Laravel Installation:-

### 1: Server Requirements

* PHP >= 7.1.3
* BCMath PHP Extension
* Ctype PHP Extension
* JSON PHP Extension
* Mbstring PHP Extension
* OpenSSL PHP Extension
* PDO PHP Extension
* Tokenizer PHP Extension
* XML PHP Extension

**1:** composer global require laravel/installer (but I have already installed composer)

**2:** By using composer create project command in cmd (composer create-project --prefer-dist laravel/laravel blog) and this one if want to install in specific version (composer create-project --prefer-dist laravel/laravel blog “5.8.\*”) in xampp/htdocs.

**3:**Can start development at built in php server http://localhost:8000 by running this command ‘php artisan serve’ in cmd but I did’nt use it.

**4:** For laravel auth i used php artisan make:auth command and for migration files I used this command php artisan migrate.

Rest API’s:- (application programming interface)

1: An **API** is an application programming interface. It is a set of rules that allow programs to talk to each other. (Way of communication between two or more applications.)

2: The developer creates the API on the server and allows the client to talk to it.

3: **REST (**Representational State Transfer**)** determines how the API will looks like.

4: These are set of rules that developers use to write api’s.

5: When a developer call an URL he gets a piece of data called Resource. URL is called a **request** while the data sent back to you is called a **response.**

Structure of Request:

1: Endpoints 2:Method 3:Headers 4:Data(Body)

1: **Endpoint**s:-

**The endpoint** (or route) is the url we request for. Root-endpoint/?

i-e Root-endpoint of [Github’s API](https://developer.github.com/v3/) is <https://api.github.com> (Any colons (:) on a path denotes a variable.)

Query parameters is the final part of endpoints that can be used to modify your request with key-value pairs and it begin with question mark (?) and Each parameter pair is then separated with an ampersand (&).In request three parameters can be used Name, Type and Description.

**Curl:** Is a tool used to transfer data suing multiple protocols i-e http

### Testing Endpoints With Curl???

**JSON** (JavaScript Object Notation) a common format for sending and requesting data through a REST API.

2: **Method**:- The method is the type of request you send to the server like Get ,Post,Put,Patch,Delete and these mehods are used for crud.

3: **Header:** Headers are used to provide information to both the client and server.Can be used for multiple purposes like Authentication and and for body information. **HTTP Headers are property-value pairs** that are separated by a colon.

4: **Data(Body):** The data (sometimes called “body” or “message”) contains information we want to be sent to the server. This option is only used with POST, PUT, PATCH or DELETE requests. Bin???

### Authentication: This Authentication use to protect your data so that any person can’t get access to your personal data. Can be done with two methods i-e email and password or by using Secret Tokens.

The token method includes [oAuth](https://oauth.net/) and we can access data by Google, facebook and more by using it.

### HTTP Status Codes And Error Messages:

HTTP status codes let you tell the status of the response.

1. **200+** means the request has **succeeded**.
2. **300+** means the request is **redirected** to another URL
3. **400+** means an **error that originates from the client** has occurred
4. **500+** means an **error that originates from the server** has occurred

### API Versions:

Developers update their APIs from time to time

You can request for a specific API version in two ways. Which way you choose depends on how the API is written.

These two ways are:

1. Directly in the endpoint
2. In a request header

GIT:

Git is defined as a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency

#### Version Control System?

You can see Git like a [Dropbox](https://www.dropbox.com/) for your programming projects. You have an online backup, where other team members can change stuff. And every change is recorded, so you can go back on time very easily, accessing any previous version.

**Repository**:

A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs. We recommend including a README, or a file with information about your project. GitHub makes it easy to add one at the same time you create your new repository. It also offers other common options such as a license file.

**Difference between Git and Github:**

**Git** is a revision control system, a tool to manage your source code history.

**Git Commands:**

**$ git init //**initialize local git repository

**$ git add <file>** //add files to index(we can add specific file by giving its name after $ git add myfile or can add all files by doing so $ git add. )

**$ git add //**Used to incrementally "add" changes to the index before using the 'commit' command (Note: even modified files must be "added");

**.$ git add . //**to add all files and for specific files add use this command “$ git add \*.html”it will add all .html files

**$ git status //check** status of working tree

**$ git commit //**commit changes in index

**$ git push //**Push to remote repository

**$ git pull //**pull latest from remote repository

**$ git clone //**clone repository into a new directry

**:wq /**/it will provide all the files that have been changed and their names

**$ clear** //to clear all

* You’ll probably want to create a .**gitignore** file right away, to indicate all of the files you don’t want to track. Use **git add .gitignore** too for adding files to gitignore file..

**$ gitignore** //if we add files or anything in gitignire file it does not commit that file

**$ git branch name //**this command will create a branch by giving name by your own desire (i-e $ git branch mybranch it will create a branch of name mybranch)

**$ git commit** // Record changes to the repository

**$ git rm** // to remove files from the working tree and the index, again before using the 'commit' command;

**$ touch file //**this is used to create file in selected project folder ($ touch index.php it will create this file in that project)

**$ git checkout login //**it used to go into login branch

**$ git checkout master** //it will switch u to master branch

**$ git merge login** //it will merge login branch to master branch

**$ git diff //**it will return uexactly what changes have you made?

**$ pwd //**To check that in which directory u r

**$ git help //**to check all commands

**$ git clone** url (i-e $ git clone <https://github.com/MShahbaz786/task.git>). Downloads a project and its entire version history.

**$ git checkout master** (to chek the status of origin master branch)

**$ git remote –v (**To view your remote repositories type**)**

**$ git pull remote name-of-branch (**This is for you to work on an up-to-date copy (it is important to do this every time you start working on a project), while you set up tracking branches. You pull from remote repositories to get all the changes made by users since the last time you cloned or pulled the project. Later, you can push your local commits to the remote repositories.**)**

**This process below is to login in git using my name and email**

Sets the name you want attached to your commit transactions.

git config --global user.name "YOUR\_USERNAME"

Then verify that you have the correct username:

git config --global user.name

Sets the email you want attached to your commit transactions. To set your email address, type the following command:

git config --global user.email "your\_email\_address@example.com"

To verify that you entered your email correctly, type:

git config --global user.email

Enables helpful colorization of command line output

$ git config --global color.ui auto

You’ll need to do this only once, since you are using the --global option. It tells Git to always use this information for anything you do on that system. If you want to override this with a different username or email address for specific projects, you can run the command without the --global option when you’re in that project.

## Check your information

To view the information that you entered, along with other global options, type:

git config --global --list

1. First I access my directory (my directory c/wamp/www/myproject)
2. Then I opened git bash in project
3. **$ git init** (for initializing git repository in c:/wamp/www/myproject/.git/)
4. **Then I signup on github and create a repository ‘task’**
5. **$ git config --global user.name “MShahbaz786”**
6. **$ git config --global user.email ‘’**[**mshahbaz1013@gmail.com**](mailto:mshahbaz1013@gmail.com)**’’**
7. **$ git config --global --list** (to view the information I entered.)
8. **$ touch .gitignore (**In this we can write files name which we want to ignore but I did’nt made it
9. **.$ git add .** (to add all files to my repository)
10. **$ git commit - m “**any msg here**” (**Record changes to the repository**)**
11. **$ git status** (to check the status for my knowledge)
12. **$ git remote add origin** [git@github.com:alexpchin/<reponame>.git](mailto:git@github.com:alexpchin/%3creponame%3e.git)
13. **$ git push -u origin master** and finally its done

**To edit or upload new file in remote repository using local project:**

**1**: Open git bash in local project and goto project directory (my directory was c/wamp/www/myproject)

2: Created file in project and make changes in it.

3: Then add it repository using command **$ git add .**

4**: $ git commit -m "new file added"** (commit to add/check changes in file)

6: **$ git push origin master** and its done(to push file in remote repository master branch)

7**: $ git status** to check its status

**To clone a repository:**

**1: $ git clone path** (i-e $ git clone <https://github.com/MShahbaz786/task.git>)

**2**: It will start cloning it to folder task given at the end of link and it done.

**Creating a pull request:**

Create a pull request to propose and collaborate on changes to a repository. These changes are proposed in a *branch*, which ensures that the master branch only contains finished and approved work.

Anyone with read permissions to a repository can create a pull request, but you must have write permissions to create a branch. If you want to create a new branch for your pull request and don't have write permissions to the repository, you can fork the repository first. For more information, see "[Creating a pull request from a fork](https://help.github.com/en/articles/creating-a-pull-request-from-a-fork)" and "[About forks](https://help.github.com/en/articles/about-forks)."

Pull requests can only be opened between two branches that are different.

**Note**: To open a pull request in a public repository, you must have write access to the head or the source branch or, for organization-owned repositories, you must be a member of the organization that owns the repository to open a pull request.