1. **Create a local repo and link it with remote repo (on github or bitbucket platform)**

A screenshot of a computer program

AI-generated content may be incorrect.

“git init” creates a new local Git repository in the current folder.

“git remote add origin <link>” links that repository to a remote one.

“git branch -M main” renames the current branch form master to main, which is the modern default branch name.

1. **Create a new txt file (with name file1.txt) on the main branch and then add some changes (do 2 commits)**

A computer screen shot of text

AI-generated content may be incorrect.

I staged (using “git add .”) and committed (using “git commit”) twice to capture 2 distinct changes. “git status” helps confirm whether the file is untracked, staged or modified before each commit.

1. **Create a new branch from main with name develop:**
   * **create a new txt file (file2.txt)**
   * **change the name of the file1.txt and then modify the content of the file1.txt**
   * **add all these changes to index area, then add a new content to file2.txt, do commit, run git log**

A screen shot of a computer code

AI-generated content may be incorrect.

Using “git checkout -b” I created and switched to a new branch “develop”. I changed the name of the “file1.txt” to “file1\_dev.txt” and then I modified its content. I intentionally modified file2.txt after staging the changes, with the scope to leave local unstaged edits. As a result, the commit includes only the staged snapshot, while the most recent edits remained in the working directory.

A computer screen shot of a computer program

AI-generated content may be incorrect.

Command “git log” displays the commit history that are reachable from develop branch. Since develop branch was created from main, its history includes all the commits from main (up to the point of branching) plus the new commits I made on develop branch.

* + **run command ls -l, check files content**

A computer screen shot of a computer code

AI-generated content may be incorrect.

“ls -l” command shows the list of files in the current directory, while “cat <filename>” displays the content of the file in the terminal.

1. **Move to main branch:**
   * **What happened? Why? How to solve this? (you must have an issue)**

A screen shot of a computer

AI-generated content may be incorrect.

Git stops switching the branch to prevent overwriting uncommitted changes. This happens because Git protects the work and moving to another branch would overwrite the uncommitted changes. It can be solved by committing or stashing the changes first.

* + **solve the issue and move to main branch,**

After committing the changes in file2.txt, the issue was solved and the command to switch to main branch worked.

A computer screen with text and images

AI-generated content may be incorrect.

* + **run git log, what do you see?**

After running git log, I observed that the main branch does not include the latest commits made on develop branch. This is because the commits exist only in develop branch until they are explicitly merged into main.

A computer screen shot of a computer code

AI-generated content may be incorrect.

* + **run command ls -l, check files content, what difference you see?**

Also, the main branch does not contain the file2.txt, since that was created on develop branch. The content of file1.txt on main remains unchanged compared to before, because the rename and modifications were done only on develop branch.

A screen shot of a computer

AI-generated content may be incorrect.

1. **Create a new file (. gitignore). Add to it the extension of the files what you want to ignore, then commit.**

**A screenshot of a computer screen

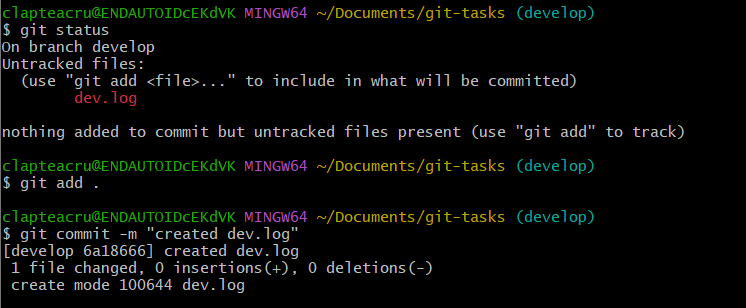
AI-generated content may be incorrect.**

I created the “.gitignore” file and added the rule to ignore all .log files.

* + **move to develop**

****

* + **create a new file that you want to ignore, commit.**

****

* + **What happened? Why?**

The new file still gets tracked and shows up in the commit, because the “.gitignore” file was created on the main branch only, and at this point it does not apply to the develop branch. In order to apply, the .gitignore commit from main must be merged into develop branch.