What's Git?

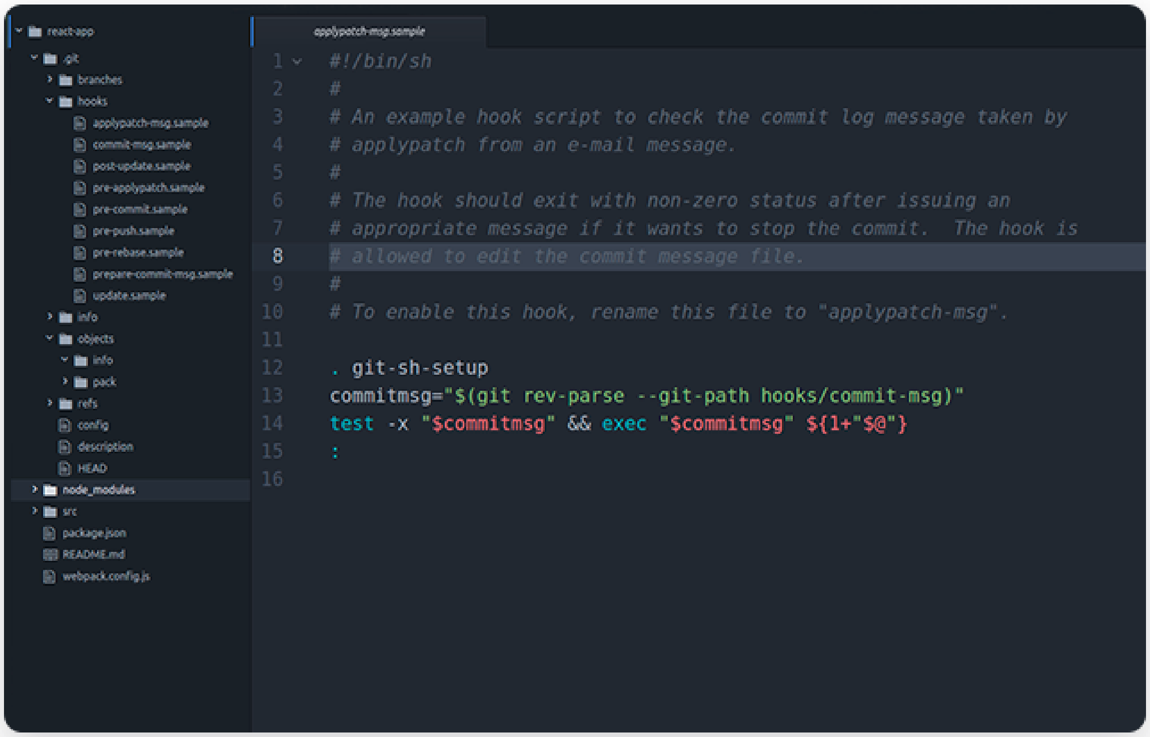
Git is a distributed version control system (DVCS). "Distributed" means that all developers within a team have a complete version of the project. A version control system is simply software that lets you effectively manage application versions. Thanks to Git, you'll be able to do the following:

- Keep track of all files in a project
- Record any changes to project files
- Restore previous versions of files
- Compare and analyze code
- Merge code from different computers and different team members.

These capabilities listed above don't tell how Git actually works, however. In all its complexity, Git works quite simply: you first need to create a local repository in your project's root directory (folder). Afterwards, Git can track project files and directories and add them to the repository. You may be thinking, "Not again! We're talking about a *repository* now?"

I know who did what, when, and why. Git

A repository is just a directory (a folder) in your project's root directory. (Throughout the entire article we'll use the term directory, not folder.) You can't see repositories in your filesystem as they're hidden. But you can still see a repository in your code editor or IDE:

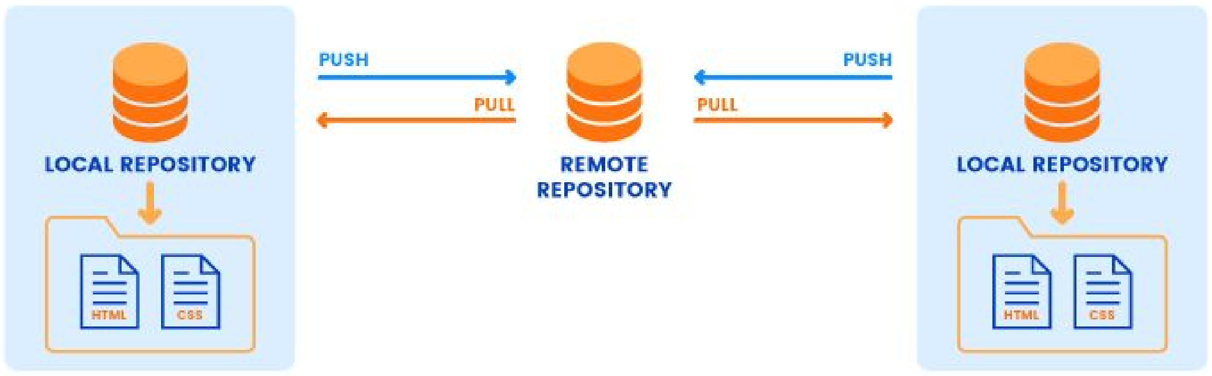


```
#!/bin/sh
#
# An example hook script to check the commit log message taken by
# applypatch from an e-mail message.
#
# The hook should exit with non-zero status after issuing an
# appropriate message if it wants to stop the commit.  The hook is
# allowed to edit the commit message file.
#
# To enable this hook, rename this file to "applypatch-msg".

. git-sh-setup
commitmsg="$(git rev-parse --git-path hooks/commit-msg)"
test -x "$commitmsg" && exec "$commitmsg" ${1+"$@"}
:

```

What's a local repository? Let's use our imagination to understand repositories.

If you store your stuff (code) at home (on a computer with a Git directory), you store your stuff locally. So a repository on your own computer will be called *local*. A *remote* repository is like a public storehouse located in a different building. You may have heard about remote repositories such as GitHub, BitBucket, and GitLab. They're like storehouses for code. Thanks to Git, you can copy your entire project to a remote repository while keeping it in a local repository as well.

Why do we use local and remote repositories? Let's explain a little bit.



In the real world, you can't have *exactly the same stuff* at home and in a storehouse. If your things disappear from home (God forbid!), you'll be able to recover copies or clones (at best) from a storehouse. And you'll still lose some valuables (the original things). With GitHub or BitBucket, however, it's a different story.

Remote storehouses (repositories like GitHub or BitBucket) store exactly the same code that you have in your local repository (on your home computer). If your code disappears from your local repository, you can restore absolutely the same code from a remote repository. A remote repository also serves as a central hub to which members of a web development team can connect to access project code.

By now it should be clear what Git is and what repositories are. But we also need to mention two methods to run Git commands. You can use programs with graphical user interfaces for Git. But you can also run terminal commands for Git. The terminal will be your paper on which you'll write Git commands. For the purpose of this course, we'll use the terminal (also called the command line) to run Git commands. The terminal is a basic tool that all developers should understand.

Lastly, you need to install Git on your computer. We will be doing that in the next lecture!