**PART 3**

**Q1) What is GitHub?**

GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. GitHub lets you work together on projects.

**Q2) When was it created?**

It was created on February 8, 2008.

**Q3) Why?**

It offers all the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

**Q4) By who?**

GitHub was developed by Chris Wanstrath, P. J. Hyett, Tom Preston-Werner and Scott Chacon.

**Q5) What similar platforms exist?**

Other platforms that are like GitHub are- Bitbucket, SourceForge, GitLab, Kiln, Codeplane, CodePlex and Beanstalk.

**Q6) Why would you use such a platform?**

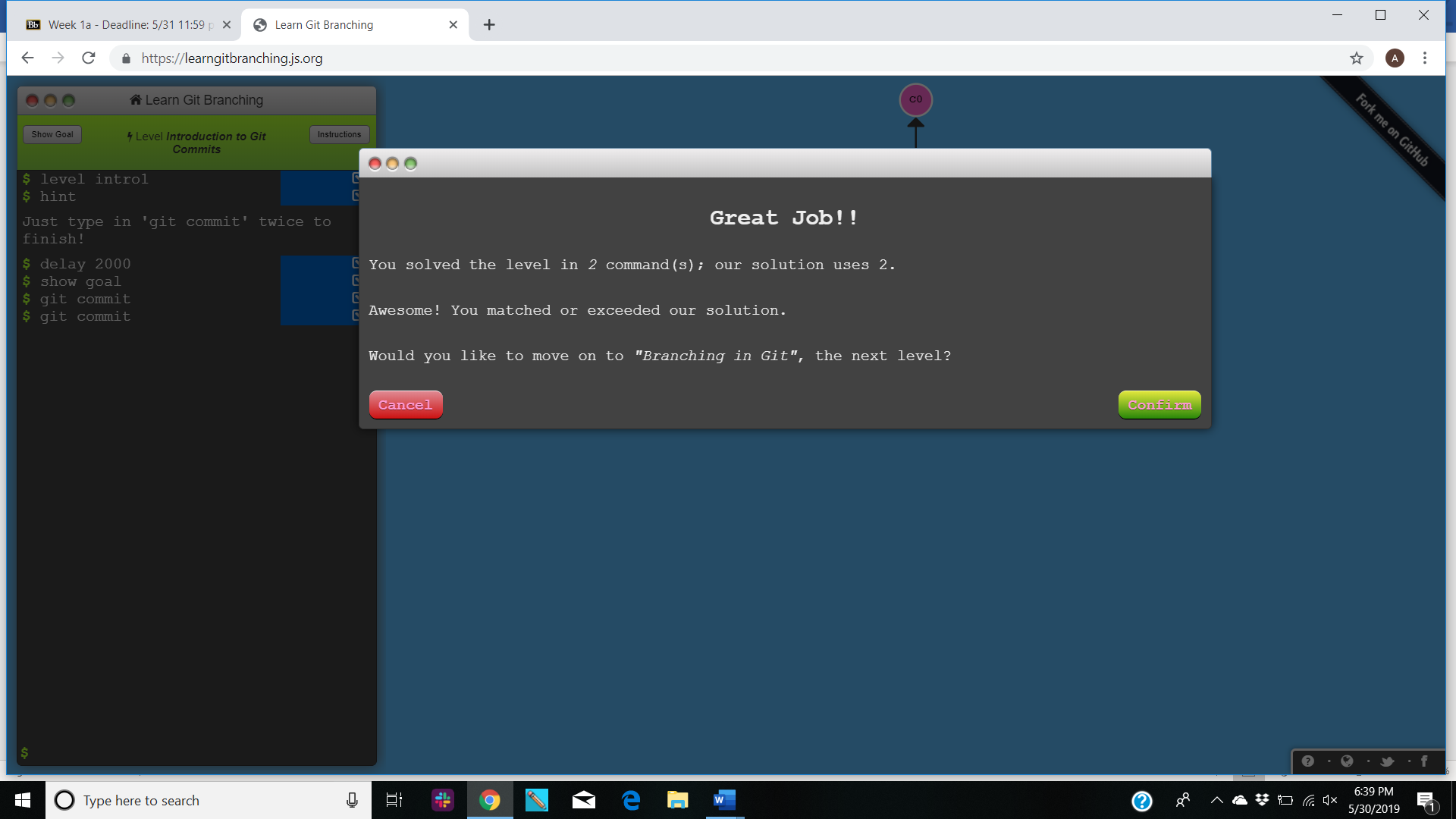
Using GitHub makes it easier to collaborate with colleagues and peers and look back at previous versions of your work. A platform like GitHub which is a repository helps users document and showcase their work. It is a good platform to socialize and host open-source software projects for free.

**PART 4**

**Introduction sequences-**

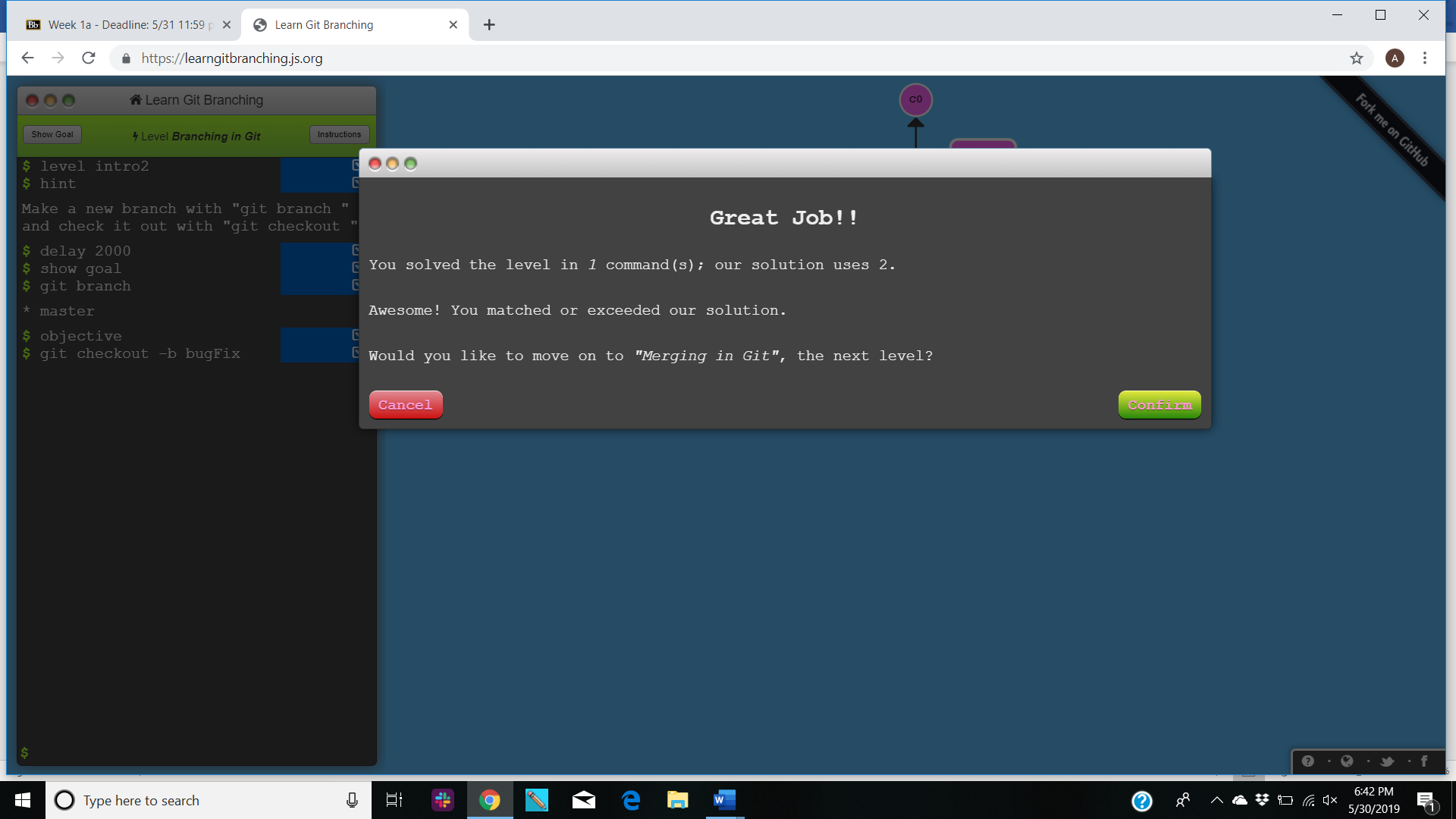
1. **Introduction to git commits-**

Command used – git commit



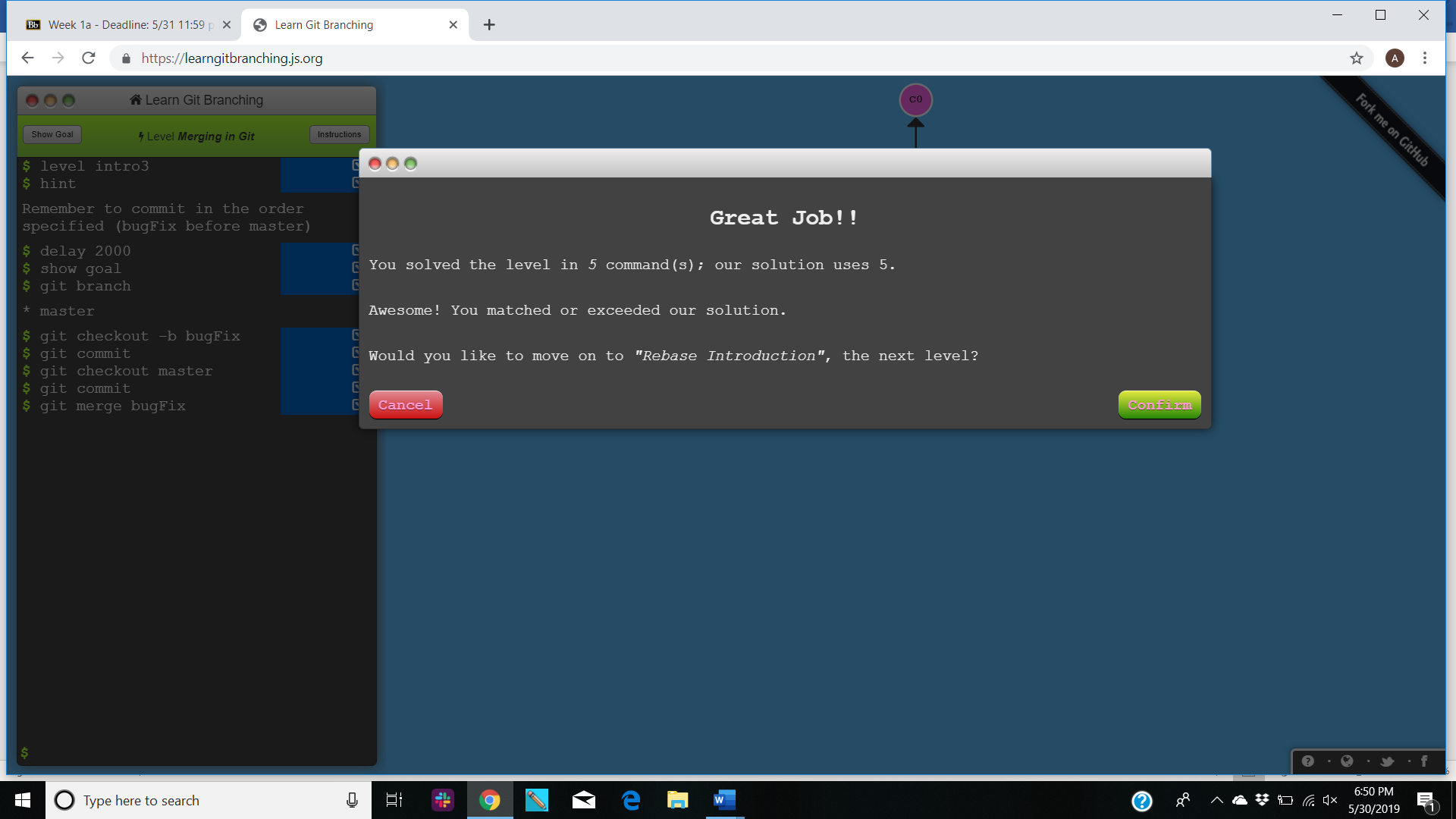
1. **Git Branches-**

Commands- git branch and git checkout.



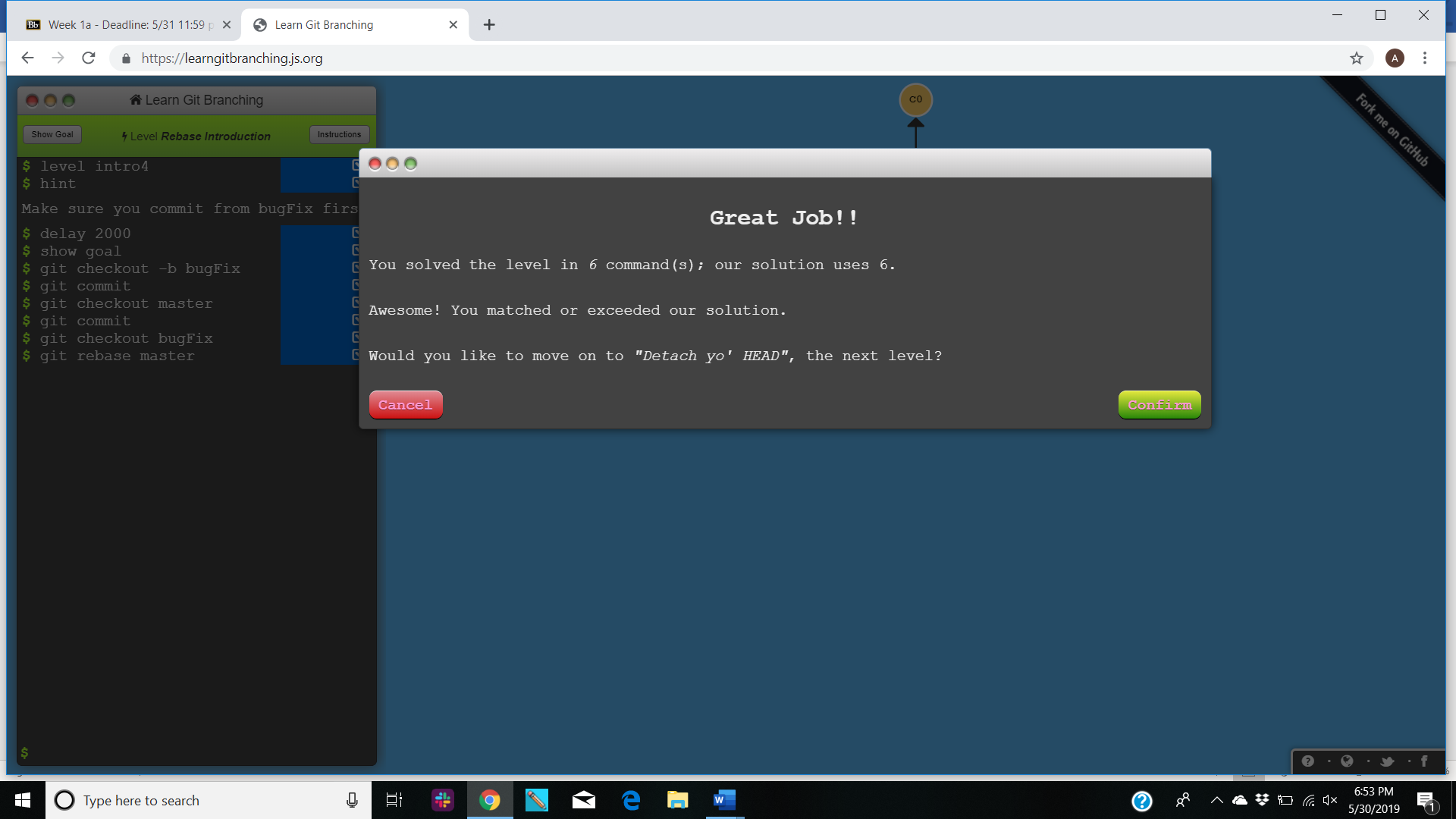
1. **Branches and Merging-**

Commands- git merge



1. **Git Rebase-**

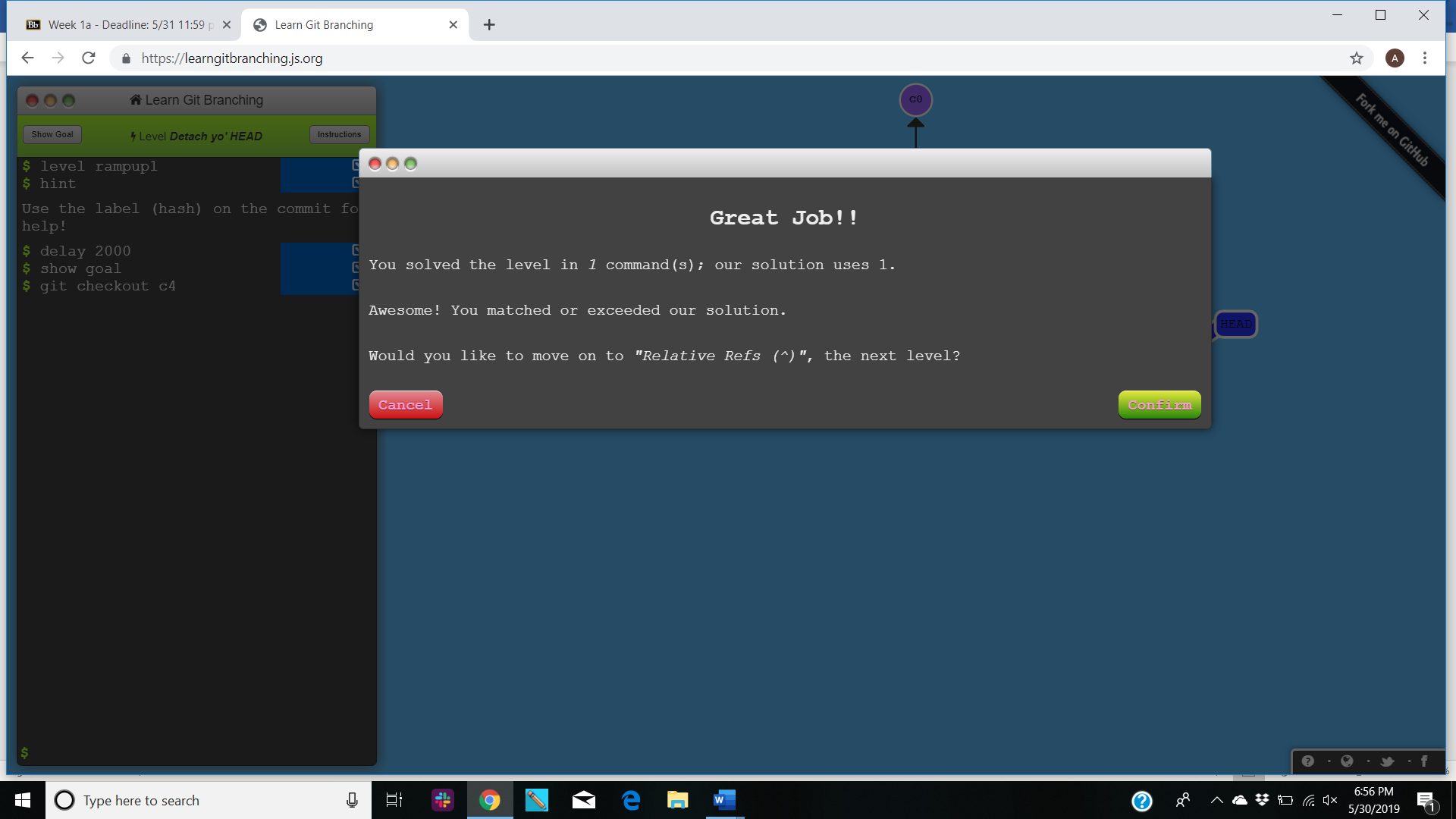
**Commands-** git rebase



**RAMPING UP-**

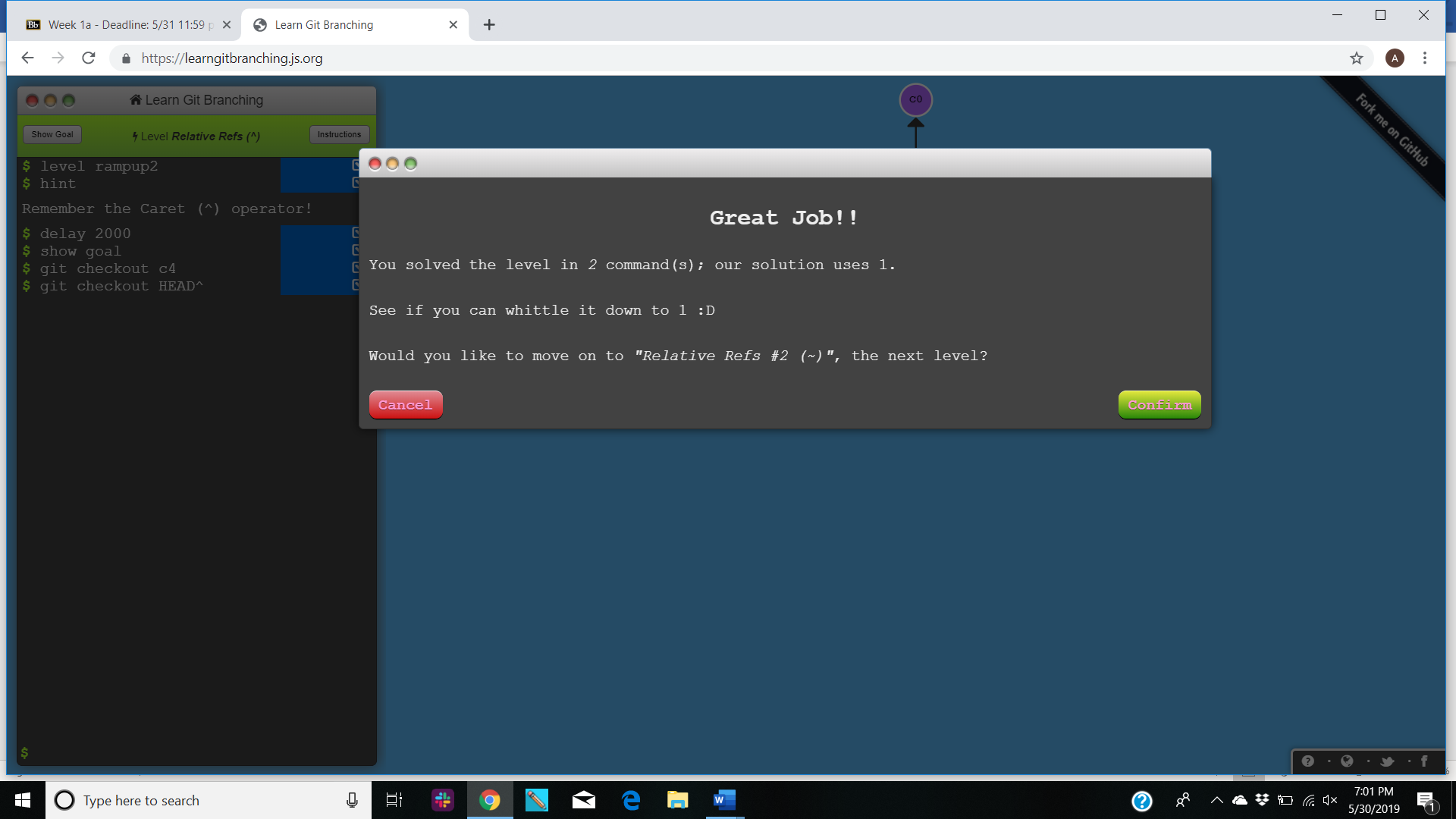
1. **Detach yo’ HEAD-**

**Commands-** git checkout



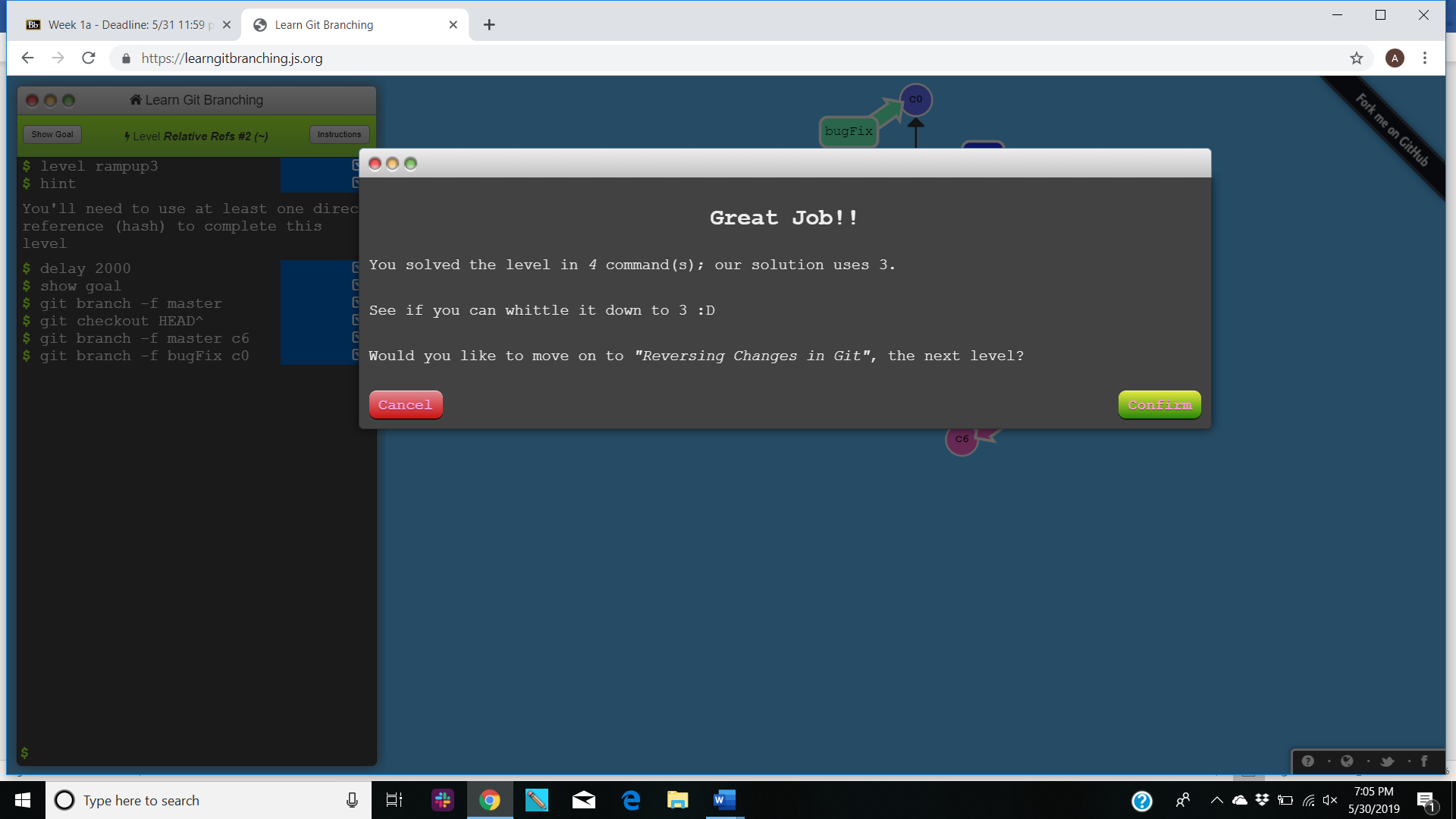
1. **Relative Refs-**

**Commands-** git checkout ^



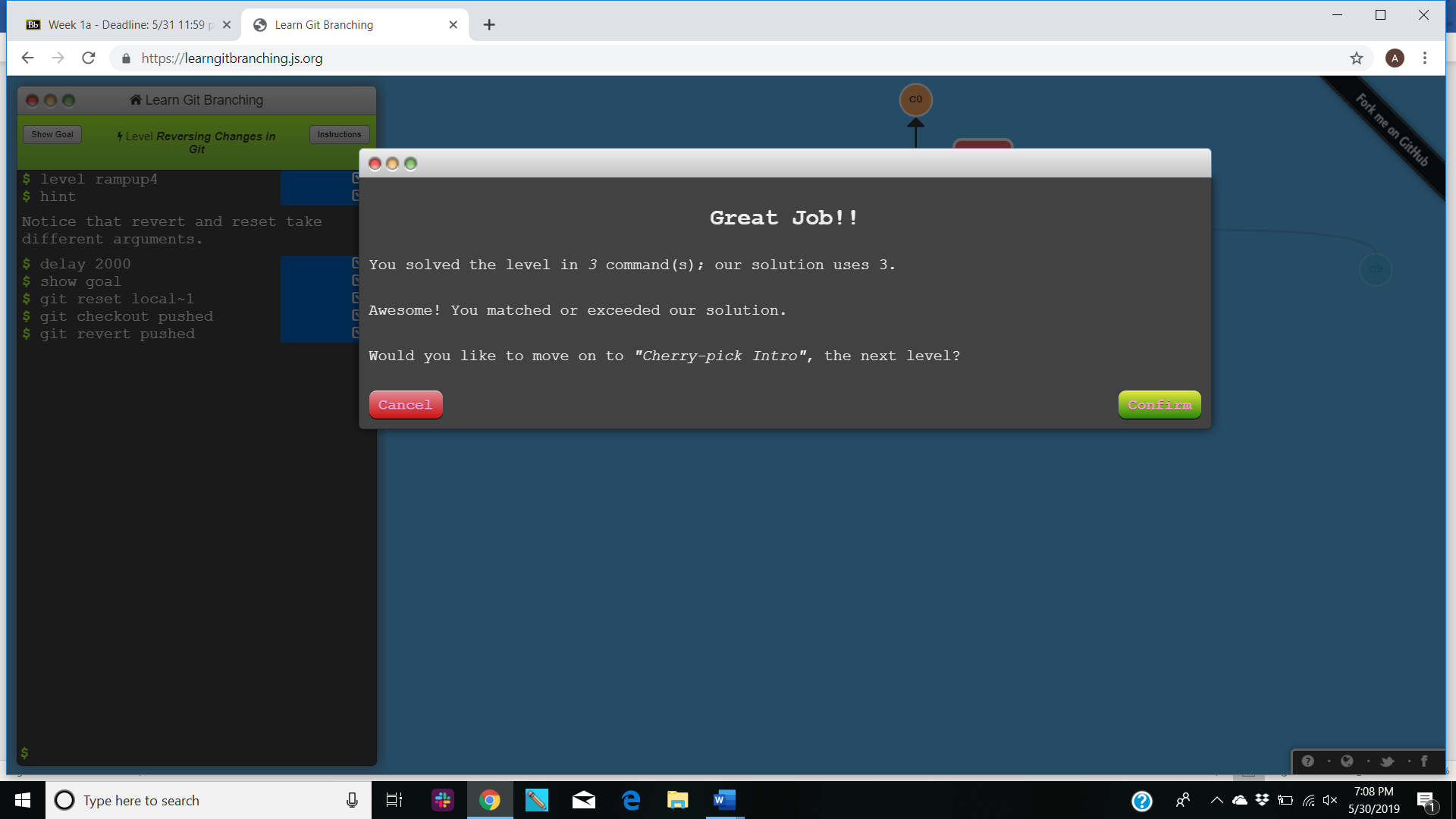
1. **Relative Ref 2-**

**Commands-** git branch and git checkout ^



1. **Reversing Changes In Git-**

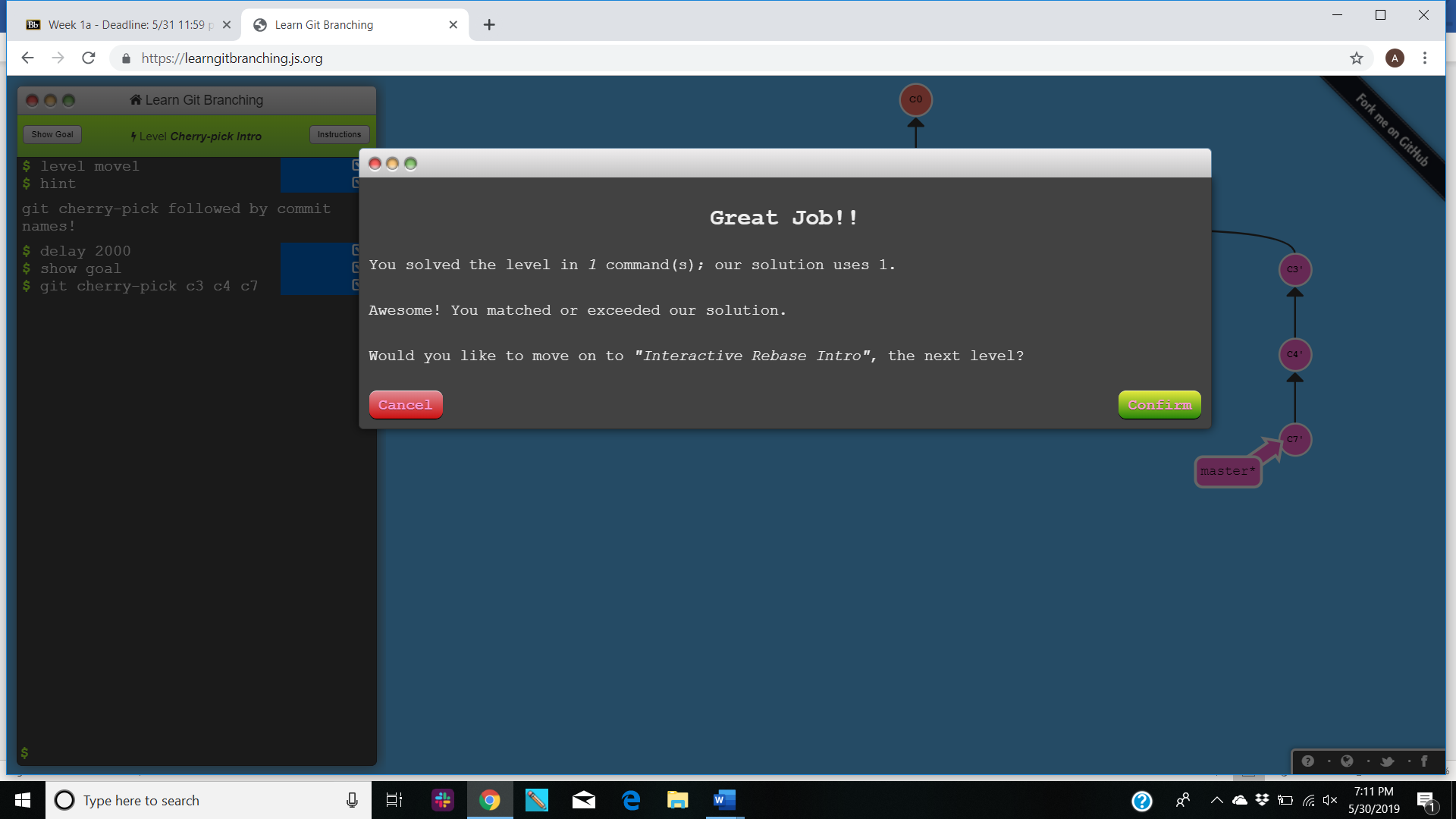
**Commands-** git reset, git checkout and git revert



**MOVING WORK AROUND-**

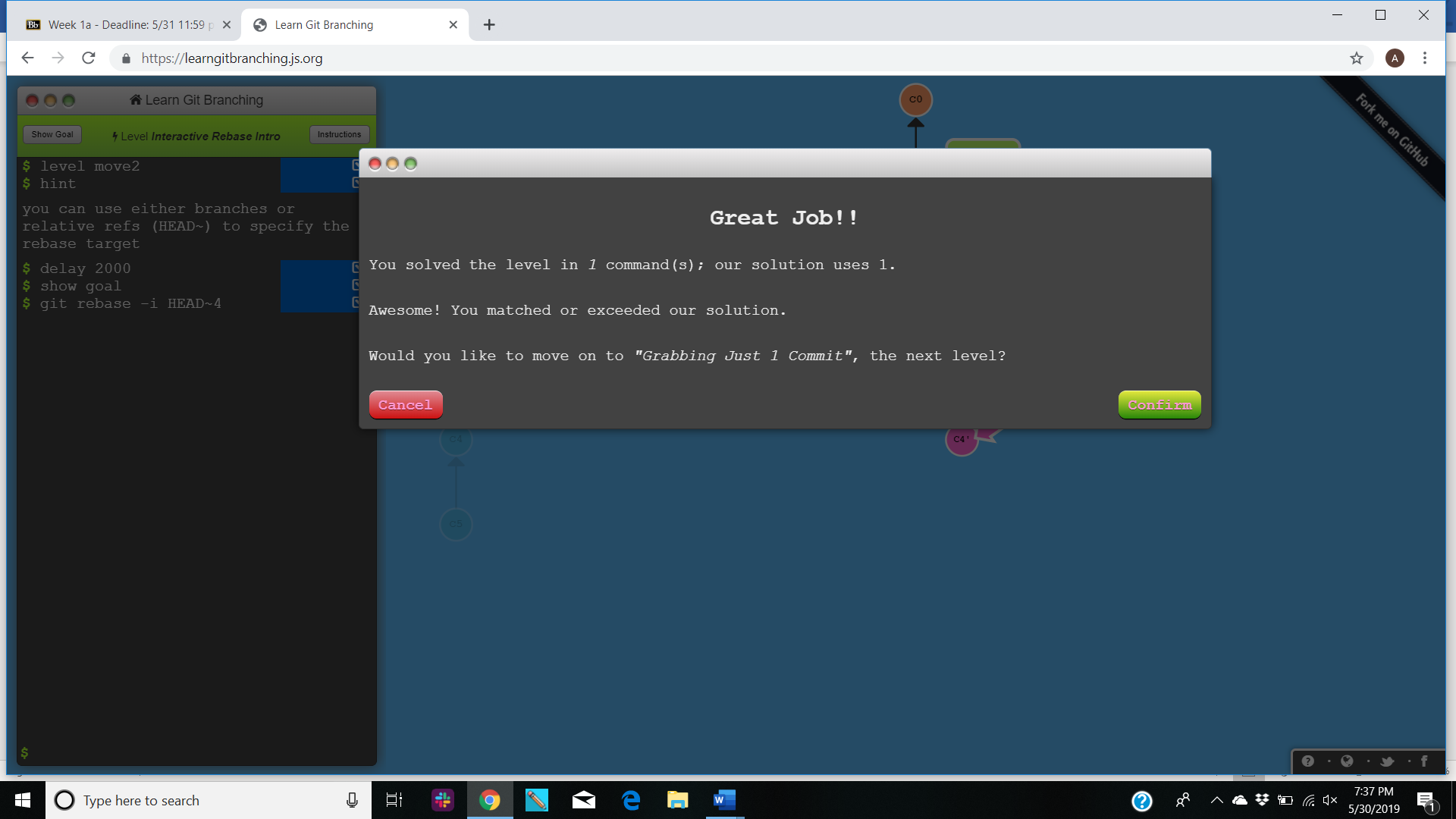
1. **Cherry-pick Intro-**

**Commands-** git cherry-pick



1. **Interactive Rebase-**

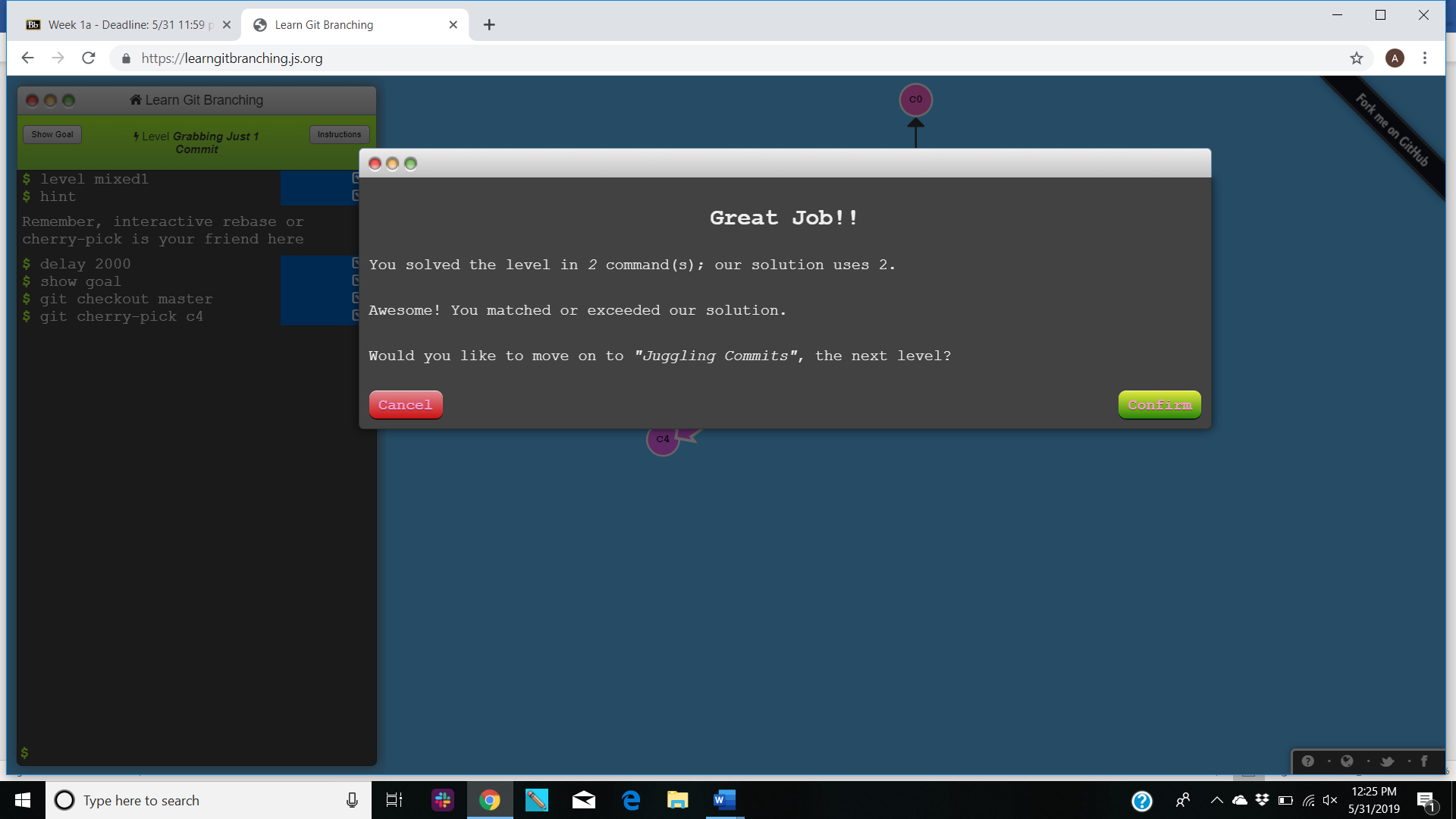
**Commands-** git rebase -i HEAD~4



**A MIXED BAG-**

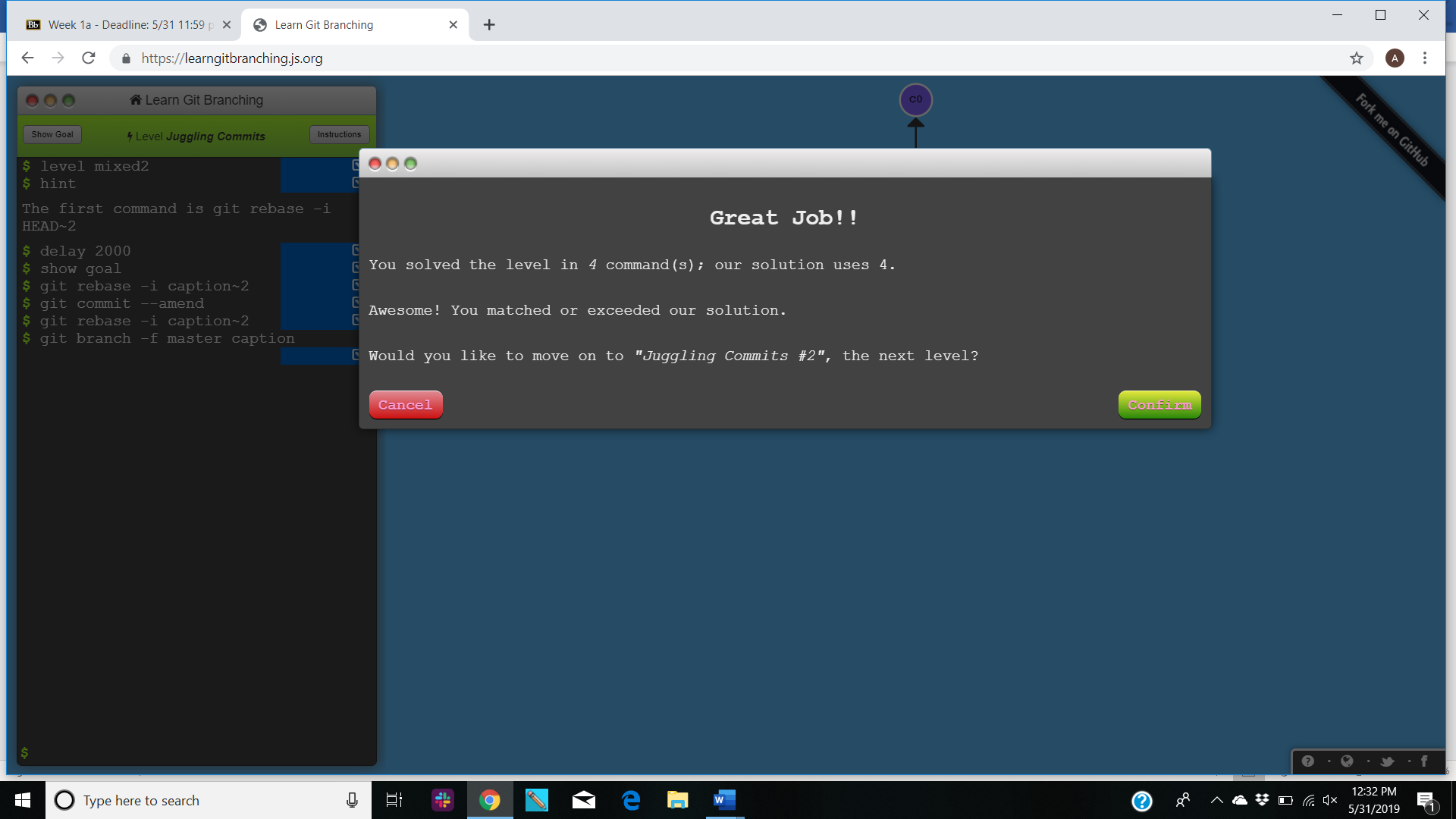
1. **Grabbing just one commit-**

**Commands-** git checkout and git cherry-pick



1. **Juggling commits-**

**Commands-** git rebase, git commit, git branch



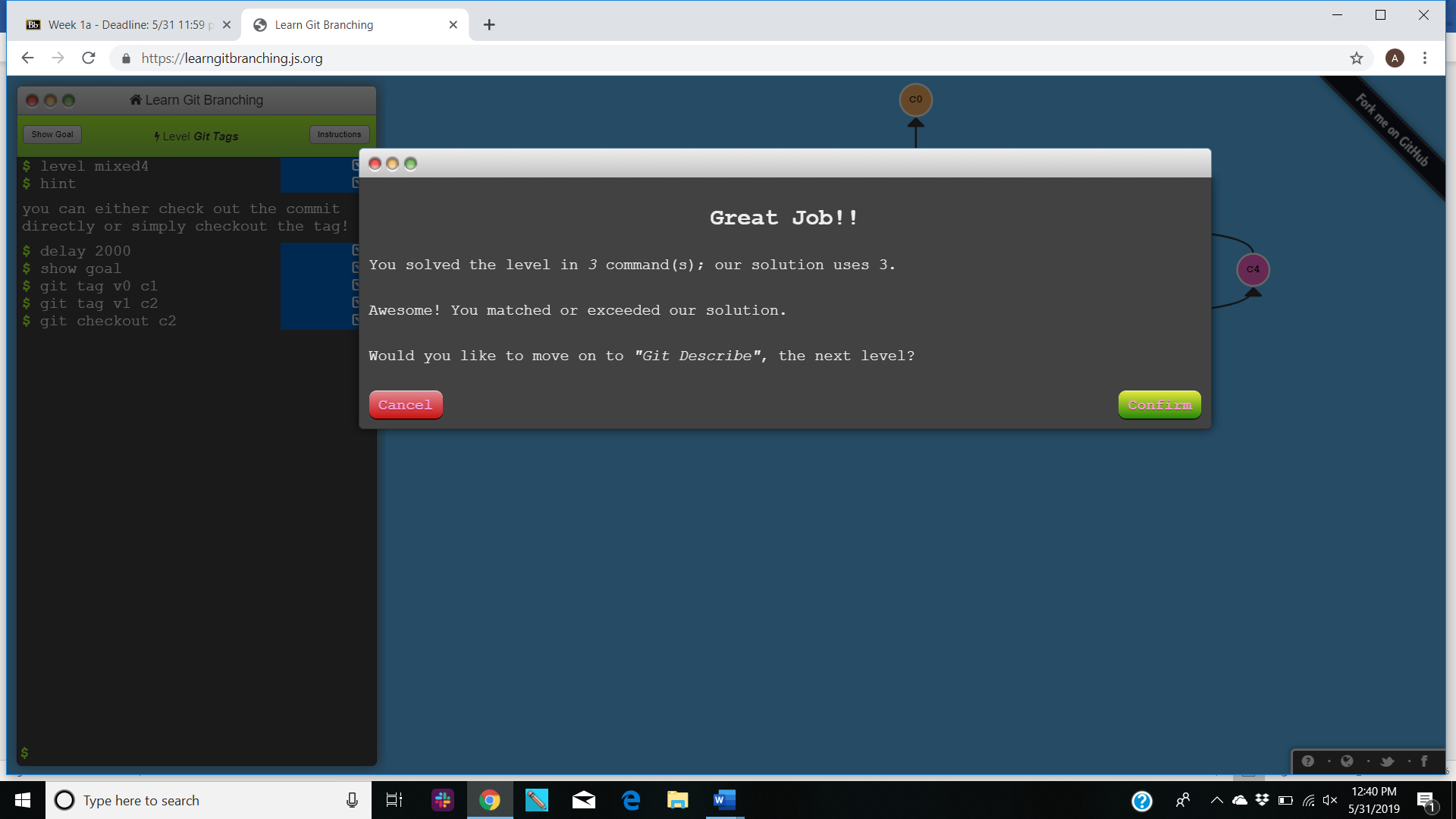
1. **Juggling commits 2-**

**Commands-** git checkout, git cherry-pick, git commit



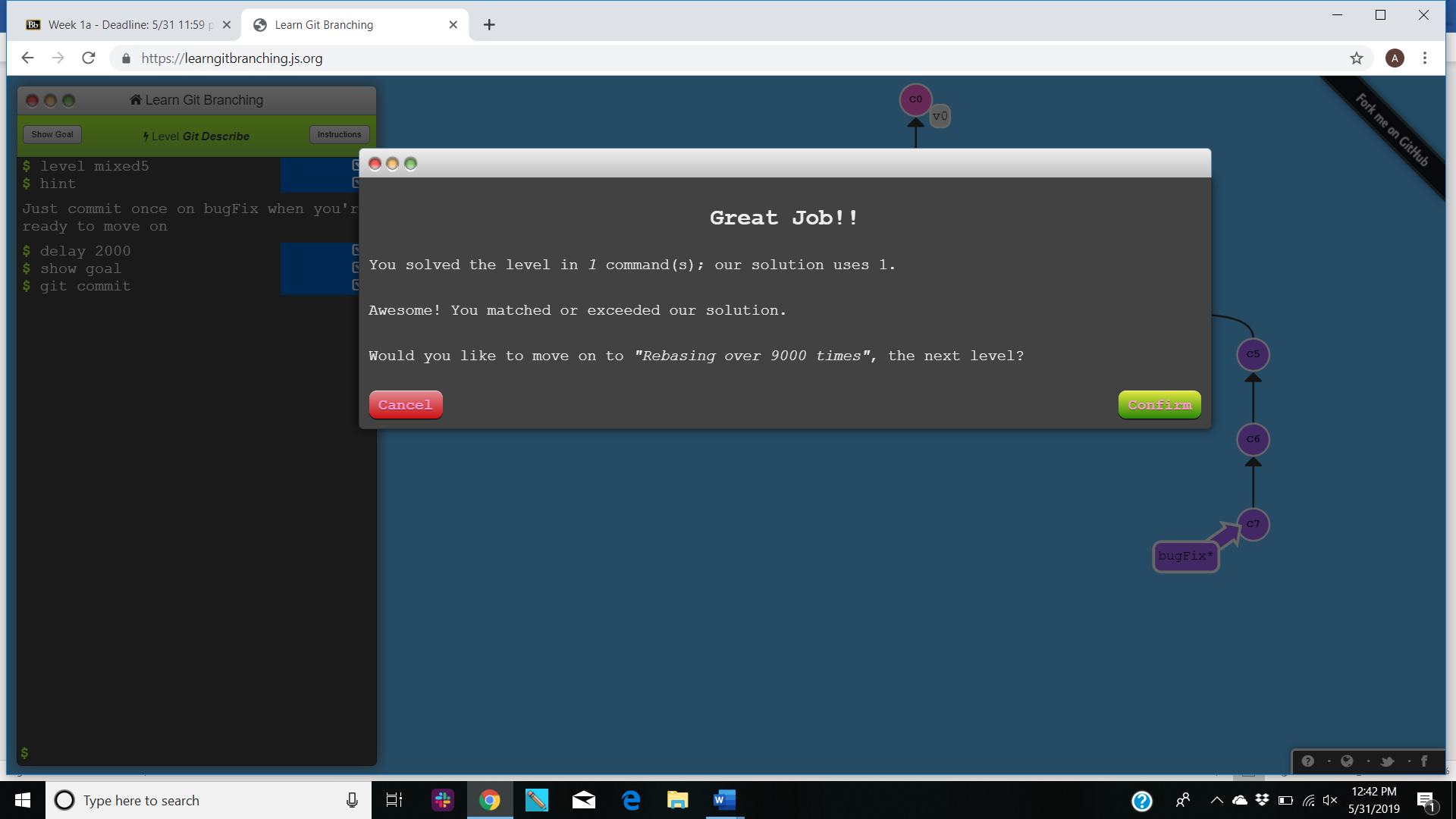
1. **Git tags-**

**Commands-** git tag, git checkout



1. **Git describe-**

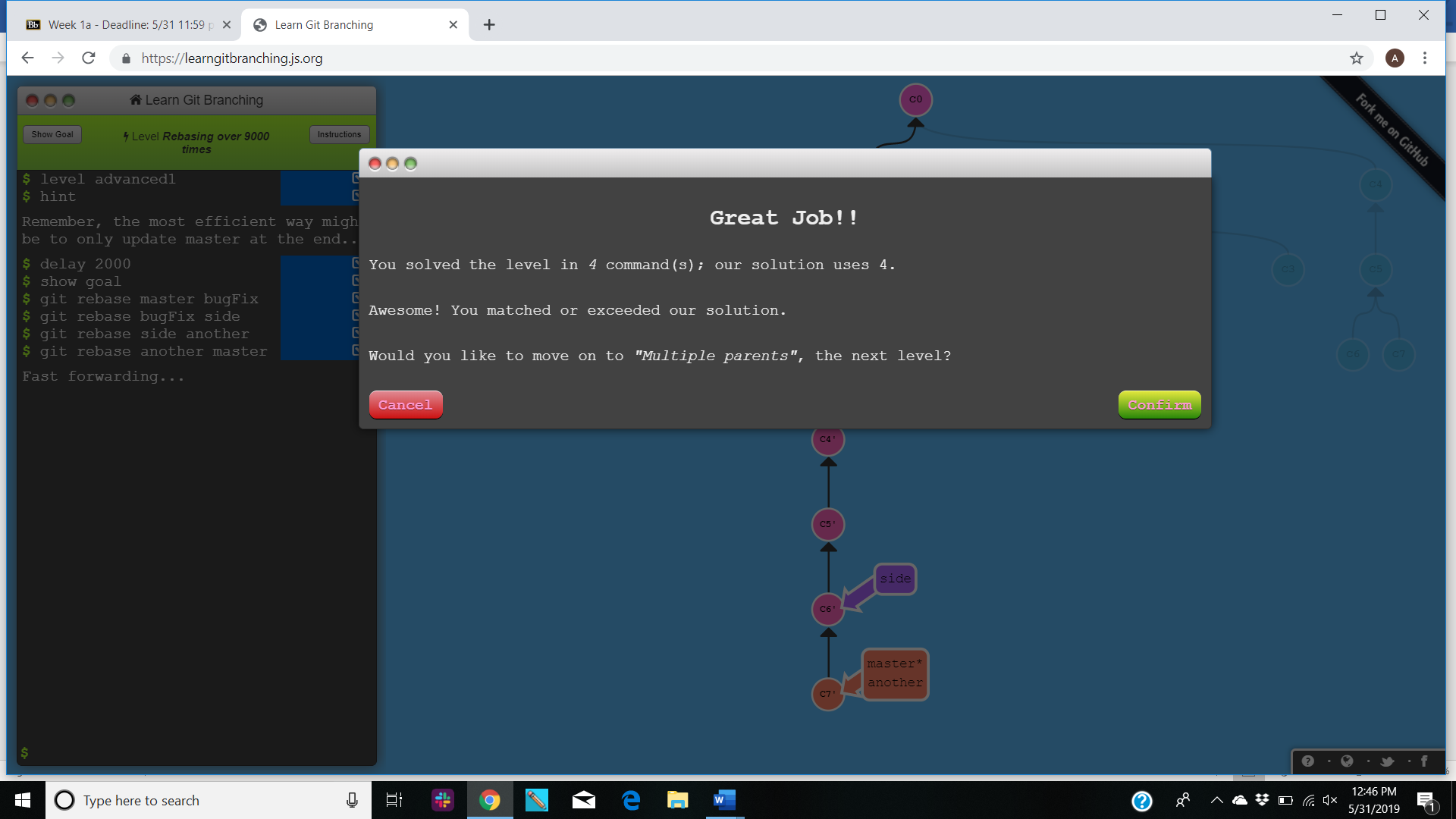
**Commands-** git commit



**ADVANCED TOPICS-**

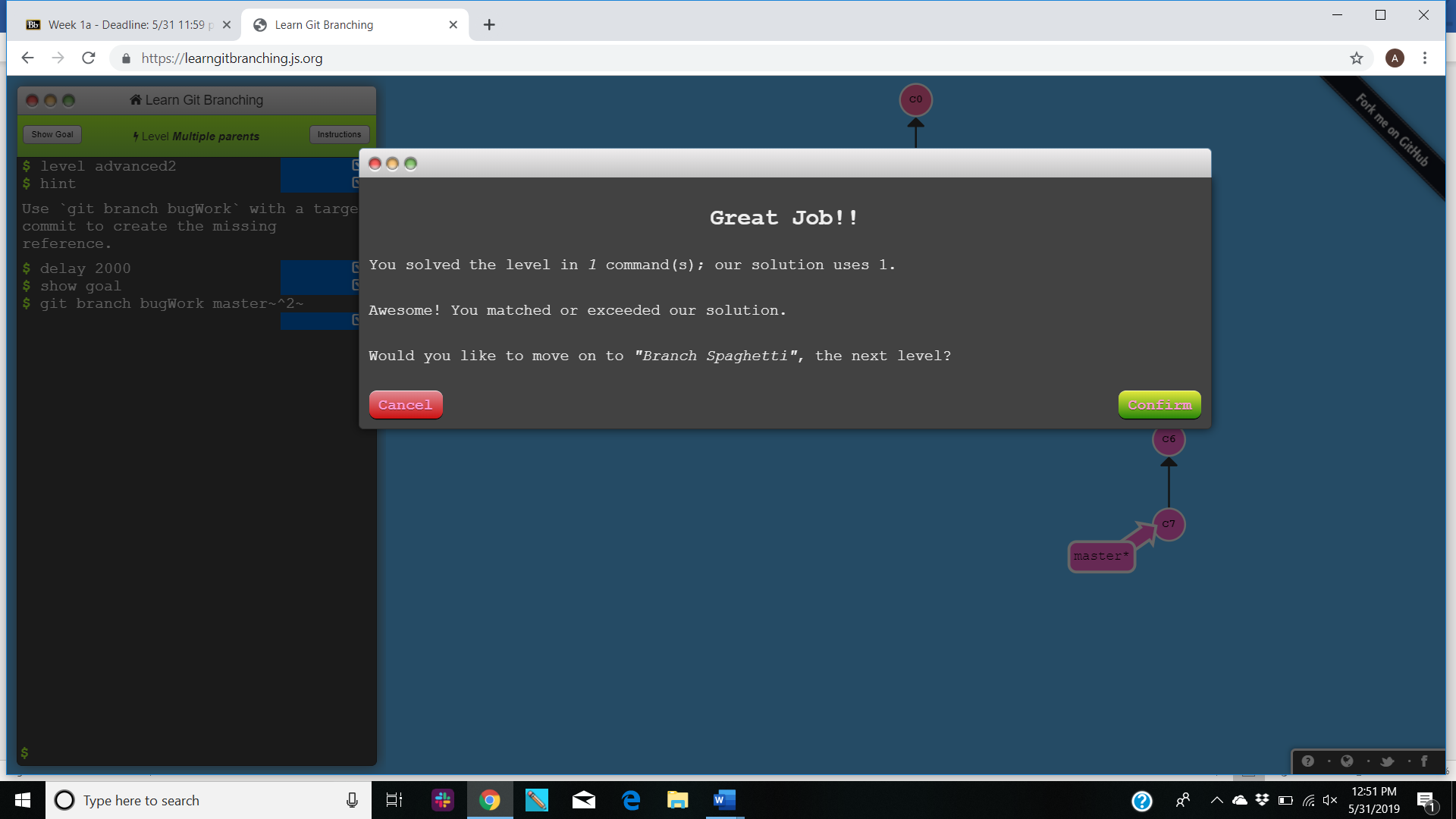
1. **Rebasing over 9000 times-**

**Commands-** git rebase



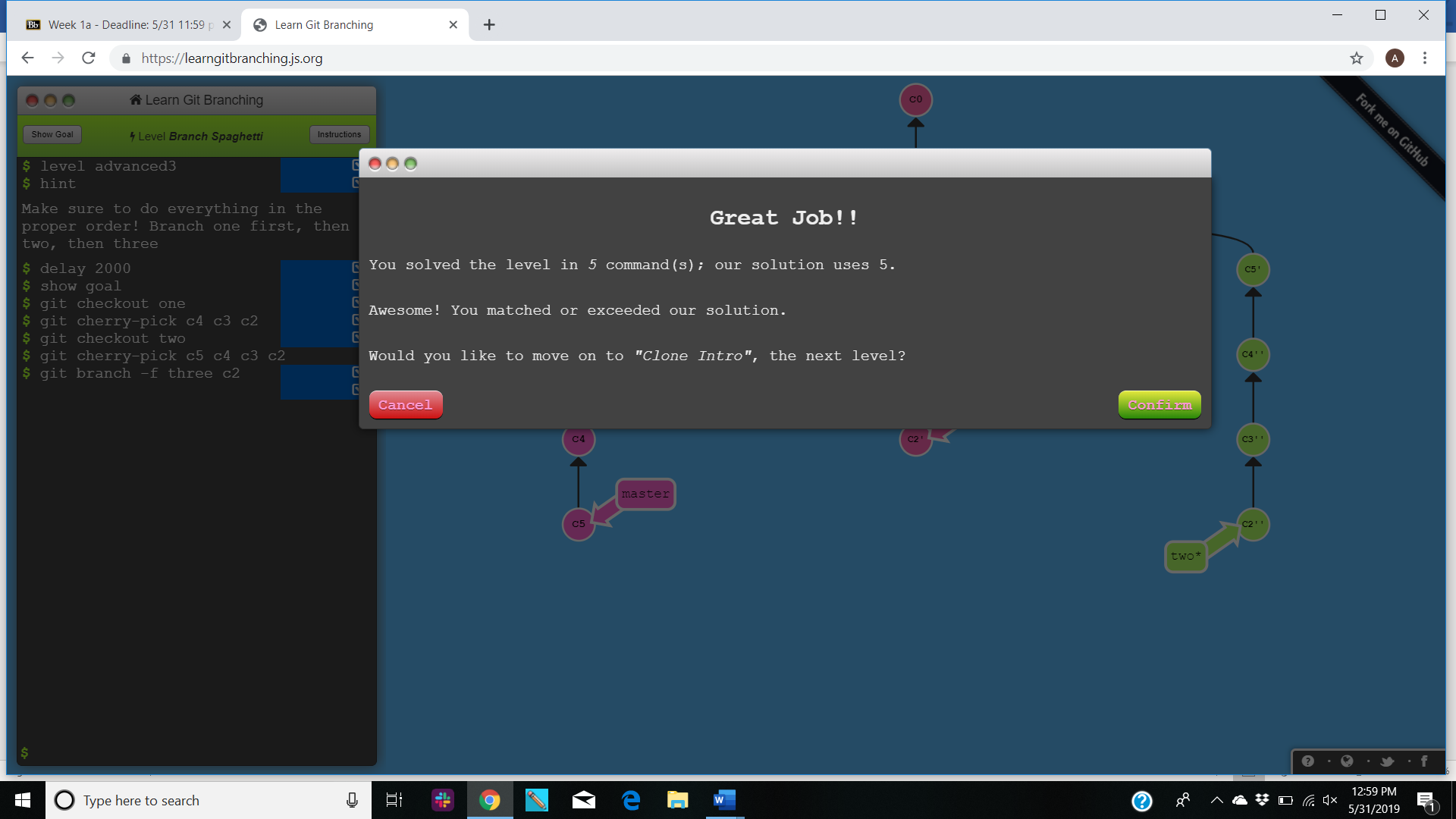
1. **Specifying parents-**

**Commands-** git branch



1. **Branch spaghetti-**

**Commands-** git checkout, git cherry-pick, git branch



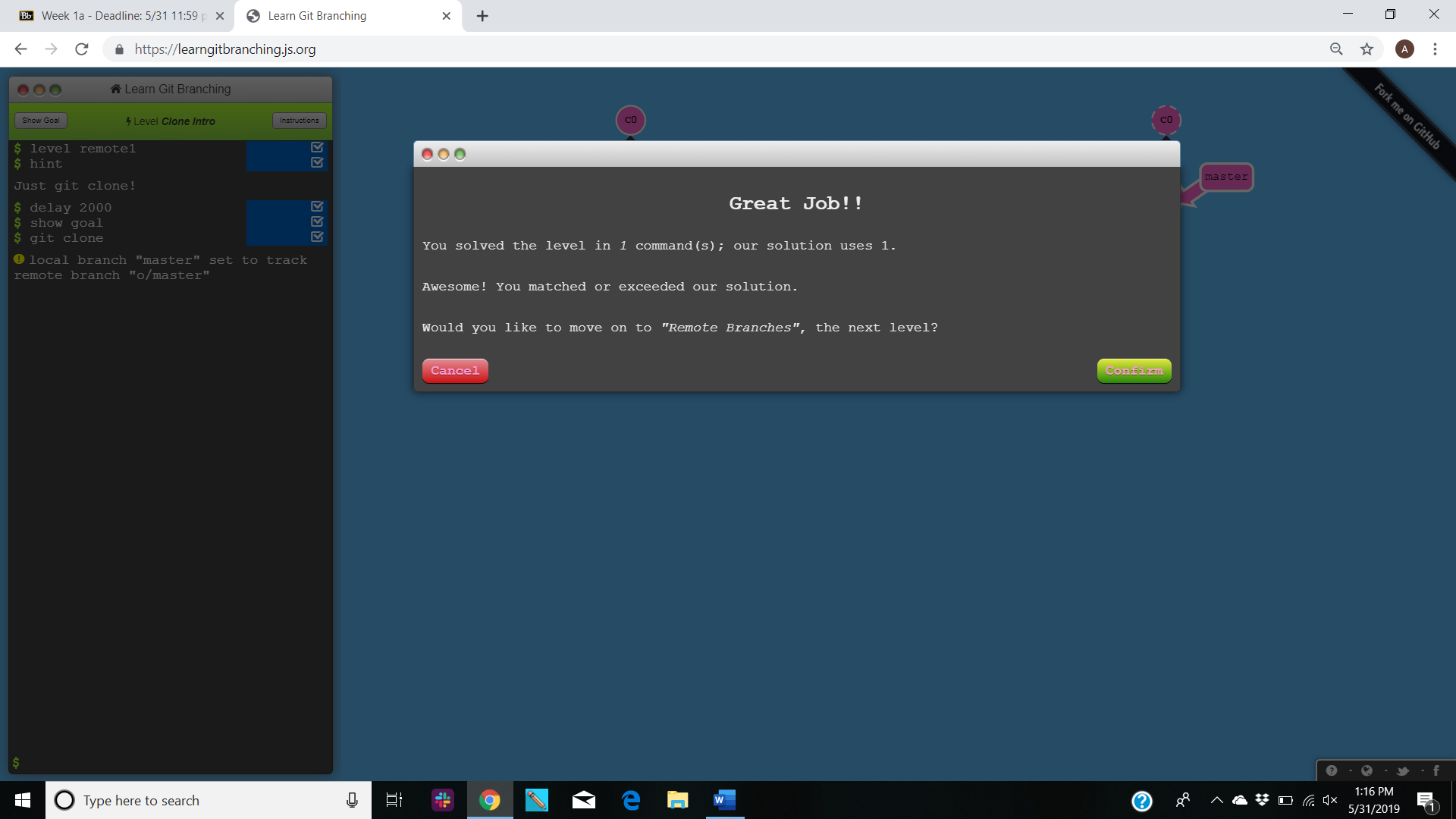
**COMPLETED LIST-**



**PUSH & PULL –GIT REMOTES-**

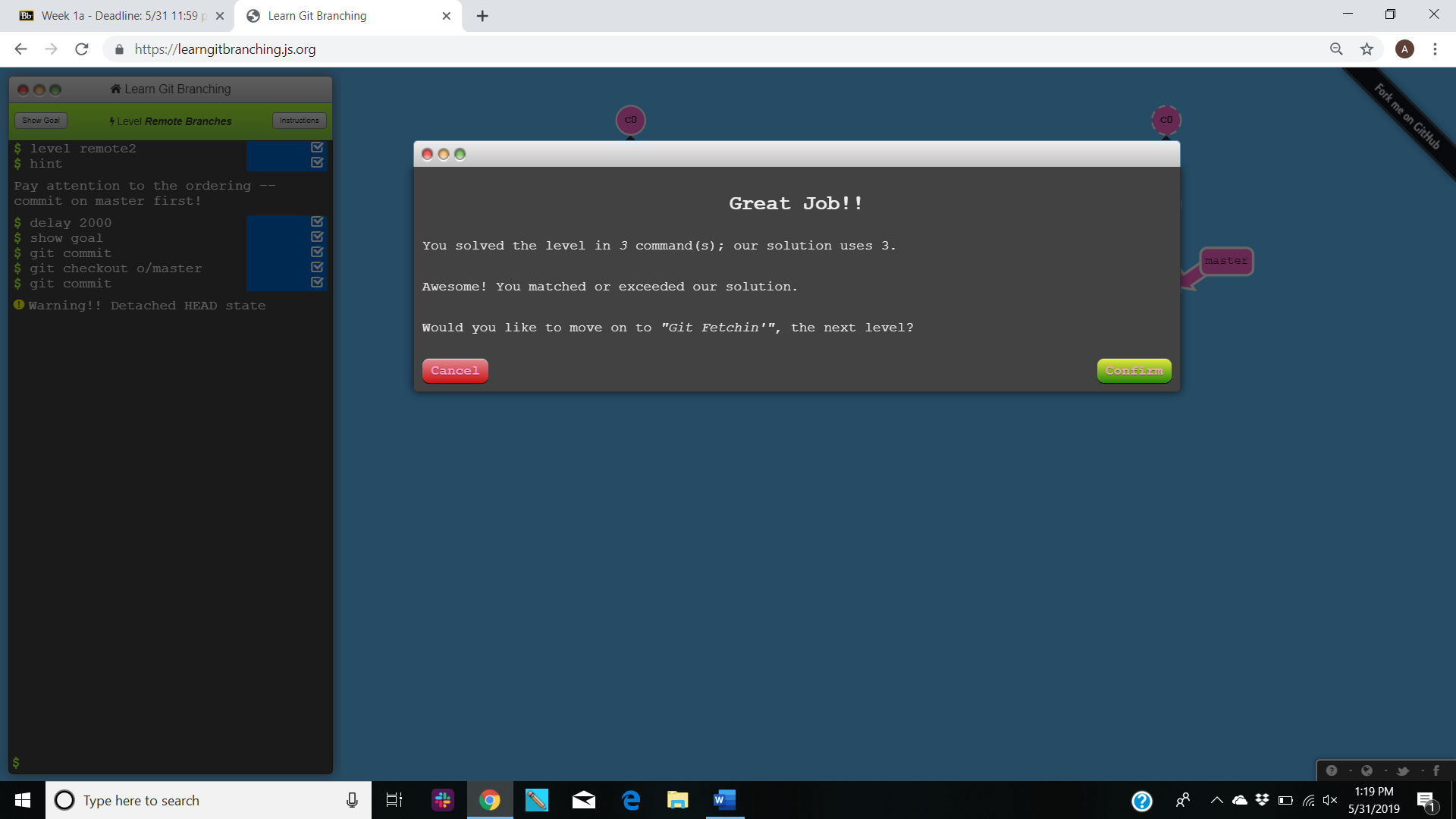
1. **Clone intro-**

**Commands-** git clone



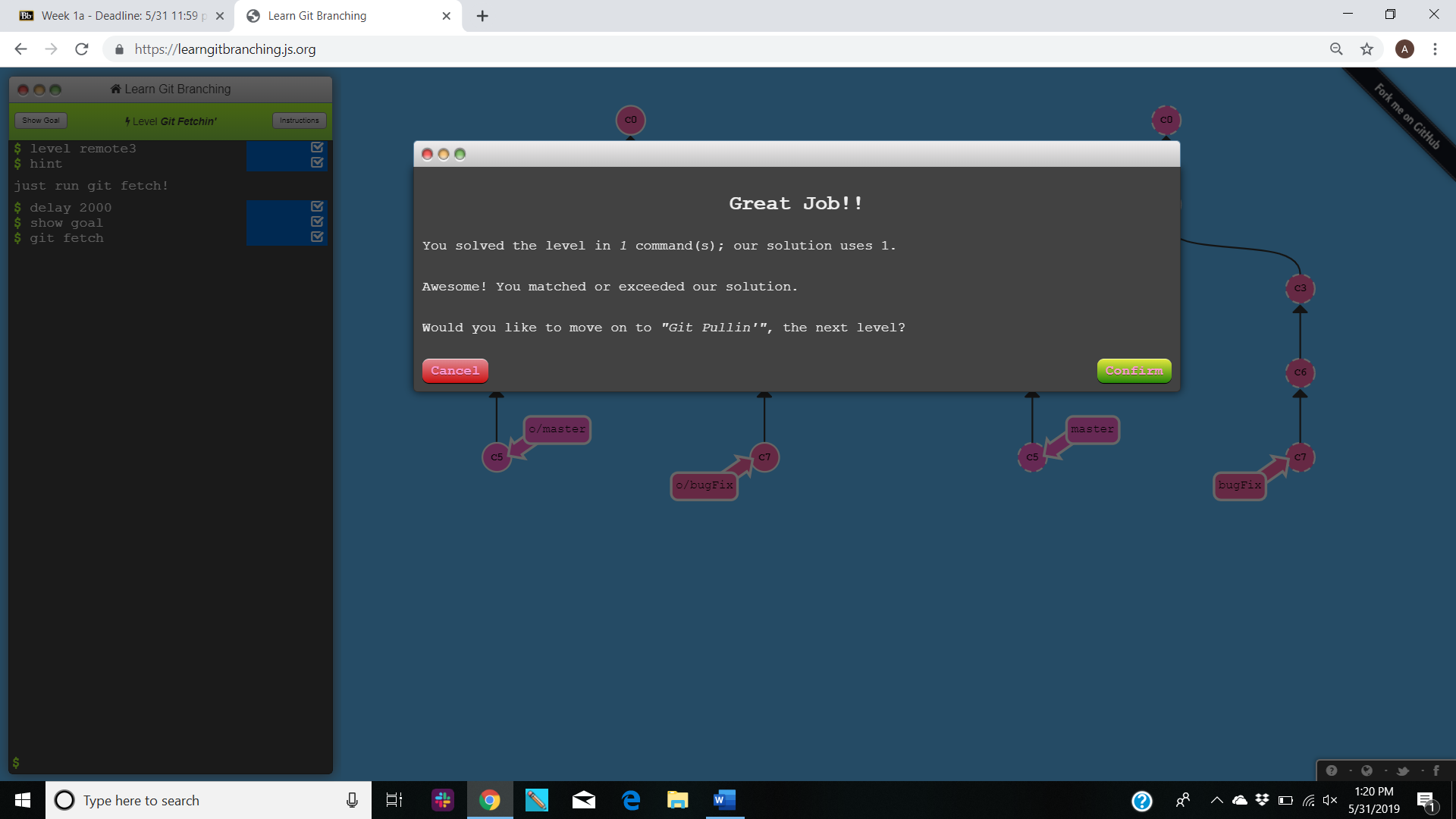
1. **Git remote branches-**

**Commands-** git commit, git checkout o/master



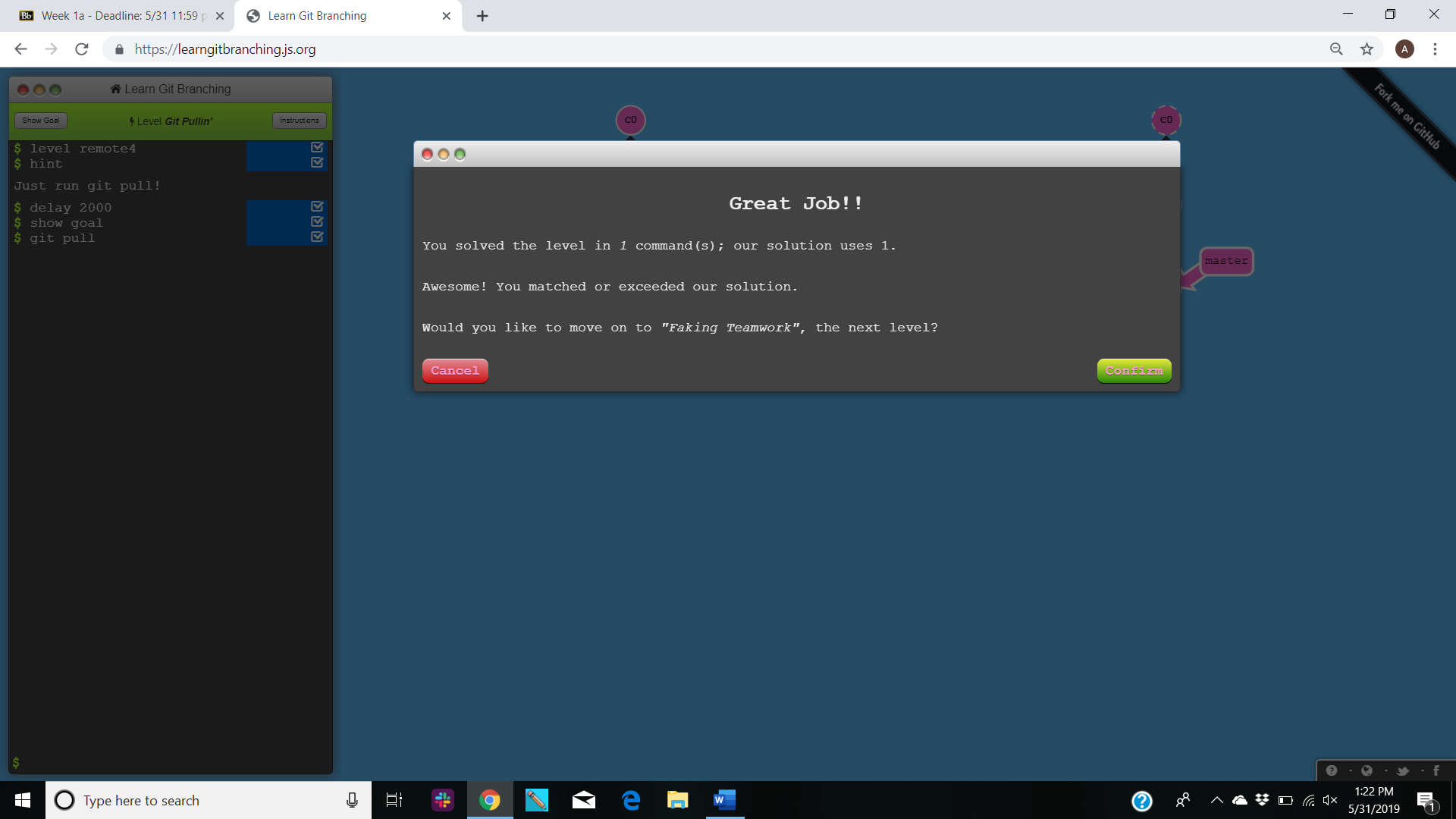
1. **Git fetch-**

**Commands-** git fetch



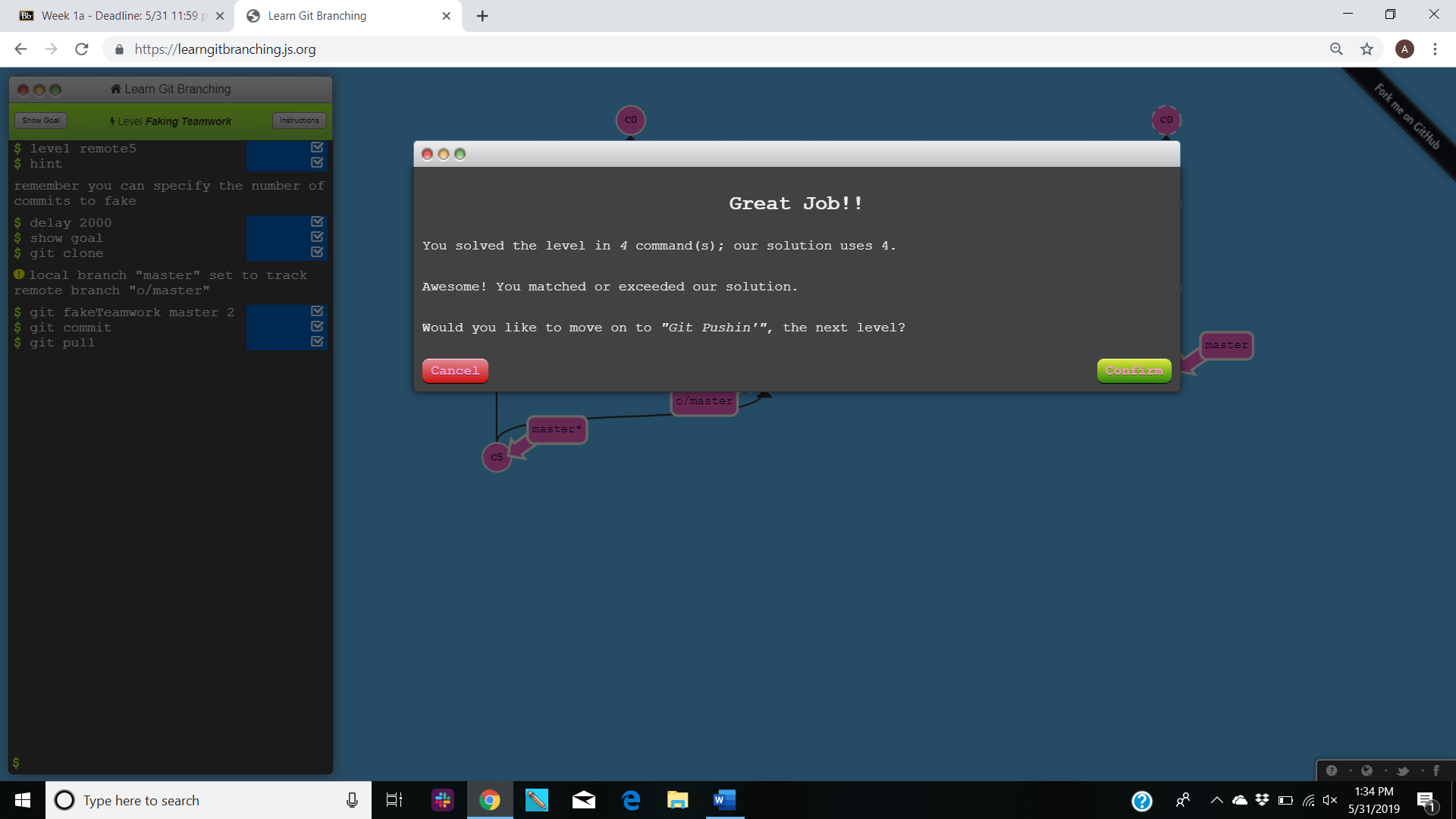
1. **Git pull-**

**Commands-**git pull (instead of git fetch and merge)



1. **Simulating collaboration-**

**Commands- git clone, git fakeTeamwork, git commit, git pull**



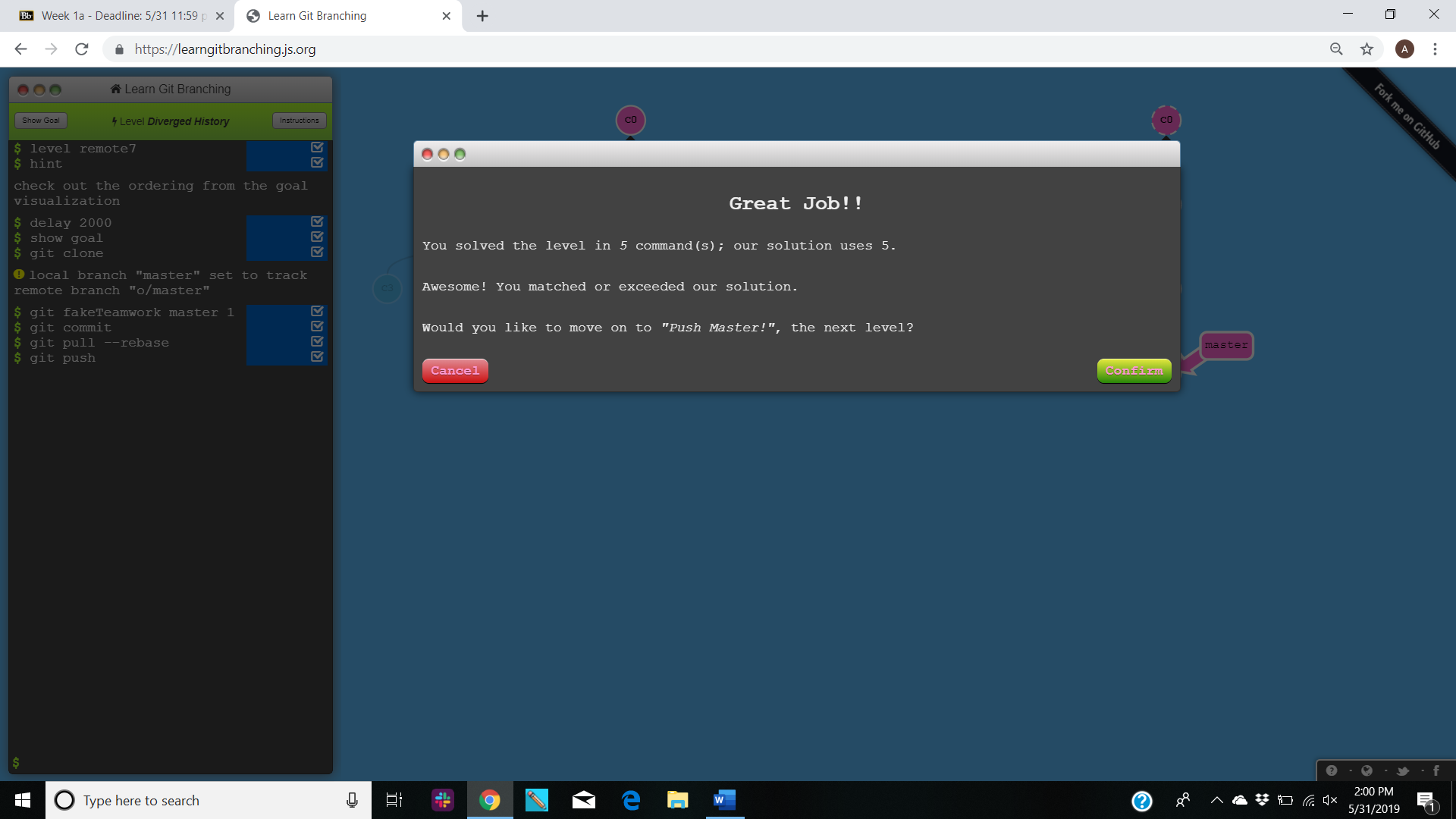
1. **Git push-**

**Commands-** git push



1. **Diverged work-**

**Commands-** git clone, git fakeTeamwork, git pull –rebase, git push



**PART 5**

Define the following terms in the context of Git (2 lines maximum):

1. **Repository-** A digital directory or storage space where you can access your project, its files, and all the versions of its files that Git saves.
2. **Commit-** It is an individual change to a file or set of files. It's like when you save a file, every time you save it creates a unique ID that allows you to keep record of what changes were made, when and by who.
3. **Push-** It refers to sending your committed changes to a remote repository.
4. **Branch-** It is essentially a unique set of code changes with a unique name. Each repository can have one or more branches.
5. **Fork-** It is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
6. **Merge-** It is a way of putting a forked history back together again. The mergecommand lets you take the independent lines of development created by git branch and integrate them into a single branch.
7. **Clone-** It is a command line utility which is used to target an existing repository and create a clone or copy of the target repository.
8. **Pull-** Thiscommand is used to fetch and download content from a remote repository and immediately update the local repository to match that content.
9. **Pull Request-** It lets you tell others about changes you've pushed to a branch in a repository. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

**PART 7**

The commands and strategy that I used to update the README.md are as follows-

1. I opened the GitHub URL provided in the browser.
2. I navigated and opened the README.md file.
3. I **forked** the repository to edit the file and inserted my name along with the date and time as a comment.
4. I **committed** the changes.
5. I created a **pull request** for my forked repository to be **merged** with the master branch.

Commands used to upload file-

1. First, I created the file with the contents and named It ChoudharyAmritGitTutorial-05-31-2019.
2. I created a repository in my GitHub account by the name CS6392019.
3. I used command prompt and cloned the repository with my file using – git clone address.
4. I then checked the status using the command – git status.
5. Next, I added the files into my repository using- git add.
6. The file is then committed to be ready to be pushed to the repository using – git commit -m “message”
7. The file is finally pushed into the repository using- git push.