Version Control With Git

Novor Rene

Advanced Information Technology Institute, Kofi Annan Centre of Excellence in ICT

renen@aiti-kace.com.gh

May 15, 2020

Overview

Version Control

A version control system helps to keep track of changes made across files in a project over the development period.

Version Control System

A Version control system is a system that records changes to a file or set of files over time so that specific versions can be recalled later.

Why Use Version Control Systems?

- Collaboration with others
- Storing versions [Snapshots of every change]
- Restoring previous version
- Informal documentation through commit messages [if messages are meaningful]
- Backup of sorts [distributed systems]

Types of Version Control Systems

- Local VCS [RCS]
- Centralized VCS [CVS,Perforce, Subversion]
- Distributed VCS [Mercurial, Git, Bazaar, Darcs]

Why Git?

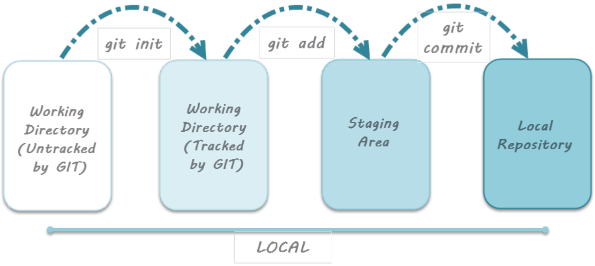
For our purposes, we will study Git. Why?

- Fast
- Supports heavy parallel development
- Easily handles large projects
- Has integrity (check-summed files using SHA-1 produces 40 character string that changes upon file change)
- Popular

The 3 areas/sections in Git?

Conceptually, there are 3 possible places a file could be in git:

- Working directory [file being worked on]
- Staging area [files going into the next commit go here]
- .git Repository [files stored]


git init
git add
git commit
Working Directory (Untracked by GIT)
Working Directory (Tracked by GIT)
Staging Area
Local Repository
LOCAL

The 3 states in Git?

A file in git can be in one of 3 possible states:

- Committed [data safely stored in database]
- Modified [change made without commit]
- Staged [files marked to be tagged in next commit]

A file could be *Untracked*, in which case git does not care about changes in it.

Installing Git

Git is available on many OSes:

- Linux
- Windows
- Mac

Initial Setup

```
$ git config
```

User config can be:

- System wide
- user confined
- repository confined

Committer's ID

- user.name
- user.email

```
$ git config --list
```

Getting help in git

```
$  git help
$  man git
```

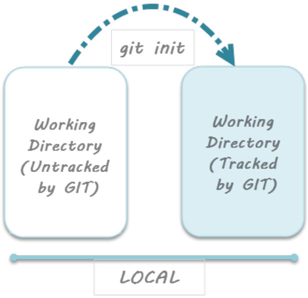
Getting a Repository

Turn a local directory into a git repo:

```
$ git init
```

Clone an existing repo:

```
$ git clone url
```


git init
Working Directory (Untracked by GIT)
Working Directory (Tracked by GIT)
LOCAL

Recording changes in a Repository

In a directory, files are either *tracked* or *untracked*
Tracked files are in one of 3 states [unmodified,modified or staged]
To check status of the repo, run in the directory of the repo:

```
$  git status
```

Start tracking untracked files:

```
$  git add nameOfFile
$  git add .
```

Stage files that have been modified

```
$  git add nameOfFile
$  git add .
```

Commit (save) your changes:

```
$  git commit
$  git commit -m "commit message" .
```

Git Workflow

Git does not try to impose its methods on the user. Hence, your normal workflow is not changed so much. When using git for Version Control, a workflow as found below is typical

```
$ git add fileName
$ git commit
$ git commit -m"Commit message"
$ git push
$ git pull
```

The last two commands only become relevant when you are collaborating with a team on a remote repository.

Parallel Development

Git allows for massive parallel development. It allows the concept of *branches*. This

```
$ git branch newBranchName
$ git checkout newBranchName
```

Inside each branch, you can do all of what was done earlier. If you want to combine two branches, git allows that: Switch to the branch you want to combine with

```
$ git checkout branchToCombineWith
$ git merge branch
```

Github

Git allows for a central server where repositories can be kept and made available to all team members with the right privileges.
Github is an online repository hosting service. It allows teams to host repos on the platform.
It also has features to help augment the features git provides.[Examples: Pull Request, Actions for workflow automation and many others].
Repositories created on Github can be public or private. Public repositories can be accessed without any credentials.
However, private repos can only be accessed with the right permissions.
The usual flow will be such that a team member hosts the code repo on Github or any other such services and gives access to the rest of the team.
The members then contribute their code to the central repo.

Collaboration

To get an already existing repo from a remote location:

```
$ git clone url
```

To get new changes from a remote repo:

```
$ git pull
```

To get make local changes available on the remote repo:

```
$ git push
```

To list the remote urls of the repo:

```
$ git remote -v
```

It should be noted that you always have to commit just like with the local only repositories.

Collaboration

To see what is going with a remote repo:

```
$ git remote show alias
```

To get a remote branch that you did not have initially:

```
$ git checkout  -b newLocal url/newRemote
$ git checkout --track url/newRemote
$ git checkout newRemote
```

The third option assumes (a) the branch doesn't already exist and (b) that the branch name matches a name on only one remote.
Previously, we used *git pull* to get remote code. It is relevant to state that *git pull* is a combination of two commands:

```
$ git fetch; git merge
```

Pull Requests

Pull requests is a way of telling others about changes you have pushed to a branch in a repository on Github.

It provides an avenue for discussion and review of the potential improvement you are contributing.

Team members might also suggest/improve what you have before it is finally merged with the base branch(branch being compared with and merged into).
Anyone who has read permissions to a repo can issue pull requests.

Issuing Pull Requests

- Navigate to main repo page.
- Select branch that has your commits.
- On the extreme right side, click on **New pull request** button.
- Select **base branch**.
- Select your branch for comparison.
- Give a title and description to your pull request.
- Click on **Create Pull Request**.

References

- Git Book (Online)
  `http://git-scm.com/book`
- Git Tutorial(Online)
  `http://schacon.github.com/git/gittutorial.html`
- GitHub Help(Online)
  `https://help.github.com/en/github/collaborating-with-issues-and-pull-requests`

This is introductory content...