

Basics of HTML and Git

Prepared By,

Tiya Jose

Table of Contents

1. HTML Basics
2. Git

HTML basics

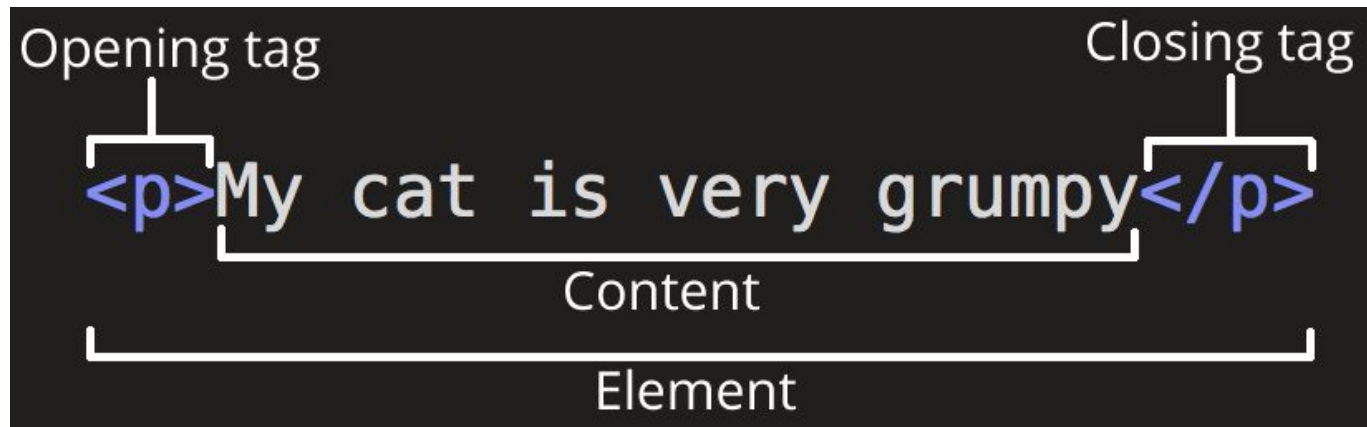
So what is HTML, really?

- HTML is not a programming language; it is a *markup language* that defines the structure of your content.
- Markup languages are designed for the processing, definition and presentation of text
- HTML stands for Hyper Text Markup Language
- HTML consists of a series of **elements**, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way.
- The enclosing **tags** can make a word or image hyperlink to somewhere else, can italicize words, and can make font bigger or smaller, and so on.

Anatomy of an HTML element

The main parts of our element are:

1. The opening tag
2. The closing tag
3. The content
4. The element



HTML Tags

HTML tags are element names surrounded by angle brackets:

```
<tagname>content goes here...</tagname>
```

- HTML tags normally come **in pairs** like <p> and </p>
- The first tag in a pair is the **start tag**, the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

Structure of HTML

- The `<!DOCTYPE html>` declaration defines this document to be HTML5
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the document
- The `<title>` element specifies a title for the document
- The `<body>` element contains the visible page content
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

<html>

- This element wraps all the content on the entire page.
- The `<html>` element is the root element of an HTML page

<head>

- The **<head>** element is a container for metadata (data about data) and is placed between the `<html>` tag and the `<body>` tag.
- HTML metadata is data about the HTML document. Metadata is not displayed.
- Metadata typically define the document title, character set, styles, links, scripts, and other meta information.

Tag	Description
<u><head></u>	Defines information about the document
<u><title></u>	Defines the title of a document
<u><base></u>	Defines a default address or a default target for all links on a page
<u><link></u>	Defines the relationship between a document and an external resource
<u><meta></u>	Defines metadata about an HTML document
<u><script></u>	Defines a client-side script
<u><style></u>	Defines style information for a document

`<body>`

- The `<body>` tag defines the document's body.
- The `<body>` element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

Anatomy of an HTML document

Here we have:

- `<!DOCTYPE html>`
- `<html> </html>`
- `<head> </head>`
- `<body> </body>`
- `<meta charset="utf-8">`
- `<title> </title>`

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Anatomy of an HTML document</title>
  </head>
  <body>
    <h1>Anatomy of an HTML document</h1>
  </body>
</html>
```

HTML Tags Ordered by Category

1. Basic HTML
2. Formatting
3. Frames
4. Images
5. Audio / Video
6. Links
7. Lists
8. Tables
9. Styles

Basic HTML

Tag	Description
<u><!DOCTYPE></u>	Defines the document type
<u><html></u>	Defines an HTML document
<u><head></u>	Defines information about the document
<u><title></u>	Defines a title for the document
<u><body></u>	Defines the document's body
<u><h1> to <h6></u>	Defines HTML headings
<u><p></u>	Defines a paragraph
<u>
</u>	Inserts a single line break
<u><hr></u>	Defines a thematic change in the content
<u><!--...--></u>	Defines a comment

Formatting

[`<address>`](#) Defines contact information for the author/owner of a document/article

[``](#) Defines bold text

[`<blockquote>`](#) Defines a section that is quoted from another source

[`<cite>`](#) Defines the title of a work

[`<code>`](#) Defines a piece of computer code

[`<pre>`](#) Defines preformatted text

HTML5 Tutorial

HTML HOME
HTML Introduction
HTML Editors
HTML Basic
HTML Elements
HTML Attributes
HTML Headings
HTML Paragraphs
HTML Styles
HTML Formatting
HTML Quotations
HTML Comments
HTML Colors
HTML CSS
HTML Links
HTML Images
HTML Tables
HTML Lists
HTML Blocks

HTML Text Formatting Elements

Tag	Description
<code></code>	Defines bold text
<code></code>	Defines emphasized text
<code><i></code>	Defines italic text
<code><small></code>	Defines smaller text
<code></code>	Defines important text
<code><sub></code>	Defines subscripted text
<code><sup></code>	Defines superscripted text
<code><ins></code>	Defines inserted text
<code></code>	Defines deleted text
<code><mark></code>	Defines marked/highlighted text



Templates

by


W3.CSS







Forms and Input

Tag	Description
<u><form></u>	Defines an HTML form for user input
<u><input></u>	Defines an input control
<u><textarea></u>	Defines a multiline input control (text area)
<u><button></u>	Defines a clickable button
<u><select></u>	Defines a drop-down list
<u><optgroup></u>	Defines a group of related options in a drop-down list
<u><option></u>	Defines an option in a drop-down list
<u><label></u>	Defines a label for an <input> element
<u><fieldset></u>	Groups related elements in a form
<u><legend></u>	Defines a caption for a <fieldset> element
<u><datalist></u>	 Specifies a list of pre-defined options for input controls
<u><output></u>	 Defines the result of a calculation


Images

Tag	Description
<code></code>	Defines an image
<code><map></code>	Defines a client-side image-map
<code><area></code>	Defines an area inside an image-map
<code><canvas></code>	 Used to draw graphics, on the fly, via scripting (usually JavaScript)
<code><figcaption></code>	 Defines a caption for a <code><figure></code> element
<code><figure></code>	 Specifies self-contained content
<code><picture></code>	 Defines a container for multiple image resources

Audio / Video

Tag	Description
<u><audio></u>	 Defines sound content
<u><source></u>	 Defines multiple media resources for media elements (<video>, <audio> and <picture>)
<u><track></u>	 Defines text tracks for media elements (<video> and <audio>)
<u><video></u>	 Defines a video or movie

Links

Tag	Description
<code><a></code>	Defines a hyperlink
<code><link></code>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<code><nav></code>	 Defines navigation links

Lists

[](#) Defines an unordered list

[](#) Defines an ordered list

[](#) Defines a list item

[<dl>](#) Defines a description list

[<dt>](#) Defines a term/name in a description list

[<dd>](#) Defines a description of a term/name in a description list











[<menu>](#) Defines a list/menu of commands

[<menuitem>](#) Defines a command/menu item that the user can invoke from a popup menu

Tables

Tag	Description
<u><table></u>	Defines a table
<u><caption></u>	Defines a table caption
<u><th></u>	Defines a header cell in a table
<u><tr></u>	Defines a row in a table
<u><td></u>	Defines a cell in a table
<u><thead></u>	Groups the header content in a table
<u><tbody></u>	Groups the body content in a table
<u><tfoot></u>	Groups the footer content in a table
<u><col></u>	Specifies column properties for each column within a <colgroup> element
<u><colgroup></u>	Specifies a group of one or more columns in a table for formatting

Styles and Semantics

Tag	Description
<u><style></u>	Defines style information for a document
<u><div></u>	Defines a section in a document
<u></u>	Defines a section in a document
<u><header></u>	 Defines a header for a document or section
<u><footer></u>	 Defines a footer for a document or section
<u><main></u>	 Specifies the main content of a document
<u><section></u>	 Defines a section in a document
<u><article></u>	 Defines an article
<u><aside></u>	 Defines content aside from the page content
<u><details></u>	 Defines additional details that the user can view or hide
<u><dialog></u>	 Defines a dialog box or window
<u><summary></u>	 Defines a visible heading for a <details> element
<u><data></u>	 Links the given content with a machine-readable translation

HTML Attributes

- All HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

The href Attribute

HTML links are defined with the **<a>** tag. The link address is specified in the **href** attribute:

Example

```
<a href="https://www.w3schools.com">This is a link</a>
```

HTML Images

HTML images are defined with the **** tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

Example

```

```


The HTML Style Attribute

Setting the style of an HTML element, can be done with the **style attribute**.

The HTML style attribute has the following **syntax**:

```
<tagname style="property:value;">
```

The **property** is a CSS property. The **value** is a CSS value.

HTML Background Color

The **background-color** property defines the background color for an HTML element.

This example sets the background color for a page to powderblue:

Example

```
<body style="background-color:powderblue;">
```

```
<h1>This is a heading</h1>
```

```
<p>This is a paragraph.</p>
```

```
</body>
```

HTML Text Color

The **color** property defines the text color for an HTML element:

Example

```
<h1 style="color:blue;">This is a heading</h1>
```

```
<p style="color:red;">This is a paragraph.</p>
```

HTML Fonts

The **font-family** property defines the font to be used for an HTML element:

Example

```
<h1 style="font-family:verdana;">This is a heading</h1>
```

```
<p style="font-family:courier;">This is a paragraph.</p>
```

HTML Text Size

The **font-size** property defines the text size for an HTML element:

Example

```
<h1 style="font-size:300%;">This is a heading</h1>
```

```
<p style="font-size:160%;">This is a paragraph.</p>
```

HTML Text Alignment

The **text-align** property defines the horizontal text alignment for an HTML element:

Example

```
<h1 style="text-align:center;">Centered Heading</h1>
```

```
<p style="text-align:center;">Centered paragraph.</p>
```

Follow the four steps below to create your first web page with Notepad or Text Editor

Step 1: Open Notepad/Text Editor (PC)

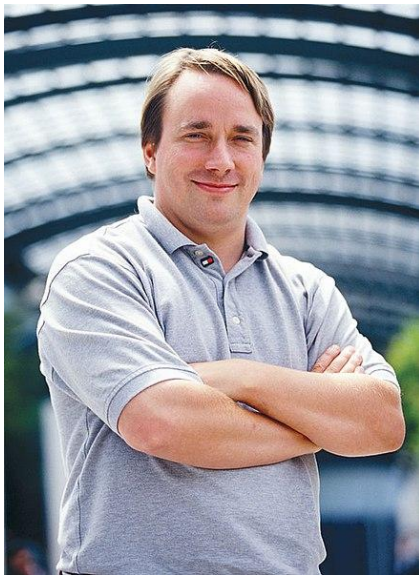
Step 2: Write HTML code

Step 3: Save the HTML Page with .html extension

Step 4: View the HTML Page in Your Browser

About Git

- Created by Linus Torvalds, creator of Linux, in 2005
- Goals of Git:
 - Speed
 - Support for non-linear development (thousands of parallel branches)
 - Fully distributed
 - Able to handle large projects efficiently



Git

- Git is a software that is used for Version Control

What is Version Control?

Version Control is the management of changes to documents, computer programs, large websites and other collection of information.

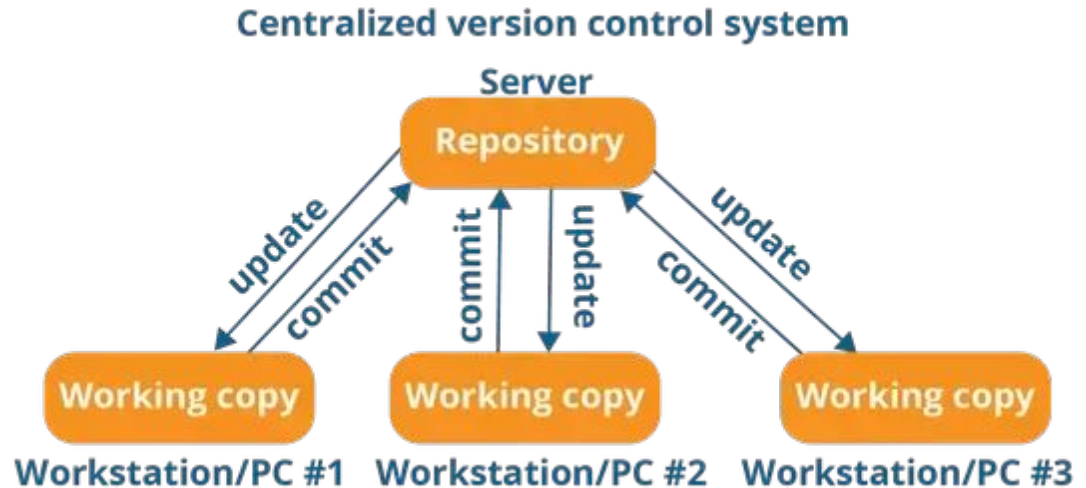
There are two types of VCS:

- Centralized Version Control System (CVCS)
- Distributed Version Control System (DVCS)



Centralized VCS

- Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration.
- It works on a single repository to which users can directly access a central server.
- Eg: Subversion (or SVN) and Perforce.



Pitfalls of Centralized VCS

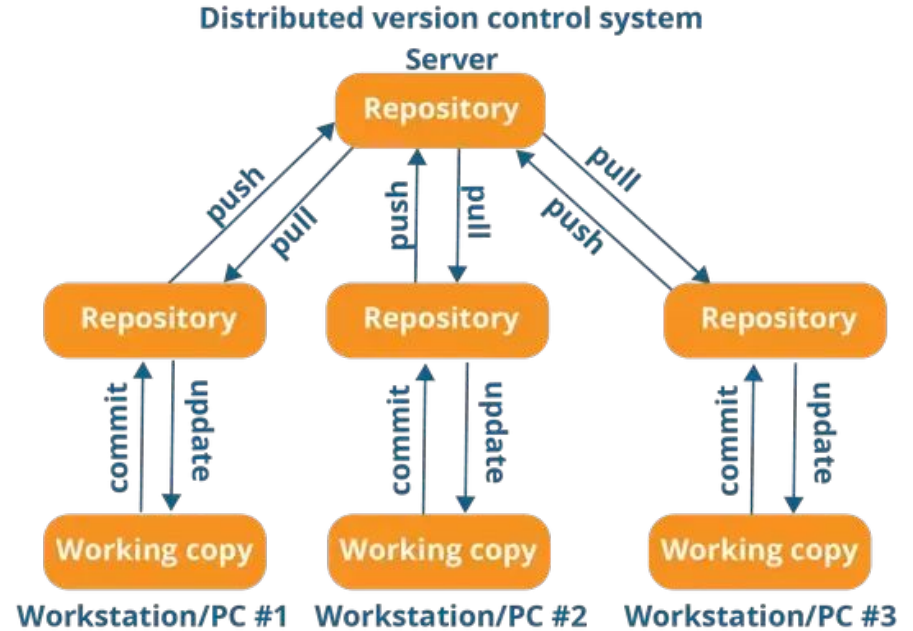
- It is not locally available; meaning you always need to be connected to a network to perform any action.
- Since everything is centralized, in any case of the central server getting crashed or corrupted will result in losing the entire data of the project.

This is when Distributed VCS comes to the rescue.

Distributed VCS

These systems do not necessarily rely on a central server to store all the versions of a project file.

In Distributed VCS, every contributor has a local copy or “clone” of the main repository i.e. everyone maintains a local repository of their own which contains all the files and metadata present in the main repository.

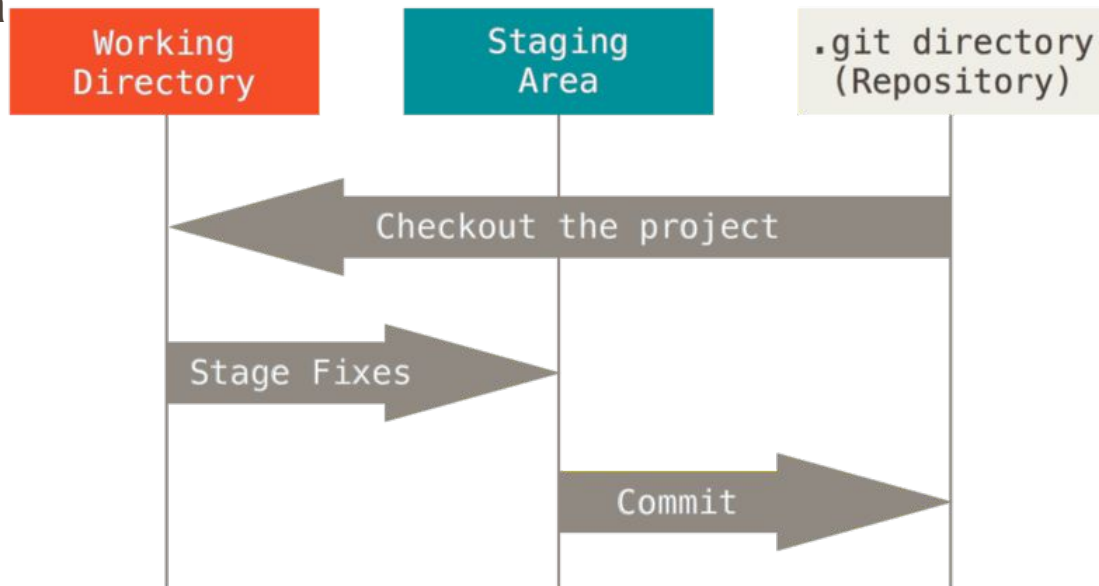


Advantages of Distributed VCS

- All operations (except push & pull) are very fast because the tool only needs to access the hard drive, not a remote server. Hence, you do not always need an internet connection.
- Committing new change-sets can be done locally without manipulating the data on the main repository. Once you have a group of change-sets ready, you can push them all at once.
- Since every contributor has a full copy of the project repository, they can share changes with one another if they want to get some feedback before affecting changes in the main repository.
- If the central server gets crashed at any point of time, the lost data can be easily recovered from any one of the contributor's local repositories.

Three states

- Committed means that the data is safely stored in your local database.
- Modified means that you have changed the file but have not committed it to your database yet.
- Staged means that you have marked a modified file in its current version to go into your next commit snapshot.



Some of the basic operations in Git

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

Setting your commit email address in Git

- GitHub uses the email address set in your local Git configuration to associate commits pushed from the command line with your GitHub account.
- You can use the `git config` command to change the email address you associate with your Git commits. The new email address you set will be visible in any future commits you push to GitHub from the command line. Any commits you made prior to changing your commit email address are still associated with your previous email address.

```
*** Please tell me who you are.
```

```
Run
```

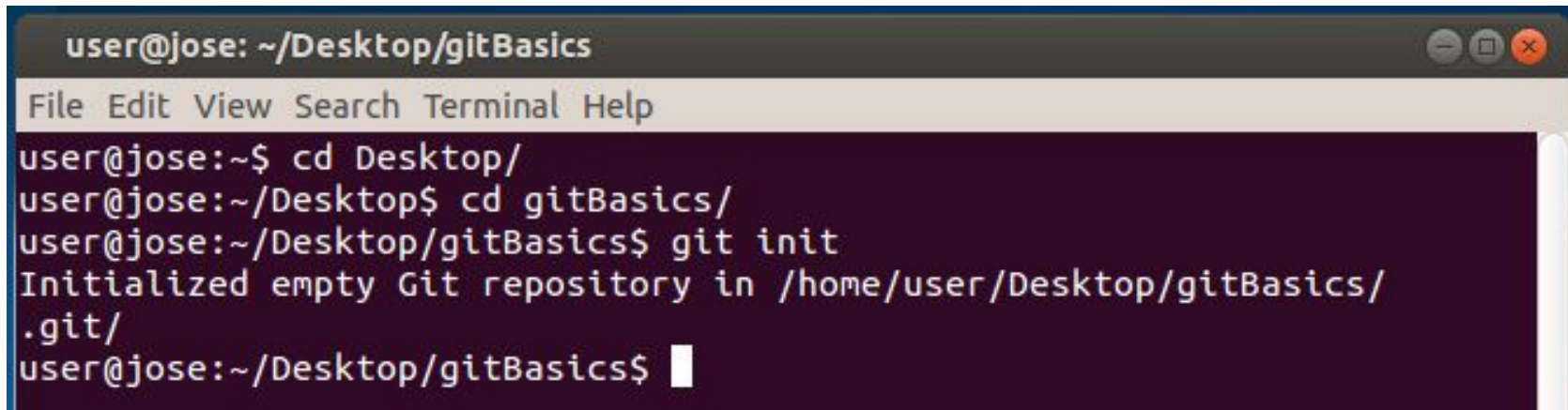
```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

```
to set your account's default identity.
```

```
Omit --global to set the identity only in this repository.
```


Initialize

If you're starting to track an existing project in Git, you need to go to the project's directory and type **git init**

A terminal window titled 'user@jose: ~/Desktop/gitBasics' with standard window controls. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal text shows the user navigating to the 'gitBasics' directory and running 'git init', which successfully creates an empty Git repository.

```
user@jose: ~/Desktop/gitBasics
File Edit View Search Terminal Help
user@jose:~$ cd Desktop/
user@jose:~/Desktop$ cd gitBasics/
user@jose:~/Desktop/gitBasics$ git init
Initialized empty Git repository in /home/user/Desktop/gitBasics/.git/
user@jose:~/Desktop/gitBasics$
```

Git status

Now that the repository is initialized, let me create some files in the directory/repository namely *git_basics_1.txt* and *git_basics_2.txt*

The **git status** command lists all the modified files which are ready to be added to the local repository.

```
user@jose: ~/Desktop/gitBasics
File Edit View Search Terminal Help
user@jose:~$ cd Desktop/
user@jose:~/Desktop$ cd gitBasics/
user@jose:~/Desktop/gitBasics$ git init
Initialized empty Git repository in /home/user/Desktop/gitBasics/.git/
user@jose:~/Desktop/gitBasics$ touch git_basics_1.txt git_basics_2.txt
user@jose:~/Desktop/gitBasics$ git status
On branch master

Initial commit

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

    git_basics_1.txt
    git_basics_2.txt

nothing added to commit but untracked files present (use "git add" to track)
user@jose:~/Desktop/gitBasics$
```

Add

- This command updates the index using the current content found in the working tree and then prepares the content in the staging area for the next commit.
- Thus, after making changes to the working tree, and before running the **commit** command, you must use the **add** command to add any new or modified files to the index.

For that, use the commands: **git add <directory>** or **git add <file>**

Add

- Let us add the files using the command **git add -A**.
- This command will add all the files to the index which are in the directory but not updated in the index yet.

```
user@jose:~/Desktop/gitBasics$ git add -A
user@jose:~/Desktop/gitBasics$ git status
On branch master

Initial commit

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

        new file:   git_basics_1.txt
        new file:   git_basics_2.txt

user@jose:~/Desktop/gitBasics$
```

Commit

It refers to recording snapshots of the repository at a given time. Committed snapshots will never change unless done explicitly. Let me explain how commit works with the diagram below:



Commit

You can commit by using the command **git commit** or **git commit -m "<message>"**

```
user@jose:~/Desktop/gitBasics$ git commit -m "Initial commit"
[master (root-commit) 152ec14] Initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 git_basics_1.txt
 create mode 100644 git_basics_2.txt
user@jose:~/Desktop/gitBasics$ git status
On branch master
nothing to commit, working directory clean
user@jose:~/Desktop/gitBasics$
```

Pull

The **git pull** command fetches changes from a remote repository to a local repository. It merges upstream changes in your local repository, which is a common task in Git based collaborations.

But first, you need to set your central repository as origin using the command:

git remote add origin <link of your central repository>

```
user@jose:~/Desktop/gitBasics$ git status
On branch master
nothing to commit, working directory clean
user@jose:~/Desktop/gitBasics$ git remote add origin https://github.com/geethujp/gitBasics.git
user@jose:~/Desktop/gitBasics$
```

Pull

Now that the origin is set, let us extract files from the origin using pull. For that use the command:

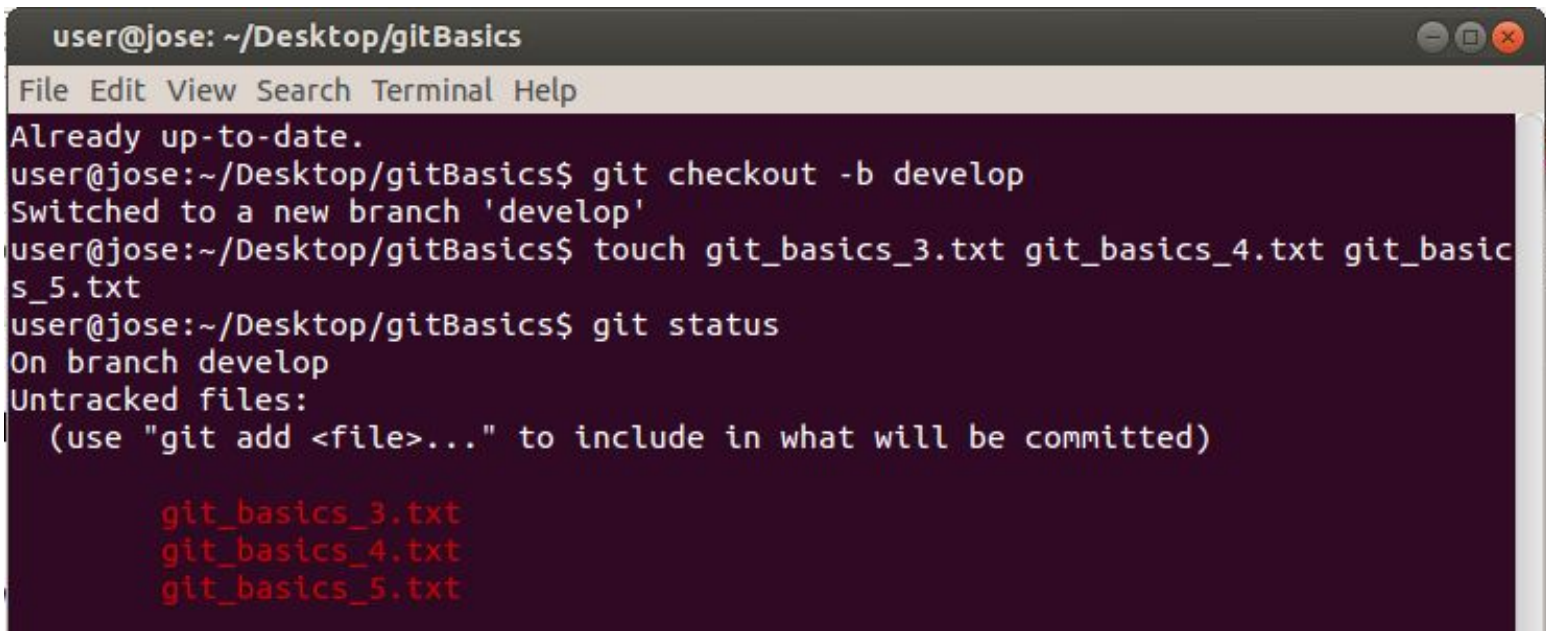
git pull origin master

```
user@jose:~/Desktop/gitBasics$ git pull origin master
From https://github.com/geethujp/gitBasics
 * branch          master      -> FETCH_HEAD
Already up-to-date.
user@jose:~/Desktop/gitBasics$
```

This command will copy all the files from the master branch of remote repository to your local repository.

Creating a New Branch

- Git is supposed to understand what files already exist on the server, unless you somehow made a huge difference to your tree and the new changes need to be sent.
- To create a new branch with a copy of your current state

A terminal window titled 'user@jose: ~/Desktop/gitBasics' with standard window controls. The terminal shows the following sequence of commands and output:

```
File Edit View Search Terminal Help
Already up-to-date.
user@jose:~/Desktop/gitBasics$ git checkout -b develop
Switched to a new branch 'develop'
user@jose:~/Desktop/gitBasics$ touch git_basics_3.txt git_basics_4.txt git_basics_5.txt
user@jose:~/Desktop/gitBasics$ git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    git_basics_3.txt
    git_basics_4.txt
    git_basics_5.txt
```

Push

- This command transfers commits from your local repository to your remote repository. It is the opposite of pull operation.
- Pulling imports commits to local repositories whereas pushing exports commits to the remote repositories .

git push origin

<branch_name>

```
user@jose:~/Desktop/gitBasics$ git add -A
user@jose:~/Desktop/gitBasics$ git commit -m "develop
> "
[develop 28e8dba] develop
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 git_basics_3.txt
create mode 100644 git_basics_4.txt
create mode 100644 git_basics_5.txt
user@jose:~/Desktop/gitBasics$ git pull origin master
From https://github.com/geethujp/gitBasics
* branch          master      -> FETCH_HEAD
Already up-to-date.
user@jose:~/Desktop/gitBasics$ git push origin develop
Username for 'https://github.com': geethujp
Password for 'https://geethujp@github.com':
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 288 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To https://github.com/geethujp/gitBasics.git
* [new branch]      develop -> develop
user@jose:~/Desktop/gitBasics$
```