This lecture talks about version control. In particular, we will talk about git hub.

I will talk about the concept of distributed version control. Git is a popular tool for version control.

Git hub is a cloud based service for git. You can also use version control services from other vendors.

This class uses git hub to distribute homework assignments. You need to use git to obtain the assignments.

You are encouraged, but not required, to use version control for your own work.

Many people have public repositories. The repositories can be helpful in job searching.

A frequently asked question is why this course does not require that students use git hub for their work. The answer is that git hub is a commercial entity. This class does not wish to force students to use a service offered by a commercial vendor.

If you decide to use git hub, or any other version control service, please make sure your work is private. If your work is public, everyone can see it.

For this course, if your work is in a public repository, you are considered violation of the academic honesty rules.

This figure shows a typical flow of using git.

First, you create a repository. For cloud-based version control service, this is done in the cloud.

The second step is to clone the repository and you have a copy on your local computer. The command is git clone.

You can make changes to your local copy. You may add files, edit files, move them around, and so on.

While you are working on the files on your computer, the repository may have changed. Changes may occur due to several reasons. One reason is that you work in a team and someone else has made the changes. Another reason is that you have several computers. Maybe you have a computer in your office and another computer at home.

You want to make sure your local copy is up-to-date. The command is git pull. This command pulls the changes in the repository.

You can keep working on your local copy of the files.

When you feel that your work is ready to take a snapshot, use the git commit command.

The git commit command changes only your local files. You have not made any changes to the repository yet.

When you are ready to change the repository, use the git push command.

Let’s look more deeply how distributed version control works. Suppose you have a repository stored in the cloud, such as git hub.com. This repository is shared by several people.

You can have a copy on your office computer, another copy on your home computer. Your teammates, Alice, Bob, Cathy, and David, can each have a local copy. The command to get a local copy is git clone.

You have made some changes to the files in your office computer. David also made some changes.

Both you and David do “git commit” to take snapshots of the changes.

Please remember that you need to do “git commit” often. If you do not do “git commit”, version control does not help you.

David has not done “git push”. Thus, the changes he has made are visible to him only. The other people cannot see his changes.

You decide that your changes are ready for others to see. Thus, you do “git push”. The repository is changed.

Next, others can do “git pull” and obtain the changes you have pushed.

For David, the changes from you are merged with the changes David has made earlier.

Please notice that the local copy at your home computer now has the changes you have made in your office computer.

To summarize, you can do “git commit” as often as you want because you do not modify the shared repository yet.

If you do “git push”, the changes you have made are visible in the repository and the others can see that changes you have made.

The “git pull” command obtains the latest changes from the repository.

Here are some suggestions how to use git effectively.

You should do “git commit” often. Without “git commit”, version control cannot help you. You should also do “git pull” often to ensure that you get the most recent changes in the repository.

If you work alone, you can do “git push” as often as you wish.

If you work in a team, however, you need to be more careful. You should push only working code. If you push buggy code, your buggy code may break the others’ work.

The design principle of version control is that you should do one thing at a time, finish it, test it. Ensure that your work is complete and then push it.

Many beginners want to make a huge amount of changes, commit all changes at once, and then push. The problem of doing so that you increase the chance of incompatibility with others’ work. If you push only a few changes each time and your teammates have something different, you can discover the problems earlier and solve them more easily.

A general rule is that you should do “git commit” and “git pull” at least once a day. You should do “git push” at least once a week. If what you are doing takes more than one week to finish, you should divide it into smaller pieces.