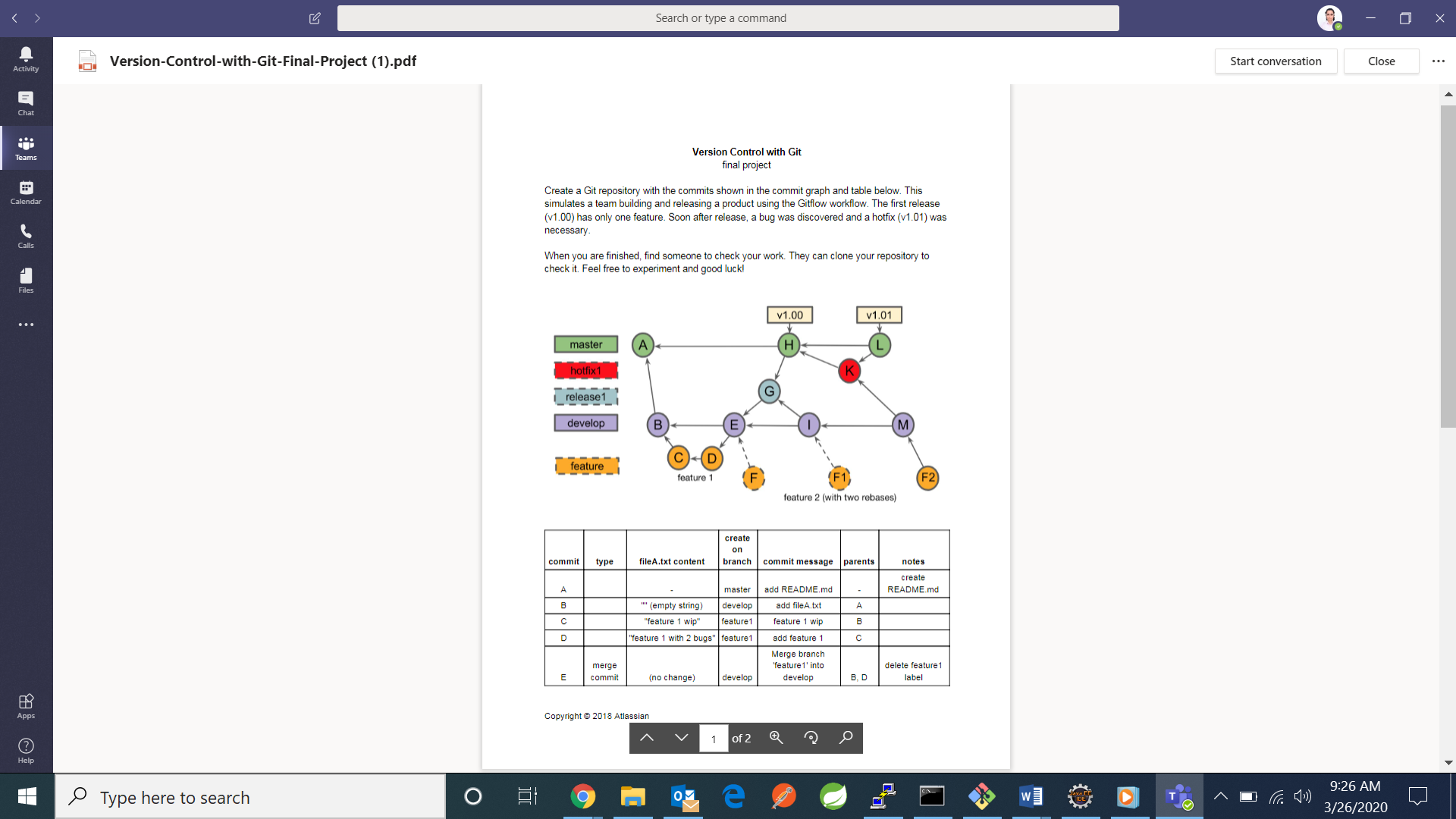
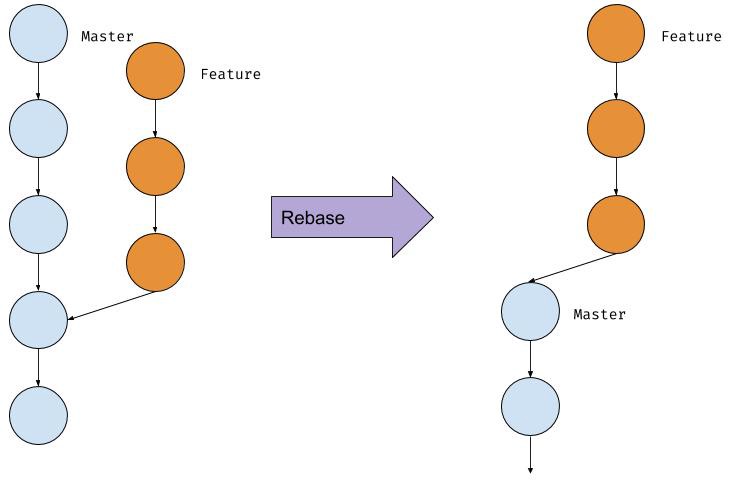
WE NEED TO MAKE A WORKFLOW OF THE FOLLOWING :

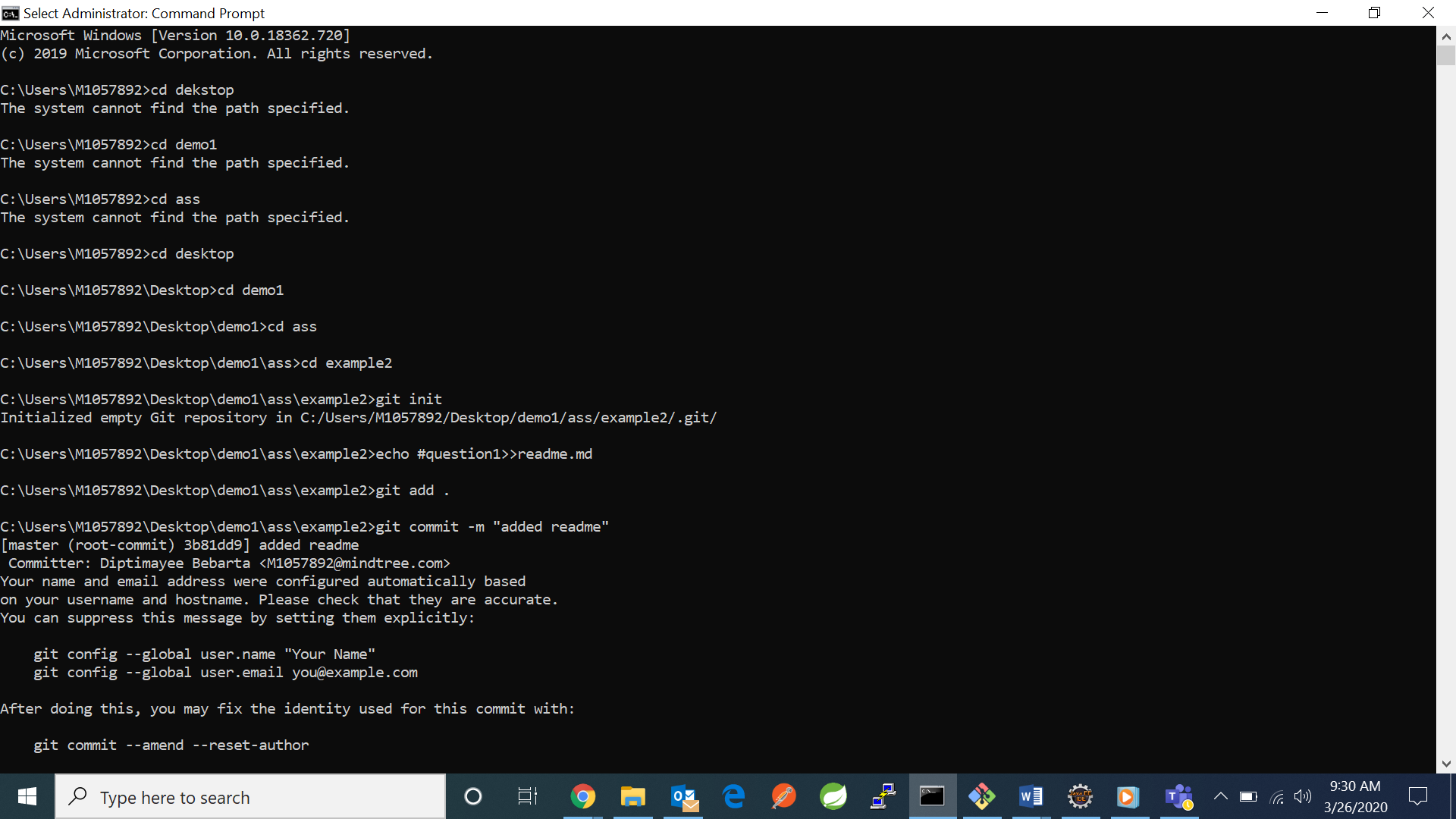


Master, hotfix1, release1, develop, feature are the branches and the circled alphabets are the commits. The F and F1 commits are temporary as these commits will be rebase commits. This is what rebase means:

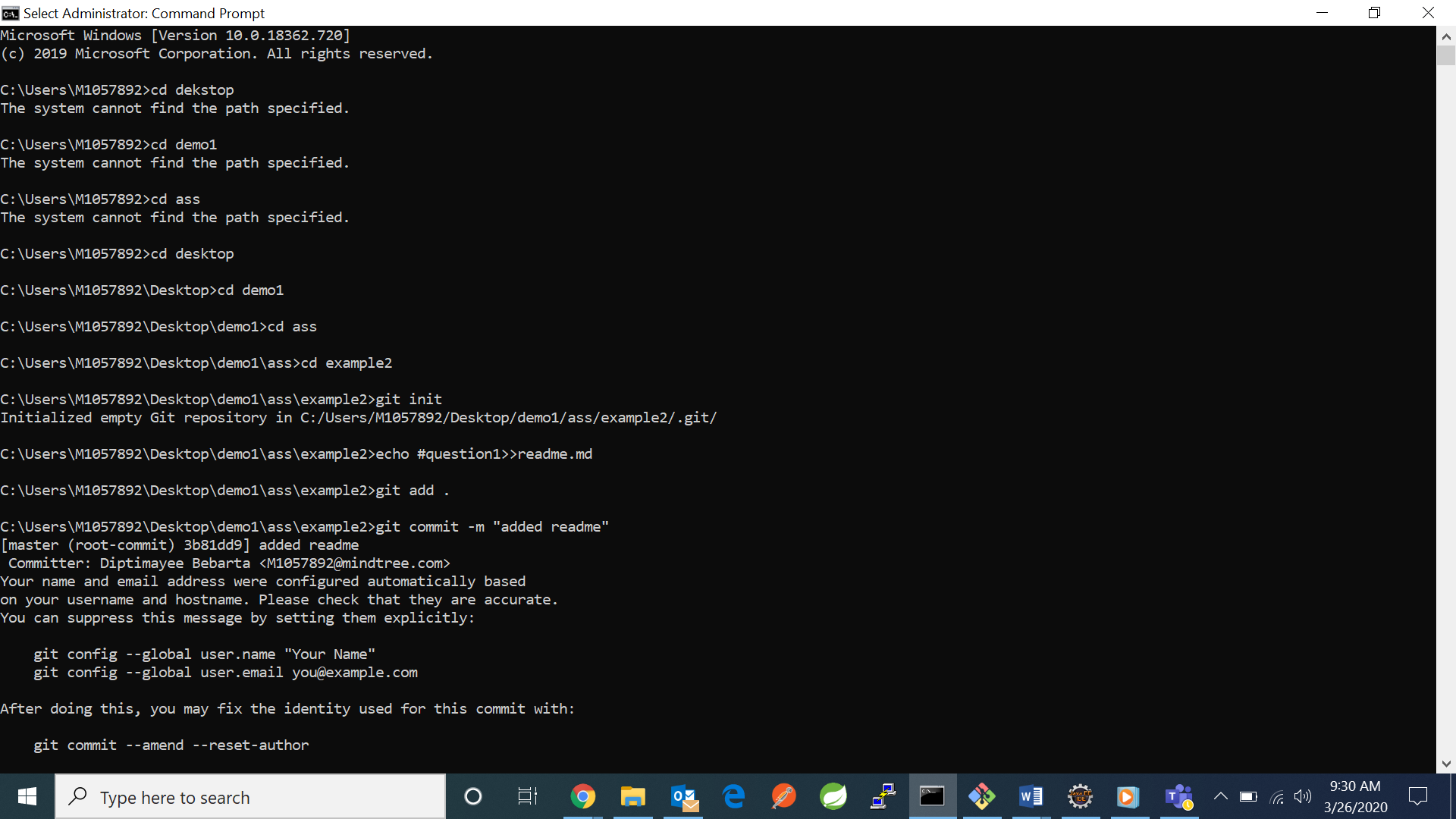


* Make a directory of your own in which you want to solve the question and switch to that directory

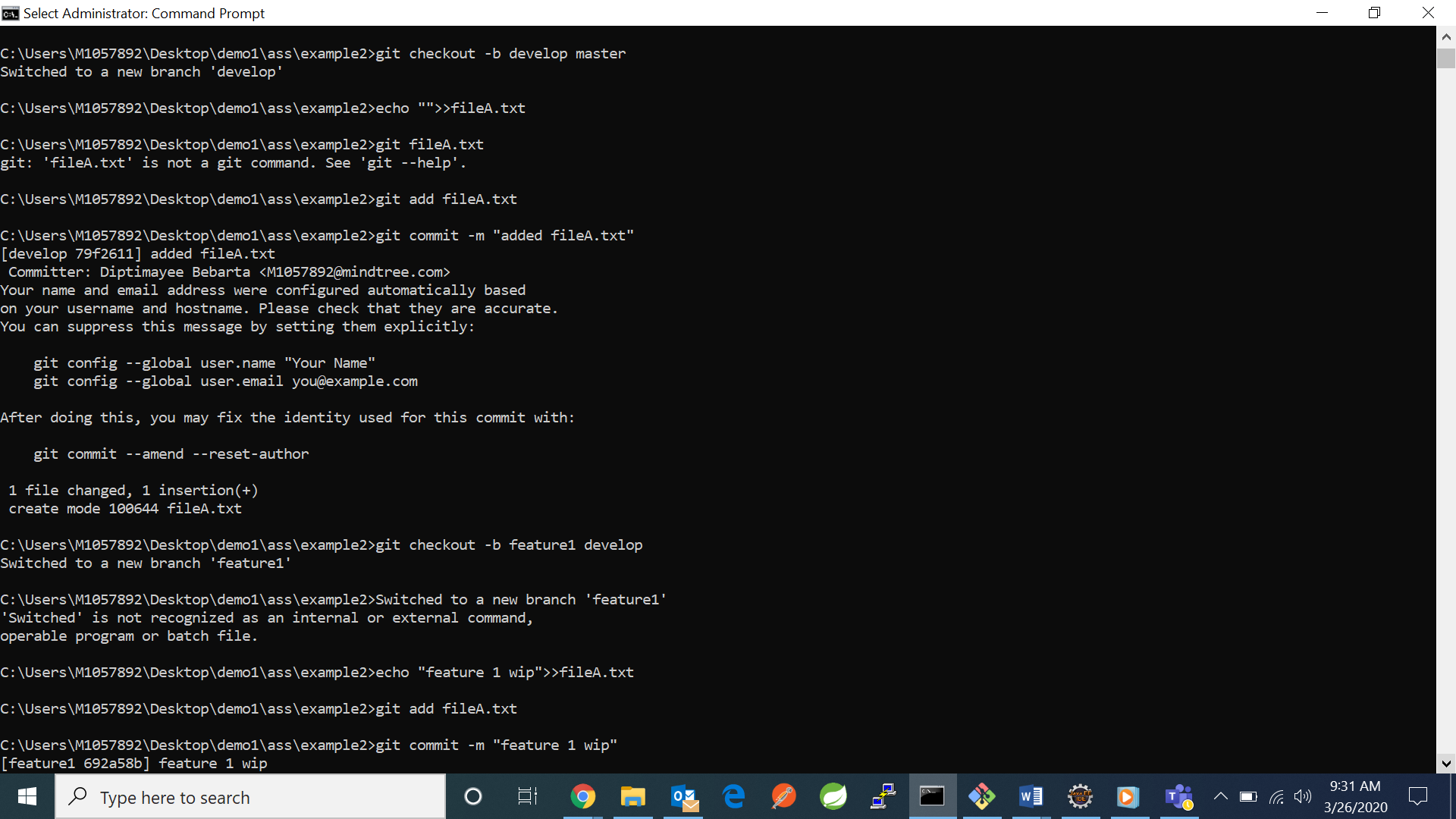
My directory is example3:



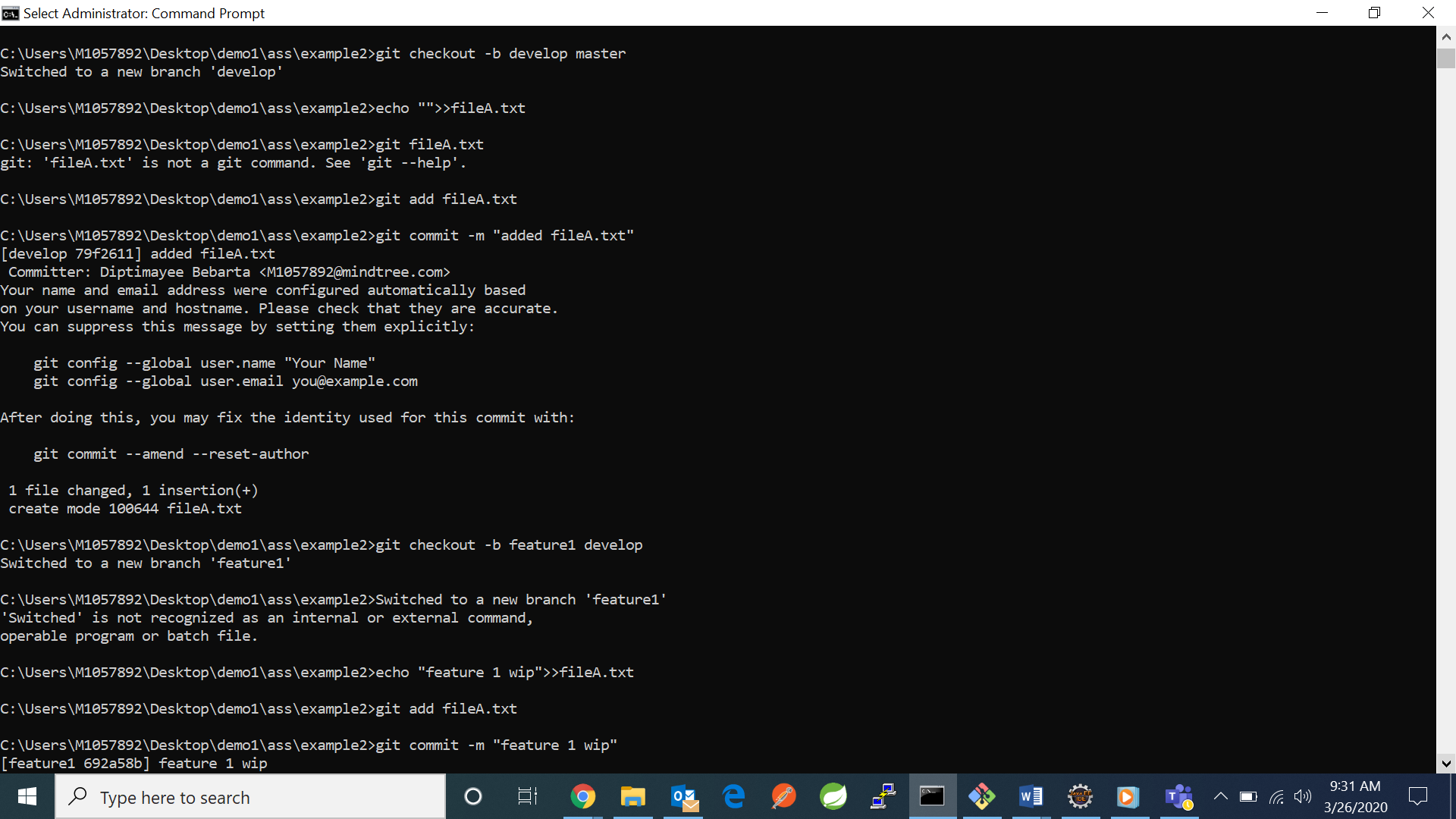
* Do the command **git init –**it initializes a local git repository. By default the master branch will be created which is the main branch.
* **COMMIT A**
  + - 1. **echo “#question1”>>readme.md**: creates a readme file and writes into it question1.
      2. **git add readme.md** : It adds the readme file to the staging area.
      3. **git commit -m “added readme.md”** – It saves the changes in the local repository along with the message. All the changes are done on master branch
      4. **git remote add origin “abc”** – it creates a origin which points to the url of github or remote repository.
      5. **git push origin master** –it pushes the contents of the master branch to the remote repo.



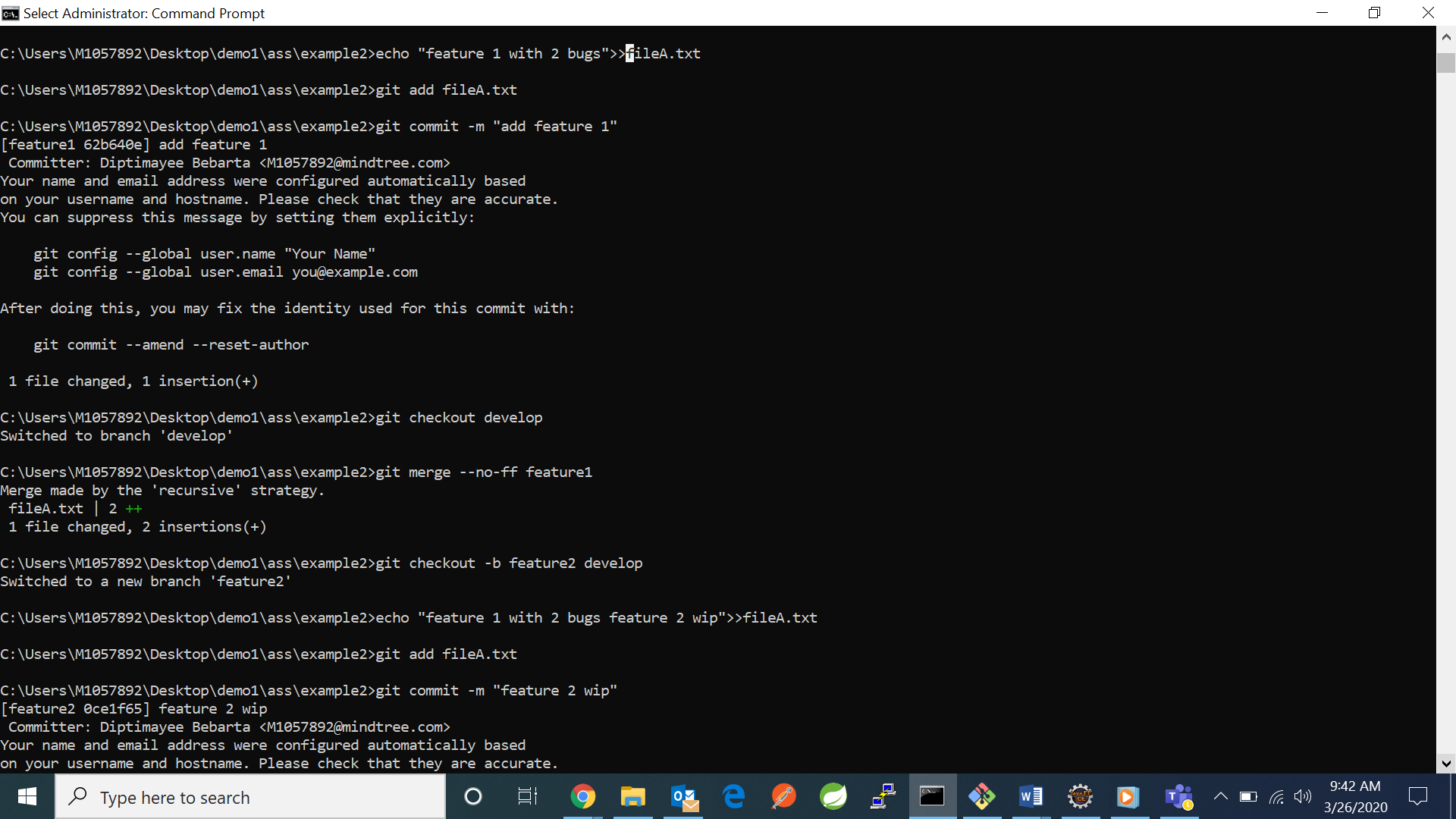
* **COMMIT B:**
  + - 1. **git checkout –b develop master**- Develop is a branch name that is created and it points to the last commit of master branch
      2. **echo “”>>fileA.txt**: an fileA.txt file is created and an empty string is written into it
      3. **git add fileA.txt :** It adds the readme file to the staging area.
      4. **git commit -m “added fileA.txt”** – It saves the changes in the local repository along with the message. All the changes are done on master branch.
      5. **git push origin develop** –it pushes the contents of the develop branch to the remote repo.



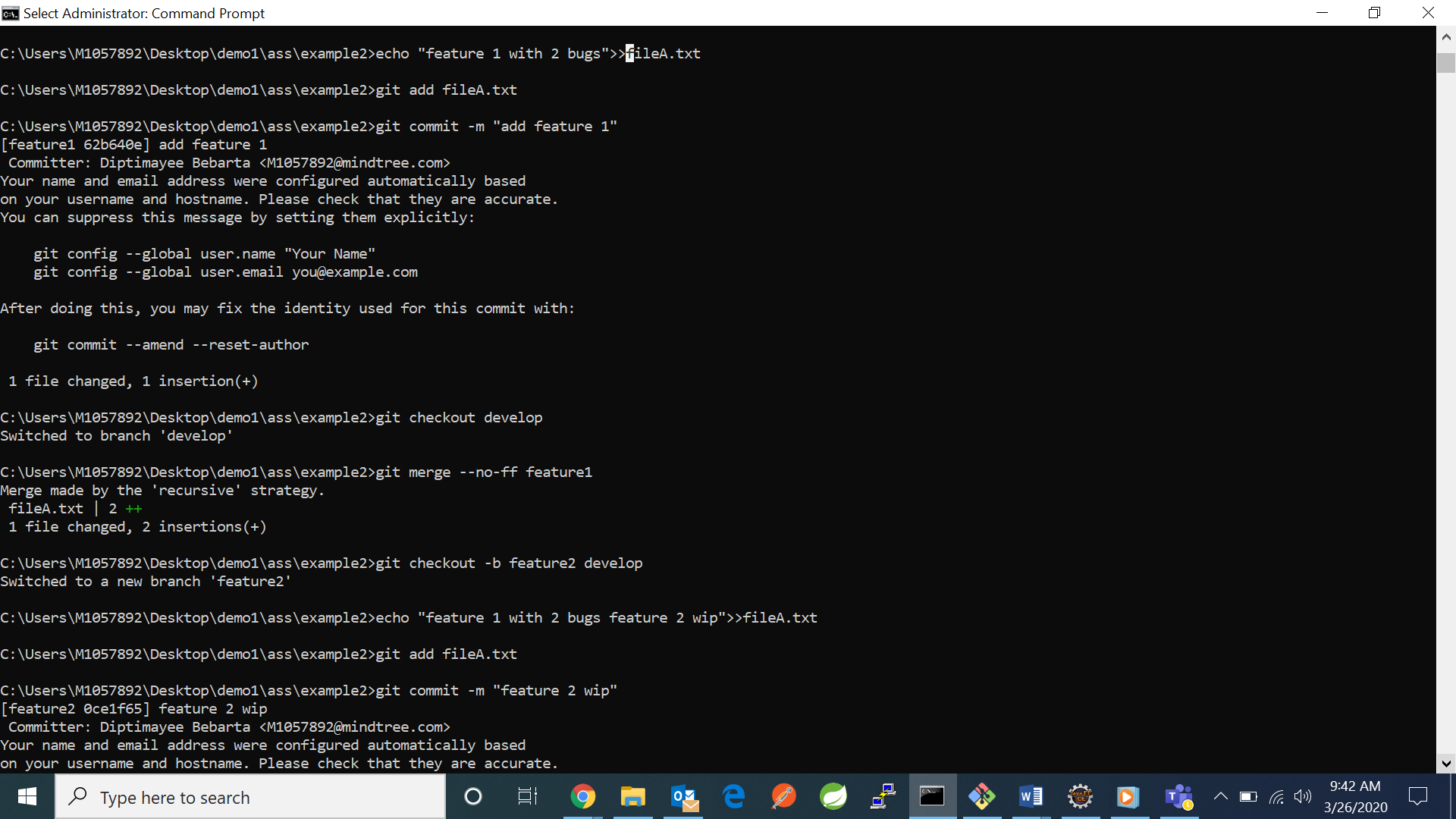
* COMMIT C:
  + - 1. **git checkout -b feature1 develop –**create a feature1 branch pointing to last commit of develop branch
      2. **echo "feature 1 wip">>fileA.txt –**adds the content to existing fileA.txt
      3. **git add fileA.txt**
      4. **git commit -m "feature 1 wip"**



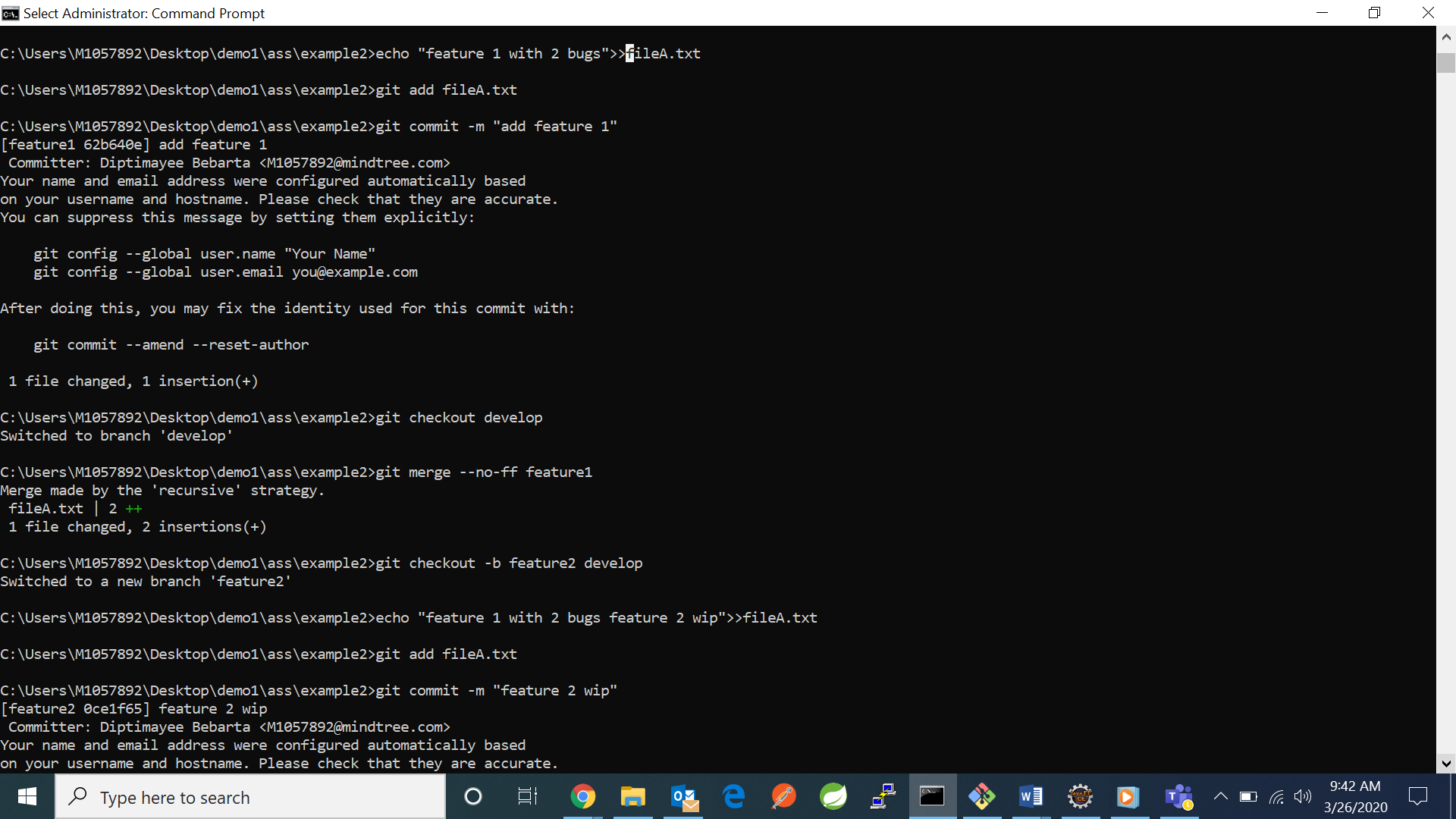
* COMMIT D:
  + - * 1. **echo "feature 1 with 2 bugs">>fileA.txt**
        2. **git add fileA.txt**
        3. **git commit -m "add feature 1"**



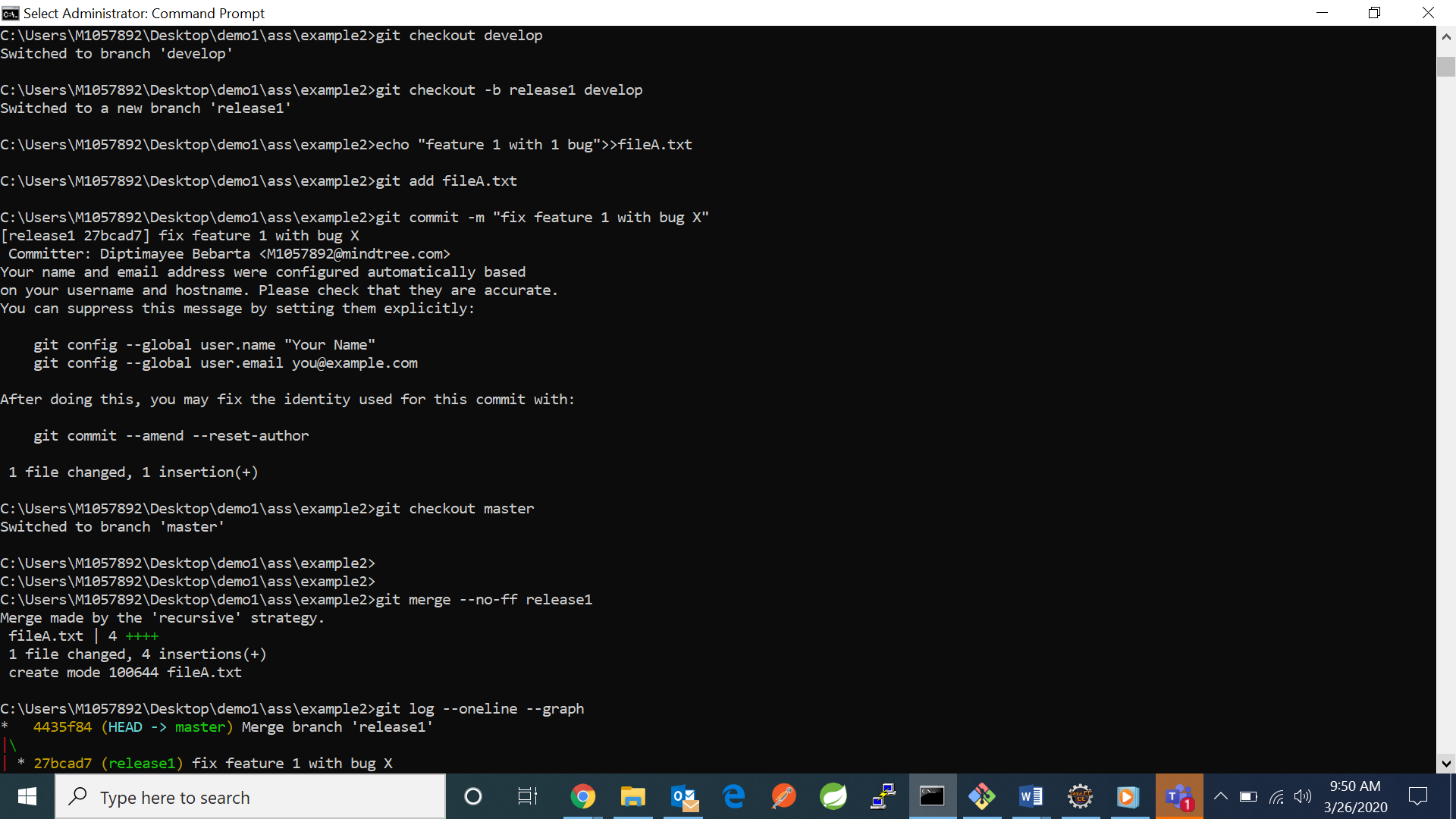
* **COMMIT E:**
  + - * 1. **git checkout develop** –switches to develop branch
        2. **git merge –no-ff feature1** – merges feature1 with develop branch



* **COMMIT F:**
  + - * 1. **git checkout -b feature2 develop –**points to last commit in develop branch and creates feature2 branch
        2. **echo "feature 1 with 2 bugs feature 2 wip">>fileA.txt**
        3. **git add fileA.txt**
        4. **git commit -m "feature 2 wip”**

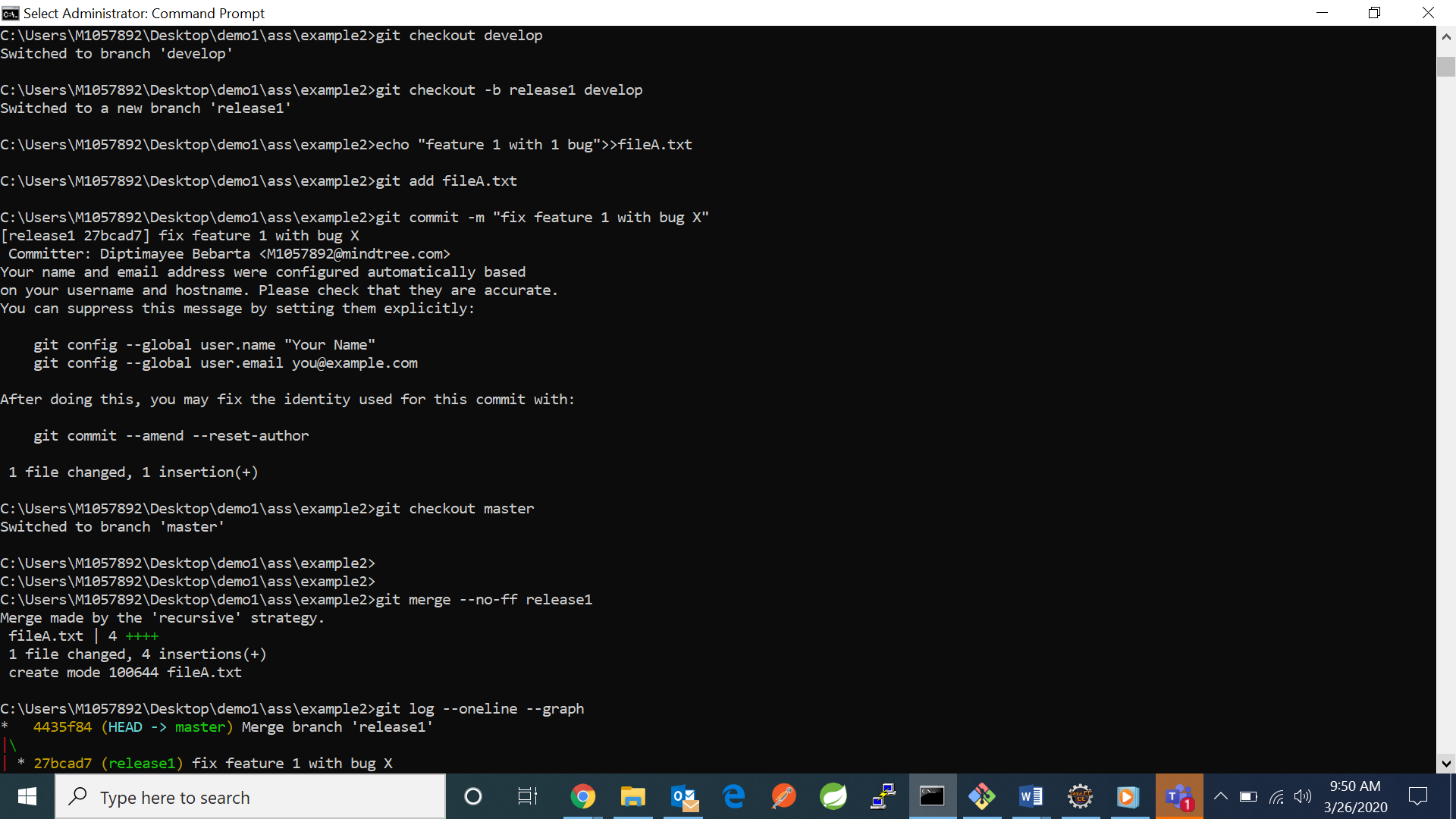


* **COMMIT G:**
  + - * 1. **git checkout develop** -switches to develop branch
        2. **git checkout –b release1 develop**  -creates a release1 branch pointing to develop last commit and switches to it
        3. **echo “feature 1 with 1 bug”>>fileA.txt**
        4. **git add fileA.txt**
        5. **git commit –m “fix feature1 bug X**”



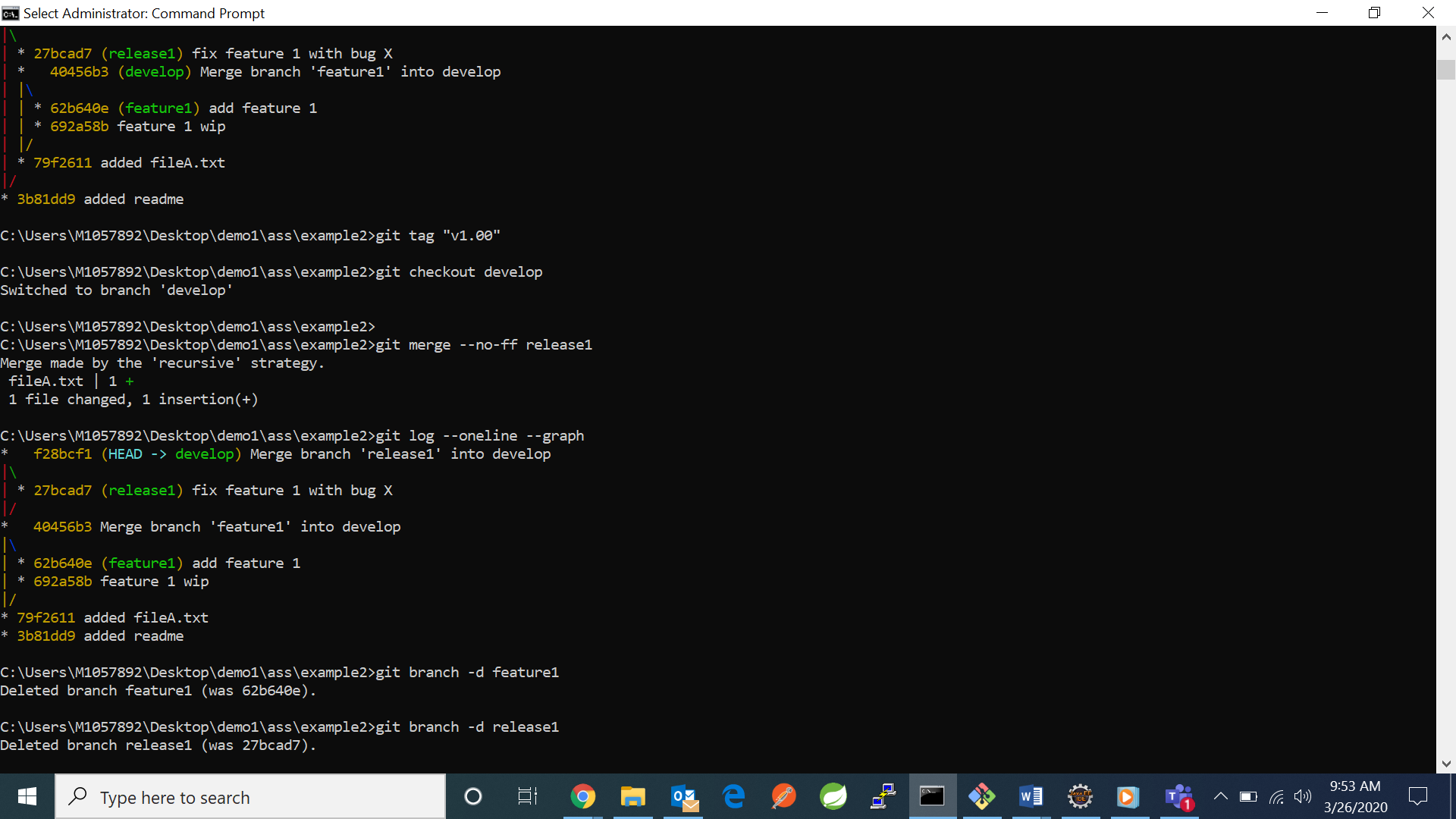
* **COMMIT H:**

1. **git checkout master** – switch to master branch
2. **git merge --no-ff release1** –merges release1 without fast forwarding to with master
3. **git tag "v1.00"** –this adds a tag to the merged commit



* COMMIT I:

1. **git checkout develop**
2. **git merge –no-ff release1** –merges release1 without fast forwarding to with master

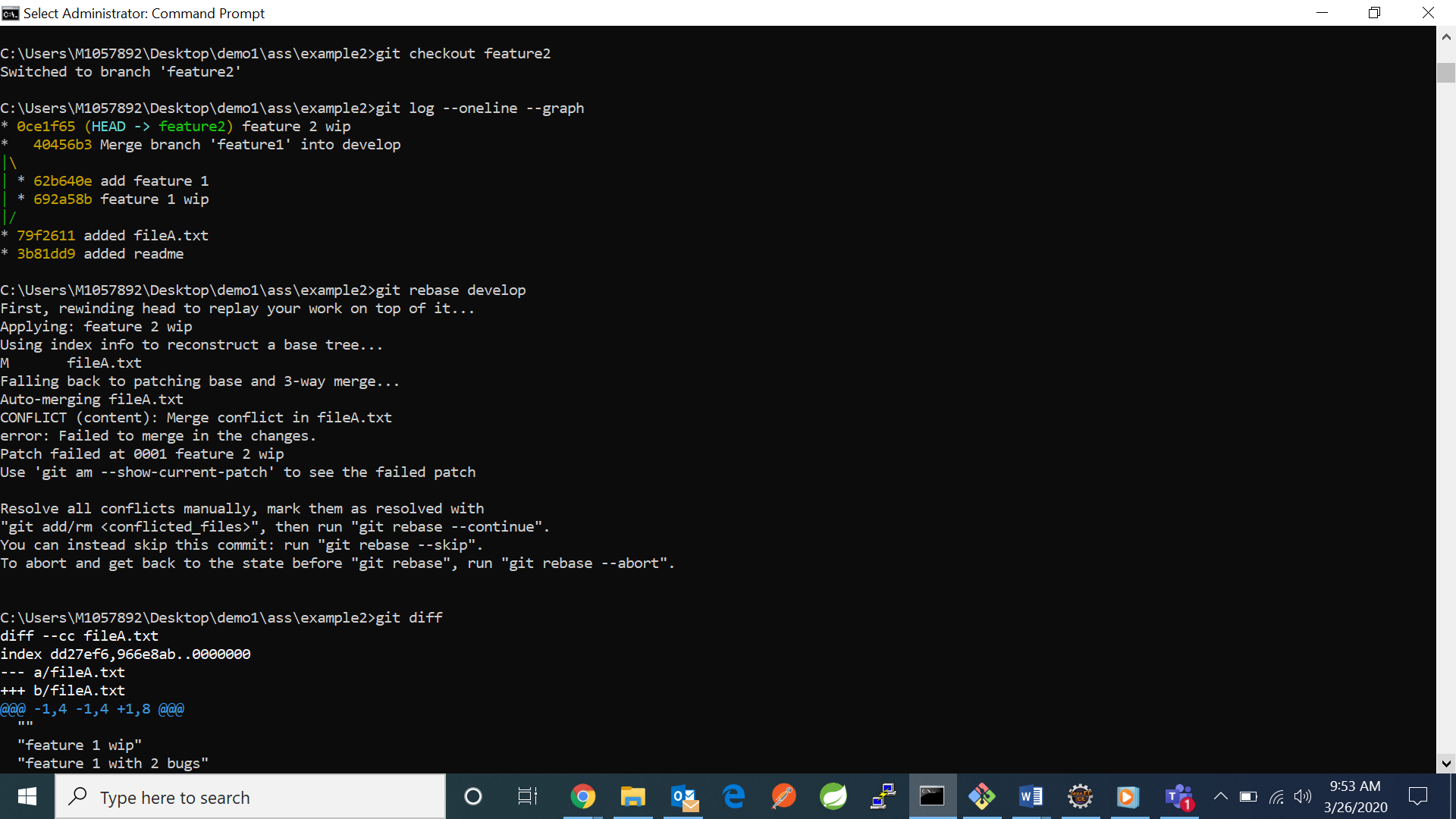


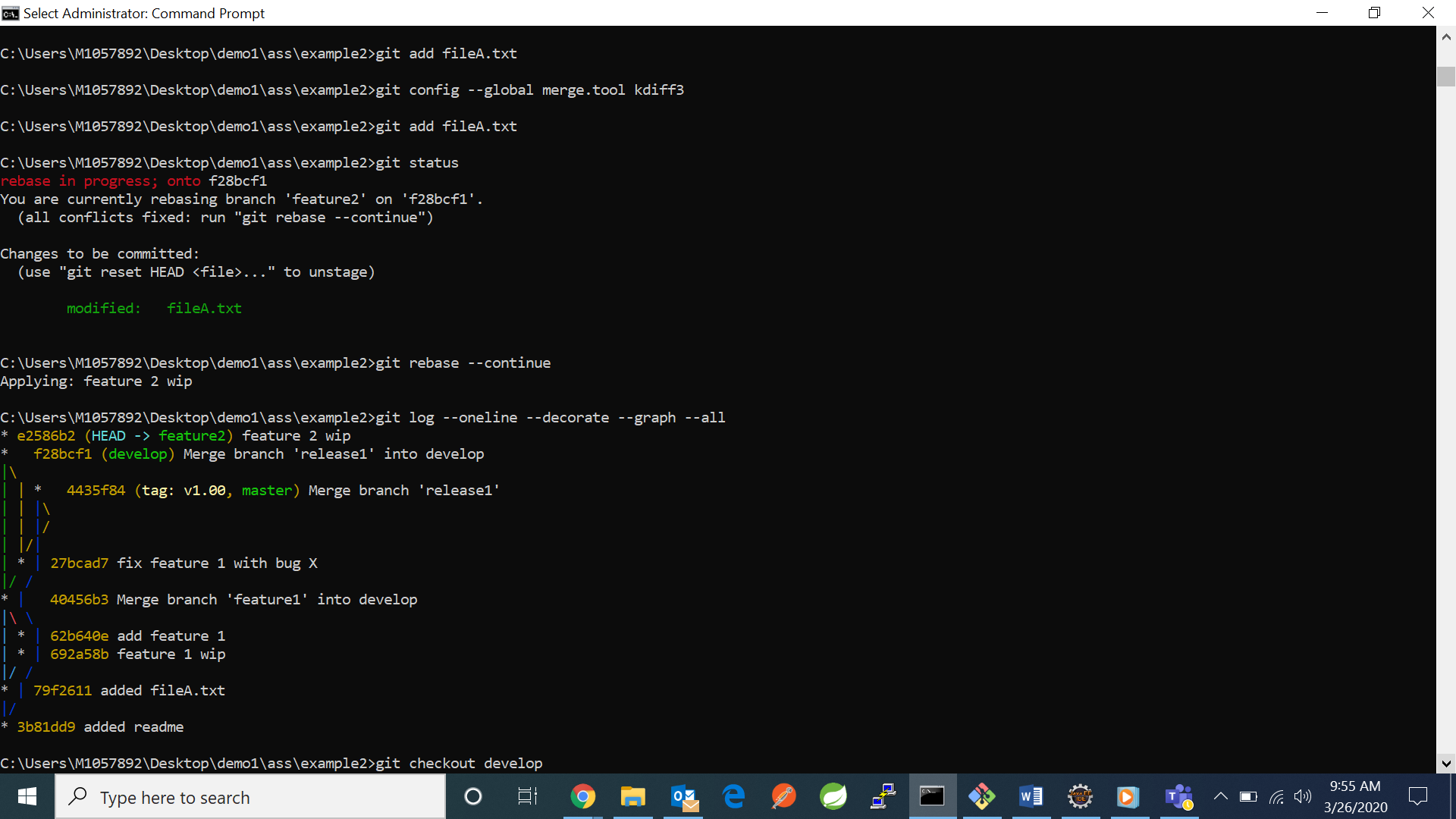
* COMMIT F1:

1. **git checkout feature2** –switches to feature2
2. **git rebase develop** –it rebases onto develop branch

It shows an conflict which needs to resolve conflict

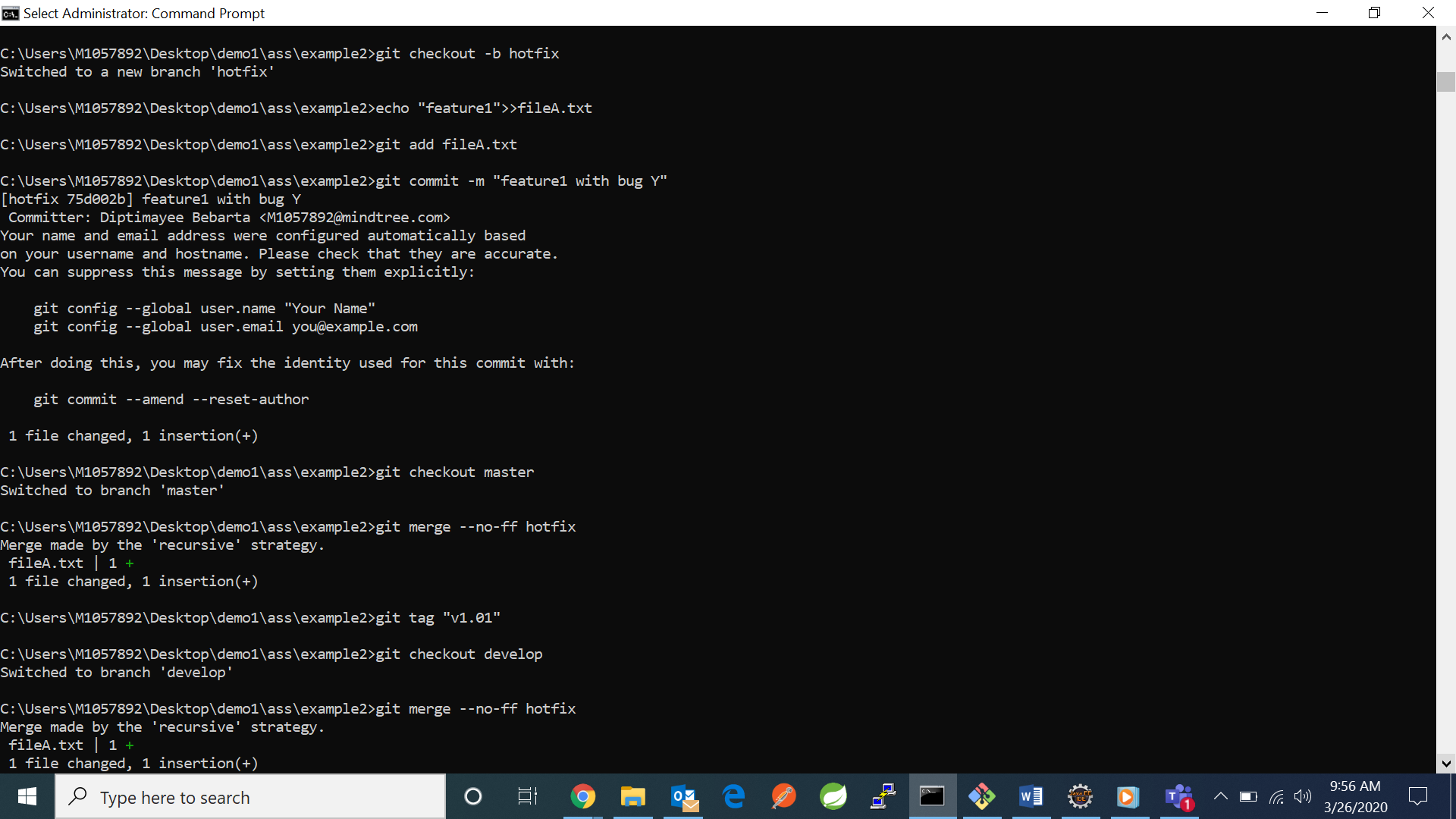
1. **git diff** –shows where conflict arises which needs to be resolved
2. **git add .** –adds file to staging area
3. **git config** **–global merge.tool kdiff3** –used to resolve merge conflict
4. **git rebase –continue** –finishes the incomplete rebase





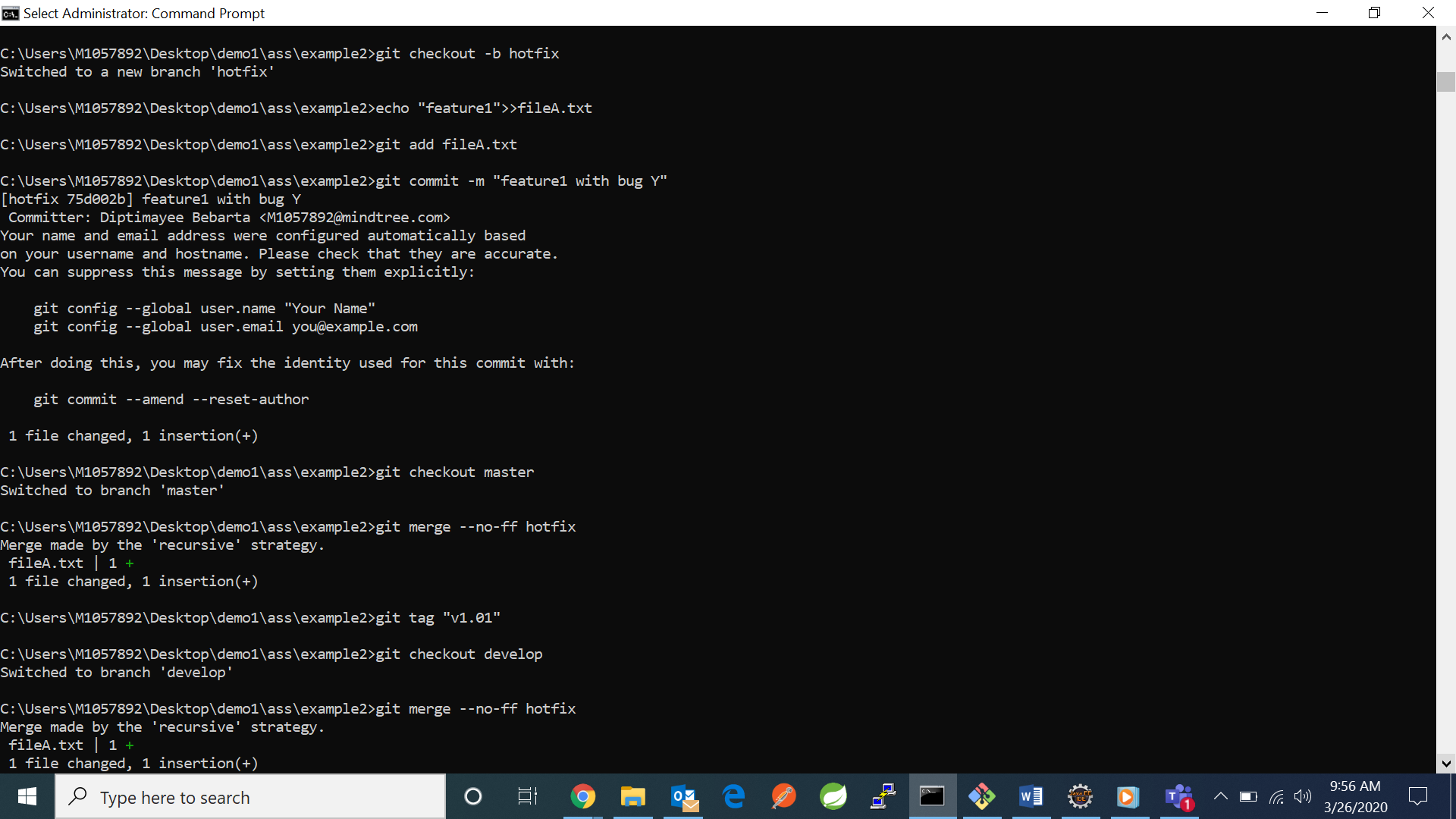
* COMMIT L:

1. **git checkout -b hotfix**
2. **echo “feature1”>>fileA.txt**
3. **git add fileA.txt**
4. **git commit -m "feature1 with bug Y"**



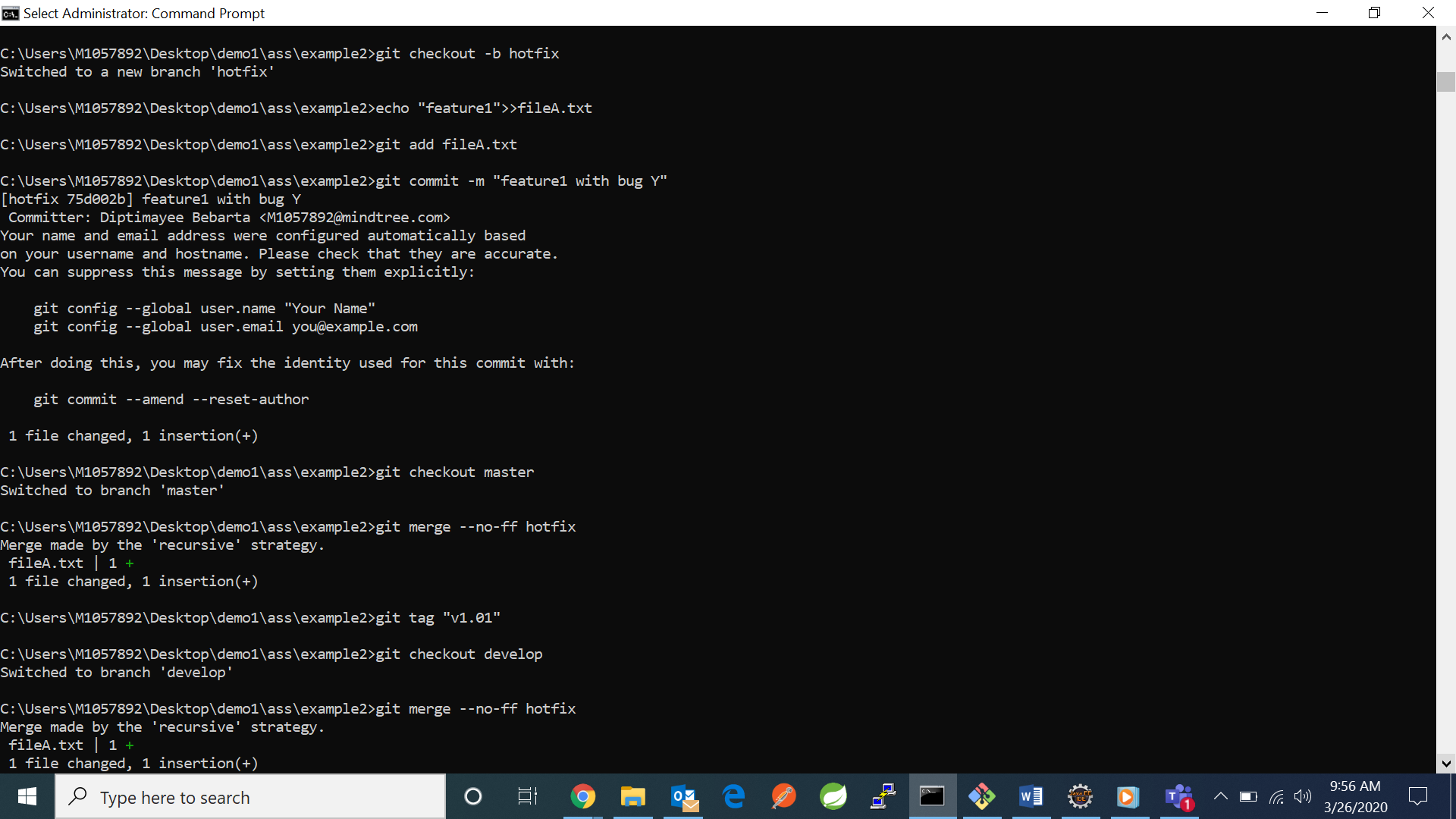
* COMMIT M:

1. **git checkout master**
2. **git merge --no-ff hotfix**
3. **git tag "v1.01"**



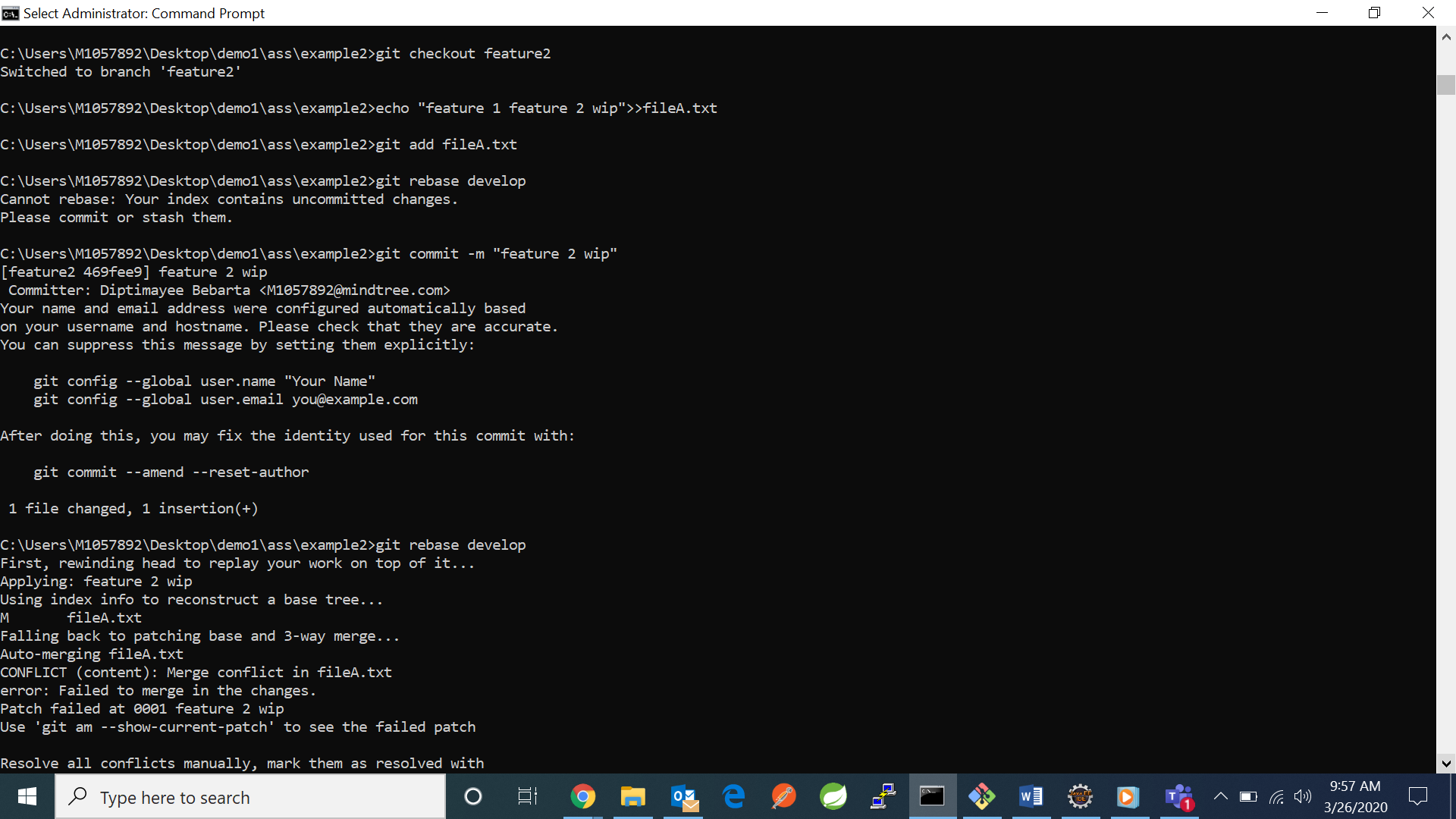
* COMMIT M:

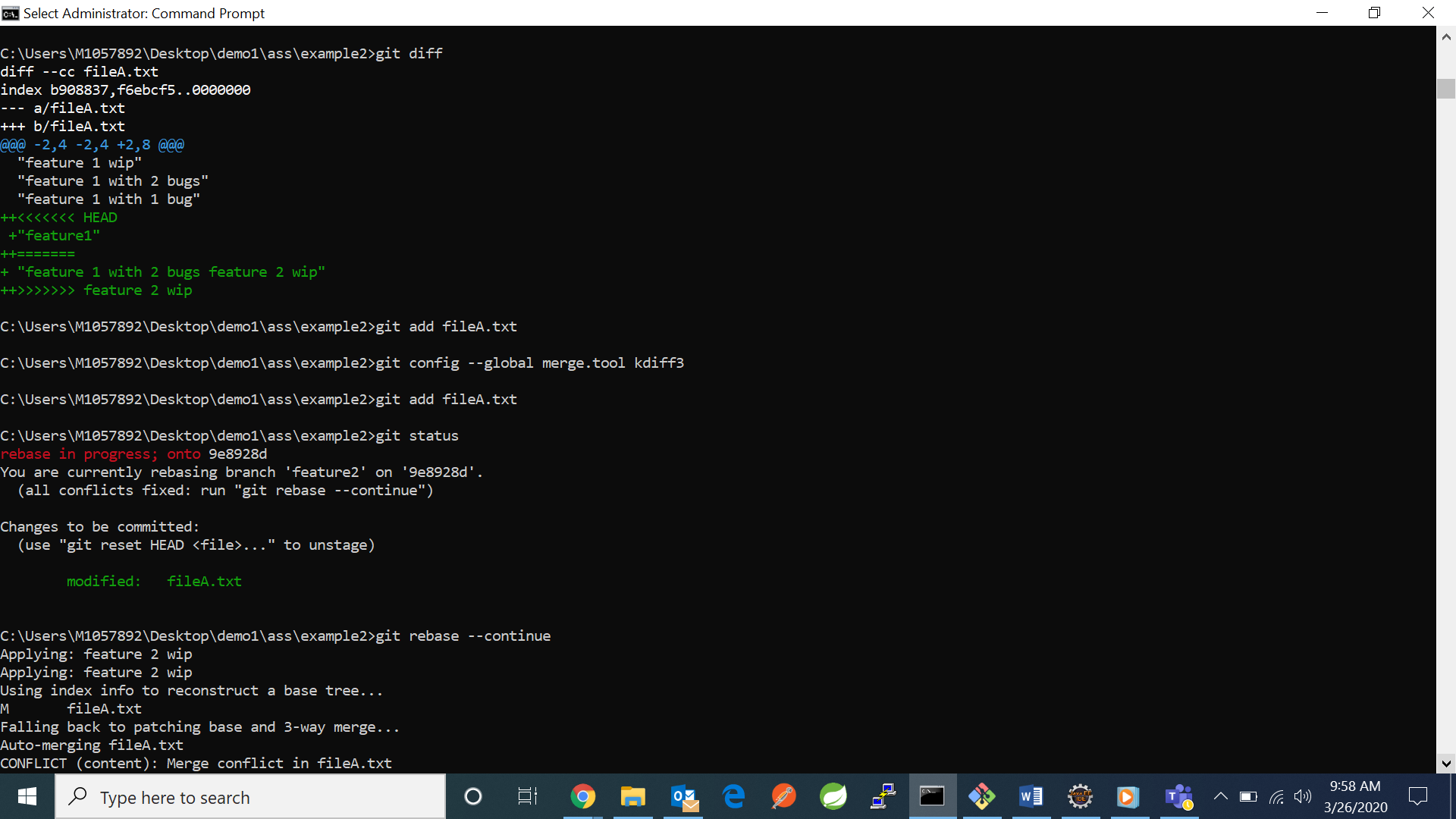
1. **git checkout develop**
2. **git merge --no-ff hotfix**

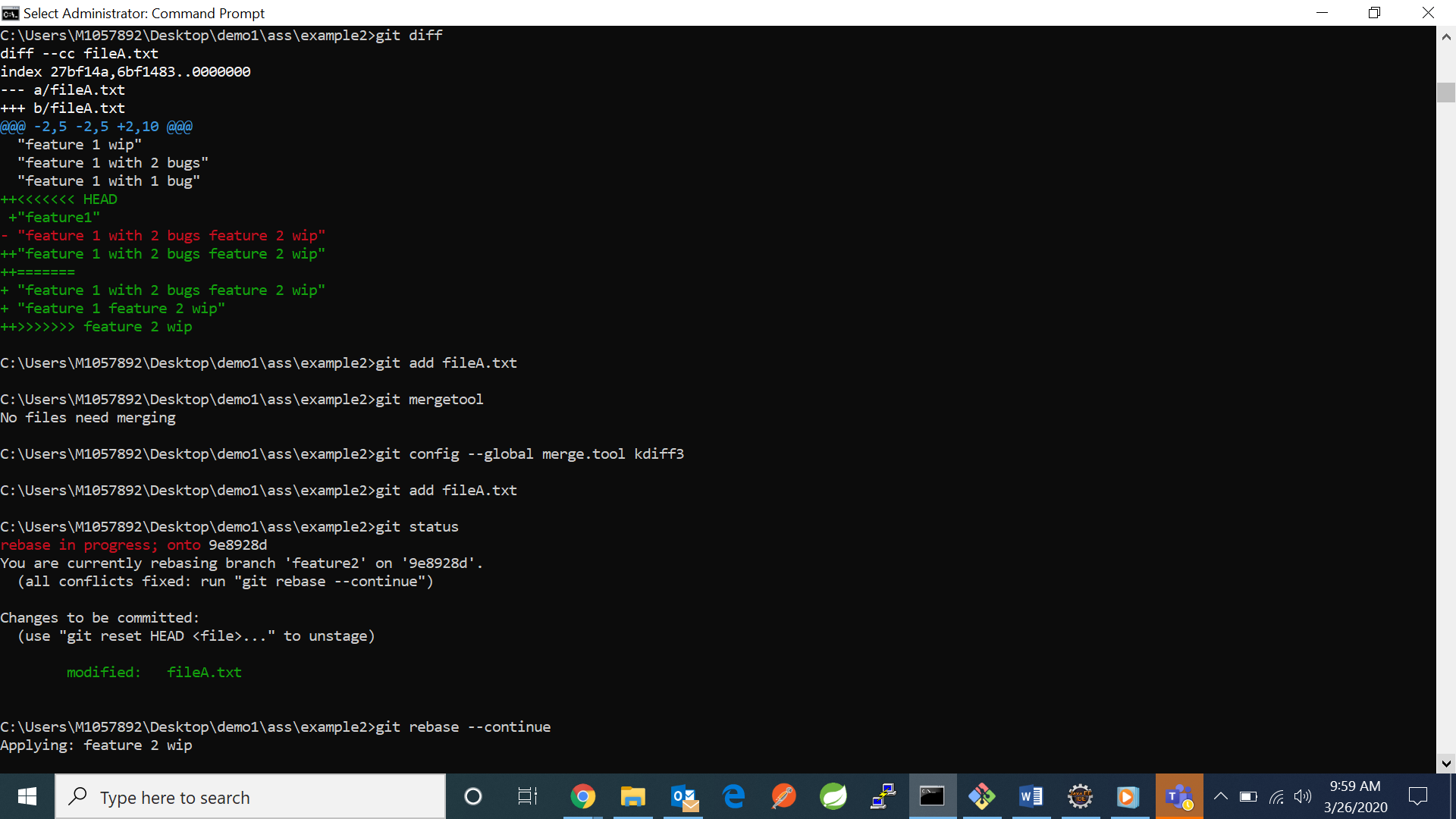


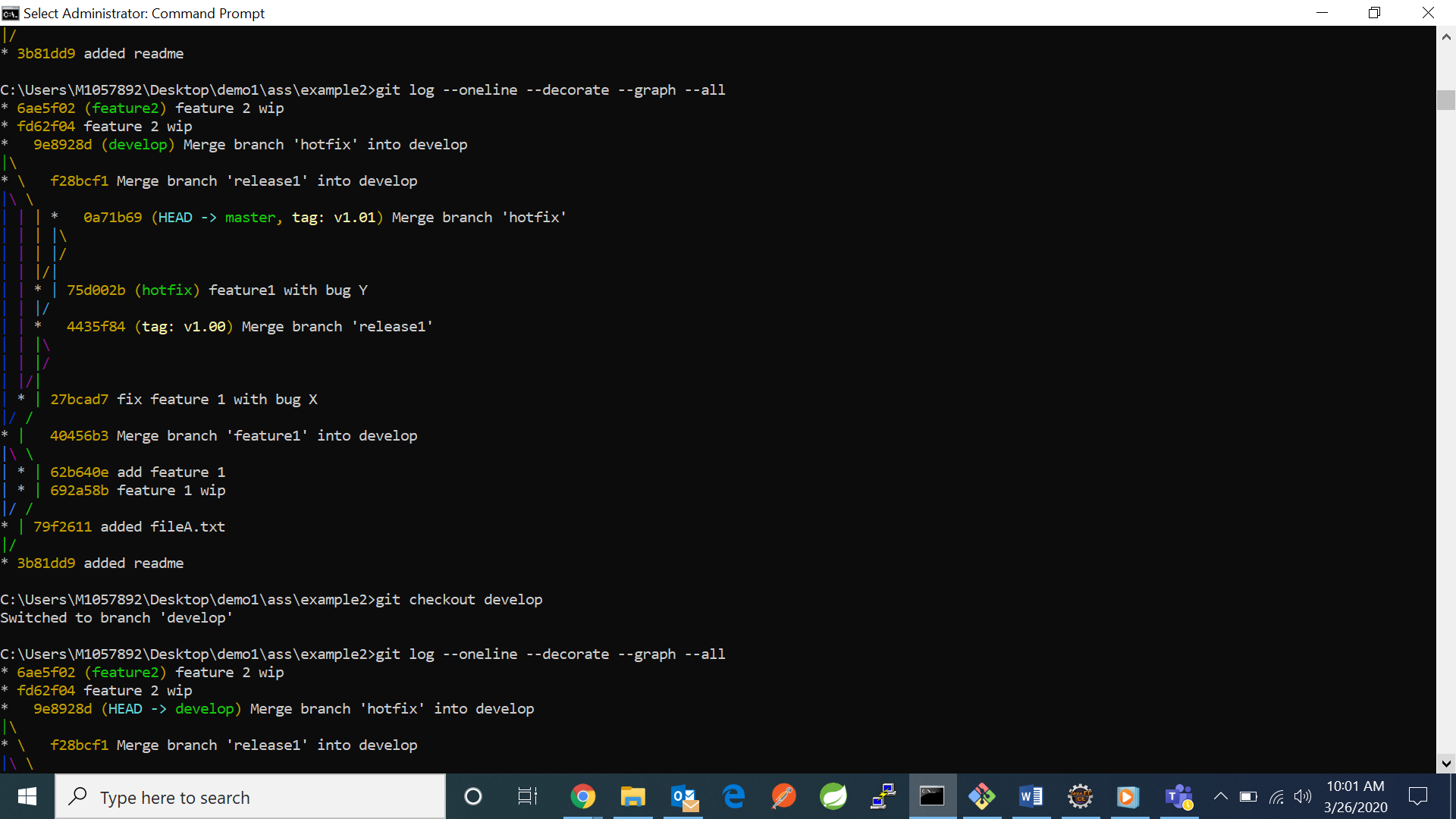
* COMMIT F2:

1. **git checkout feature2**
2. **echo "feature 1 feature 2 wip">>fileA.txt**
3. **git add fileA.txt**
4. **git commit -m "feature2 wip"**
5. **git rebase develop**
6. **git diff**
7. **git add .**
8. **git config –global merge.tool kdiff3**
9. **git add .**
10. **git rebase –continue –** This will create another conflict which needs to be resolved again by doing the same process.
11. **git diff**
12. **git add .**
13. **git config –global merge.tool kdiff3**
14. **git add .**
15. **git rebase --continue**







THIS IS HOW THE GRAP LOOKS:  


node{  
stage('SCM Checkout'){  
git '[https://github.com/Devil0730/MVC1'](https://github.com/Devil0730/MVC1%27)  
}  
stage('Compile Package'){  
def mvnHome= tool name: 'Maven 3', type: 'maven'  
sh "${mvnHome}/bin/mvn package"  
}  
}