Git & GitHub:

Git:

Git is a Distributed Version Control tool that is used to store different versions of a file in a remote or local repository. It is used to track changes in the source code. Git favors both programmers and non-technical users by keeping track of their project files. It allows multiple developers to work together.

GitHub:

GitHub is a project management and a code version control system as well as a social network platform made for developers. But what is GitHub used for? Well, amongst all, it allows you to work collaboratively with other people around the world, plan your projects and track your work.

Repository:

A GitHub **repository** can be used to store a development **project**.

It can contain **folders** and any type of **files** (HTML, CSS, JavaScript, Documents, Data, Images). A GitHub repository should also include a **licence** file and a **README** file about the project. A GitHub repository can also be used to store ideas, or any resources that you want to share.

Branch:

A GitHub branch is used to work with different **versions** of a repository at the same time. By default, a repository has a **master** branch (a production branch). Any other branch is a **copy** of the master branch (as it was at a point in time).

New Branches are for bug fixes and feature work separate from the master branch. When changes are ready, they can be merged into the master branch. If you make changes to the master branch while working on a new branch, these updates can be pulled in.

Commits:

At GitHub, changes are called commits. Each commit (change) has a description explaining why a change was made.

Pull Requests:

Pull Requests are the heart of GitHub **collaboration**.

With a pull request you are **proposing** that your changes should be **merged** (pulled in) with the master. Pull requests show content **differences**, changes, additions, and subtractions in **colors** (green and red). As soon as you have a commit, you can open a pull request and start a discussion, even before the code is finished.

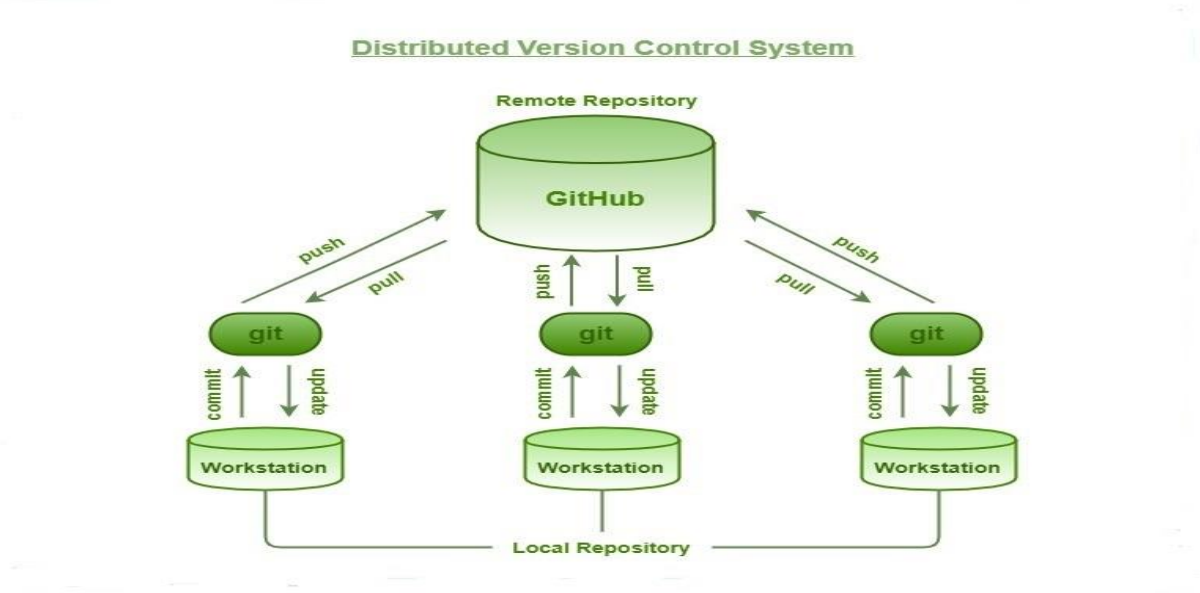
Push:

Git push is mostly used to publish uploaded local changes to a central repository. After a local repo has been modified a push is executed to share the modifications.

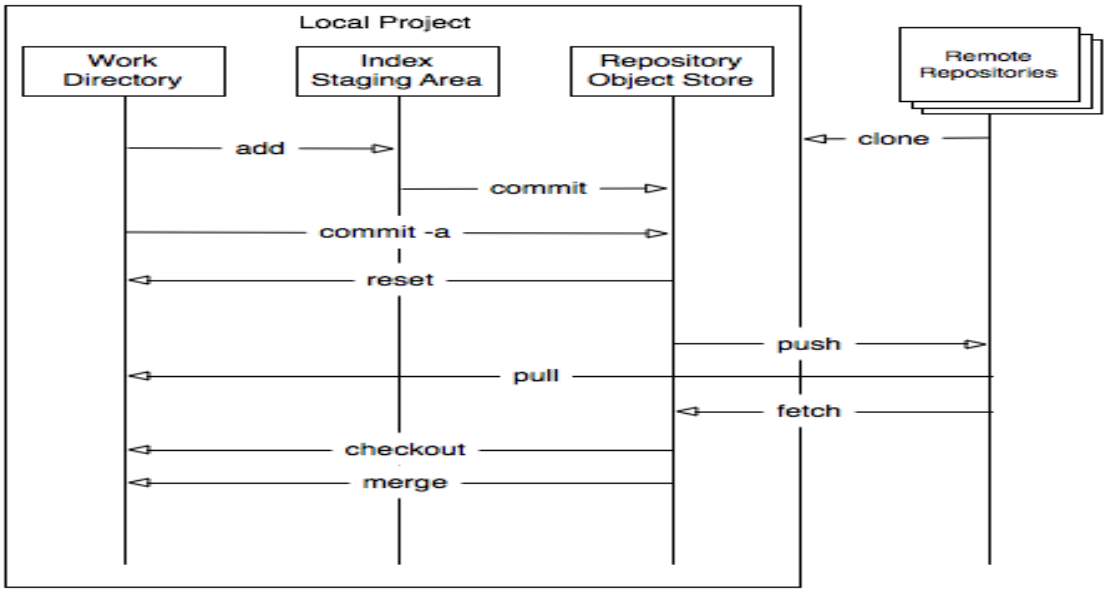
Forking a Repository:

Forking a repository means you make a new project based on the existing repository. In plain English, forking a repo means you copy an existing repository, make some changes needed, store the new version as a new repository, and call this your own project.

This is a great feature that boosts project development. Because it is a totally new project, the central repository won't be affected. If the 'master' repository is updated, you can also apply the update to your current fork.



GIT & GITHUB Flow:



Command:

Command	Origin	Destination	Description
`git clone REPO_URL`	Personal Github	Local	Creates a local copy of a Github repo. The URL can be copied from Github.com by clicking the `Clone or Download` button.
`git add README.md`	Working Dir	Staging Area	Add "README.md" to staging area.
`git commit`	Staging Area	Local	Commits changes to files to the local repo.
`git commit -a`	Working Dir	Local	adds and commits all file changes to the local repo.
`git pull`	Personal Github	Local	Retrieve any changes from a Github repo.
`git push`	Local	Personal Github	Sends commited file changes to Github repo.
`git merge`	Other branch	Current branch	Merge any changes in the named branch with the current branch.
`git checkout -b patch1`	NA	NA	Create a branch called "patch1" from the current branch and switch to it.
`git init`	NA	NA	Initialise a directory as a Git repo.
`git log`	NA	NA	Display the commit history for the current repo
`git status`	NA	NA	See which files are staged/unstaged/changed
`git diff`	NA	NA	See the difference between staged uncomitted changes and the most recent commit
`git stash`	NA	NA	Save uncommitted changes in a temporary version and revert to the most recent commit

Installation and configuration Git:

```
# yum install git
# git config --global user.name "nabi"
# git config --global user.email "paged.us@gmail.com"
# git --version
# git config --list
```

First Code Push to GitHub:

```
# vim page.txt
# git init
# git status
# git add page.txt
# git status
# git commit -m "First Commit"
# git status
# git remote add origin https://github.com/cnabi/pagecloud.git
# git push -u origin master
```

Code Modify and Push to GitHub:

```
# cat page.txt
# git status
# git add page.txt
# git status
# git commit -m "Modified"
# git push
```

Create Git Branch:

```
# git branch
# git branch devops
# git branch
```

Check Out to New Branch and commit new Code:

```
# git checkout devops
# git branch
# vim devops.txt
# git branch
# git add devops.txt
# git commit -m "DevOps File"
# git status
# git push -u origin devops
# git branch
# git push -u origin devops
```

Marge a Code with Master:

```
# git checkout master
# git branch
# git merge devops
```

Create a new branch and Best Practice to Merge:

```
# git branch production
# git checkout production
# git branch
# vim production.txt
# git add production.txt
# git commit -m "Production"
# git push -u origin production
# git checkout master
# git branch
```

***Go to GUI and merge Production with Master ***

*** Go to another directory and clone the whole repository****