**GITHUB**

* **WHAT IS GITHUB?**

GitHub is a hosting service for Git repositories. GitHub is the service for projects that use Git.

Your remote repository, is basically your source code being stored on a third party service making it accessible to anyone across the world, much like GitHub does.

So once you push your code from your local system to GitHub, it is now also stored on their servers. Now anyone can clone or fork your repository and make changes to it.

The various services apart from just being a remote repository offered by GitHub can be read up at: [GitHub Services](https://en.wikipedia.org/wiki/GitHub#Services).

GitHub was written using [Ruby on Rails](http://rubyonrails.org/) .GitHub's backend services for using Git as a datastore are written in Erlang ([Introducing BERT and BERT-RPC](https://github.com/blog/531-introducing-bert-and-bert-rpc) )

They also have some other infrastructure services written in Ruby.

Git is an Open Source DISTRIBUTED VERSION CONTROL SYSTEM. It is mainly used for source code management in software development. It is a Command Line Interface(CLI) tool and can be mastered easily.

GitHub is web-based hosting service for Version Control using Git. It offers all functionalities of git as well adding many of its own. You get a much clean web UI providing great visuals and don’t need to master any CLI.

You can comment on changes/revisions made by others, contribute to open source projects, fork projects, download projects, upload your projects, make gists, and much much more.

* **HOW GITHUB WORKS?**

In other version control systems such as CVS, Subversion, and Clearcase, the server is centralized — there’s a clear separation between the server and clients.

When developers work on projects that use these systems, they first send a “checkout” request to the server, then retrieve a “snapshot” of the current version — usually the most recent one. Everyone has to go through the central server in order to work on the same project, sending “commits” or creating branches.

With Git, things are different. When you ask for a project, you clone it locally on to your machine.

In other words, Git copies all the project files to your hard drive, then allows you to work on the project autonomously. All operations run locally on your machine. You don’t even need a network connection, except to synchronize with the source code by “pushing” or “pulling.”

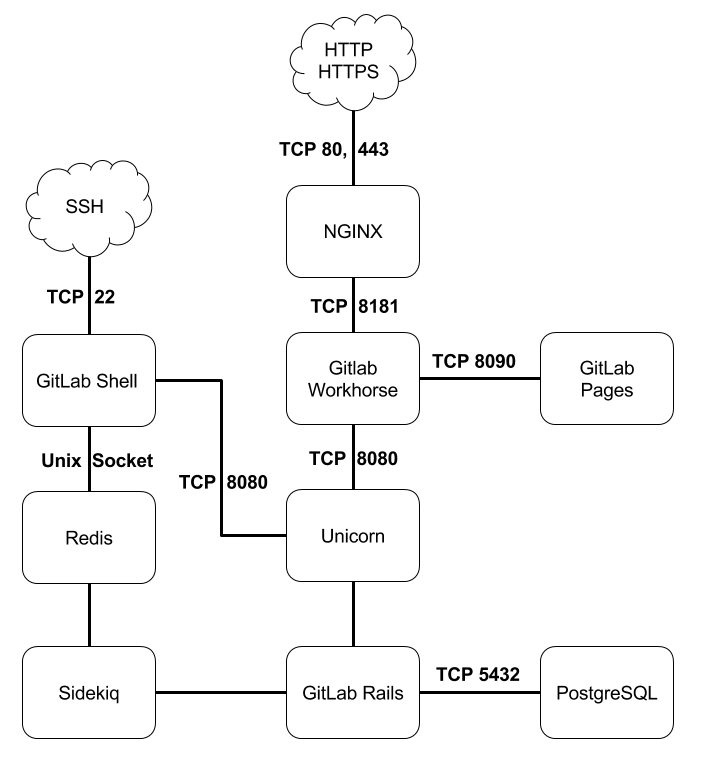
Working steps:

1. **Create a new repository -**You start by creating a repository which will be the place where you store all your files.
2. **Workflow -** Your local repository consists of three "trees" maintained by git.
   1. The **first**one is your Working Directory which holds the actual files.
   2. The **second**one is the Index which acts as a staging area.
   3. The **third**HEAD which points to the last commit you've made.
3. **Add & commit -**You can propose changes (add it to the Index) using git command **add**. This is the first step in the basic git workflow. To actually commit these changes use another command **commit.**
4. Now the file is committed to the HEAD, but not in your **remote**repository yet.

* **COMMANDS IN GITHUB**

| **Command** | **Description** |
| --- | --- |
| git init | Initialize a local Git repository |
| git clone ssh://git@github.com/[username]/[repository-name].git | Create a local copy of a remote repository |
| Command | Description |
| git status | Check status |
| git add [file-name.txt] | Add a file to the staging area |
| git add –A | Add all new and changed files to the staging area |
| git commit -m "[commit message]" | Commit changes |
| git rm -r [file-name.txt] | Remove a file (or folder) |
| git branch | List branches (the asterisk denotes the current branch) |
| git branch –a | List all branches (local and remote) |
| git branch [branch name] | Create a new branch |
| git branch -d [branch name] | Delete a branch |
| git push origin --delete [branch name] | Delete a remote branch |
| git checkout -b [branch name] | Create a new branch and switch to it |
| git checkout -b [branch name] origin/[branch name] | Clone a remote branch and switch to it |
| git checkout [branch name] | Switch to a branch |
| git checkout - | Switch to the branch last checked out |
| git checkout -- [file-name.txt] | Discard changes to a file |
| git merge [branch name] | Merge a branch into the active branch |
| git merge [source branch] [target branch] | Merge a branch into a target branch |
| git stash | Stash changes in a dirty working directory |
| git stash clear | Remove all stashed entries |
| git push origin [branch name] | Push a branch to your remote repository |
| git push -u origin [branch name] | Push changes to remote repository (and remember the branch) |
| git push | Push changes to remote repository (remembered branch) |
| git push origin --delete [branch name] | Delete a remote branch |
| git pull | Update local repository to the newest commit |
| git pull origin [branch name] | Pull changes from remote repository |
| git remote add origin ssh://git@github.com/[username]/[repository-name].git | Add a remote repository |
| git remote set-url origin ssh://git@github.com/[username]/[repository-name].git | Set a repository's origin branch to SSH |
| git log | View changes |
| git log –summary | View changes (detailed) |
| git diff [source branch] [target branch] | Preview changes before merging |

* **ARCHITECTURE OF GITHUB**

****

The core of Github is the open-source software git, which is a distributed version control and source code management software. Github adds many features onto git's functionality, including issue tracking, wiki, access control, etc.

However, Github is proprietary software, and many of it's internals remain unknown to general public.

In order to find out how pieces are fit together, I suggest following the leads below:

* [Github's official Engineering Blog](https://github.com/blog/category/engineering), which covers a wide range of topics of the design of the platform.
  + especially [this article](https://github.com/blog/530-how-we-made-github-fast), an extensive overview written by Github CTO.
* [Projects that power Github](https://github.com/showcases/projects-that-power-github), an officially curated list of open-source projects that power Github. Here are a few highlights:
  + rails - web backend
  + jekyll - blog engine behind *Github Pages*
  + elasticsearch - search engine
  + hubot - a chat bot for task automation
  + ace - browswer based code editor
  + d3 - js lib for graphs and charts
  + resque - background works