

**Intro to Git and GitHub**

Teacher Assistant Supplemental Material:

**Slide Notes:**

**Slide 7: Example Workflows: Non Linear**

1. Support for multiple users and sets of changes.

**Slide 8: Example Workflows: Subversion- Style (Centralized)**

1. A very common workflow (Subversion, CVS and CM Synergy support only this workflow), especially for people transitioning from a centralized system, is a centralized workflow. A distributed VCS will not allow you to push if someone has pushed since the last time you fetched, so a centralized model where all developers push to the same server works just fine.

**Slide 9: Example Workflow: Integration Manager**

1. Another common workflow is where there is an integration manager: that is, a single person who commits to the 'blessed' repository. There are a number of developers who clone from that repository, push to their own independent repositories and ask the integrator to pull in their changes. This is the type of development model you often see with open source or GitHub repositories.

**Slide 10: Example Workflow Dictator and Lieutenants**

1. For more massive projects, you can set up your developers similarly to the way the Linux kernel is run: certain people are in charge of a specific subsystem of the project (*lieutenants*) and they merge in all changes that have to do with that subsystem. Then another integrator (the *dictator*) can pull changes from only his/her lieutenants and then push to the *blessed* repository that everyone then clones from again.

**Slide 11: Version Control Distribution:**

1. Here is a survey from 2016 and here are the most used version-controlled system.

**Slide 12: Without Version Control:**

1. This is what it would look like without version control. You have numerous files with different names, with similar content and minor changes

**Slide 13: With Version Control:**

1. Unlike the previous side, this is why a version control is so effective.
   1. Only one big file
   2. You have a date when it was changed
   3. There was authors listed.
   4. What was changed specifically, no need for numerous of files with different names
   5. Also code, where you can search for different changes.

**Slide 14: Installation**

1. <https://help.github.com/articles/set-up-git/>
2. Download and Install: <https://git-scm.com/downloads>
3. Set your username in Git: <https://help.github.com/articles/setting-your-username-in-git/>
   1. Open terminal
   2. Set a Git Username
      1. $ git config --global user.name “Mona Lisa”
   3. Confirm that you have set the Git username correctly:
      1. $ git config –global.user.name
         1. It should give you “Mona Lisa”

**Your first Local Repository:**

1. Go to the Home Directory and Type in the bash terminal:
   1. Type in: **cd ~**
2. Now, let’s create a “working directory”
   1. Next, type: **mkdir my-first-repo**
   2. Directory is similar to files and folders on our computer
3. Then, type: **cd my-first-repo**
   1. Now you should see something similar to: *my-first-repo donitm$* in the terminal and on top of the terminal
4. Now, let’s initialize repository with Git
   1. Type in: **git init**
   2. You should now see *“Initialized empty Git repository in /Users/donitam/my-first-repo/.git/”*
5. It’s important to always check the status of our repo, to see any changes and etc
   1. Now, type: **git status**
   2. It should say *“no commits yet”,* which mean we have not committed anything to the repo, as of yet.
6. Now, lets add some files to our repo:
   1. Type: **$ echo "# This is my first repo's text" >> hello\_world.txt**
   2. Now we have a file, it should say *“Untracked Files”*
7. Now, it’s time to track the files:
   1. Type: **$ git add hello\_world.txt**
8. Now, let’s check the status:
   1. Type**: $ git status**
   2. it should say “*new file: hello\_world.txt” in green”*, which means it’s tracking.
9. Now let’s set it up to add so more text to our file:
   1. Type in: **git add hello\_world.txt and press enter**
10. Let’s start staging and committing changes
    1. Type: **git commit –m “First commit. Added hello world to repository.”**
       1. *You should now see something similar to: “[master (root-commit) 0e17..] // 1 file changed, 1 insertion.*
11. Now let’s look at our progress and see what we just did:
    1. Type in: **git log**
    2. You should see
       1. *Commit [hash here]*
       2. *Author: Your name*
       3. *Date: [Date Here]*
12. So nobody’s perfect, let’s undo local changes:
    1. Let’s first check our status: git status
    2. Then type: git checkout hello\_world.txt
    3. Lastly, check our status again.
13. Still nobody’s perfect, let’s undo staged changes
    1. Type: git add hello\_world.txt
    2. Type: git reset HEAD hello\_world.txt
    3. Type: gitl checkout hello\_world.txt
    4. Type: git status