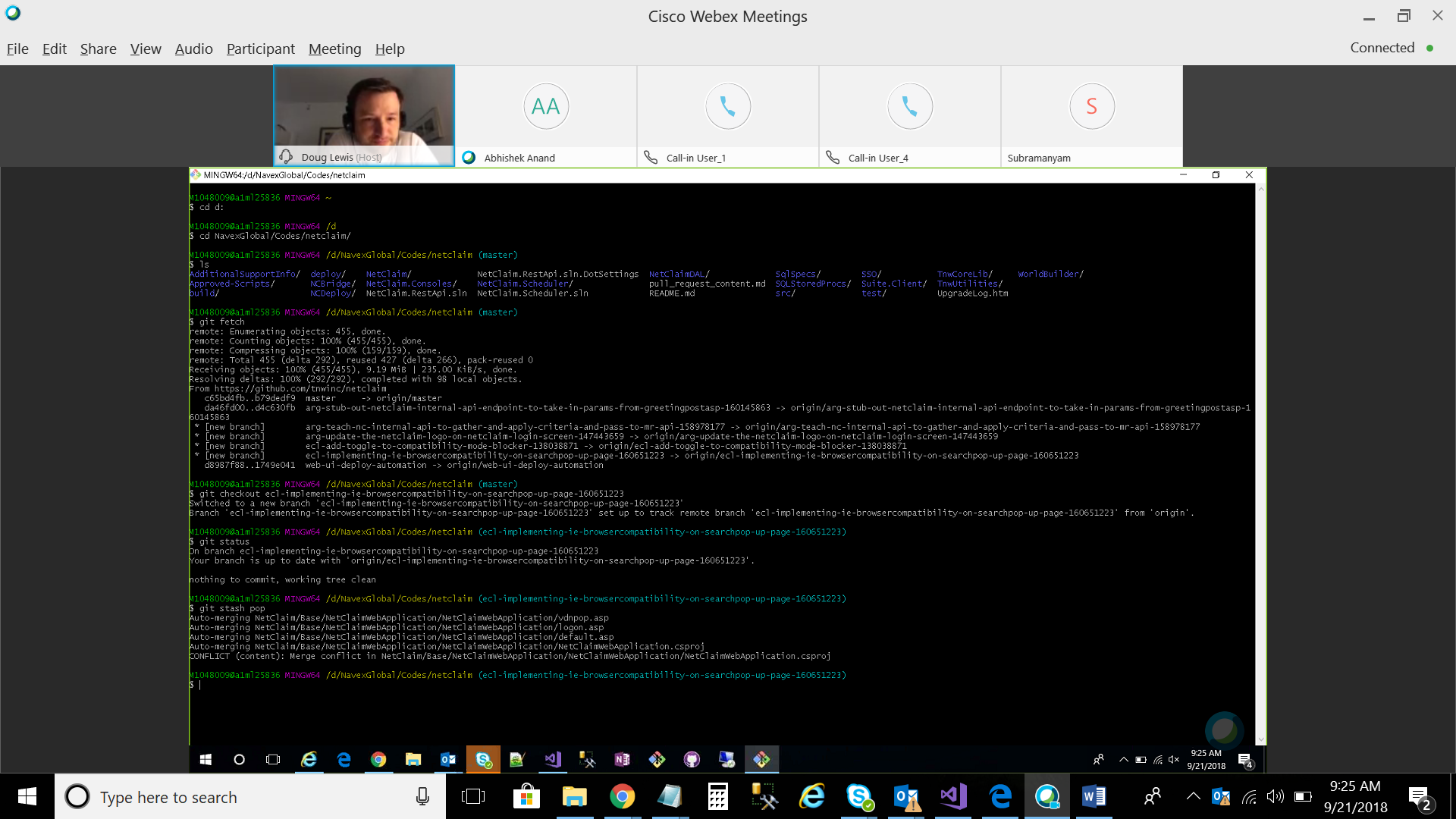
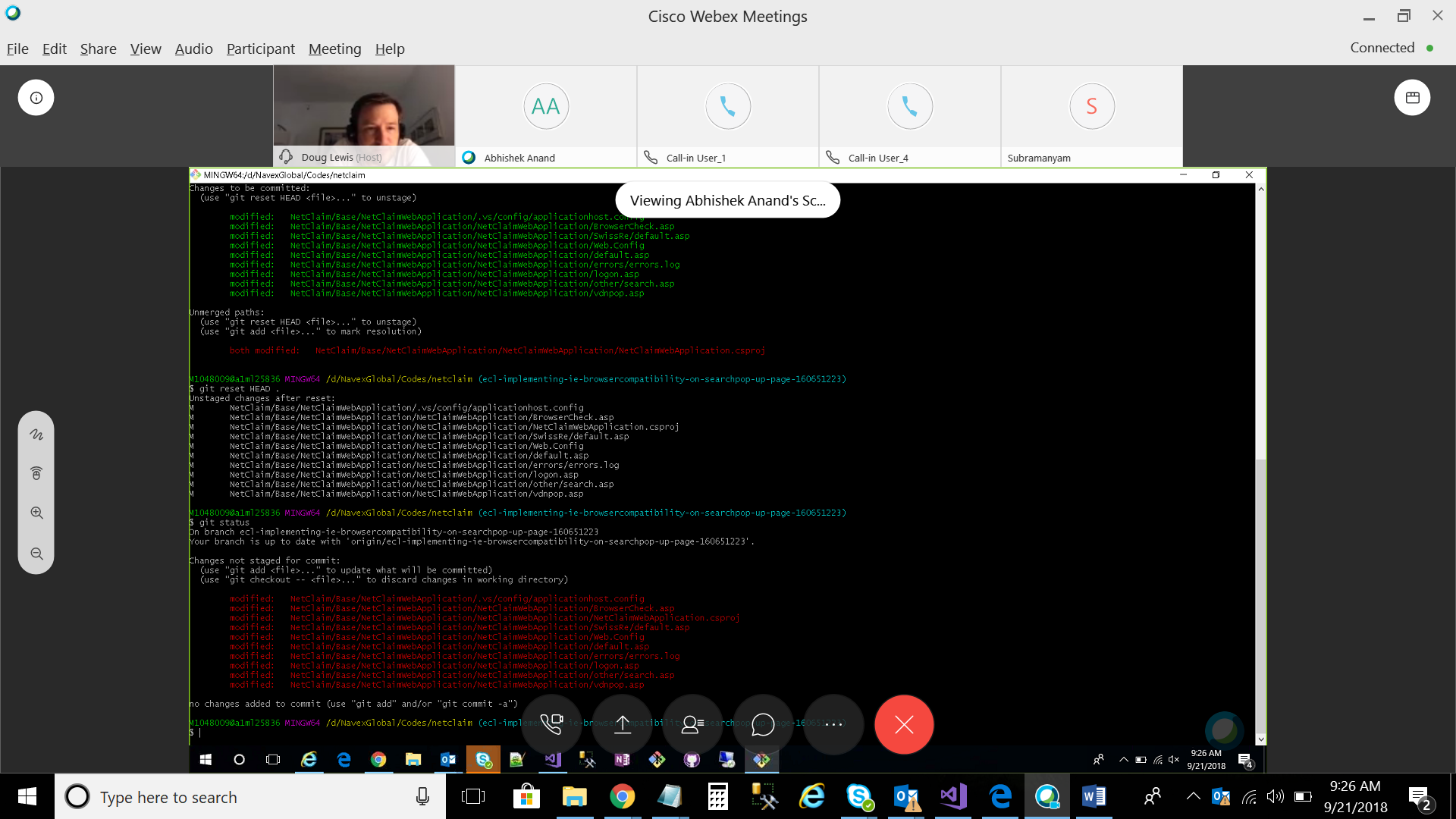
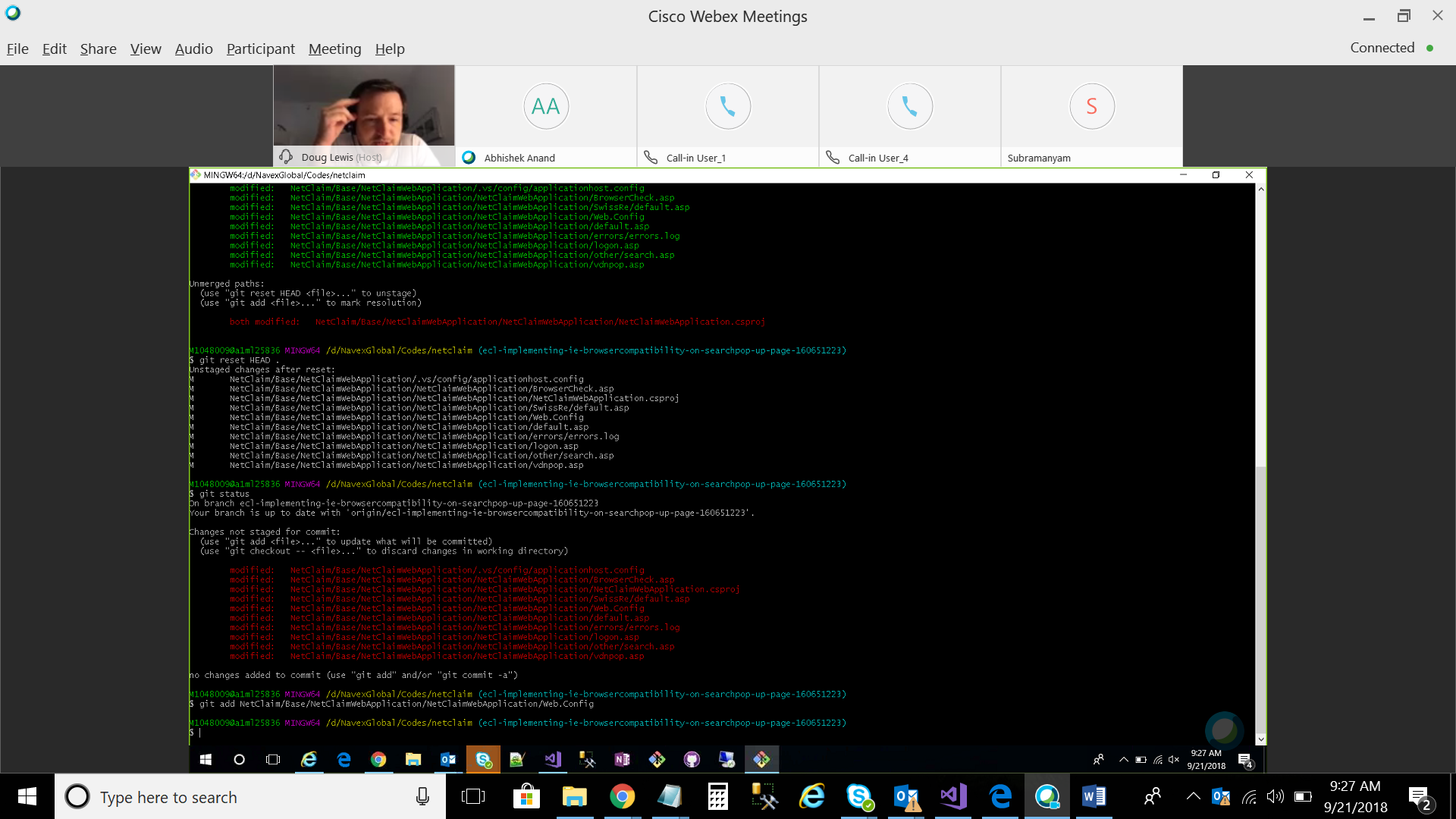
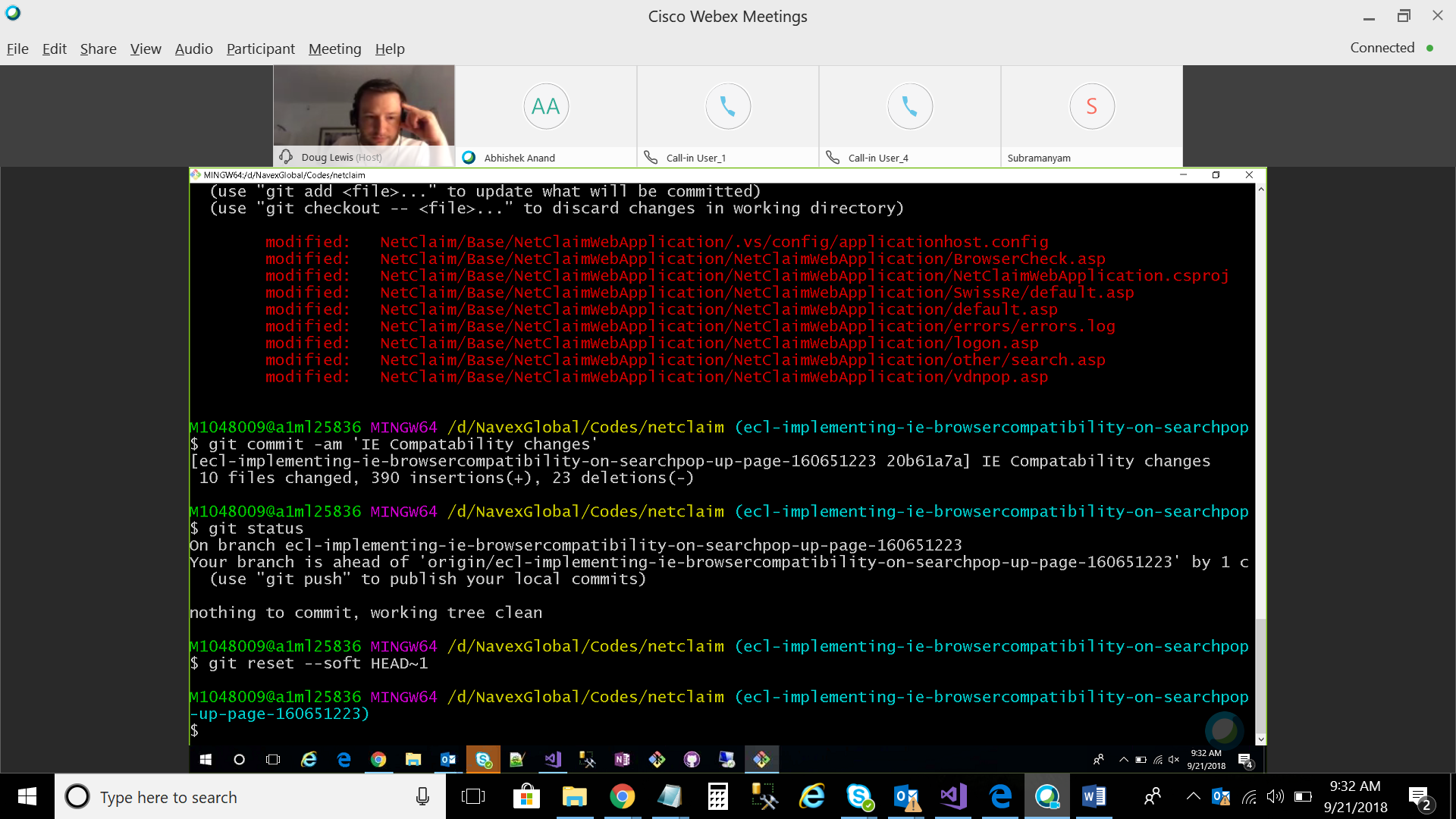
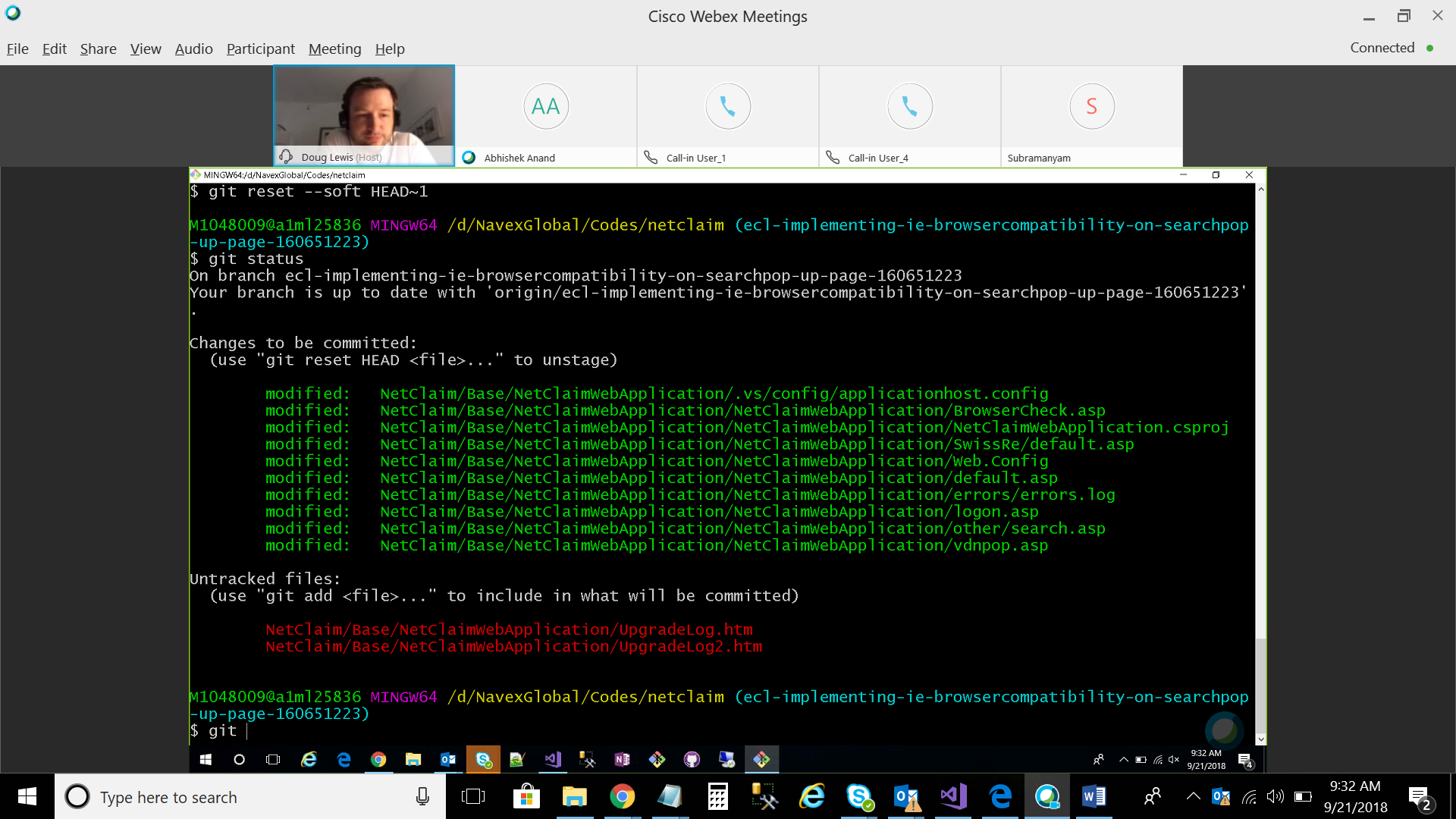
Tfs not work on branch git is good with merging stuff

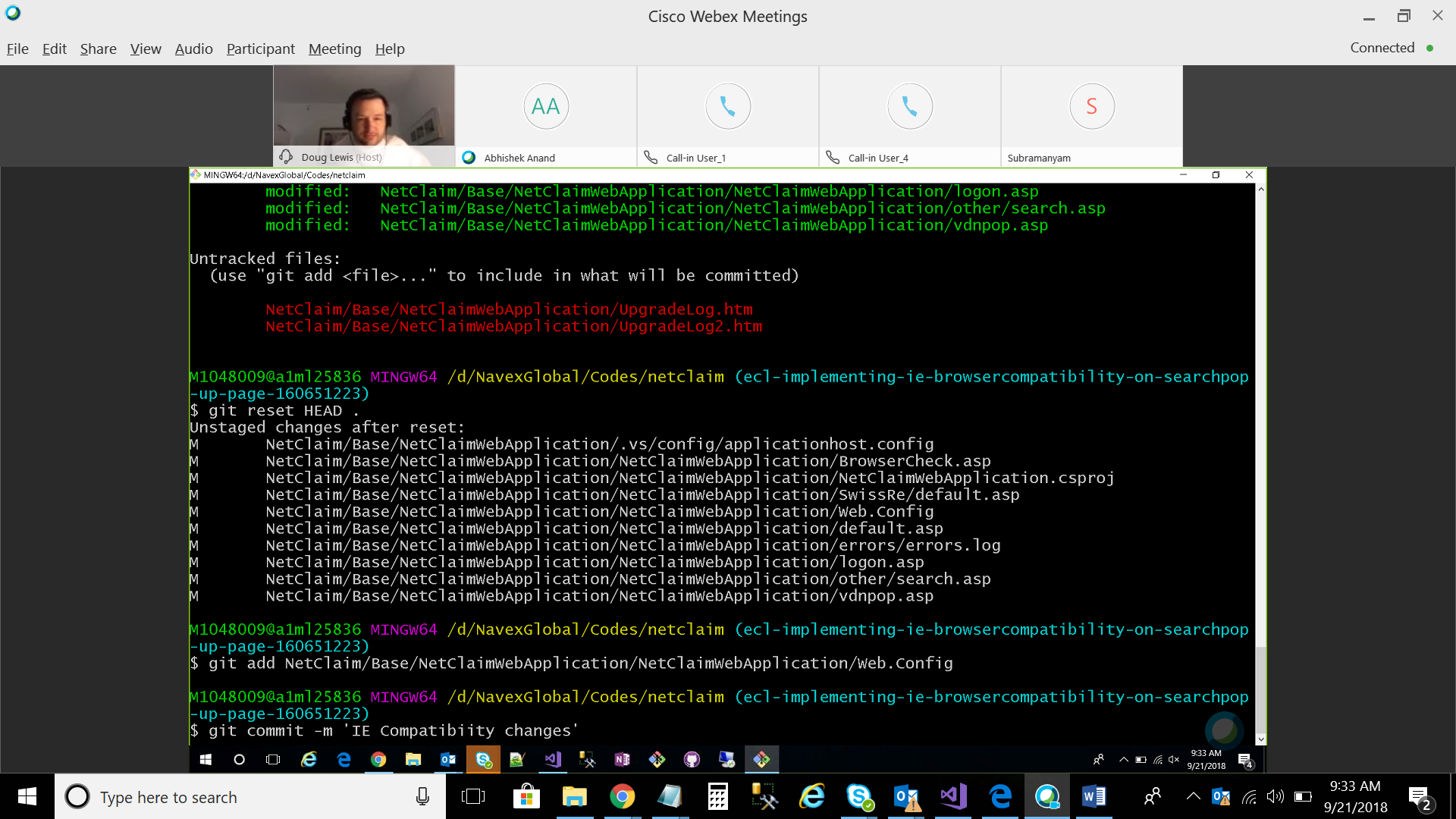


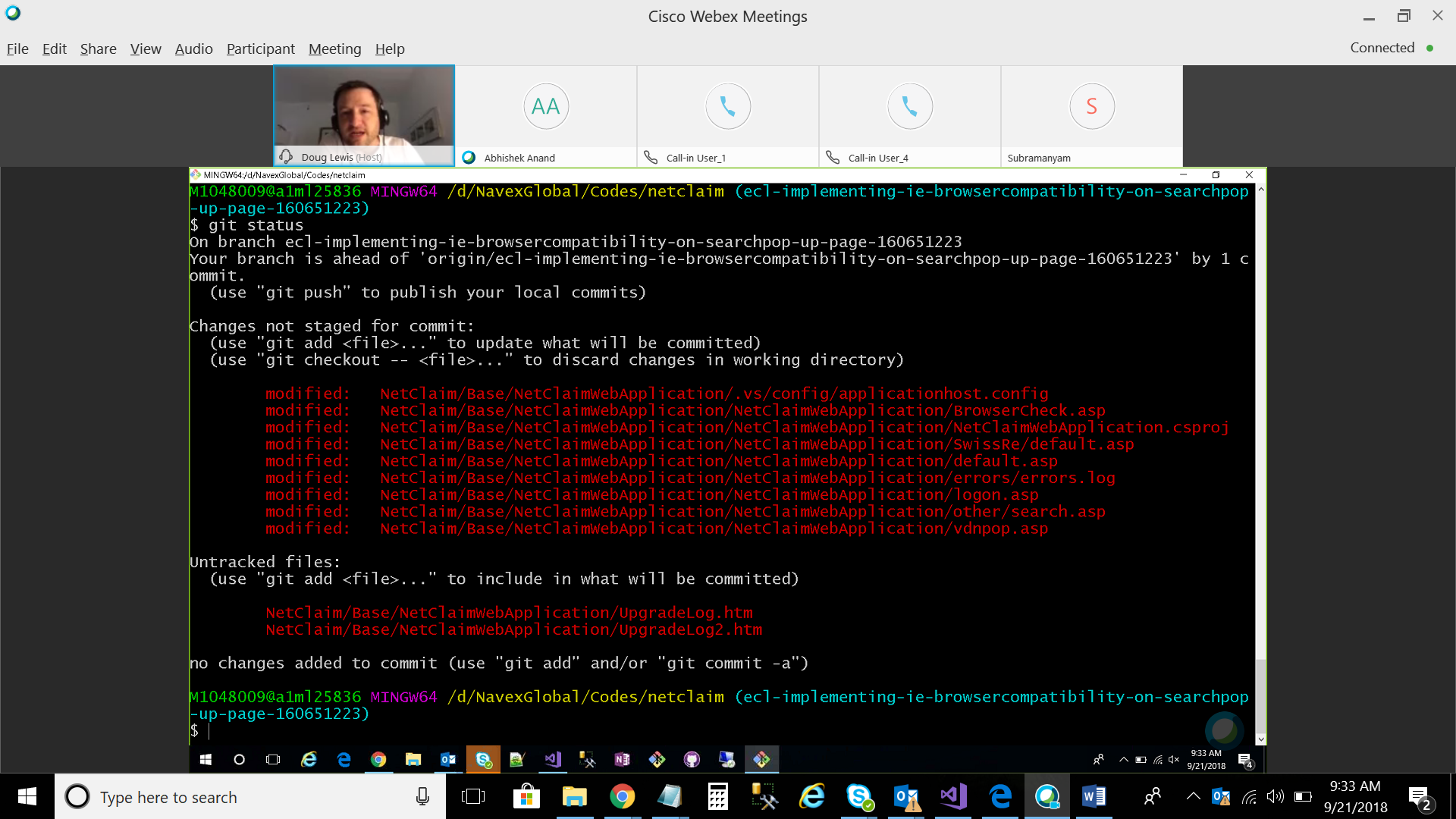


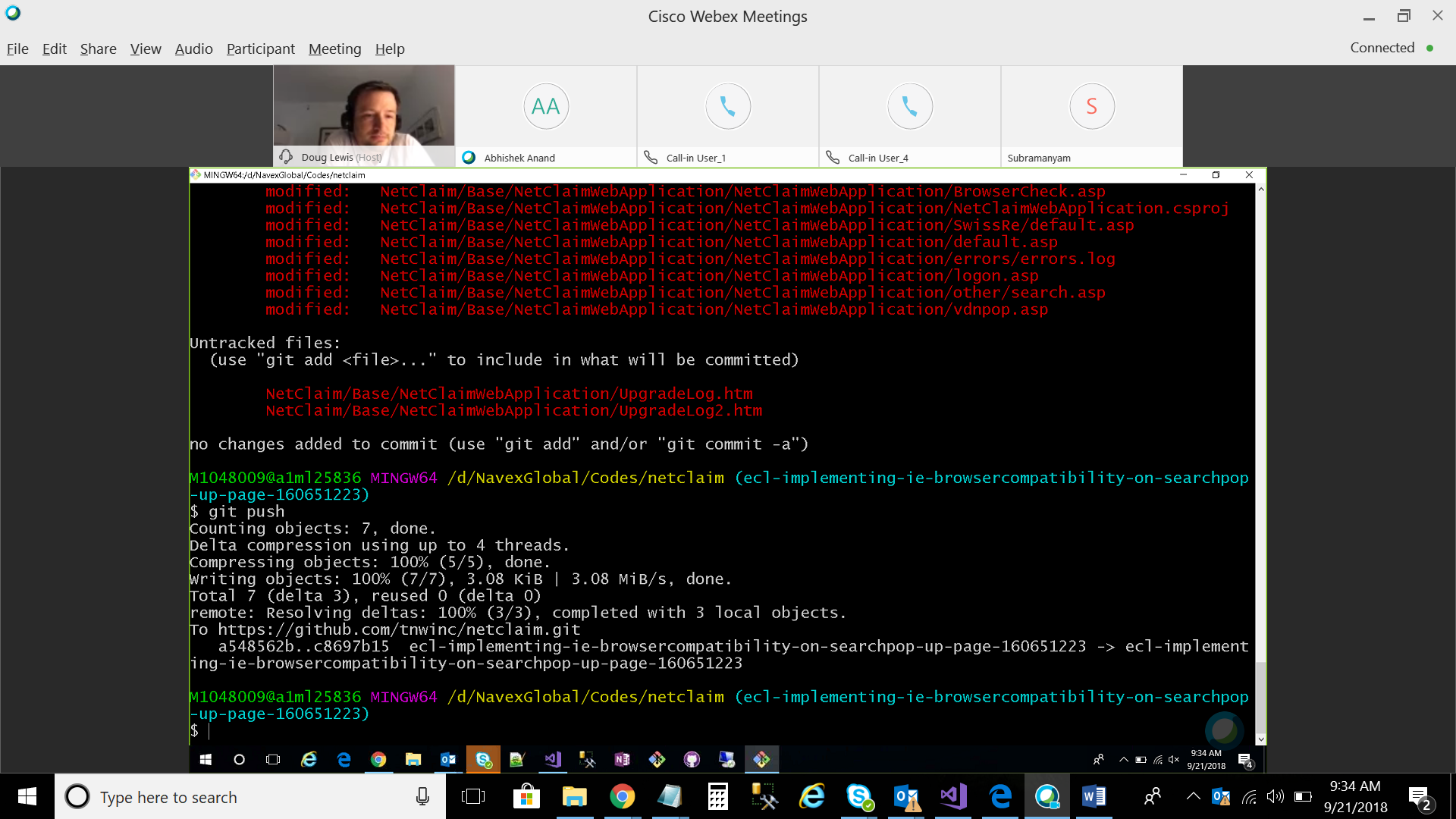


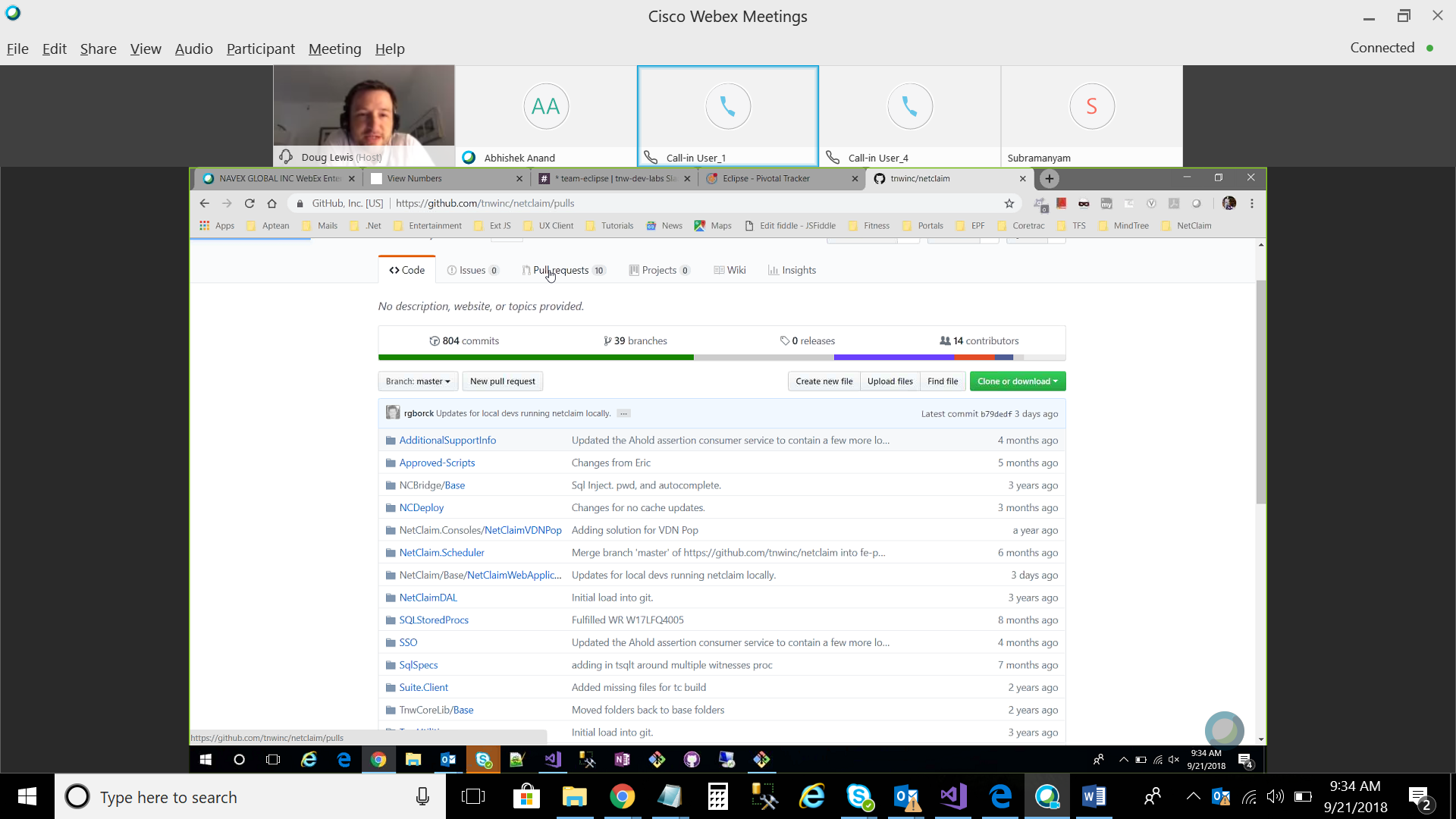


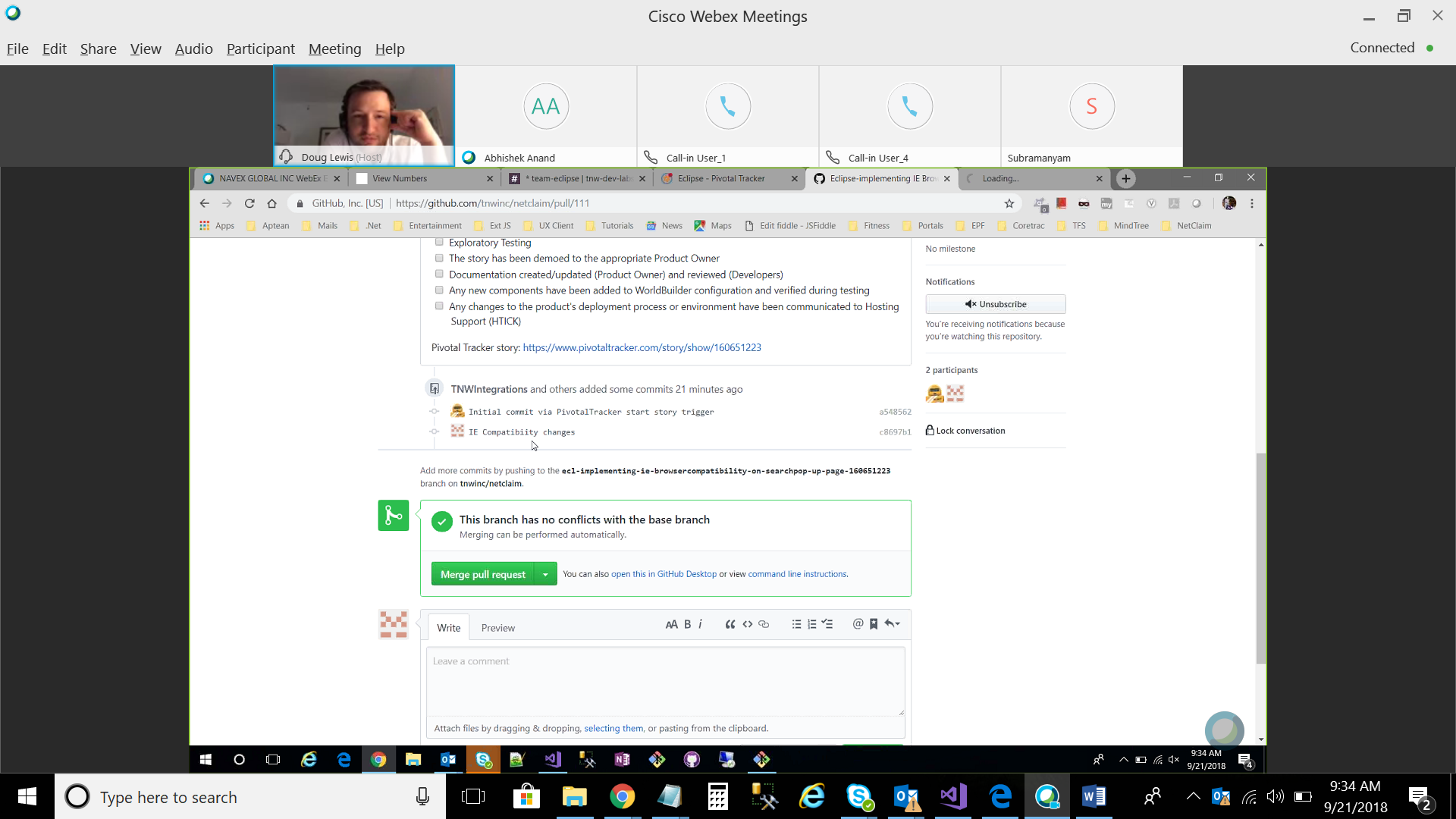


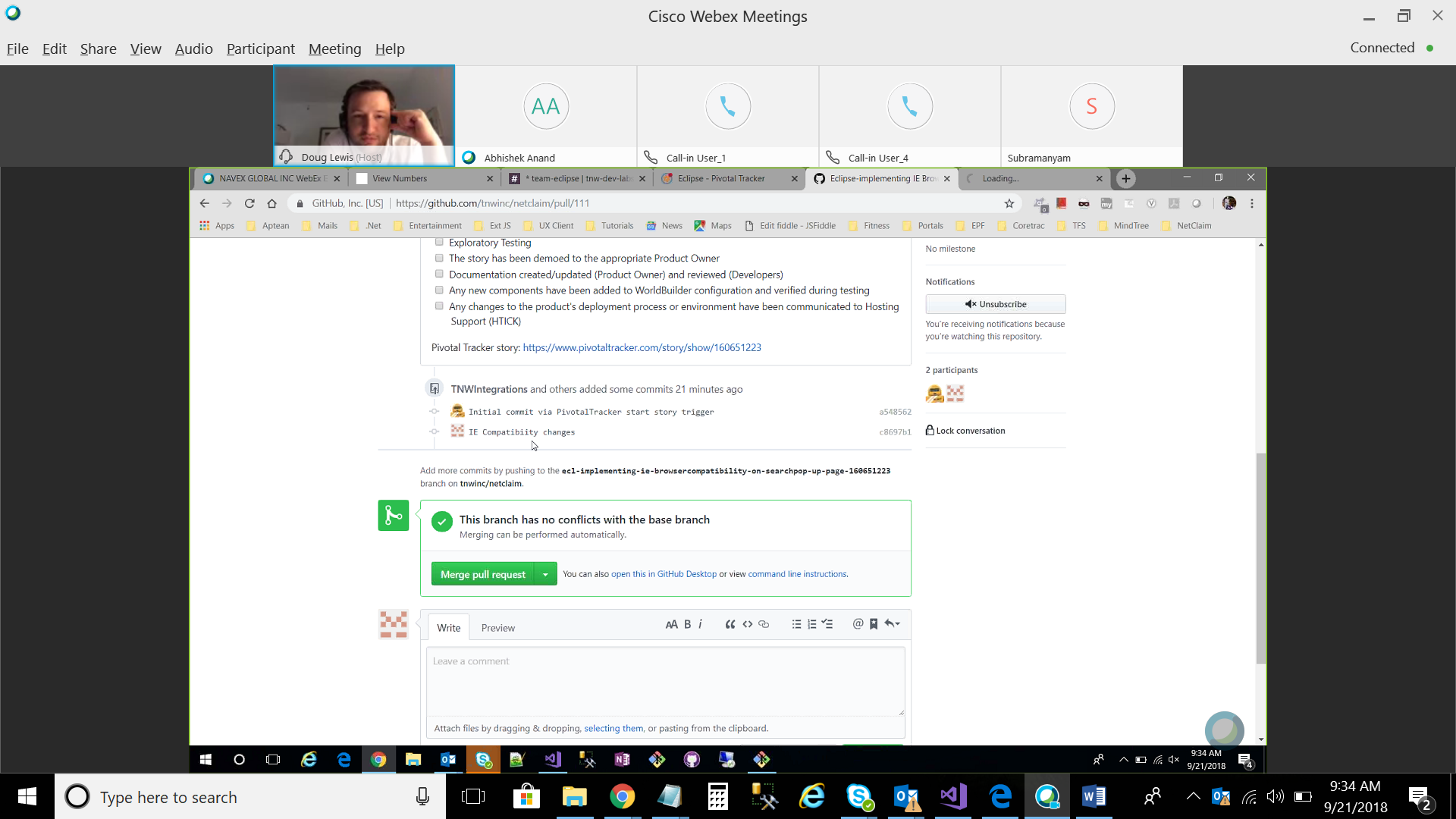


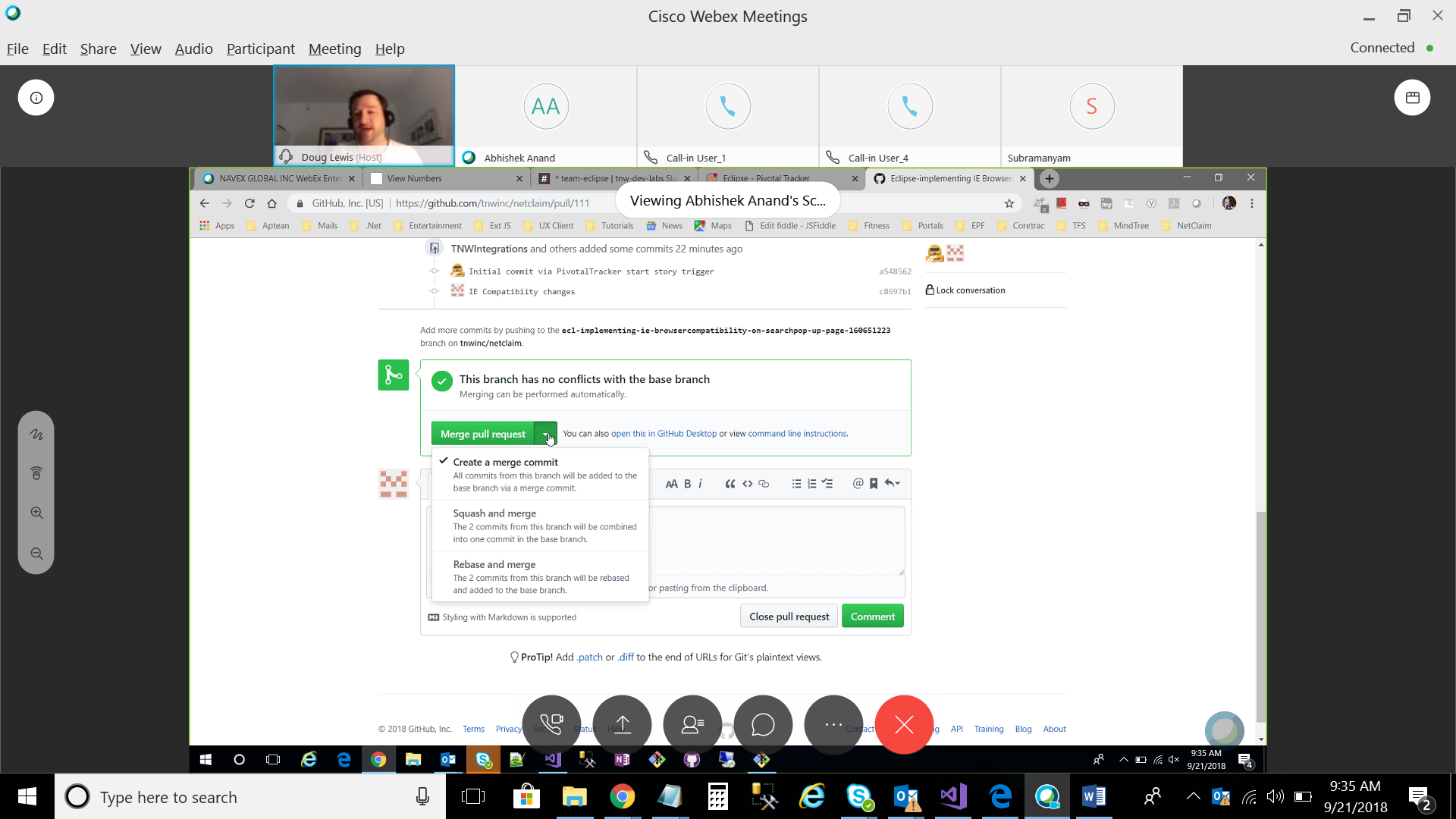












There isn't a way to select multiple files for deletion using the Web UI at the moment, but I'll let the team know you'd like us to add this feature in the future.

In the meantime, the way we recommend editing or deleting multiple files is by working with a local repository. You would then be able to delete the files in your local clone, commit that change to your local repository, and then push that change to the remote repository on GitHub.

The steps for doing this are:

* In the command-line, navigate to your local repository.
* Ensure you are in the default branch:  
  git checkout master
* The rm -r command will recursively remove your folder:  
  git rm -r folder-name
* Commit the change:  
  git commit -m "Remove duplicated directory"
* Push the change to your remote repository:  
  git push origin master

----------------------------------------------------------------------------------------------

* **Remove directory from git and local**
* You could checkout 'master' with both directories;
* git rm -r one-of-the-directories
* git commit -m "Remove duplicated directory"
* git push origin <your-git-branch> (typically 'master', but not always)
* **Remove directory from git but NOT local**
* As mentioned in the comments, what you usually want to do is remove this directory from git but not delete it entirely from the filesystem (local)
* In that case use:
* git rm -r --cached myFolder

First, you create your branch locally:

git checkout -b <branch-name> # Create a new branch and check it out

The remote branch is automatically created when you push it to the remote server. So when you feel ready for it, you can just do:

git push <remote-name> <branch-name>

Where <remote-name> is typically origin, the name which git gives to the remote you cloned from. Your colleagues would then just pull that branch, and it's automatically created locally.

Note however that formally, the format is:

git push <remote-name> <local-branch-name>:<remote-branch-name>

But when you omit one, it assumes both branch names are the same. Having said this, as a word of **caution**, do not make the critical mistake of specifying only :<remote-branch-name> (with the colon), or the remote branch will be deleted!

So that a subsequent git pull will know what to do, you might instead want to use:

git push --set-upstream <remote-name> <local-branch-name>