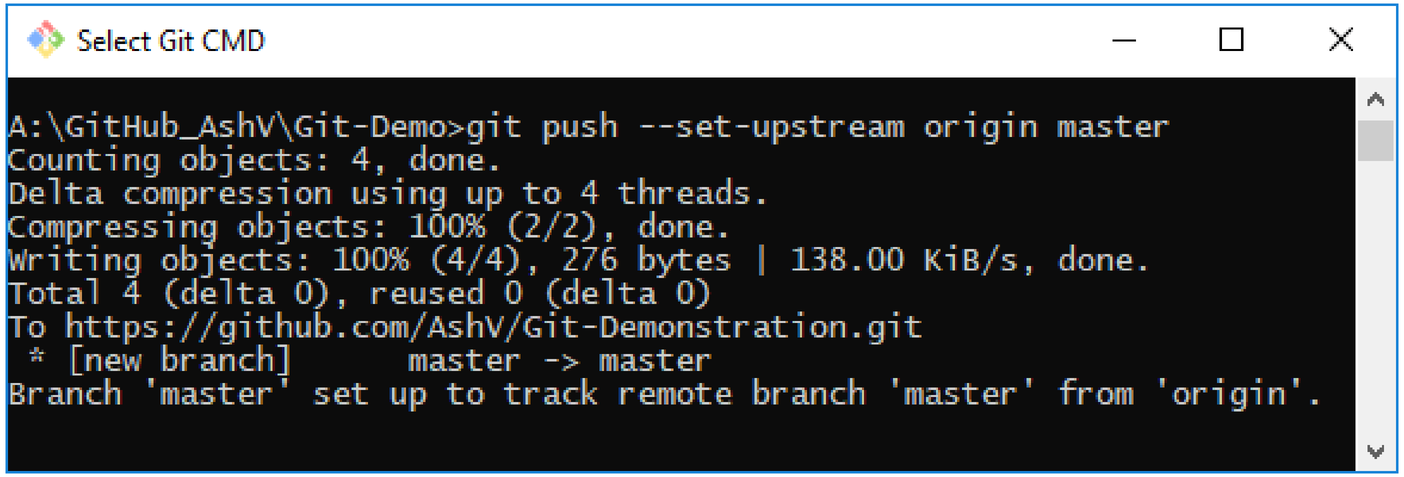
Link Remote Repository

- With git remote now our repository knows where our remote server is, but to push changes there we need to run git push, while pushing to remote for the first time we need to set remote as upstream for the current branch, which is master by default. (branching in git is very popular and useful feature, as this article is intended for beginners, so I will be covering them in further article). Run below command to set upstream.

$ git push --set-upstream origin master

- Here is the short version of above with a more descriptive command.

$ git push -u origin master

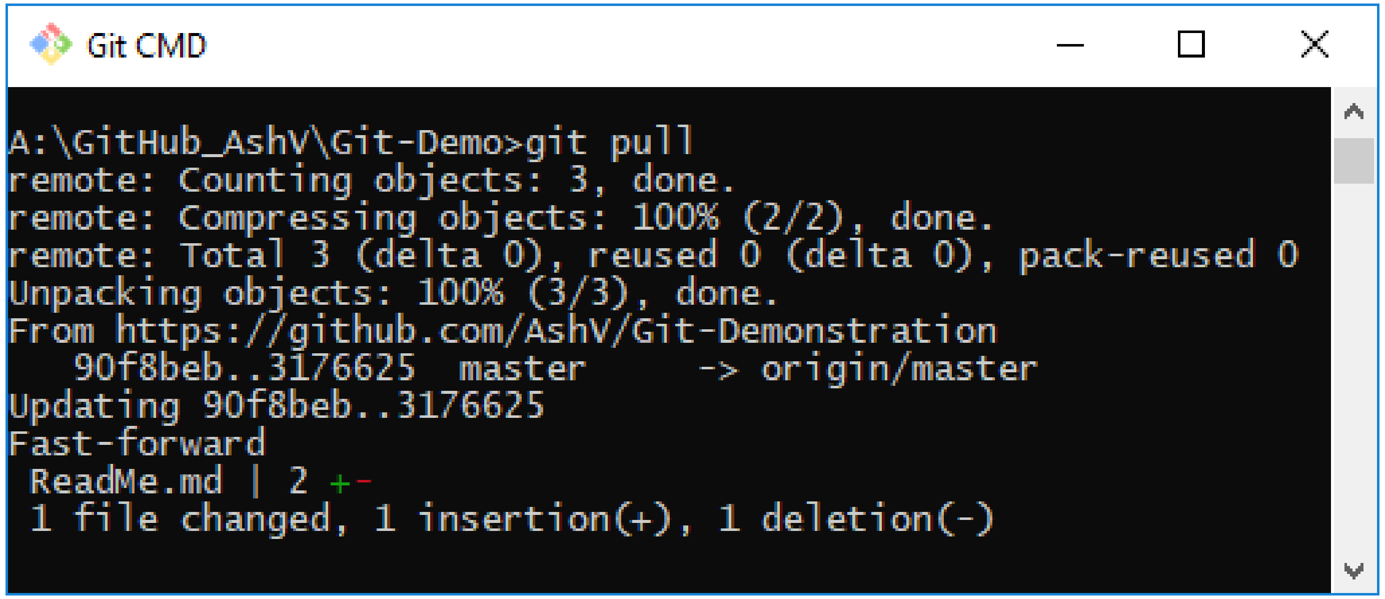


```
A:\GitHub_AshV\Git-Demo>git push --set-upstream origin master
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 276 bytes | 138.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/AshV/Git-Demonstration.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Taking latest from Server

- While working on a project with multiple collaborators, then it's always a good idea to take the latest before starting to make changes in the repository to avoid conflicts and warnings while pushing. 'git pull' does this job, I'm making some change in the remote repo to show demo, then I'll pull them in local.

$ git pull



```
A:\GitHub_AshV\Git-Demo>git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/AshV/Git-Demonstration
   90f8beb..3176625  master     -> origin/master
Updating 90f8beb..3176625
Fast-forward
 ReadMe.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- You can see the latest changes made -- ReadMe.md files are retrieved to local.

Conclusion

A quick recap, we have explored these commands in this article.

1. git config
2. git init
3. git status
4. git add
5. git rm
6. git commit
7. git remote
8. git push
9. git pull

You can start using git with the command line with them. Further commands will be covered in upcoming articles. You can explore commits made in this article at *https://github.com/AshV/Git-Demonstration/commits/master*