GIT defn:

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

Git purpose:

The **purpose of Git** is to manage a project, or a set of files, as they change over time. **Git** stores this information in a data structure called a repository. A **git** repository contains, among other things, the following: A set of commit objects

**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. It allows multiple developers to **work** together.

Git has three main states that your files can reside in: *modified*,

*staged*, and *committed*:

• Modified means that you have changed the file but have not committed it to your database yet.

• Staged means that you have marked a modified file in its current version to go into your next commit snapshot.

• Committed means that the data is safely stored in your local database.

**three main sections of a Git project**



* **The working tree** is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.
* **The staging area** is a file, generally contained in your Git directory, that stores information about what will go into your next commit. Its technical name in Git parlance is the “index”, but the phrase “staging area” works just as well.
* **The Git directory** is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you *clone* a repository from another computer.

**Git is a version control system**

VCS:**Version control systems** are a category of software tools that help a software team manage changes to source code over time.

Software developers working in teams are continually writing new source code and changing existing source code.

Git installation

Git setup

git config

**Your Identity**

The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it’s immutably baked into the commits you start creating:

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

**Getting a Git Repository**

**Initializing a Repository in an Existing Directory**

If you have a project directory that is currently not under version control and you want to start controlling it with Git, you first need to go to that project’s directory. If you’ve never done this, it looks a little different depending on which system you’re running:

for Linux:

$ cd /home/user/my\_project

for macOS:

$ cd /Users/user/my\_project

for Windows:

$ cd /c/user/my\_project

**Cloning an Existing Repository**

If you want to get a copy of an existing Git repository — for example, a project you’d like to

contribute to — the command you need is git clone. If you’re familiar with other VCS systems such as Subversion, you’ll notice that the command is "clone" and not "checkout".

$ git clone <https://github.com/libgit2/libgit2>

**Recording Changes to the Repository**

Tracked files are files that were in the last snapshot; they can be unmodified, modified,

or staged. In short, tracked files are files that Git knows about.

Untracked files are everything else — any files in your working directory that were not in your last snapshot and are not in your staging area. When you first clone a repository, all of your files will be tracked and unmodified because Git just checked them out and you haven’t edited anything.

As you edit files, Git sees them as modified, because you’ve changed them since your last commit.

As you work, you selectively stage these modified files and then commit all those staged changes,and the cycle repeats.

**Checking the Status of Your Files**

The main tool you use to determine which files are in which state is the git status command. If you

run this command directly after a clone, you should see something like this:

$ git status

On branch master

Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean

**Tracking New Files**

In order to begin tracking a new file, you use the command git add. To begin tracking the README

file, you can run this:

$ git add README

git add is a multipurpose command — you use it to begin tracking new files,

**Short Status**

While the git status output is pretty comprehensive, it’s also quite wordy. Git also has a short

status flag so you can see your changes in a more compact way. If you run git status -s or git

status --short you get a far more simplified output from the command:

$ git status -s

M README

MM Rakefile

A lib/git.rb

M lib/simplegit.rb

?? LICENSE.txt