slide 2: **What is the difference between Git and GitHub**? Git is version control software that runs on your local computer. GitHub is an online collaboration platform interacts remotely with your local Git. A key component is a **repository**: copies can exist on your local computer and in GitHub.

slide 3: Key components of the local Git system: **staging area** and **repository** with processes: **add file** for tracking changes and **commit** files to the version record.

slide 5: Cloned local copies of a repo on local computers can interact with the remote repo on GitHub by **pulling** changes down and **pushing** changes up. Unlike cloud services like DropBox and Google Drive synching is not automatic. You can have local repos on several computers that are synched with the remote.

slide 6: A **branch** represents the history of a line of work on a project. **Commits** are frozen snapshots of the condition of the project a particular moment in time. They represent a version that can be examined, compared to, or rolled back to.

slide 7: The **work cycle** ensures that the local copy of the repo and the copy of the repo in GitHub are synched. If you are working by yourself on a single computer, you can get away with being sloppy about it, but not if the repository is downloaded onto several computers.

slide 8: Collaborators can develop together by making commits to the same branch on the remote. They should strictly adhere to the work cycle, but may create **version conflicts** if they are working on the same part of the text at the same time.

slide 9: An **issue** is a defined problem to be solved. There is usually one or more commits that represent the solution of that problem. The **issues tracker** feature of GitHub helps collaborators keep track of what needs to be done and can be used to keep a record of what was done to solve issues.

slide 10: Collaborators with write access to the same repo on GitHub can avoid version conflicts by working on **separate branches**. This is called the Shared Repository model.

slide 11: Changes made in a branch can be **merged** into the main development branch by a process called a **pull request** (not to be confused with pulling changes from the remote). This process is part of a paradigm for code management called "GitHub flow".

slide 12: The more complex Open Source model when contributors do not have write access to the repo. Contributors **fork** the repository and open a pull request to ask the maintainers to merge their suggested changes.

slide 13: **GitHub Pages** is a system built-in to GitHub that allows you to manage a website using Git and GitHub. Jekyll is a website generator that is integrated into GitHub. It turns Markdown into HTML.

slide 14: The process of creating a local copy of a **remote** repository that's on GitHub is called **cloning**.