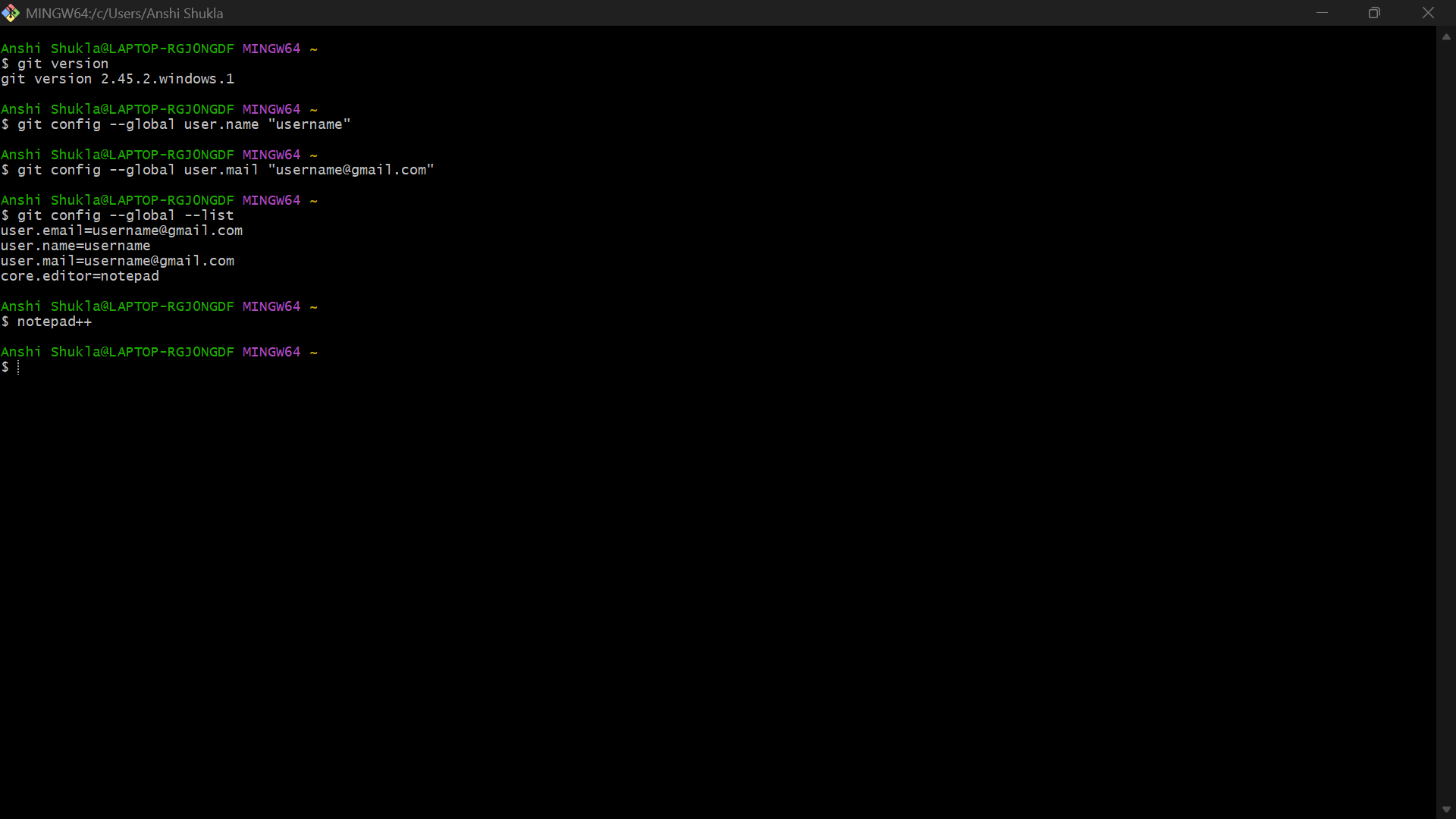
Week -8

1 -In this hands-on lab, you will learn how to

* Setup your machine with Git Configuration
* Integrate notepad++.exe to Git and make it a default editor
* Add a file to source code repository



2 –Create a **“.log”** file and a **log folder** in the working directory of Git. Update the **.gitignore** file in such a way that on committing, these files (.log extensions and log folders) are ignored.

Verify if the git status reflects the same about working directory, local repository and git repository.

echo This is a new file for GitNewBranch" > debug.log

mkdir log

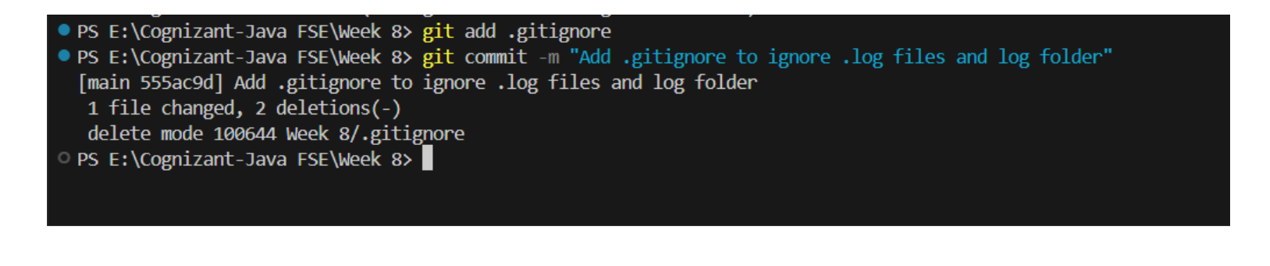
echo “Folder log content”

notepad ++ .gitignore

git status

git add .gitignore

git commit -m “Add .gitignore to ignore .log files and log folder”



3 –**Branching:**

1. Create a new branch **“GitNewBranch”.**
2. List all the local and remote branches available in the current trunk. Observe the “\*” mark which denote the current pointing branch.
3. Switch to the newly created branch. Add some files to it with some contents.
4. Commit the changes to the branch.
5. Check the status with **“git status”** command.

**Merging:**

1. Switch to the master
2. List out all the differences between trunk and branch. These provide the differences in command line interface.
3. List out all the visual differences between master and branch using **P4Merge tool**.
4. Merge the source branch to the trunk.
5. Observe the logging after merging using **“git log –oneline –graph –decorate”**
6. Delete the branch after merging with the trunk and observe the git status.

git checkout -b GitNewBranch

git brach

echo “This is a new feature file”

git add feature.txt

git commit -m “Add feature.txt in GitNewBranch”

git status

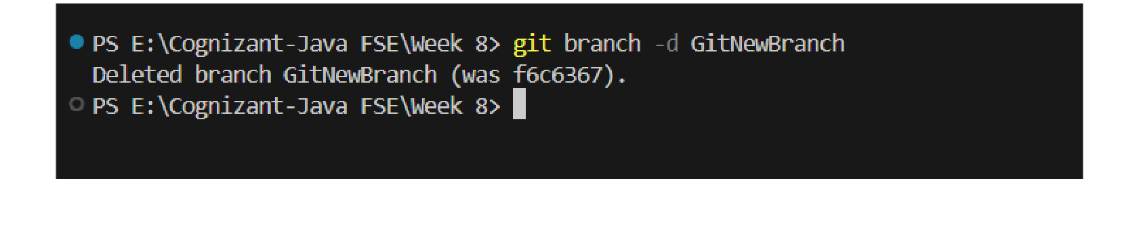
git checkout main

git diff main..GitNewBranch

git merge GitNewBranch

git log

git branch -d GitNewBranch



4 –

Please follow the instructions to complete the hands-on. Each instruction expect a command for the Git Bash.

1. Verify if master is in clean state.
2. Create a branch **“GitWork”.** Add a file “hello.xml”.
3. Update the content of “hello.xml” and observe the status
4. Commit the changes to reflect in the branch
5. Switch to master.
6. Add a file **“hello.xml”** to the master and add some different content than previous.
7. Commit the changes to the master
8. Observe the log by executing **“git log –oneline –graph –decorate –all”**
9. Check the differences with Git diff tool
10. For better visualization, use P4Merge tool to list out all the differences between master and branch
11. Merge the bran to the master
12. Observe the git mark up.
13. Use 3-way merge tool to resolve the conflict
14. Commit the changes to the master, once done with conflict
15. Observe the git status and add backup file to the .gitignore file.
16. Commit the changes to the .gitignore
17. List out all the available branches
18. Delete the branch, which merge to master.
19. Observe the log by executing **“git log –oneline –graph –decorate”**

git checkout main

git status

git checkout -b GitWork

echo “<message>Hello from GitWork<.messagre>” >hello.xml

git status

echo “<note>Update from GitWork Branch</note>”>> hello.xml

git add hello.xml

git commit -m “Add hello.xml with content from Gitwork”

echo “<note>Update from GitWork branch</note>” >> hello.xml

git add hello.xml

git commit -m “Add hello.xml with content from GitWork”

echo “<note>Update from GitWork branch</note>” >> hello.xml

git add hello.xml

git commit -m “Add hello.xml with content from GitWork”

git log

git diff Gitwork

git merge GitWork

notepad++ hello.xml

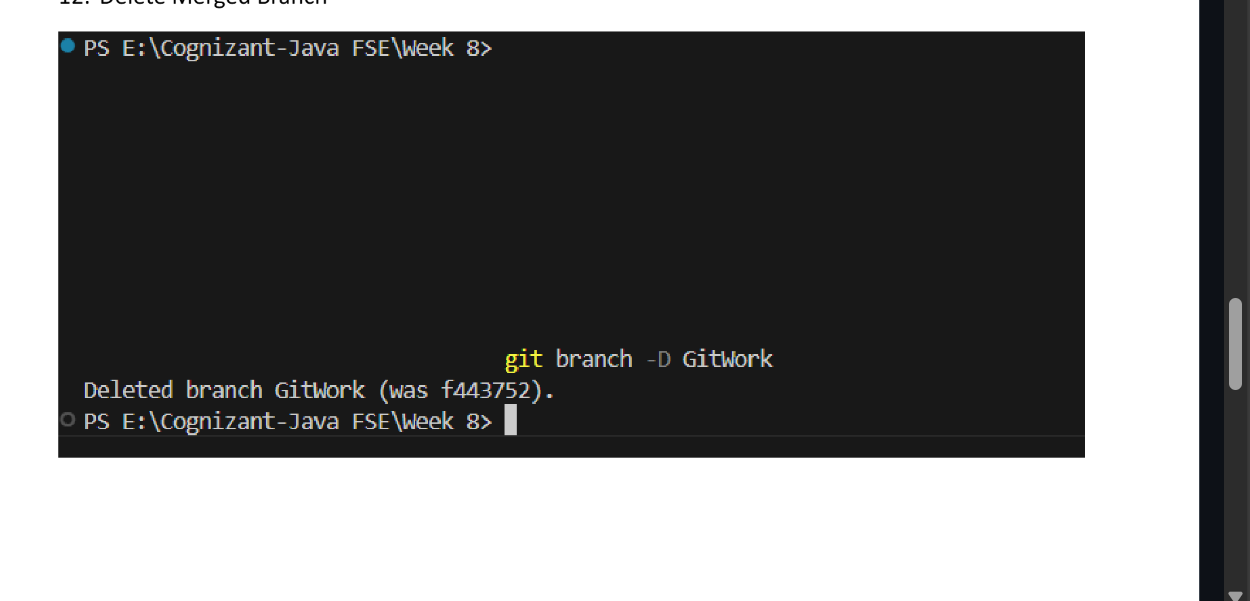
git add hello.xml

git commit -m “Resolve merge conflict in hello.xml”

git status

git branch

git branch -D GitWork



5 - Please follow the instructions to complete the hands-on. Each instruction expects a command for the Git Bash.

1. Verify if master is in clean state.
2. List out all the available branches.
3. Pull the remote git repository to the master
4. Push the changes, which are pending from **“Git-T03-HOL\_002”** to the remote repository.
5. Observe if the changes are reflected in the remote repository.

git checkout

git branch -a

git pull origin main

git push origin main