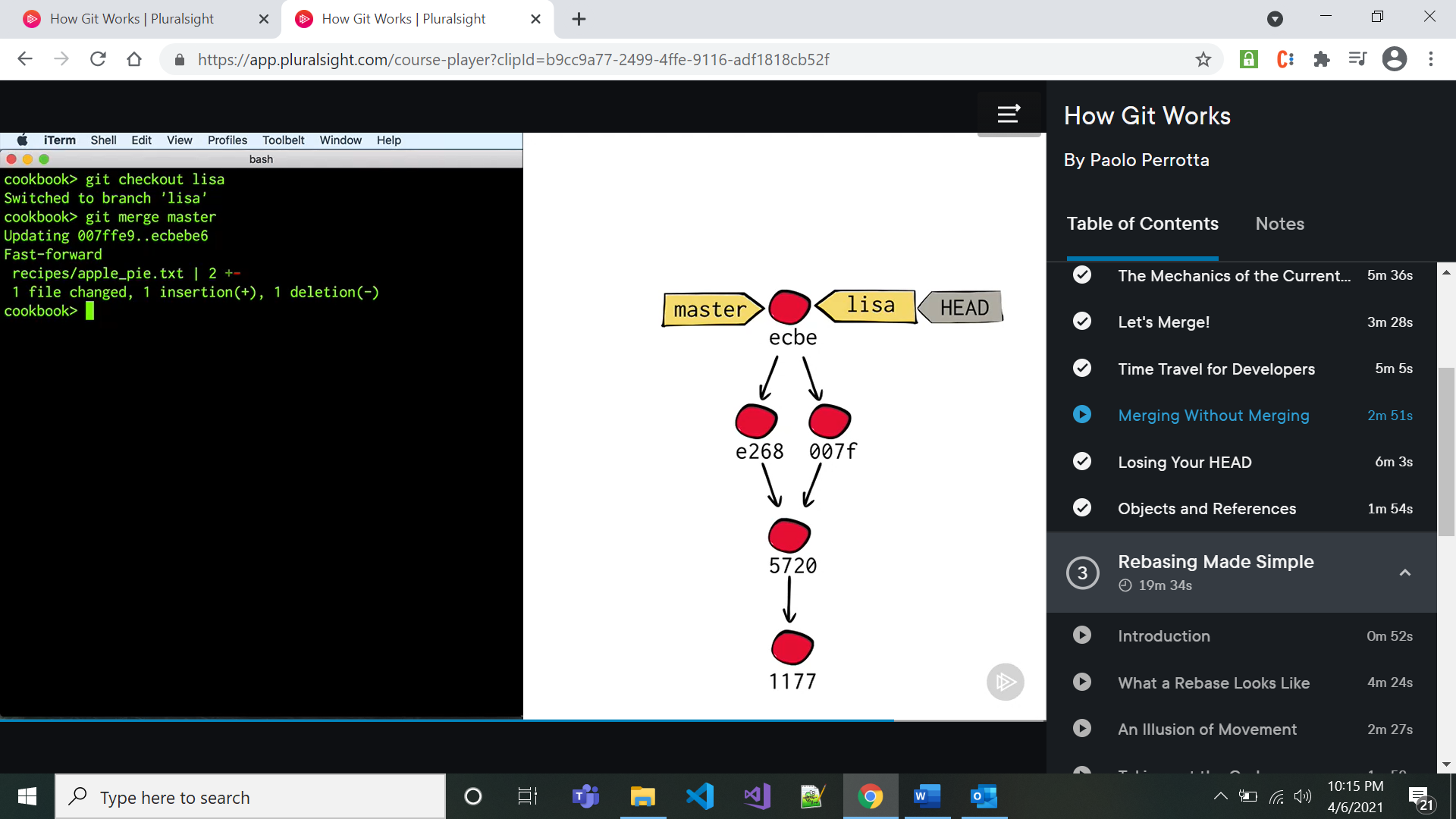
**How git works-**

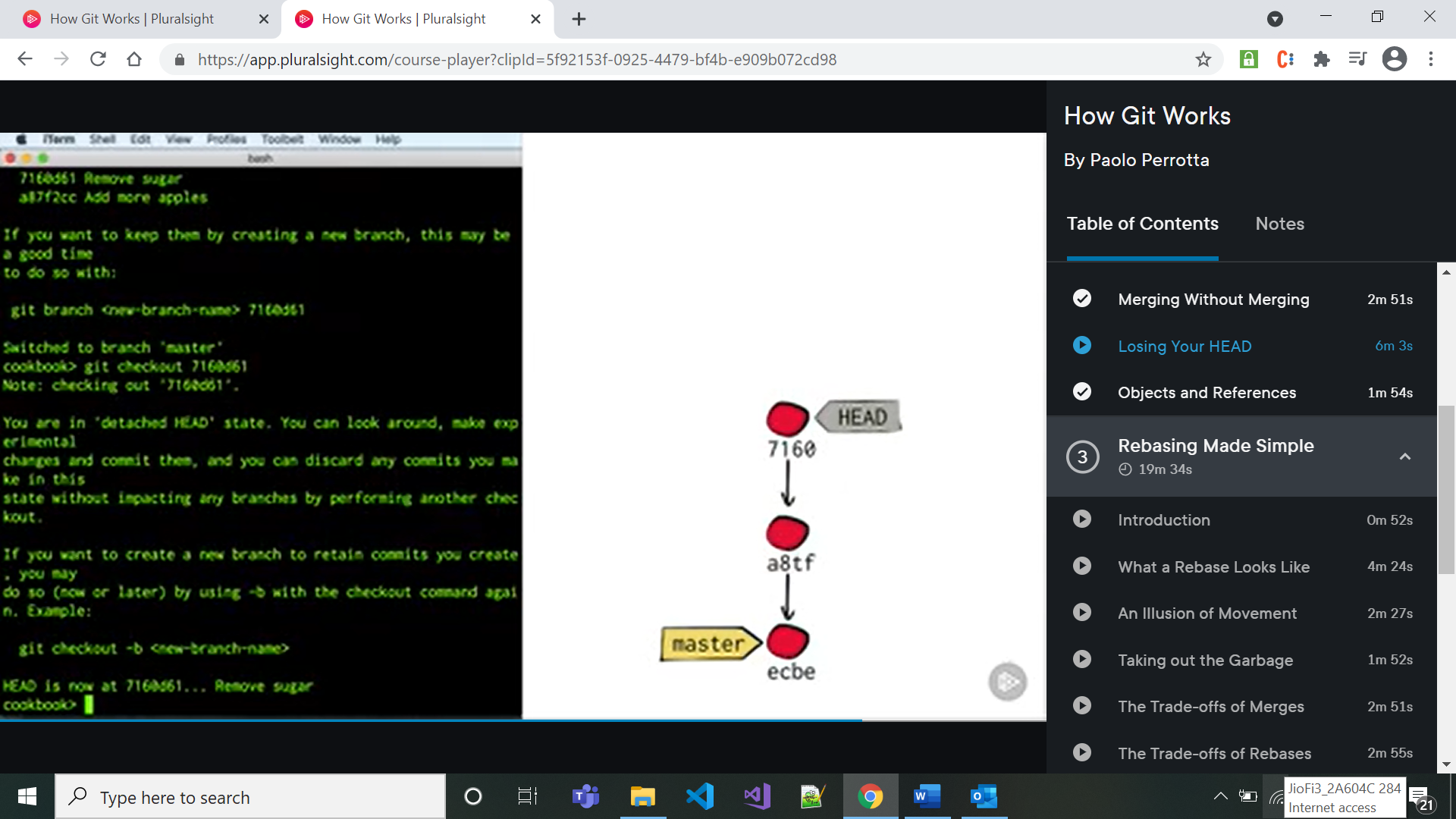
* when changes are committed then a hash object is created which is compressed object of commit msg/tree/file content. For eg- 2872344970ef162779153adff531f02271ed8fdc in this if you open .git/objects then a folder 28 will be there which is first two numbers of hash and in that remaining numbered file will be available that file may be tree/blob . Tree is directory and blob is file we can check the content of these hash object with below comments-

git cat-file -p 2872344970ef162779153adff531f02271ed8fdc

* In .git folder inside ref folder our branches are stored and HEAD file contains the current branch which has been checked out in the working directory.
* When branch is merged in other branch then if that commit is checked with git cat-file -p commit\_no then we can see it contain two parents normal commit has single parent but merge commit has two commits
* There are some corner cases in git - 1) when branch say lisa is merged into master with all latest changes and again if we try to merge master into lisa then it will point to same commit and this is called fast forward



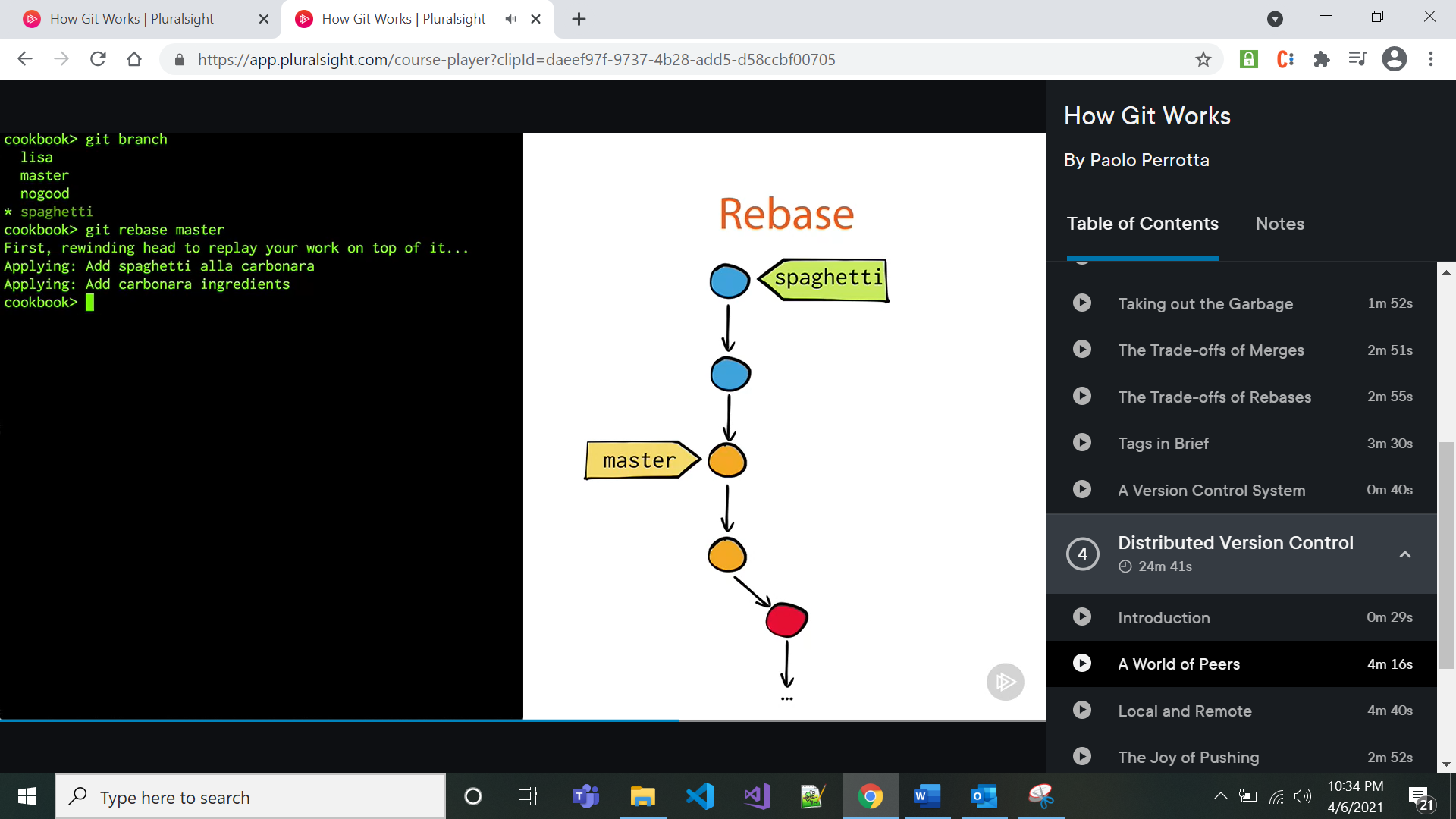
* When we checkout to a particular commit then that time HEAD is get detached and it will no more point to any branch. If we do commits and checked out to any branch then those commits will be not pointed by HEAD and will be released by garbage collector. If we want to save those commits then we can create new branch there.



* git rebase branch\_name -----------it will apply all the commits which are present in specified branch to checked out branch where they are not available. In short to get changes of one branch into another branch without merging we can do it with the rebase.

Diagram

Description automatically generated



* Merges preserve history
* Rebase clears history
* Tags are just branch that doesn’t move

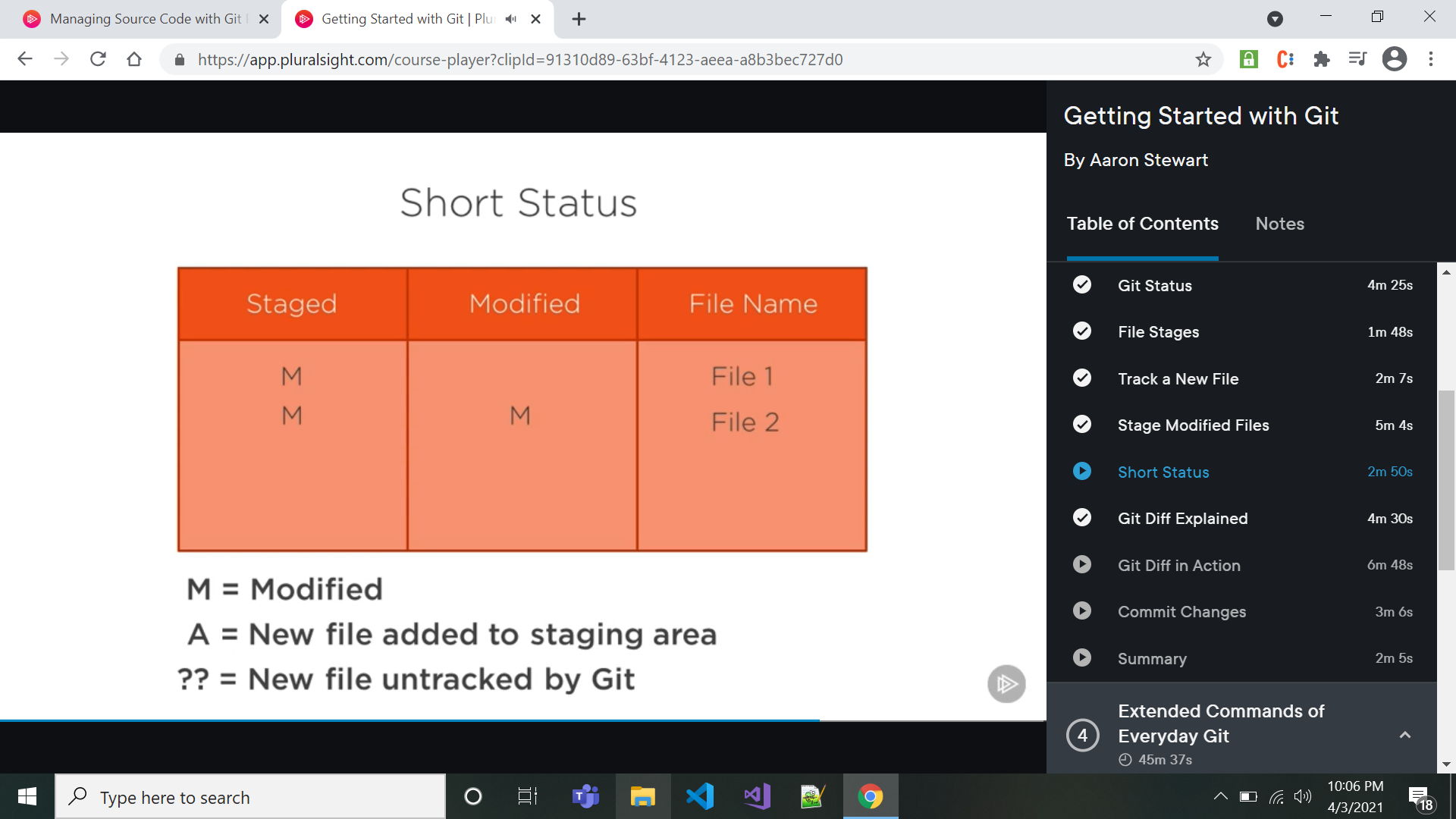
**Git Commands-**

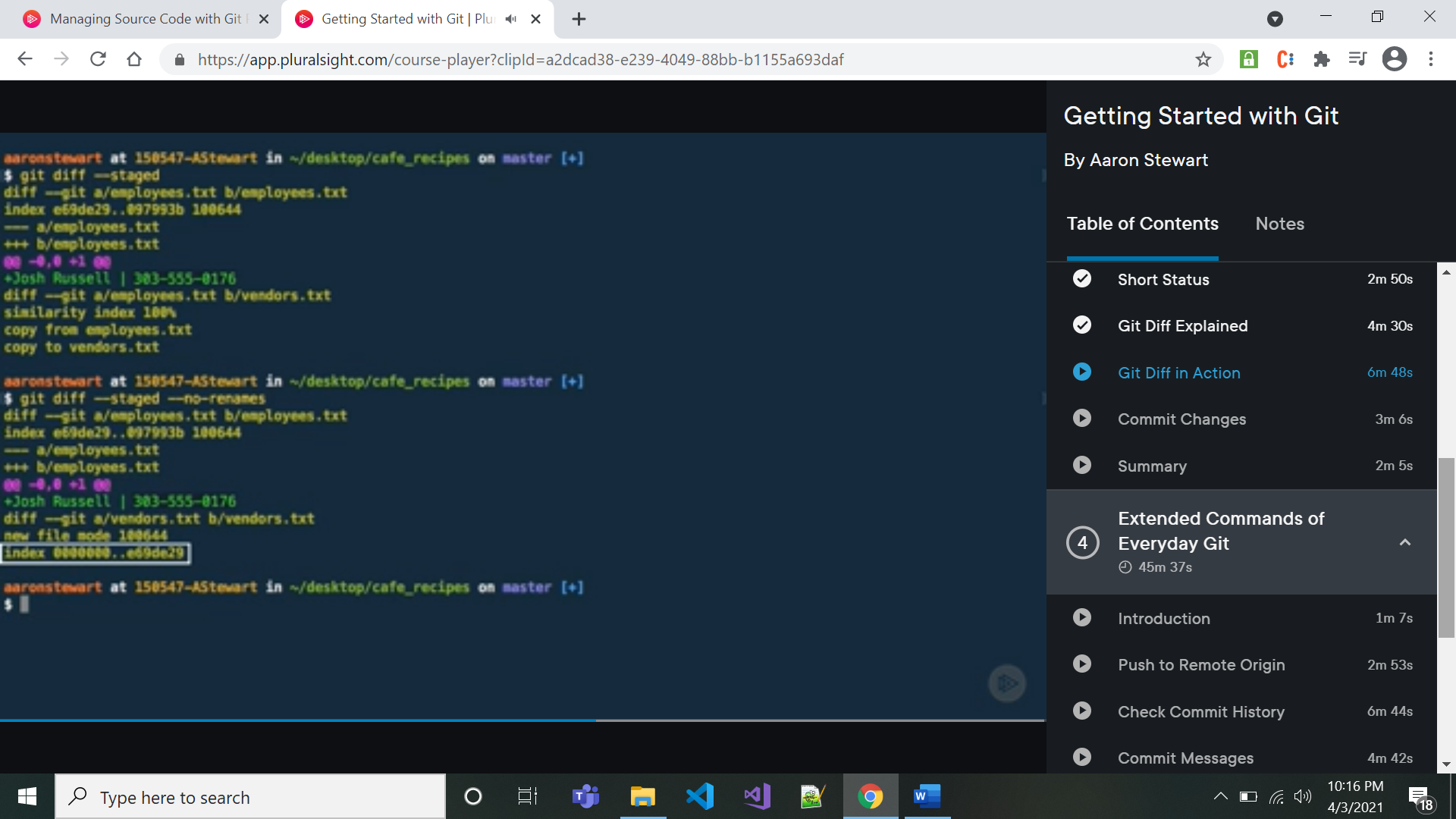
* git init- to initialize local repo
* **Git Configuration commands-**

1. git config --list: To see different configurations
2. git config --global user.name “name”: to set username at global scope
3. git config -- global user.email
4. git config --local user.name – to set username project level

* git remote add origin https://github.com/AngularDemoProjects/NodeJs-Blog-app.git blog –

to give reference of remote repo where we wanted to push our local repo

* **git remote set**-url **origin** https://github.com/Tekfinity/repoName.git-to change remote
* git add . – to add all files
* git status -------------gives status if anything need to be committed
* git status --short --------to give short status
* 
* git commit -m “commit message”
* git diff --staged -----------it compares changed/untracked files with last commit changes



git diff --staged does not recognize if new file which has same content with existing file then git shows for that a similarity index with 100%. So for that git diff --staged –no-renames can be used.

* git diff commit1\_unique\_no commit1\_unique\_no-----it shows changes and difference in those two commits
* git checkout commit\_unique\_no --------to get that commit change repo
* git log – to chk all commits
* git log --oneline ------------to get the logs with its msg only
* git push -u origin master
* git rm file\_name -----------------to delete file from project and git
* git rm --cached file\_name----------------to remove that file from git and not from project to stop tracking it
* git checkout -b new\_branch ---------------here -b will create new branch and then checkout to that branch
* git branch -----------------it will list all branches
* git branch -m <oldBranchName> <newBranchName> ------------to rename branch
* git branch -d <branchName> --------- to delete branch
* git branch -D <branchName>--------to forcefully delete the branch

**For Stashing**

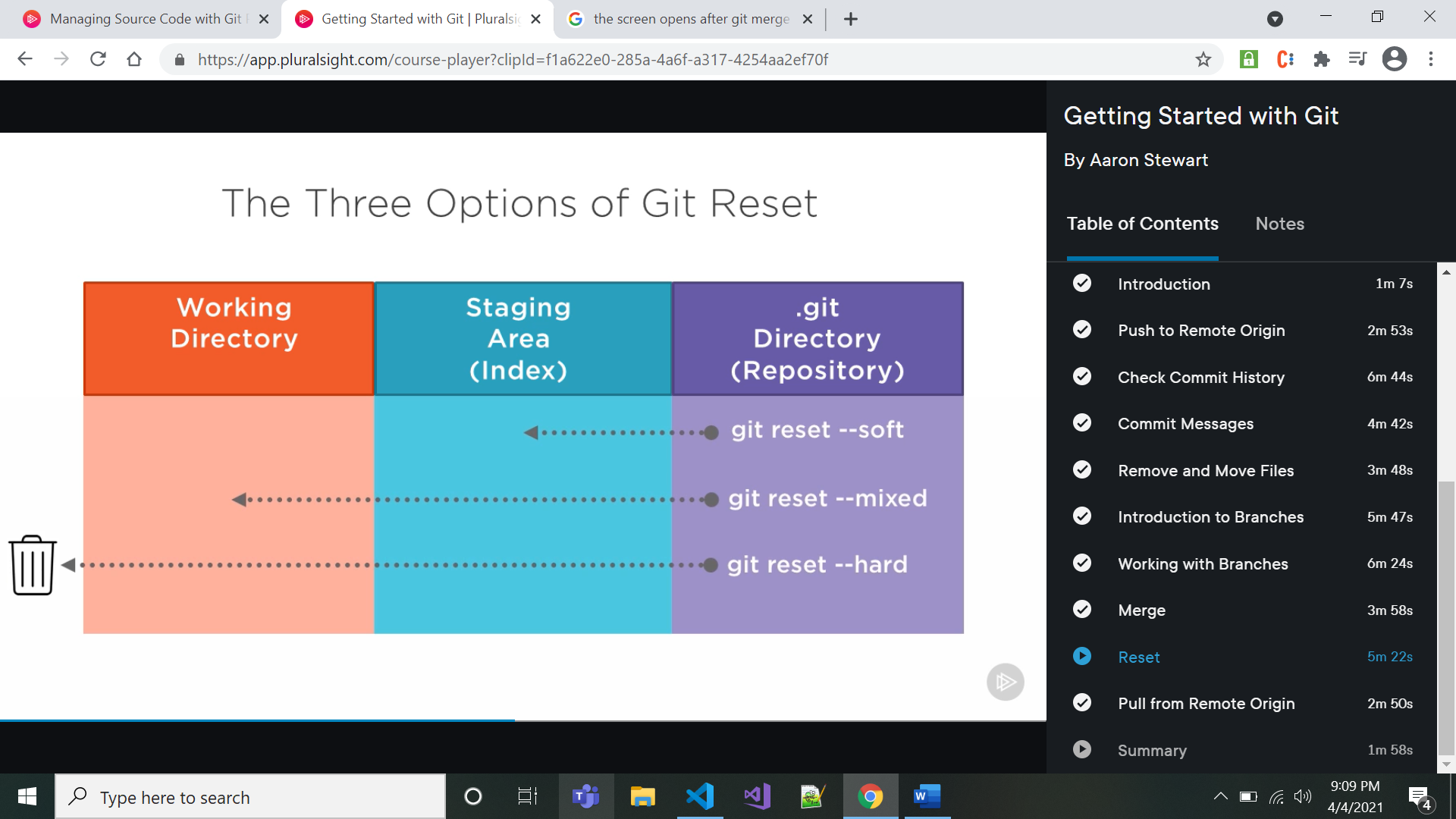
* git add .
* git stash ----to stash the changes
* git stash pop ------to apply stash

**Merging**

* git merge branch\_name -------it will merge specified branch into checked out branch

after this command it will or wont open tool called VIM, if it opens then just press I for interactive and then press escape and type :wq to save and quit.

**Resetting**

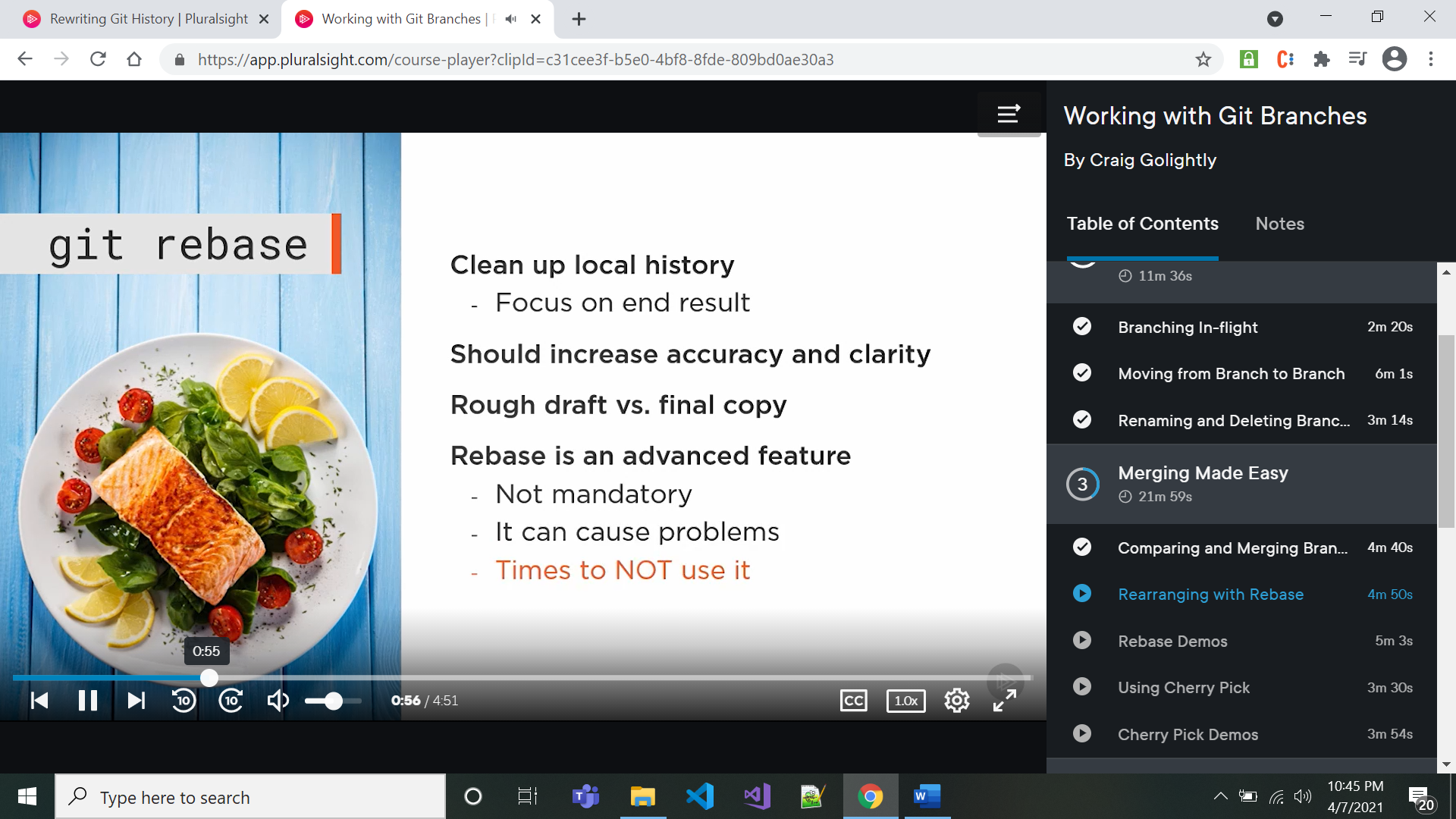


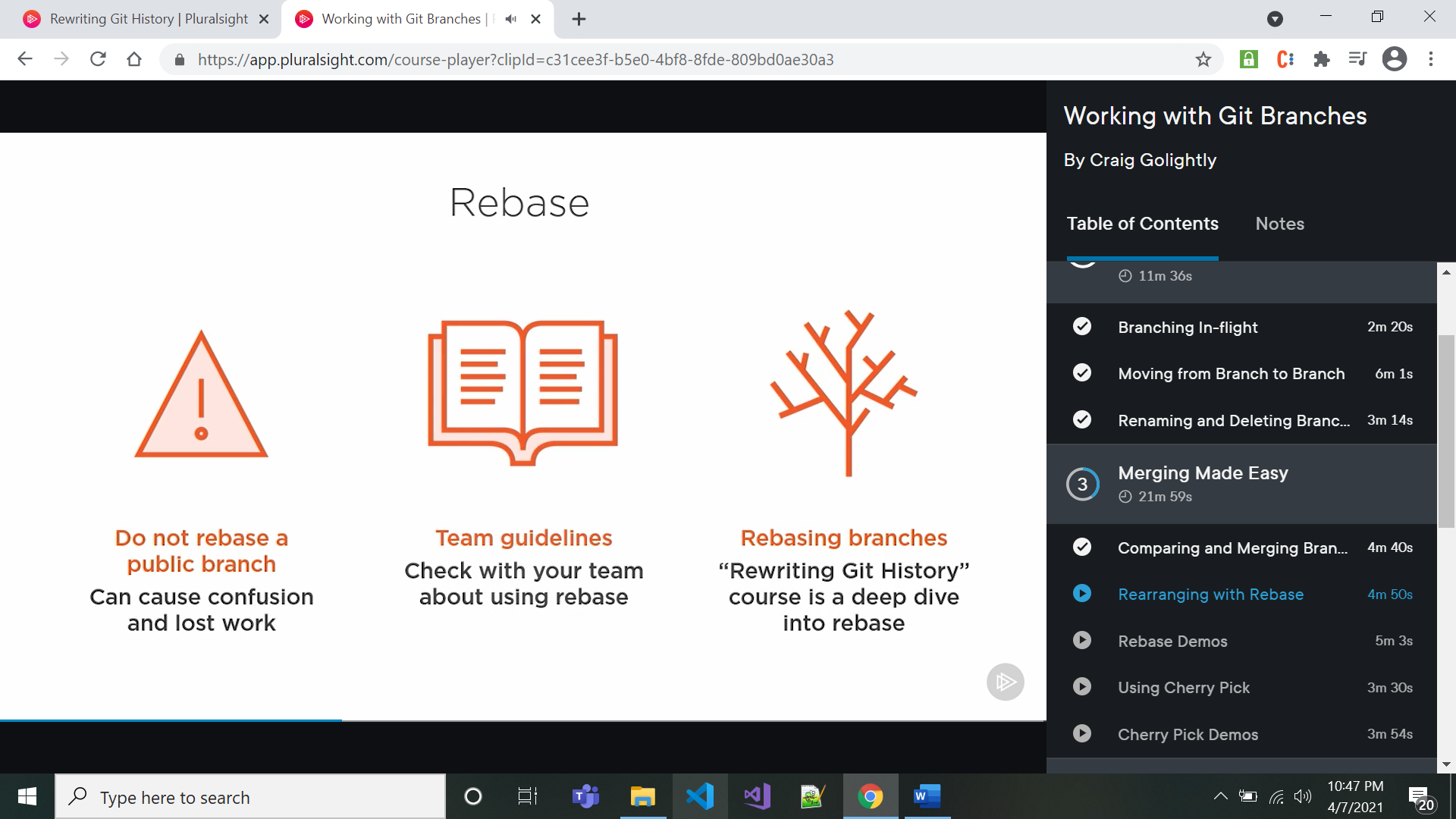
* git reset --mixed specific\_commit\_number ----------------it will get that commit changes in to working directory and remove that commit number changes

Initial Intro---------------------------------------------------

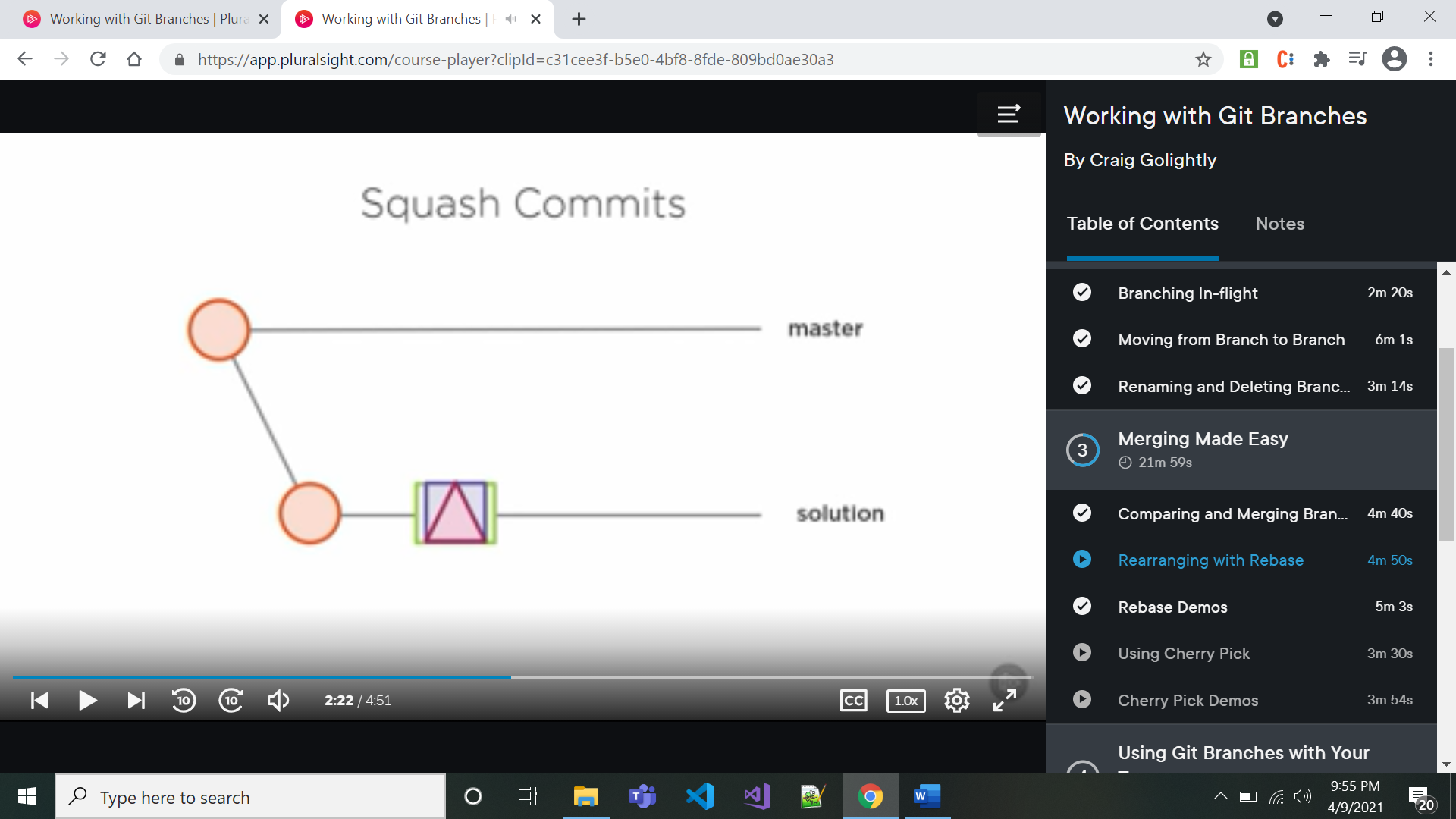
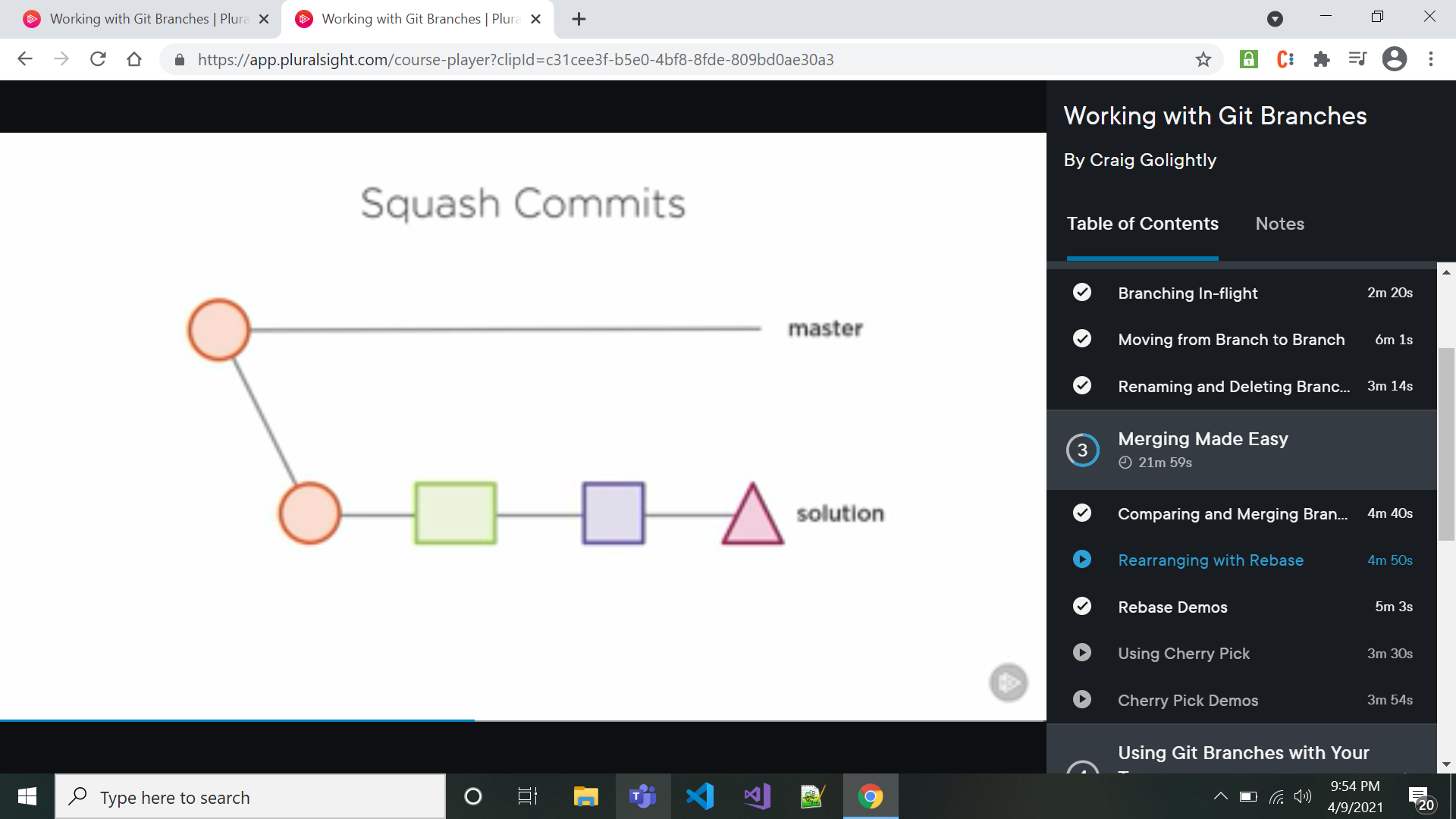
* git log- to see logs of commits
* git log --grep=”Init” : it will find all commits which contains init in their commit msg.

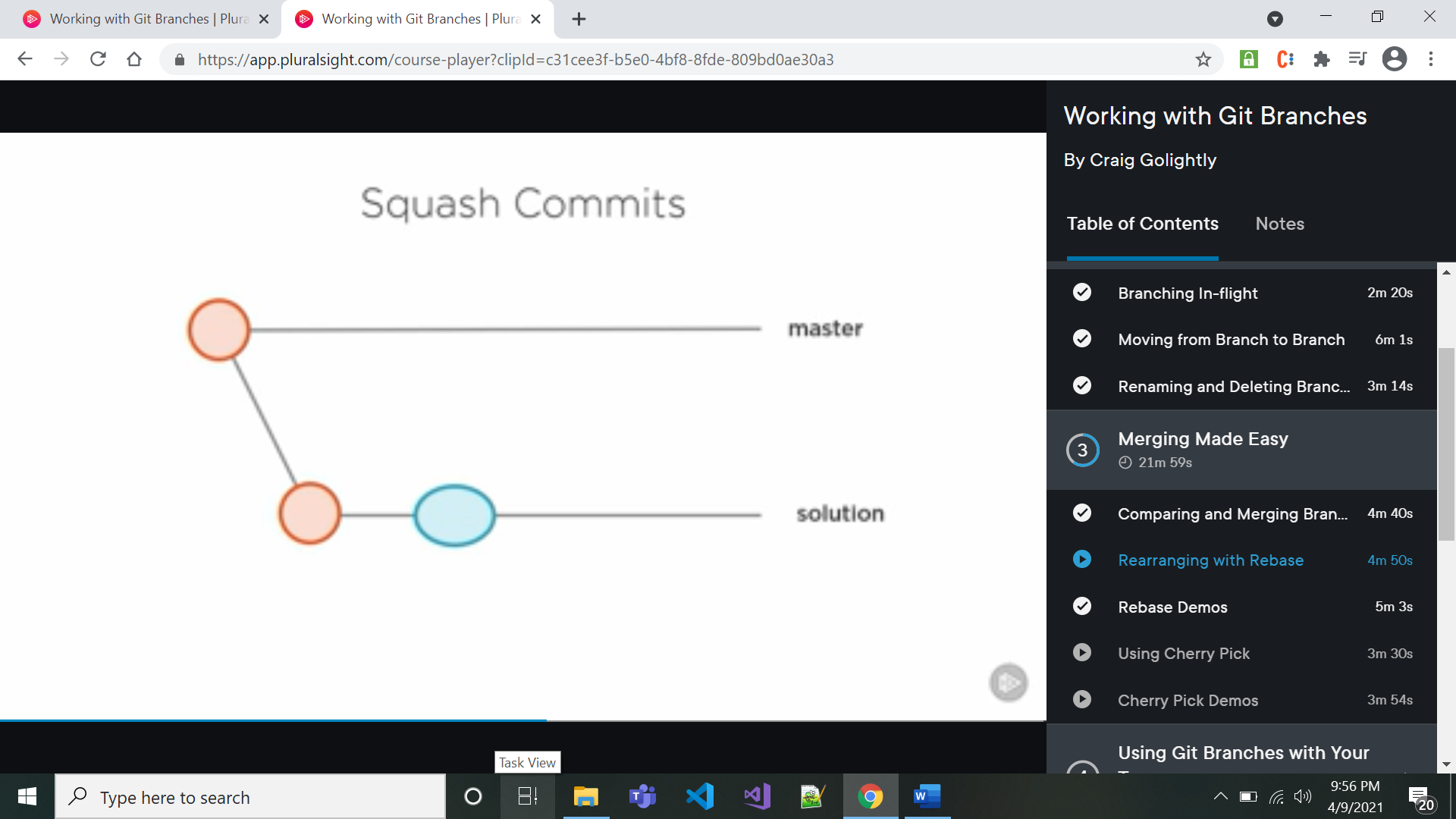
Rebase------------------------------------------------------------------------------------------------



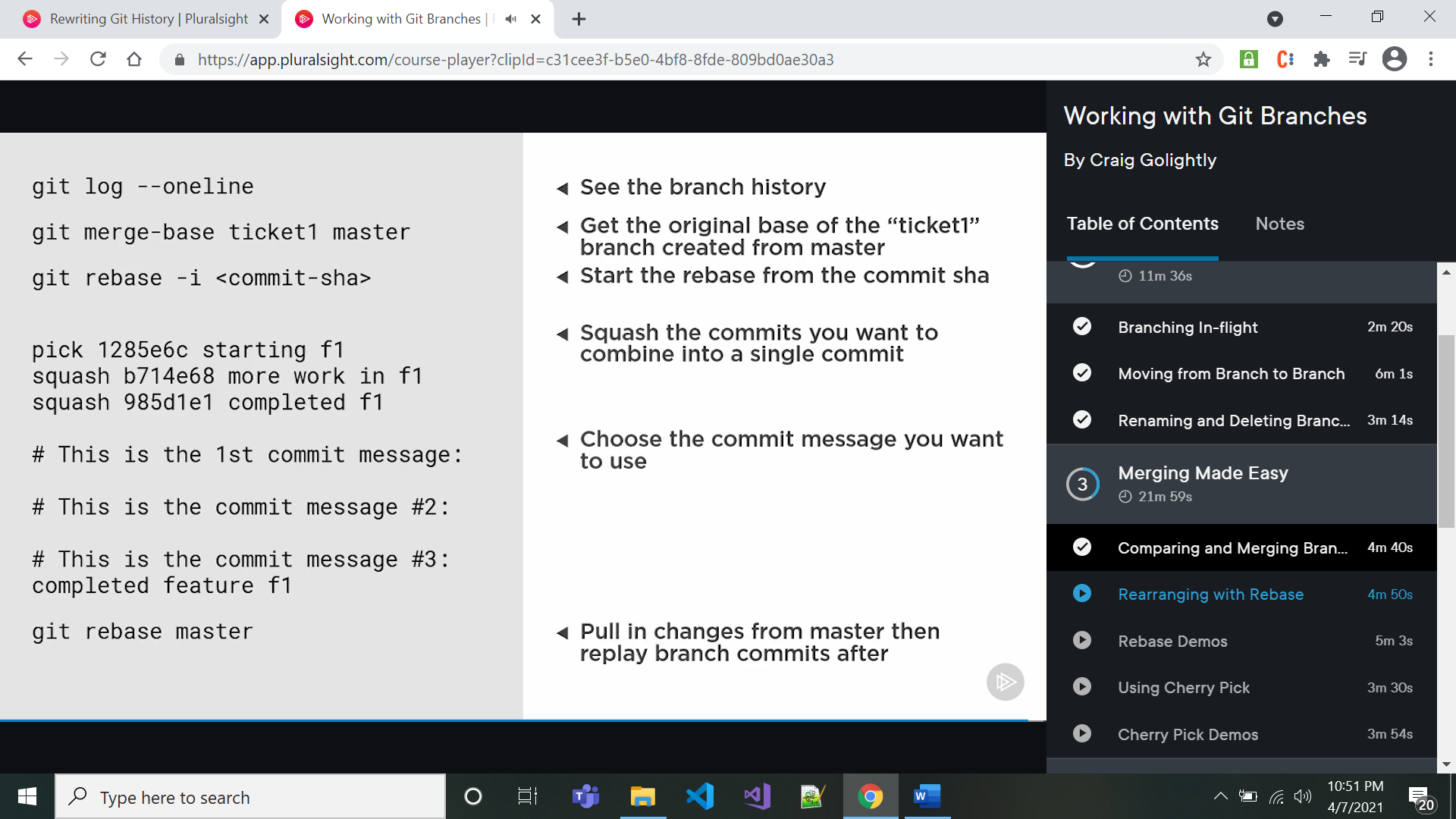


* Squash Commits—





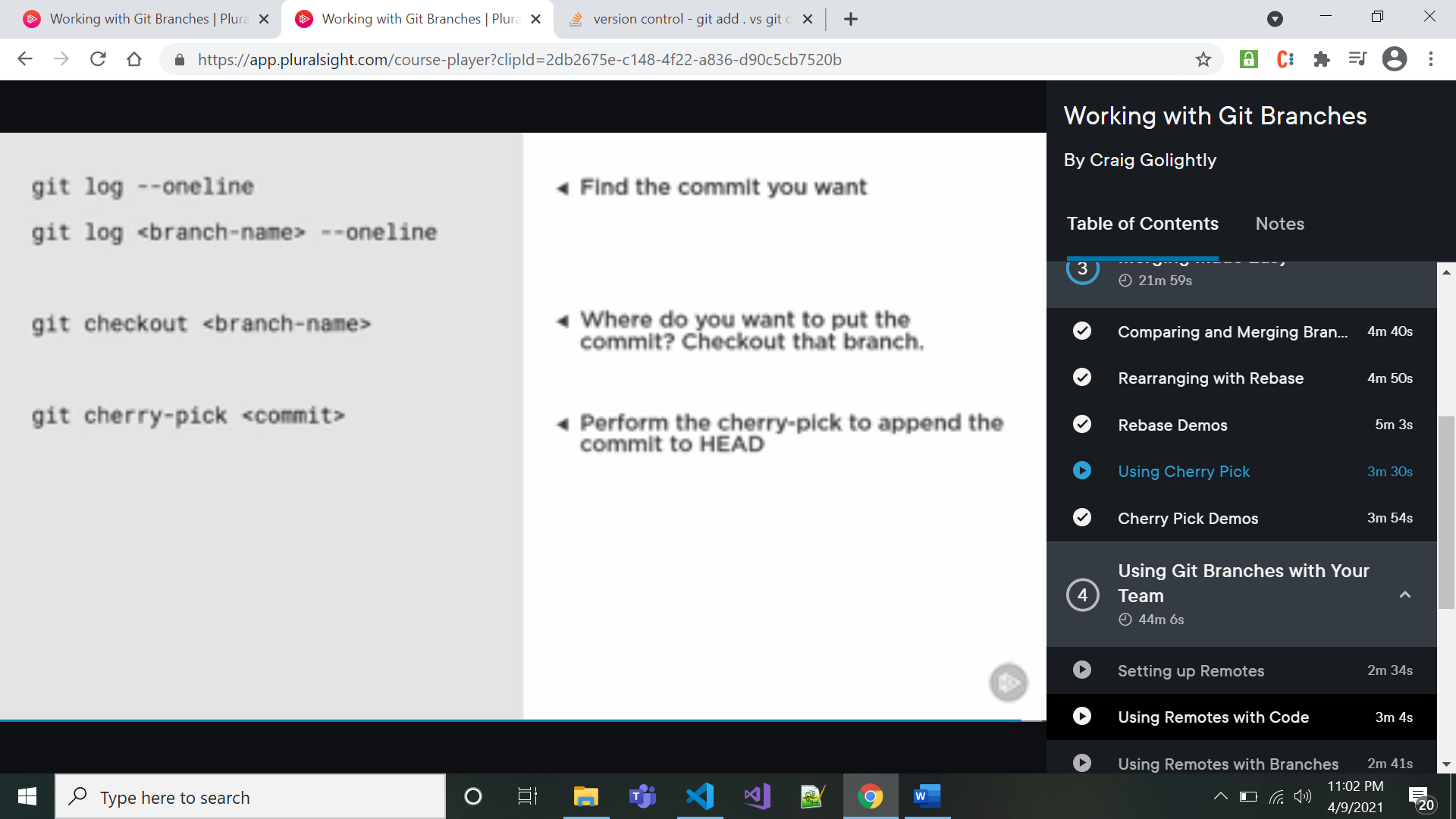
Squash commits means it combine commits as shown in above first two images and git creates new commit as shown in above image and we can choose commit msg for it

* 

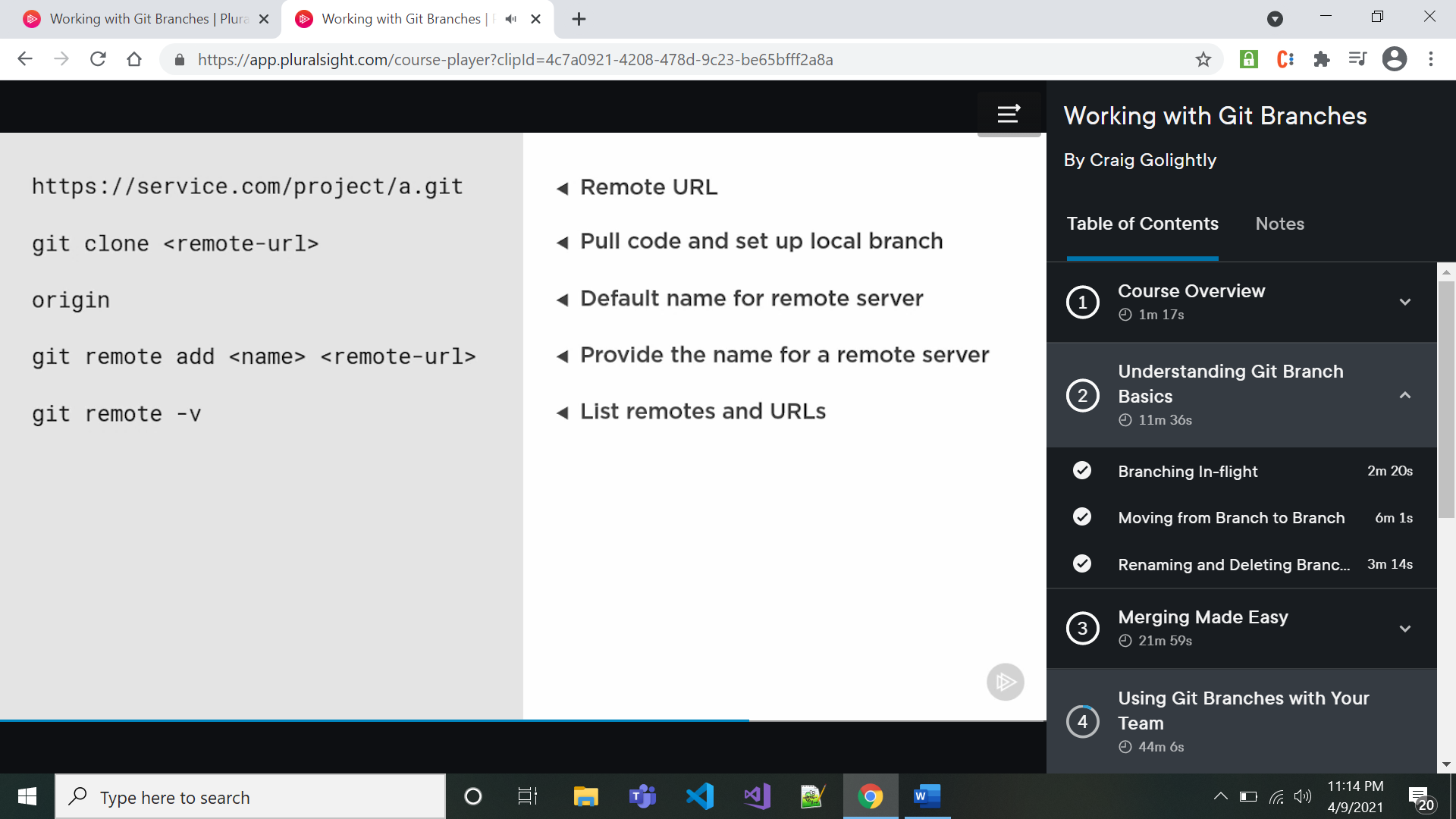
In above what happening is – first base commit of the ticket1 branch created from master is identified.after that commits are squashed into one commits with the help of rebase of commits. So after this step if you check log then that branch commits are combined into one commit who will have different SHA than previous one same commit msg. If we do git rebase master then ticket1 branch commit will appear on the top of master branch commit.

* git reflog ------to check process during and after the rebase

**Git Cherry-Pick------------------------**



**Setting Remote----------**



* to change the author of previous commits

1. change the editor and set to notepad or notepad++

change the editor by git config --global core.editor "'{path to editor}' -n -w"

1. git rebase -i –root

this will open our editor. Then add command below respective commit

exec git commit --amend --author “Ash1# <>”

save file and close it

in cmd prompt it will execute command and finally chk result by ‘git log’