SYNERGETICS GET IT RIGHT | Microsoft Partner
Gold Cloud Platform
Silver Learning

Version Control using Git

Git is an open source, distributed version control system designed for speed and efficiency

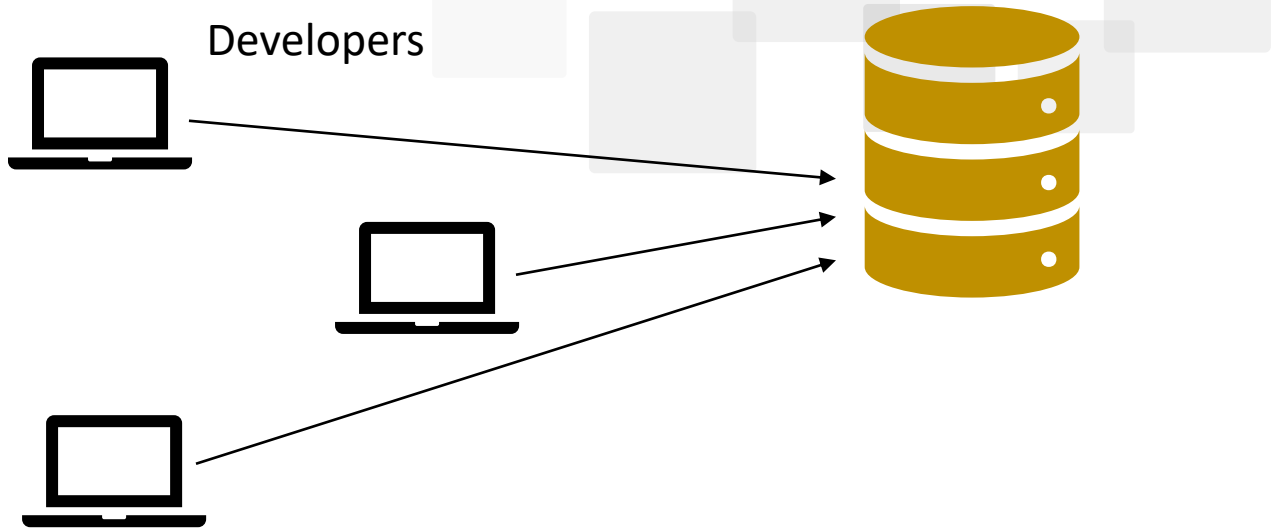
What is Git?

Git is a distributed version control and source code management(SCM) system
- Speed, Data integrity and support for distributed, non-linear workflows.

Initially designed and developed by Linus Torvalds for Linux kernel development in 2005

Full-fledged repository with complete history
Full version-tracking capabilities, independent of network access or a central server.

Git is free software distributed under the terms of the GNU General Public License version 2.



Version control systems

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

		Strengths	Best for
Centralized Version Control	Check-in Check-out	• Fine level permission control • Allows usage monitoring	• Large integrated codebases • Control and auditability over source code down to the file level
	Edit Commit	• Offline editing support • Easy to edit files outside Visual Studio or Eclipse	• Medium-sized integrated codebases • A balance of fine-grained control with reduced friction
Distributed Version Control (DVCS)		• Fast offline experience • Complete repository with portable history • Flexible advanced branching model	• Modular codebases • Integrating with open source • Highly distributed teams

Design Base

- Git's design was inspired by BitKeeper and Monotone.
- Originally designed as a low-level version control system engine on top of which others could write front ends, such as Cogito.
- The core Git project has since become a complete version control system that is usable directly.
- While strongly influenced by BitKeeper, Torvalds deliberately attempted to avoid conventional approaches, leading to a unique design.

Git Characteristics

- Strong support for non-linear development
  - Rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history.
  - Branches in git are very lightweight: A branch in git is only a reference to a single commit. With its parental commits, the full branch structure can be constructed.
- Distributed development
  - Like Darcs, BitKeeper, Mercurial, SVK, Bazaar and Monotone, Git gives each developer a local copy of the entire development history, and changes are copied from one such repository to another. These changes are imported as additional development branches, and can be merged in the same way as a locally developed branch.
- Compatibility with existing systems/protocols
  - Repositories can be published via HTTP, FTP, rsync, or a Git protocol over either a plain socket, or ssh. Git also has a CVS server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories. Subversion and svk repositories can be used directly with git-svn.

Git Characteristics

- Efficient handling of large projects
- Cryptographic authentication of history
- Toolkit-based design
- Pluggable merge strategies
- Garbage accumulates unless collected
- Periodic explicit object packing

Git Server

- As git is a distributed version control system, it can be used as server out of the box. Dedicated git server software helps, amongst other features, to add access control, display the contents of a git repository via web, and help managing multiple repositories.
- Remote file store and shell access
- Gitdaemon, instaweb
- Gitolite
- Gerrit
- Gitblit
- Gitiles
- Bonobo GitServer
- Commercial solutions

Git : Benefits

- Distributed : local & remote repositories
- No Network Needed for
  - Performing Diff
  - Viewing file history
  - Committing changes
  - Merging branches
  - Switching branches
- Immutable
- Snapshots over patches

Git Basics

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

Commit

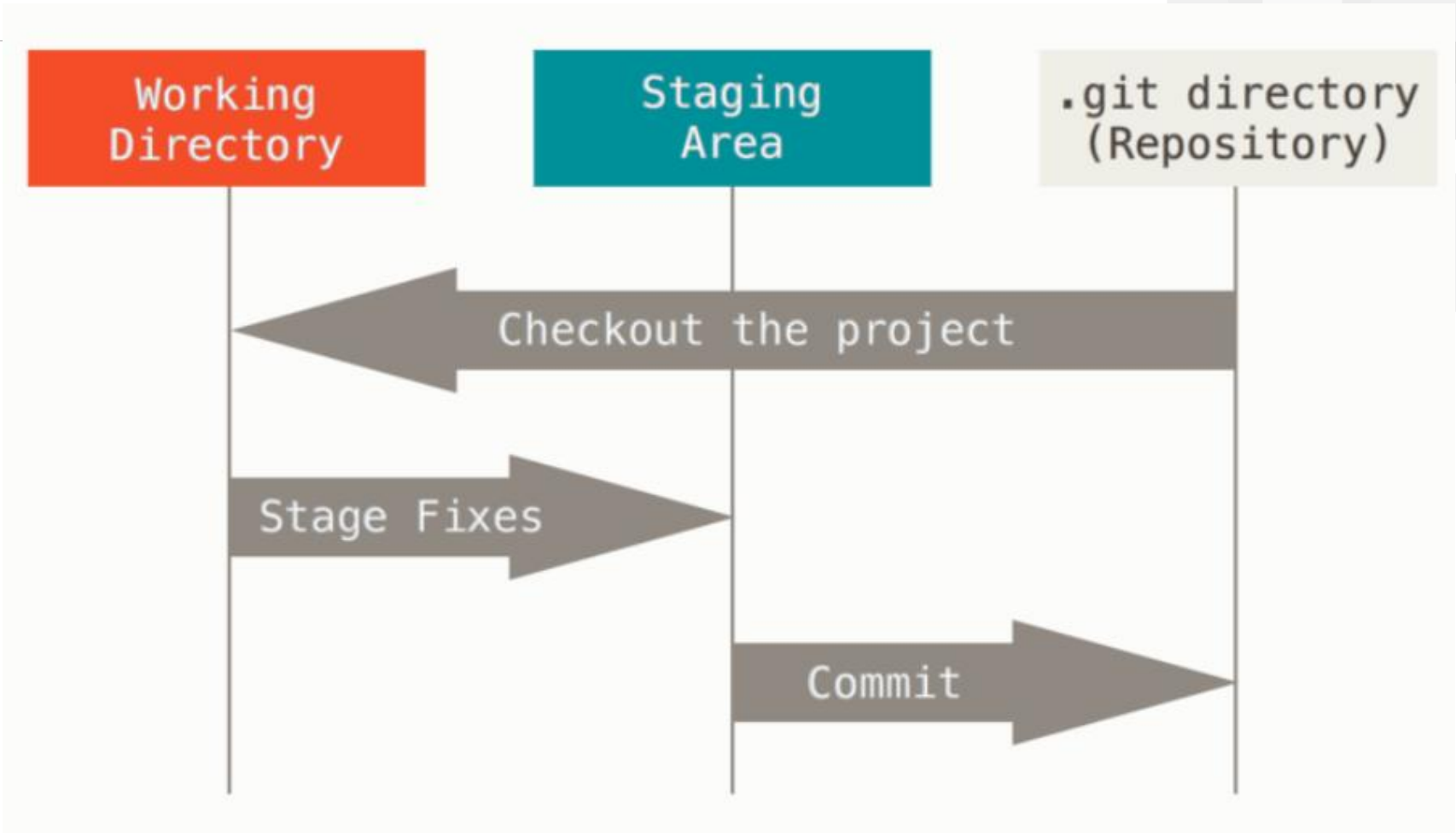
Branch

Tags

Local Repository

Remote Repository

Git : Workspace



Git : Installation

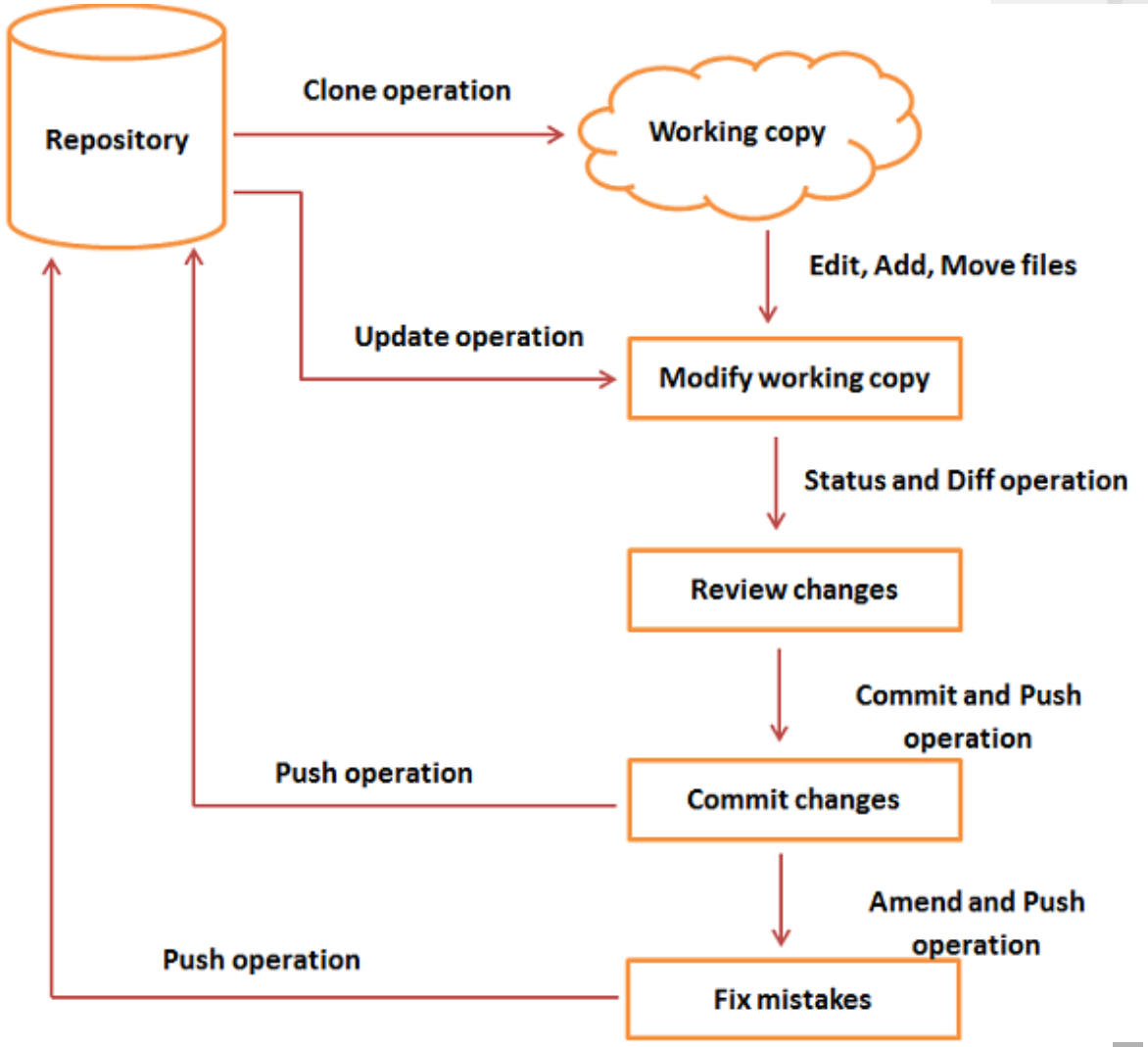
- Available on Linux
- Available on MacOS
- Available on Windows
  - Git for Windows (Installer)
  - Portable Git
  - Both Versions available at
    https://git-scm.com/download/win

Git : One time setup

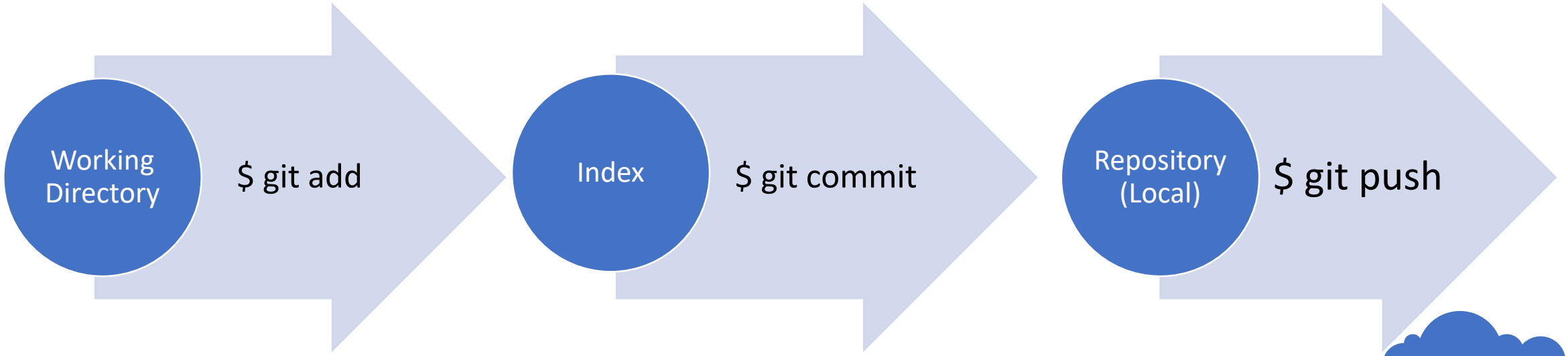
- Being a distributed version control system, need to identify current user through config:

```
$ git config --global user.name "omprakash"
$ git config --global user.email "omspandey@hotmail.com"
```

Git Work flow



Git : Typical Workflow



Git Working with new repository

- Create a new empty repository

```
$ git init
```

- Add files

Create files using favourite editor and save in current directory

- Stage / Index files

```
$ git add .
```

- Commit changes

```
$ git commit –m “Initial change!”
```

Git Hub Registration

- A hosted service for remote git repositories.
- Unlimited PUBLIC repositories for Open source projects.
- Widely popular with Open Source community.
  - Even Linus Torvalds has complete Linux Kernel source on GitHub!
- Visit https://github.com

Git Cloning

- Clone remote repository as local one

```
$ git clone <remote-repo-url>
```

- Save changes to local

```
$ git commit –m “Updated stuff!”
```

- Upload local changes to remote (Need user credentials)

```
$ git push
```

Git Branches

- A "branch" is an active line of development.
- The most recent commit on a branch is referred to as the tip of that branch.
- The tip of the branch is referenced by a branch head, which moves forward as additional development is done on the branch.
- A single git repository can track an arbitrary number of branches, but your working tree is associated with just one of them (the "current" or "checked out" branch), and HEAD points to that branch

Git Branches

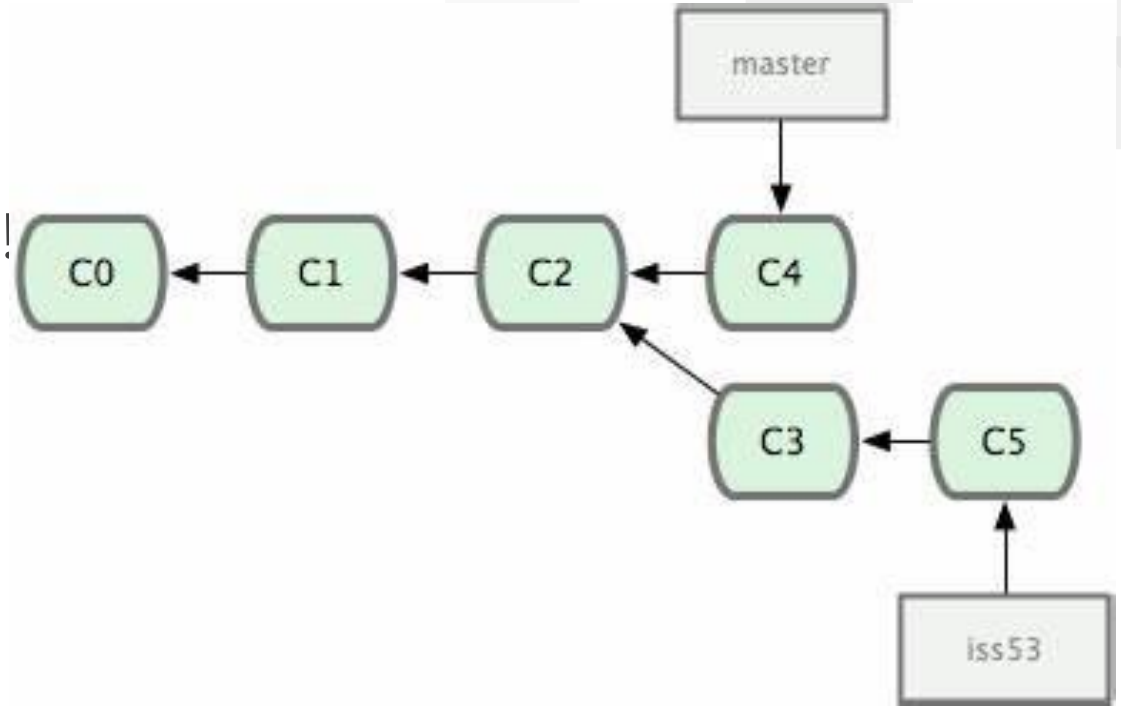
- Getting different versions of project:
- Git is best thought of as a tool for storing the history of a collection of files. It stores the history as a compressed collection of interrelated snapshots of the project's contents. In git each such version is called a commit.
- Those snapshots aren't necessarily all arranged in a single line from oldest to newest; instead, work may simultaneously proceed along parallel lines of development, called branches, which may merge and diverge.
- A single git repository can track development on multiple branches. It does this by keeping a list of heads which reference the latest commit on each branch; the git-branch(1) command shows you the list of branch heads:
- $ git branch * master
- A freshly cloned repository contains a single branch head, by default named "master", with the working directory initialized to the state of the project referred to by that branch head.
- Most projects also use tags. Tags, like heads, are references into the project's history, and can be listed using thegit-tag(1) command:
- $ git tag -l

Git Branching

- Isolate work units
- Long running topics

- Branch is JUST a POINTER to Commit!!
- Default branch name : “master”

- Commands:
  - $ git branch
  - $ git checkout
  - $ git merge



Understanding History - Commits

- Every change in the history of a project is represented by a commit. The git-show(1) command shows the most recent commit on the current branch:
- $ git show
- Every commit (except the very first commit in a project) also has a parent commit which shows what happened before this commit. Following the chain of parents will eventually take you back to the beginning of the project.
- However, the commits do not form a simple list; git allows lines of development to diverge and then reconverge, and the point where two lines of development reconverge is called a "merge".
- The commit representing a merge can therefore have more than one parent, with each parent representing the most recent commit on one of the lines of development leading to that point.
- The best way to see how this works is using the gitk(1)command; running gitk now on a git repository and looking for merge commits will help understand how the git organizes history.
- In the following, we say that commit X is "reachable" from commit Y if commit X is an ancestor of commit Y. Equivalently, you could say that Y is a descendant of X, or that there is a chain of parents leading from commit Y to commit X.

Git Commands

- Git Commands

- git --help
- git --version

- mkdir ompcgrepos
- cd ompcgrepos
- git init
- git add file1.html
- git status
- git commit file1.html -m "added file1"
- git log

- git config --global user.name "omspandey"
- git config --global user.email "omspandey@hotmail.com"
- git config --global color.ui true
- git config --global color.status auto
- git config --global color.branch auto
- git config --global core.editor vim
- git config --global merge.tool vimdiff
- git config -l

- git add file2.html
- git status

- git remote add origin https://github.com/oppandey/cgrepo.git
- git clone https://github.com/oppandey/cgrepo.git
- cd cgrepo
- git add file1.html
- git commit file1.html -m "file 1 added"
- git push
- git status
- git show 01b89
- git diff 01b89 0abcd

- git branch airoli_branch
- git branch
- git checkout airoli_branch

Git : Other common commands

- Git status
- Git diff
- Git log
- Git fetch
- Git pull