Маламуж Володимир(Editor of this part) Геннадiй Румянцев

The most common use case for Git is developers working on source code files, but it could be used to manage updates to files of any type. Git is also the version control standard for GitHub and other source code management systems. One of the biggest advantages of Git is its branching capabilities.  
 Top 20 Git Commands that you will be using frequently while you are working with Git:

* git config - sets the author name and email address respectively to be used with your commits.
* git init - is used to start a new repository.
* git clone - is used to obtain a repository from an existing URL.
* git add - adds a file to the staging area.
* git commit - records or snapshots the file permanently in the version history.
* git diff - shows the file differences which are not yet staged.
* git reset - unstages the file, but it preserves the file contents.
* git status - lists all the files that have to be committed.
* git rm - deletes the file from your working directory and stages the deletion.
* git log - is used to list the version history for the current branch.
* git show - shows the metadata and content changes of the specified commit.
* git tag - is used to give tags to the specified commit.
* git branch - lists all the local branches in the current repository.
* git checkout - is used to switch from one branch to another.
* git merge - merges the specified branch’s history into the current branch.
* git remote - is used to connect your local repository to the remote server.
* git push - sends the committed changes of master branch to your remote repository.
* git pull - fetches and merges changes on the remote server to your working directory.
* git stash - temporarily stores all the modified tracked files.

Hennadiy Rumiantsev(Editor of this part)

Malamuzh Volodymyr

Git **commit** creates a commit, which help you to save your including files in real time. These commits are like snapshots of your entire repository at specific times. You have to make new commits often, based around logical units of change. By the way, commits can tell a story of the history of your repository and how it came to be the way that it currently is, that`s really useful if you working in groups, you can do your part of work, and your colleague do that, with time you mix your project and upload to a directory and if something went work, you have a copy of your project and files which was mixed. Commits also include lots of metadata in addition to the contents and message, like the author, timestamp, and more

Shortly: That`s your save point in working with files which independent from each other

