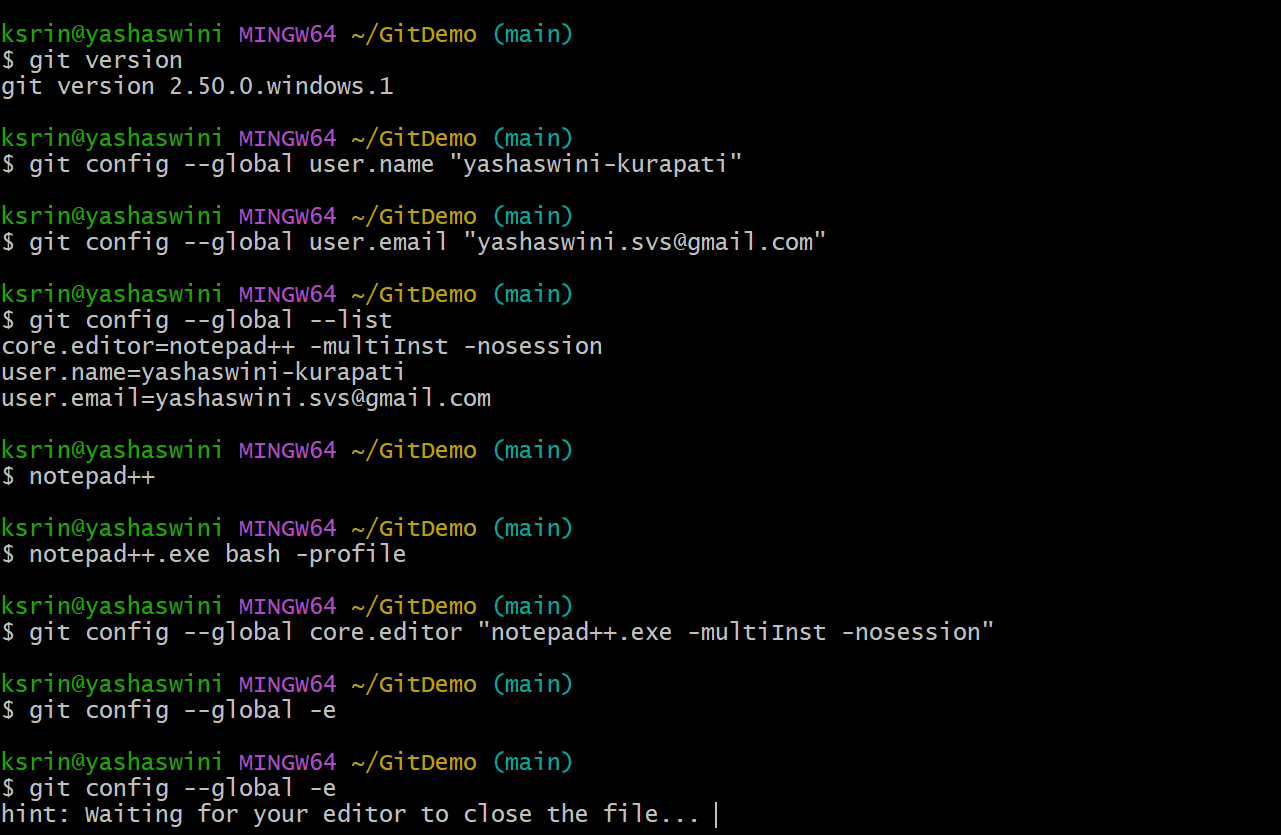
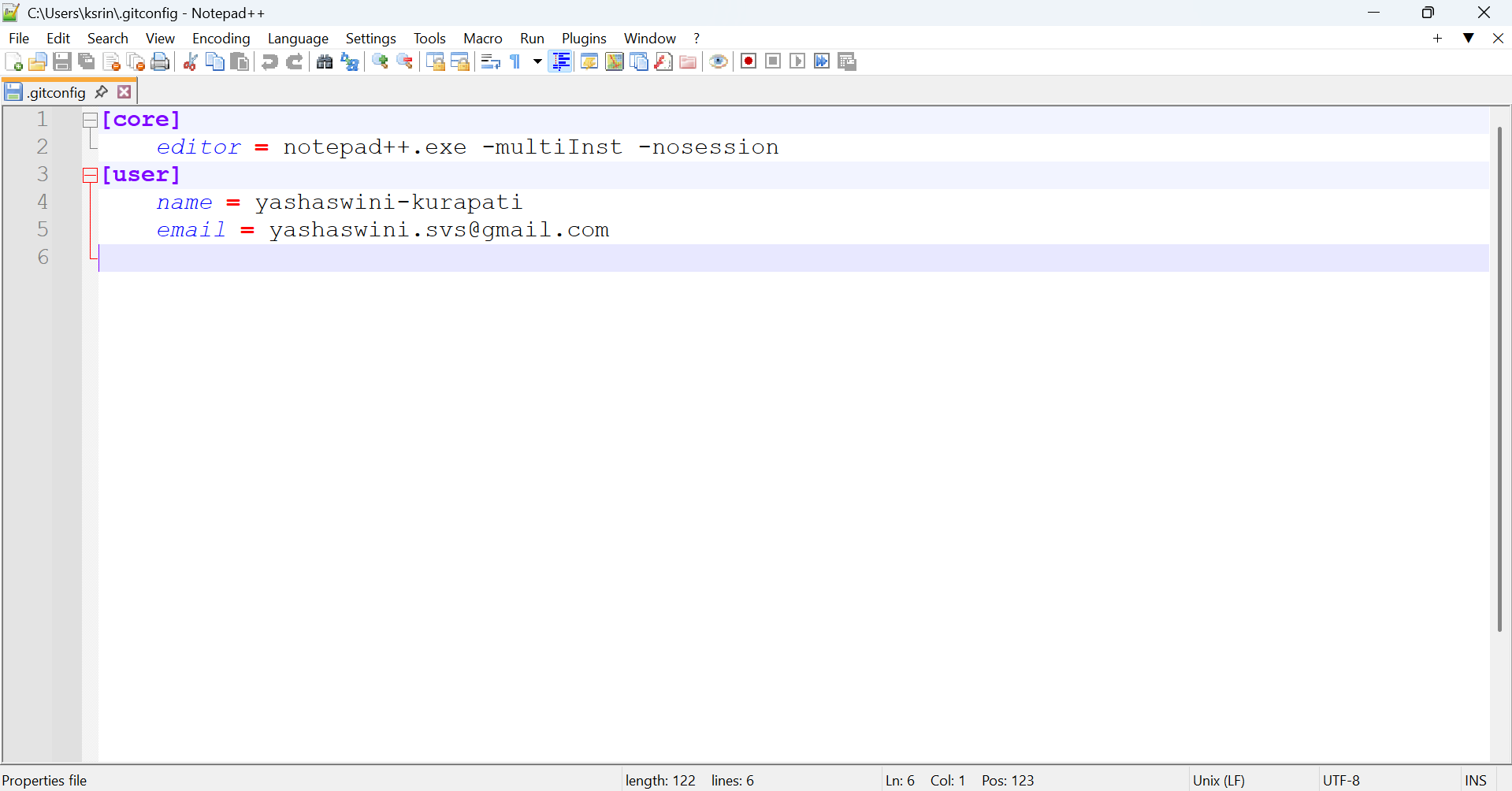
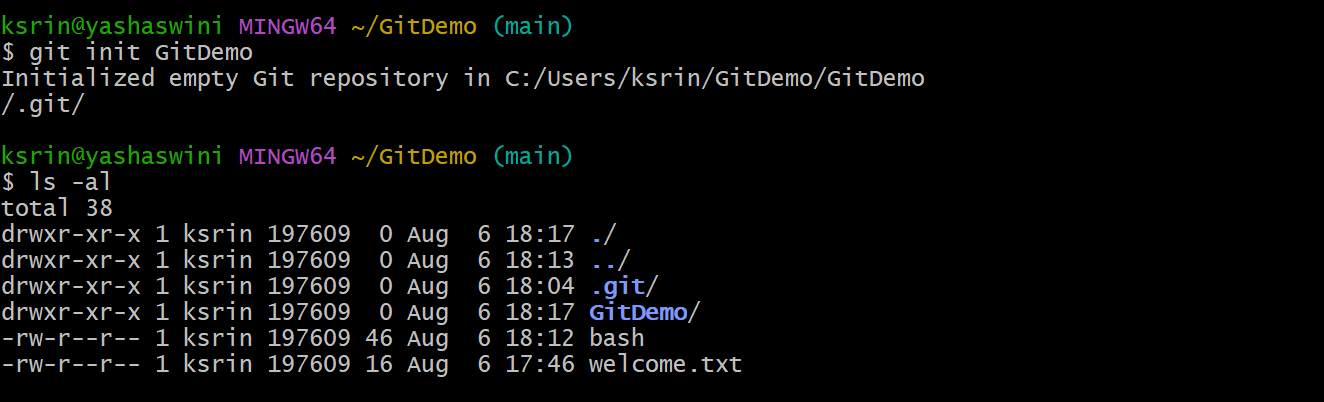
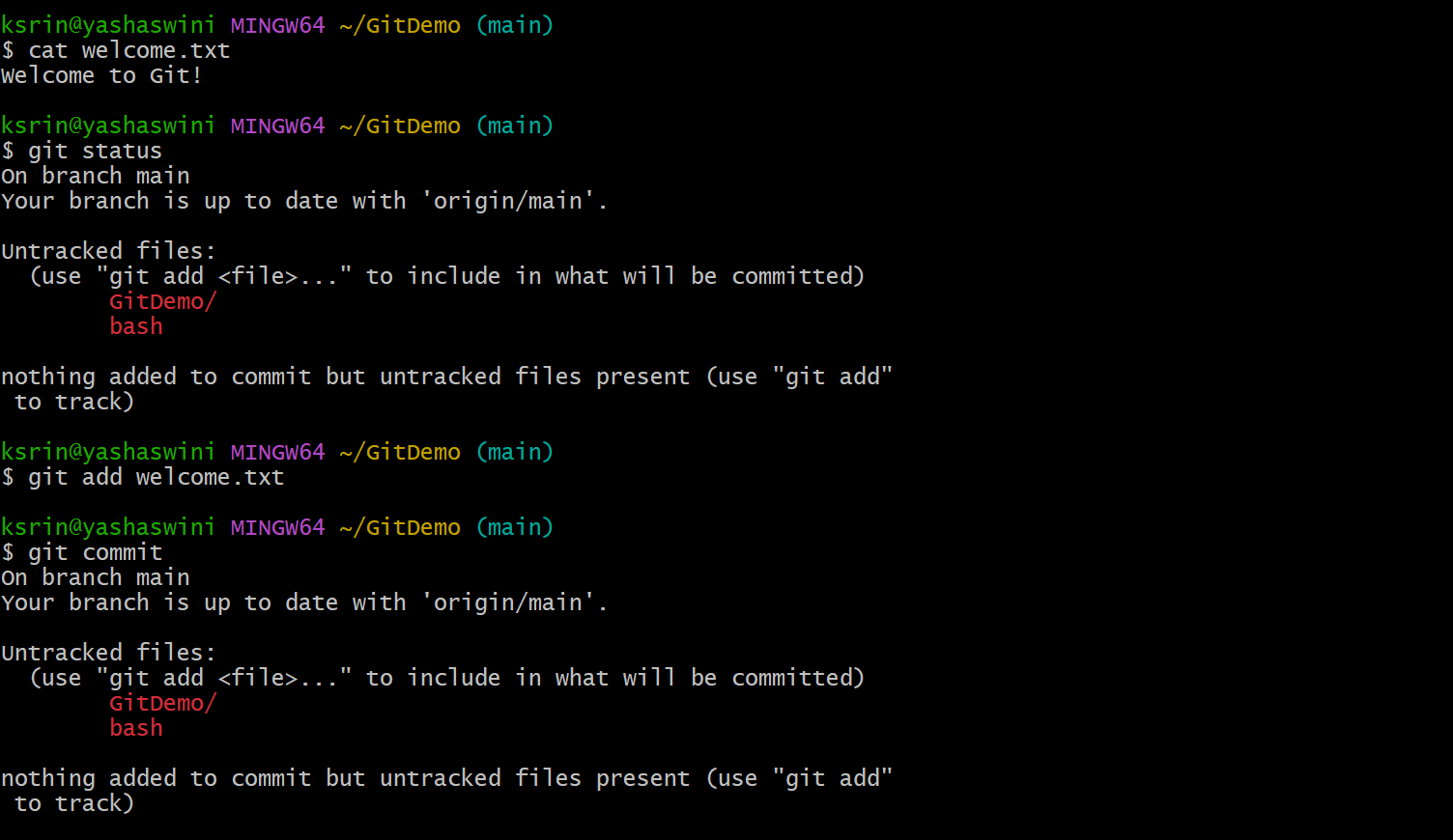
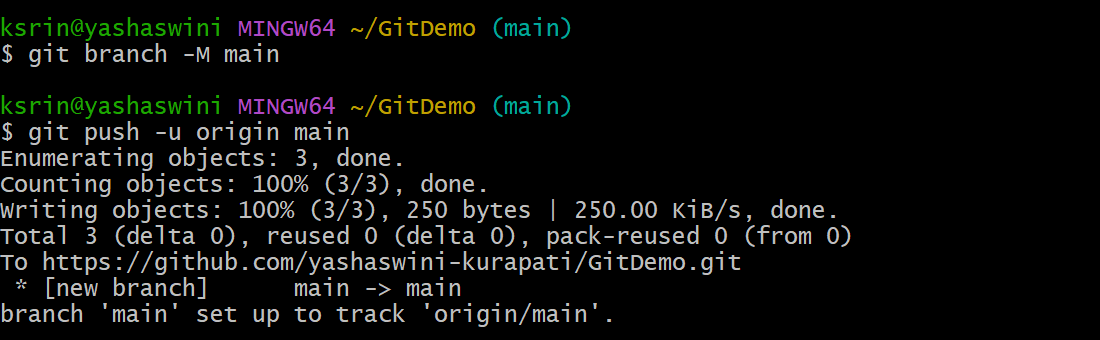
1. **Git-HOL**

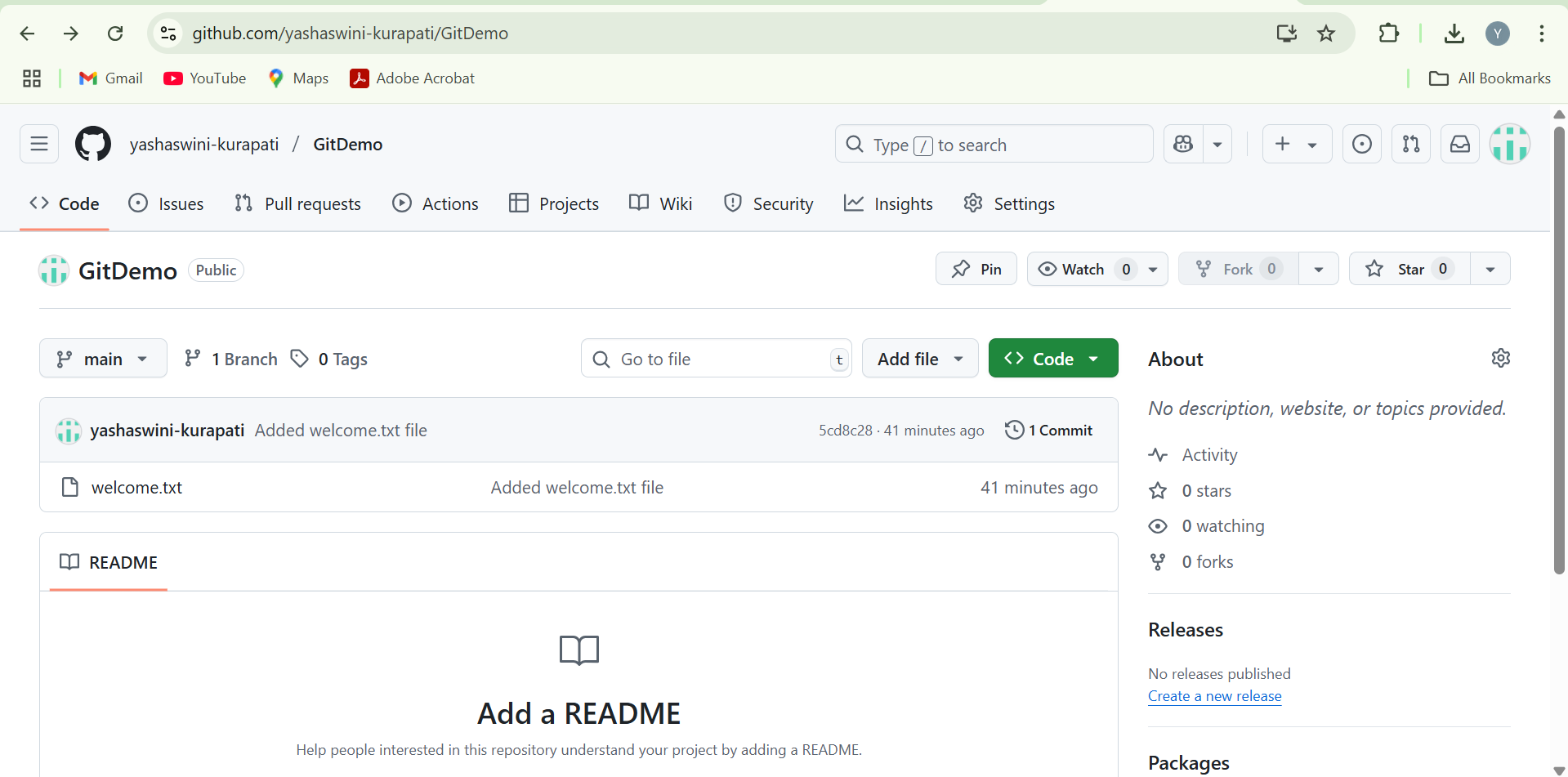


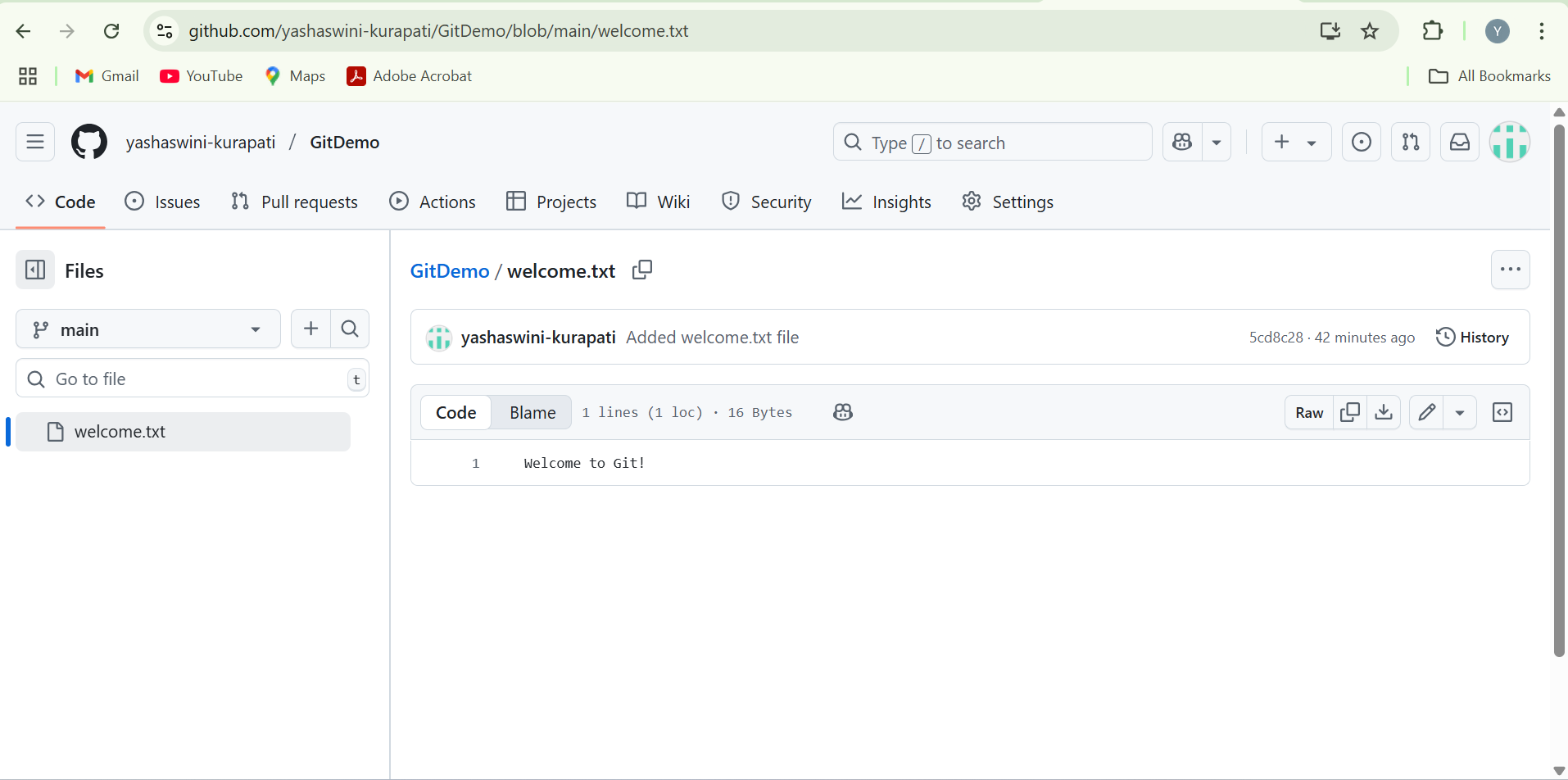












1. **Git-HOL**

**What is .gitignore?**

.gitignore is a special file in a Git repository that tells Git which files or directories to **ignore** — i.e., they will not be tracked or committed.

This is useful for:

* Ignoring temporary files (.log, .tmp, .class, etc.)
* Skipping system files (e.g., Thumbs.db, .DS\_Store)
* Avoiding committing environment-specific files or build artifacts

**How to Ignore Files Using .gitignore**

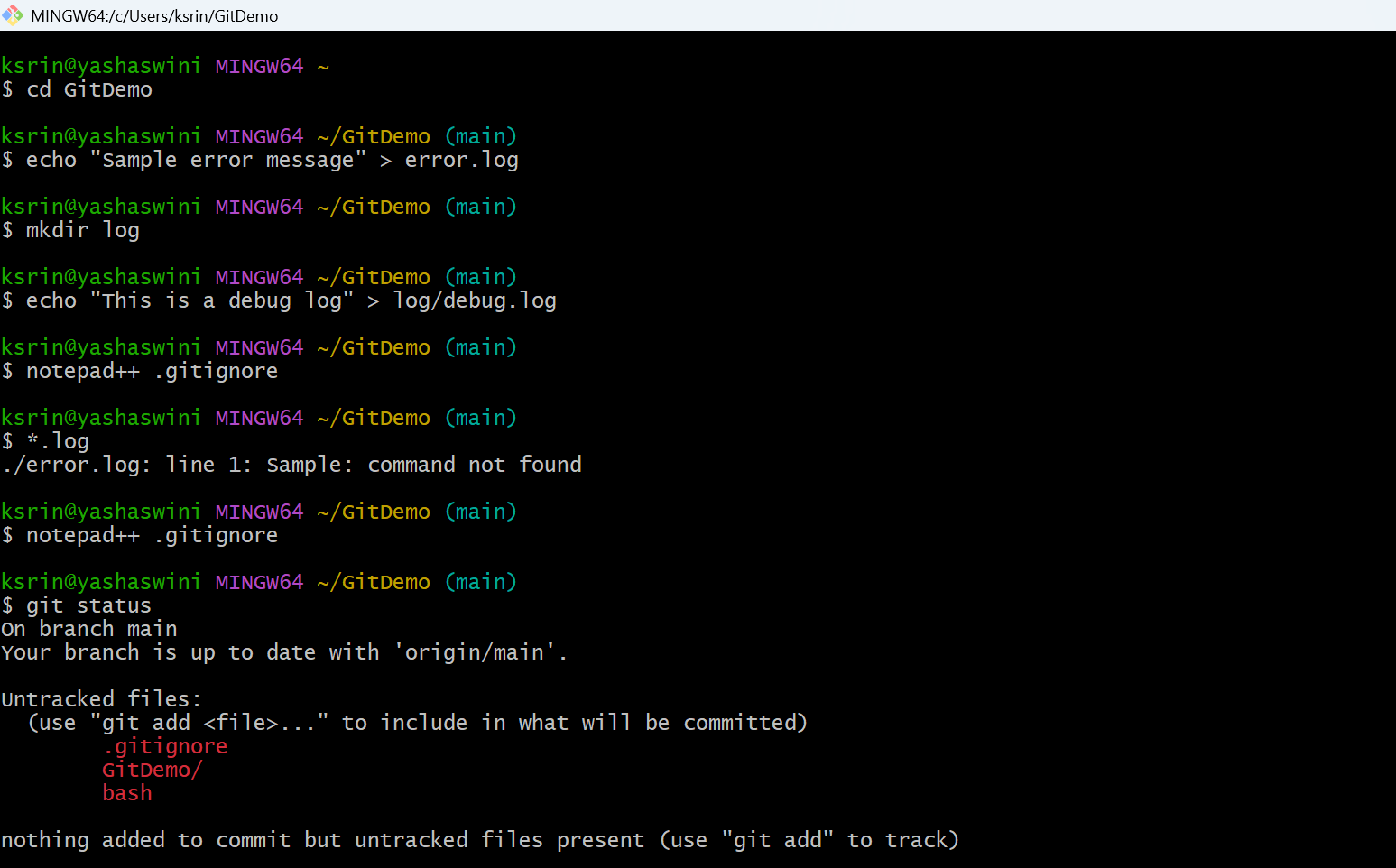
You can create a .gitignore file in your Git repository and list the file names, extensions, or folder names that Git should ignore.

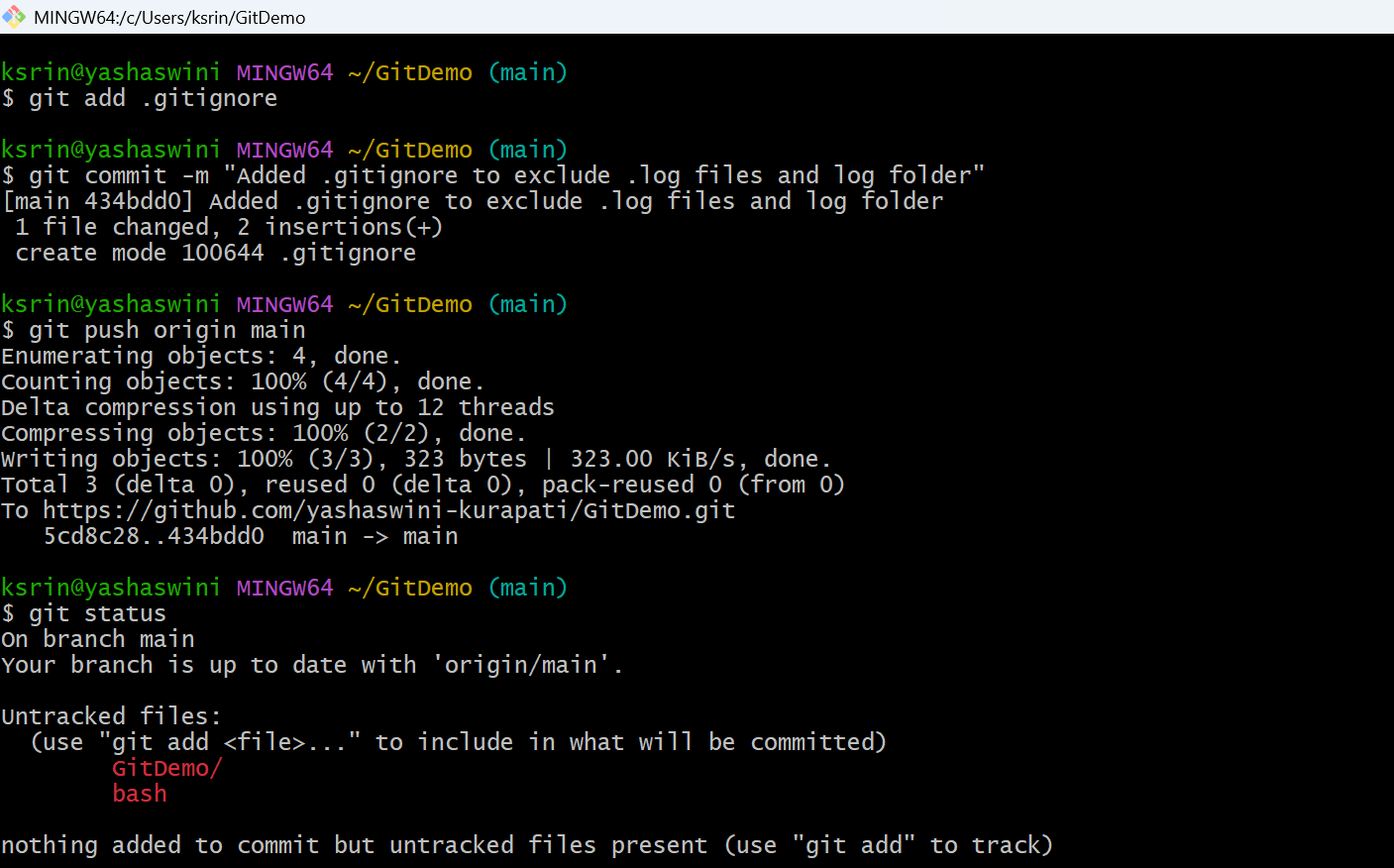
# Ignore all .log files

\*.log

# Ignore entire log folder

log/





1. **Git-HOL**

**1. Explain Branching and Merging**

**Branching** in Git is a powerful feature that allows developers to diverge from the main line of development and work on features or fixes without affecting the main (or master) branch. Each branch represents an independent line of development.

**Merging** is the process of integrating changes from one branch into another. Typically, once the feature development is complete on a separate branch, it is merged back into the main branch.

**Example Use Case:**

* Developer creates a new branch called feature-login.
* Makes changes, commits, and tests the login feature.
* Once verified, the feature-login branch is merged into the main branch.

**2. Explain About Creating a Branch Request in GitLab**

A **branch request** in GitLab usually refers to **creating a new branch** to begin work on a new feature or bug fix. This is typically done directly from the GitLab web interface or via Git CLI.

**Steps to create a branch in GitLab:**

* Go to your GitLab repository.
* Click on **Repository > Branches**.
* Click **New Branch**.
* Name your branch (e.g., feature-x) and choose the base branch (usually main).
* Click **Create Branch**.

This helps in organizing tasks and isolating code changes.

**3. Explain About Creating a Merge Request in GitLab**

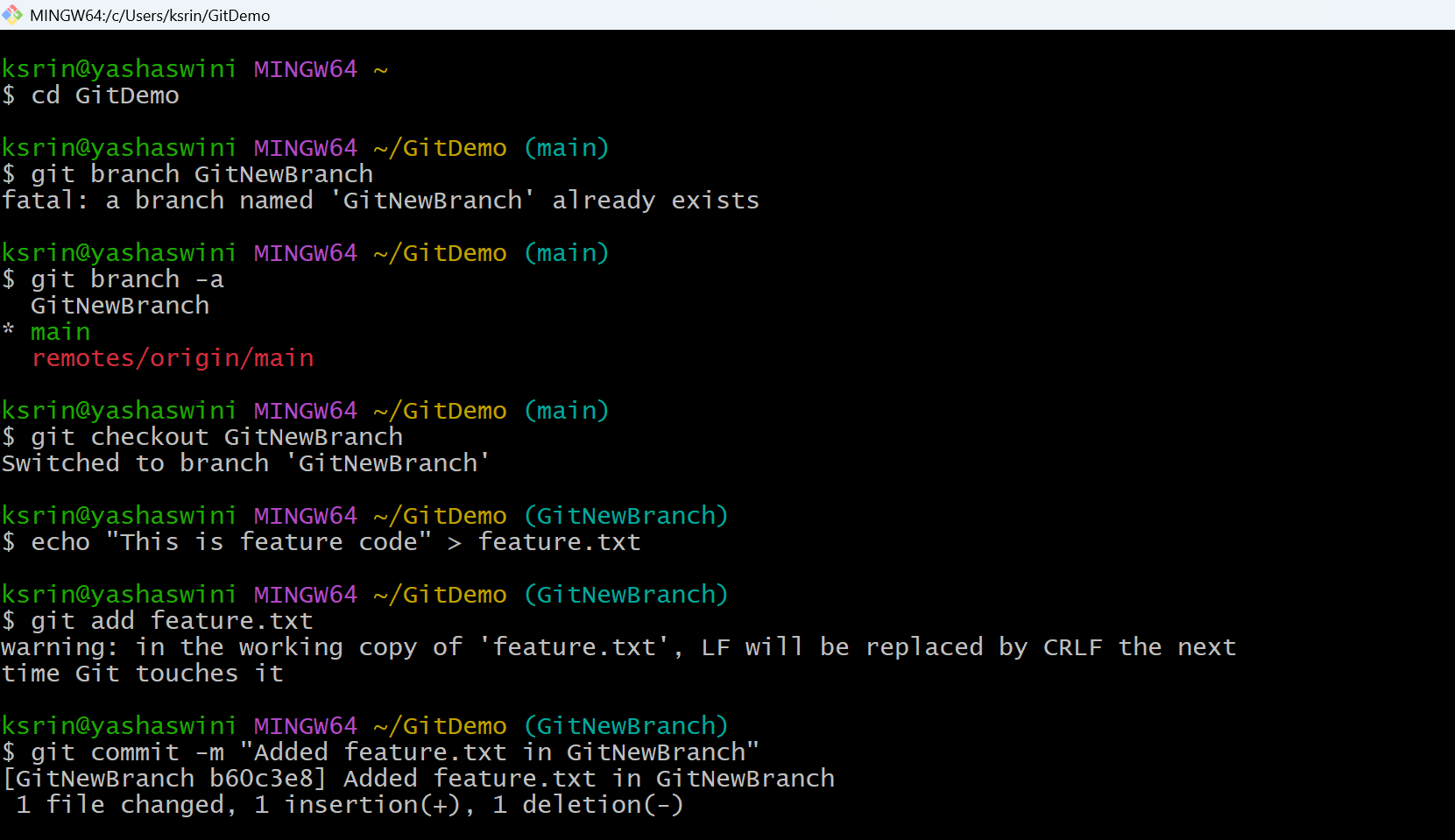
A **merge request (MR)** in GitLab is a request to merge changes from one branch (usually a feature or bugfix branch) into another (typically the main or develop branch). It is used to review and approve changes before merging.

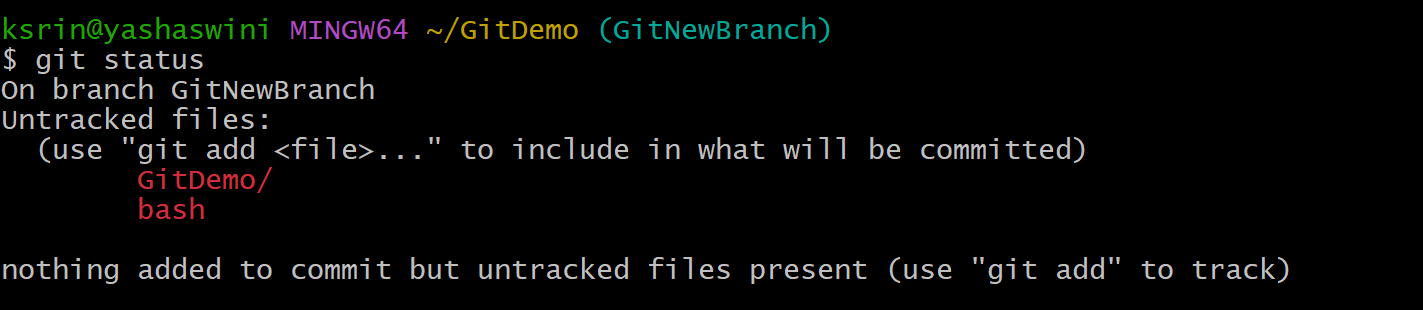
**Steps to create a merge request in GitLab:**

* Push your branch to GitLab.
* Go to **Merge Requests > New Merge Request**.
* Select the source branch (your branch) and target branch (main or develop).
* Click **Compare branches and continue**.
* Add a title and description for the merge request.
* Assign reviewers and click **Submit Merge Request**.

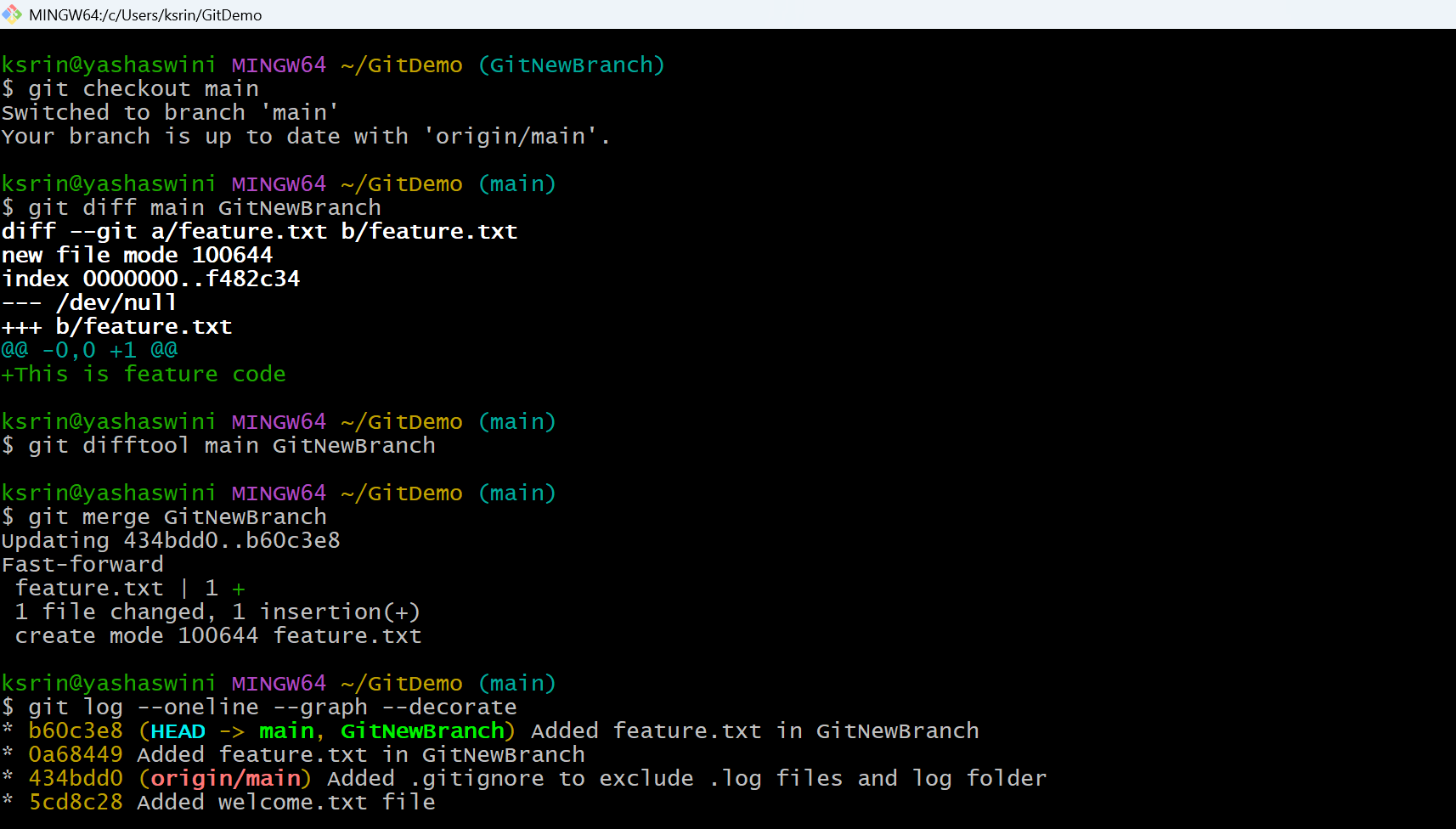
This is crucial for **code review**, **collaboration**, and **maintaining code quality**.

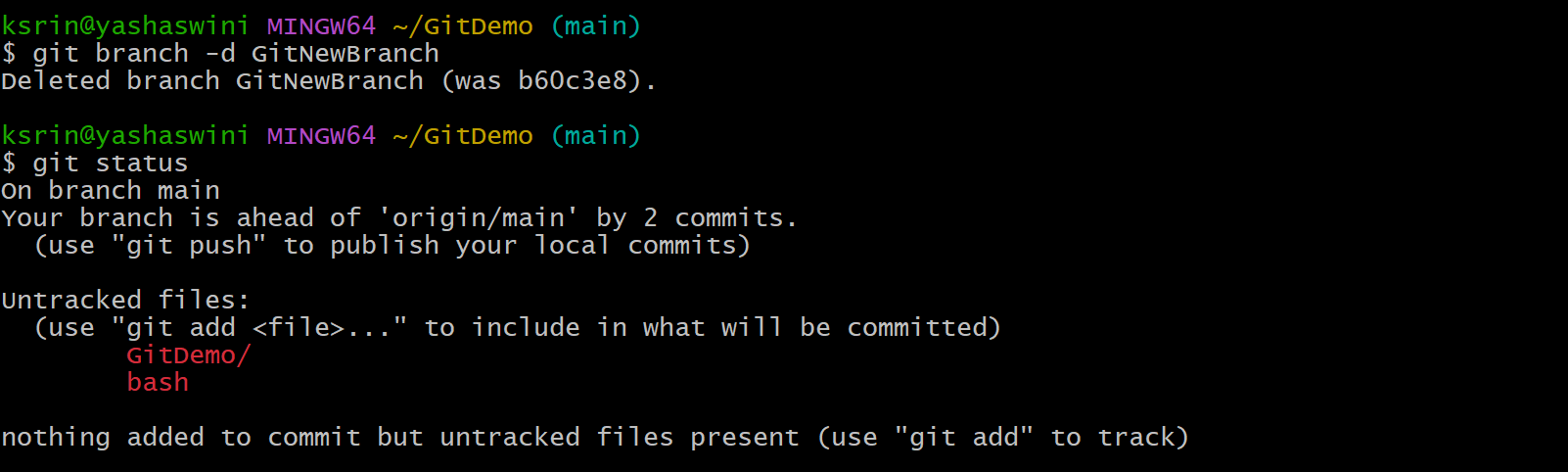
**Branching**





**Merging**



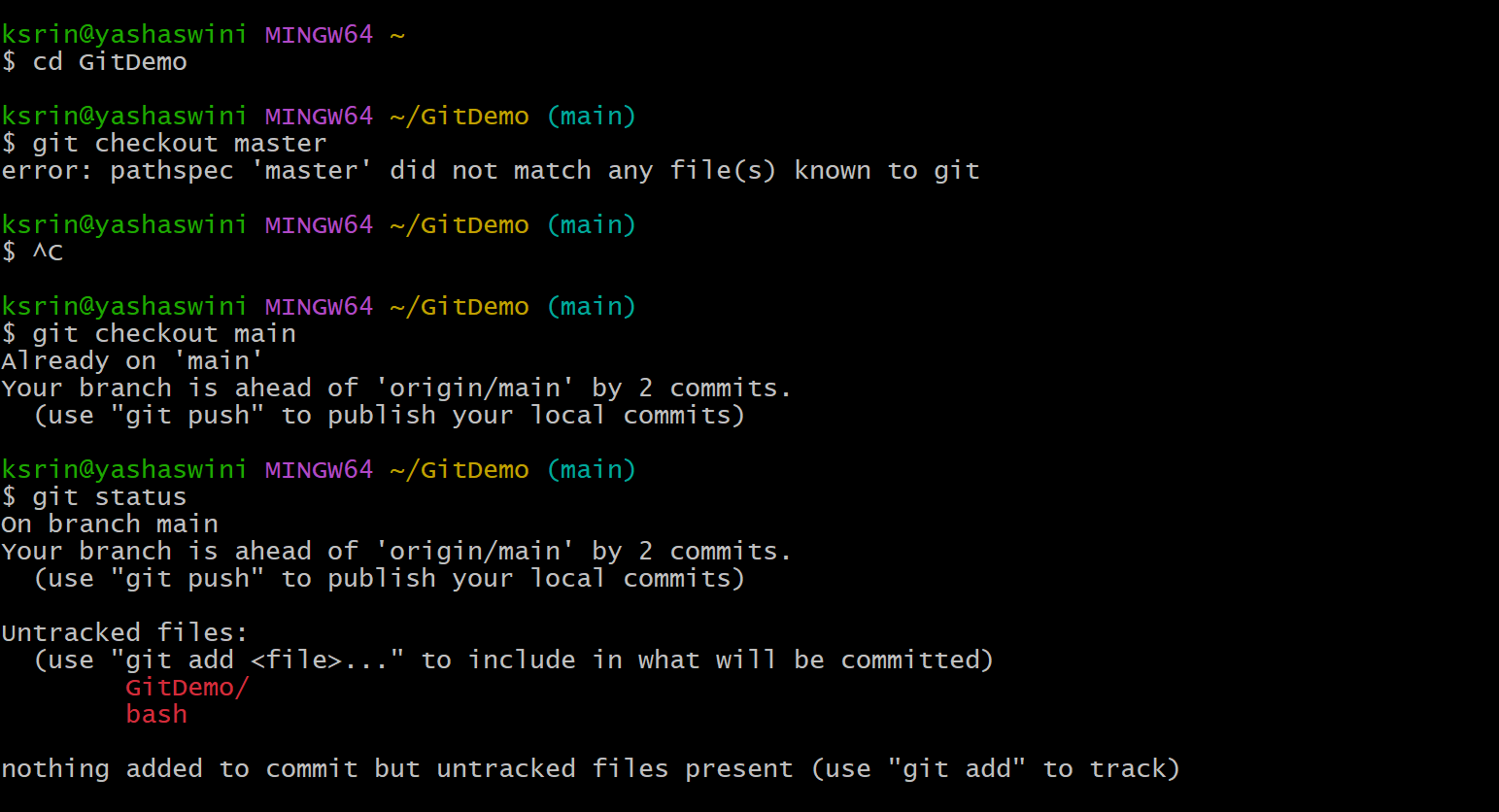


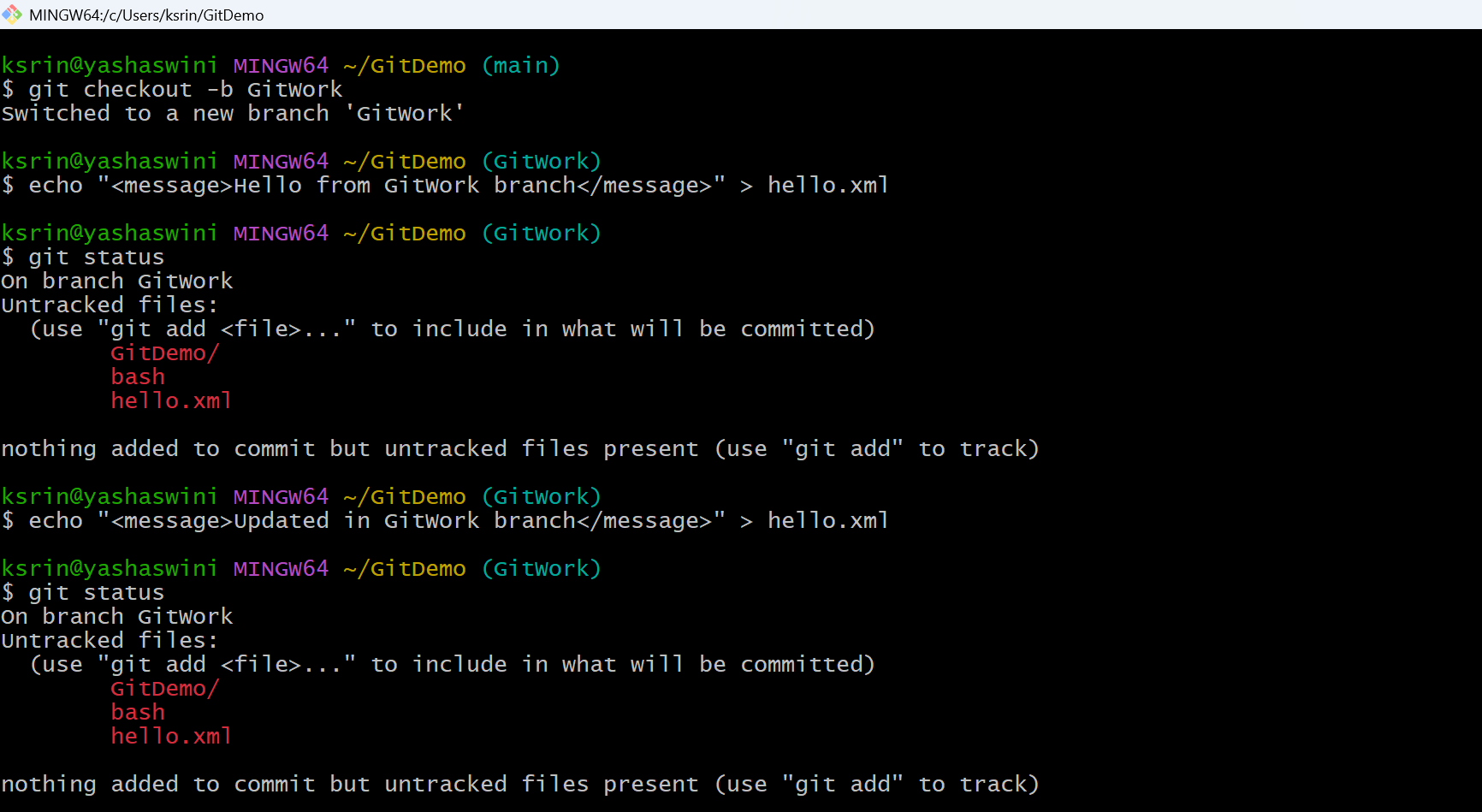
4. **Git-HOL**

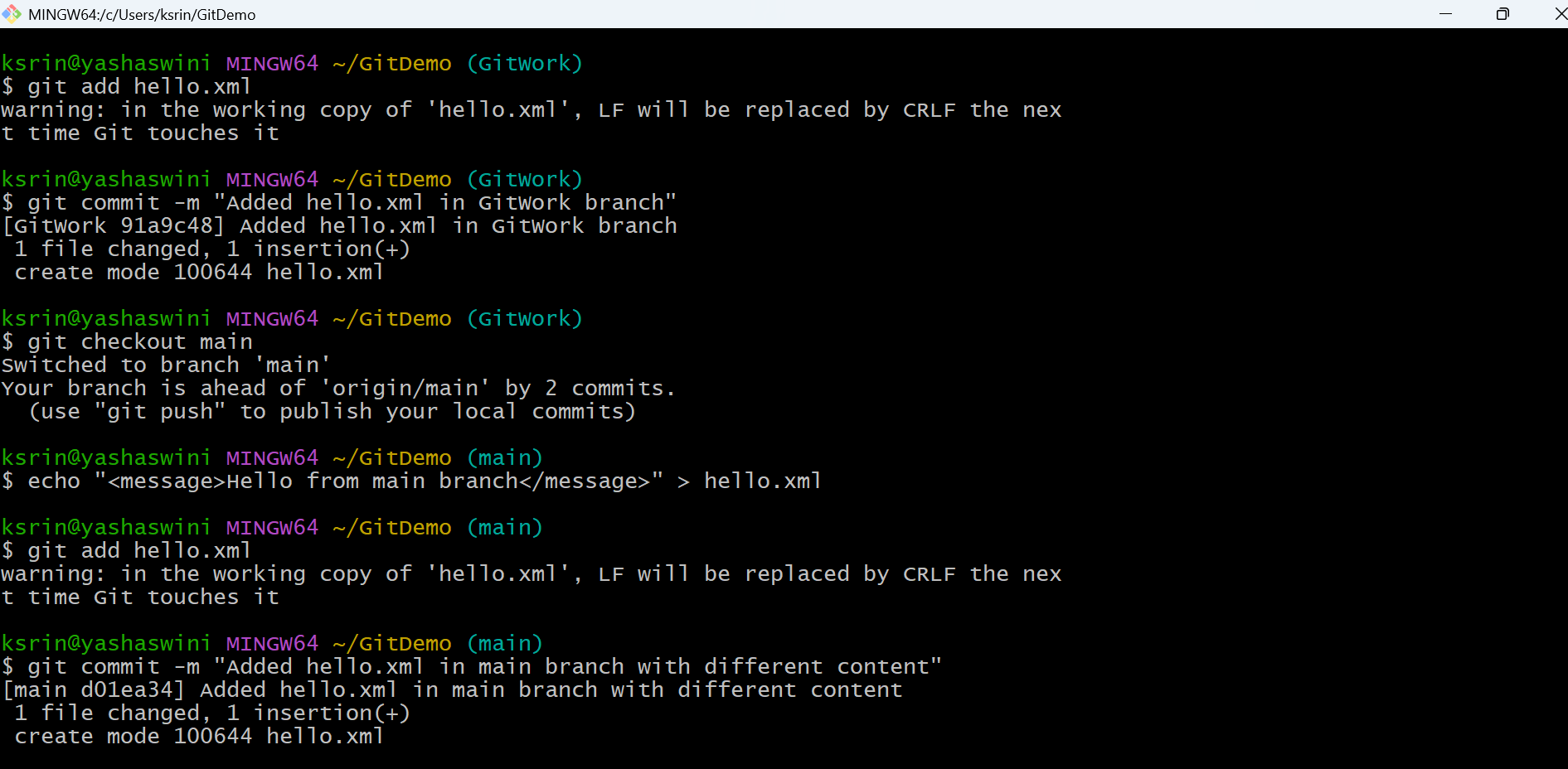
**Explain how to resolve the conflict during merge.**

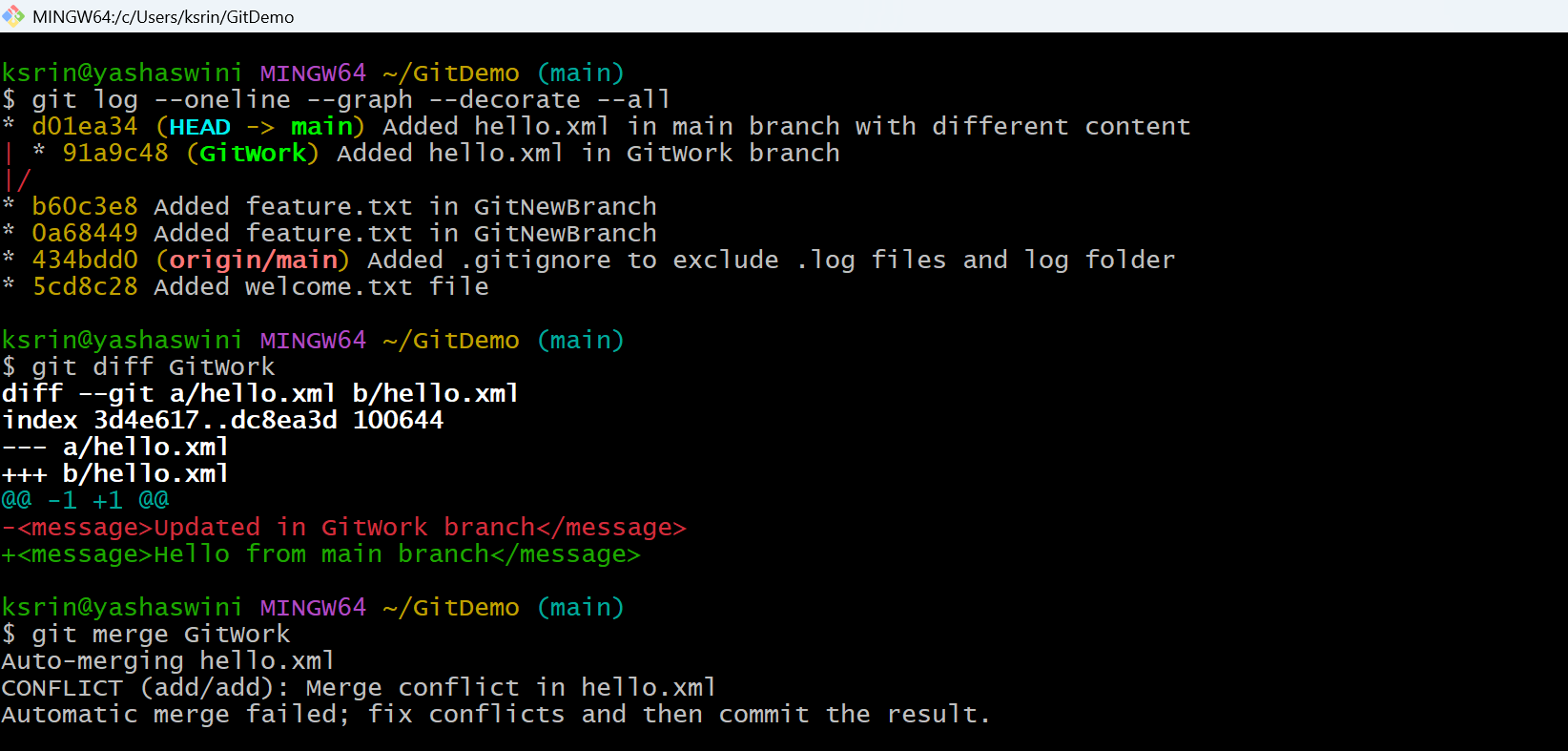
To resolve a merge conflict in Git:

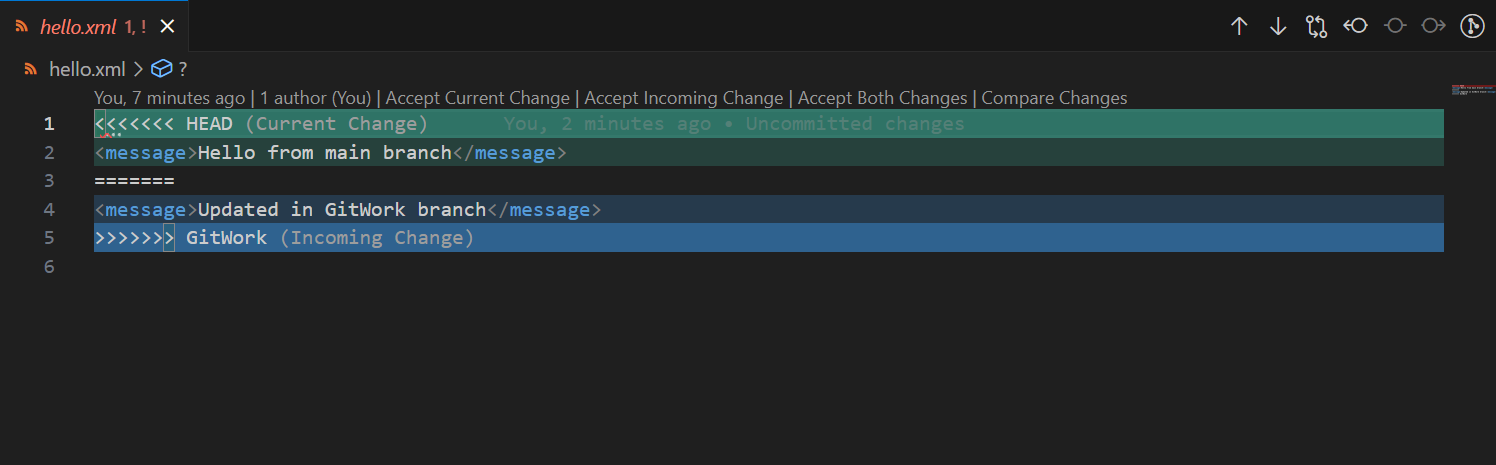
When Git shows conflict markers (<<<<<<<, =======, >>>>>>>), open the affected file, decide which changes to keep (or combine them), remove the markers, save the file, then run git add <file> and git commit to complete the merge.

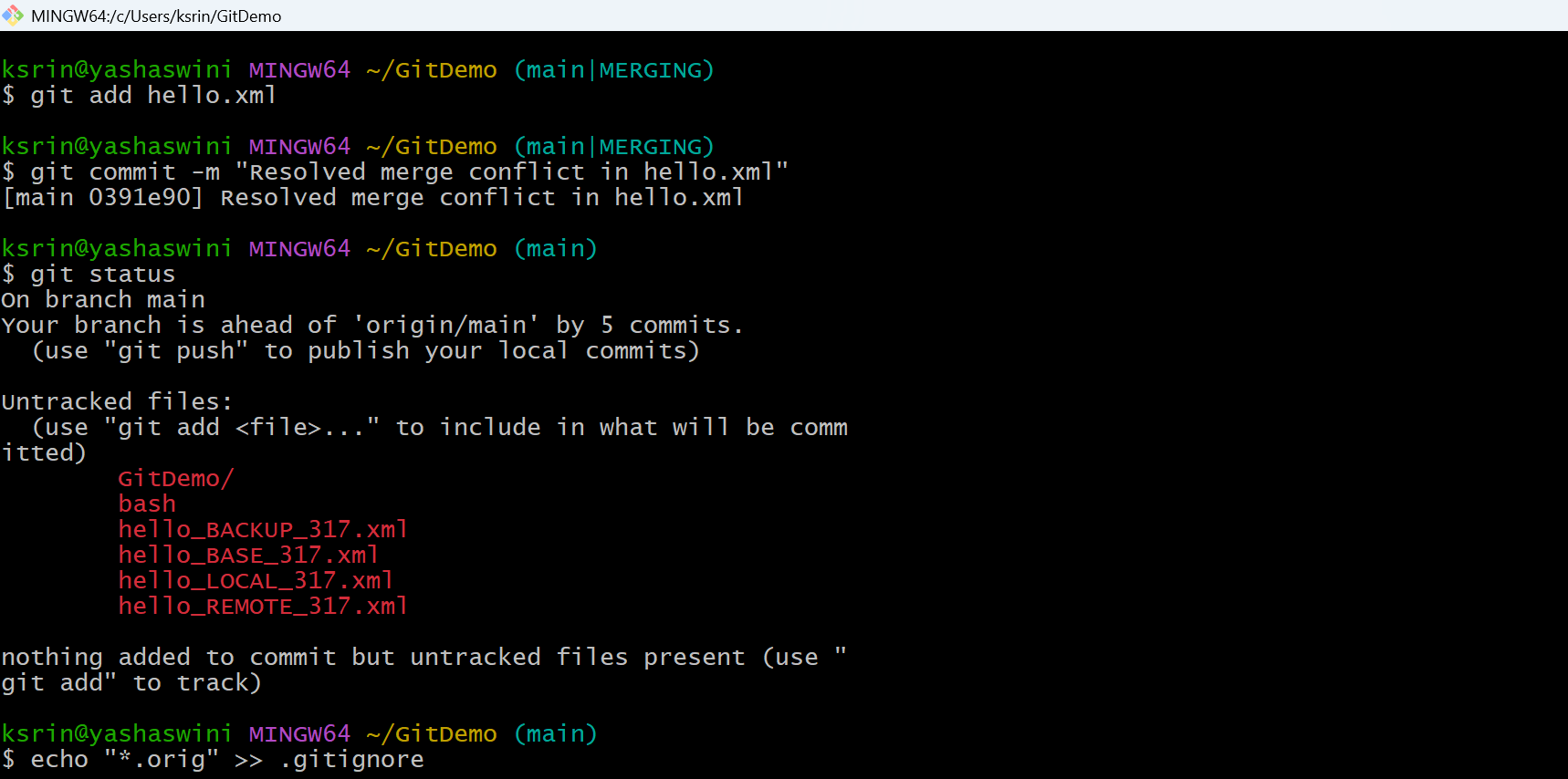


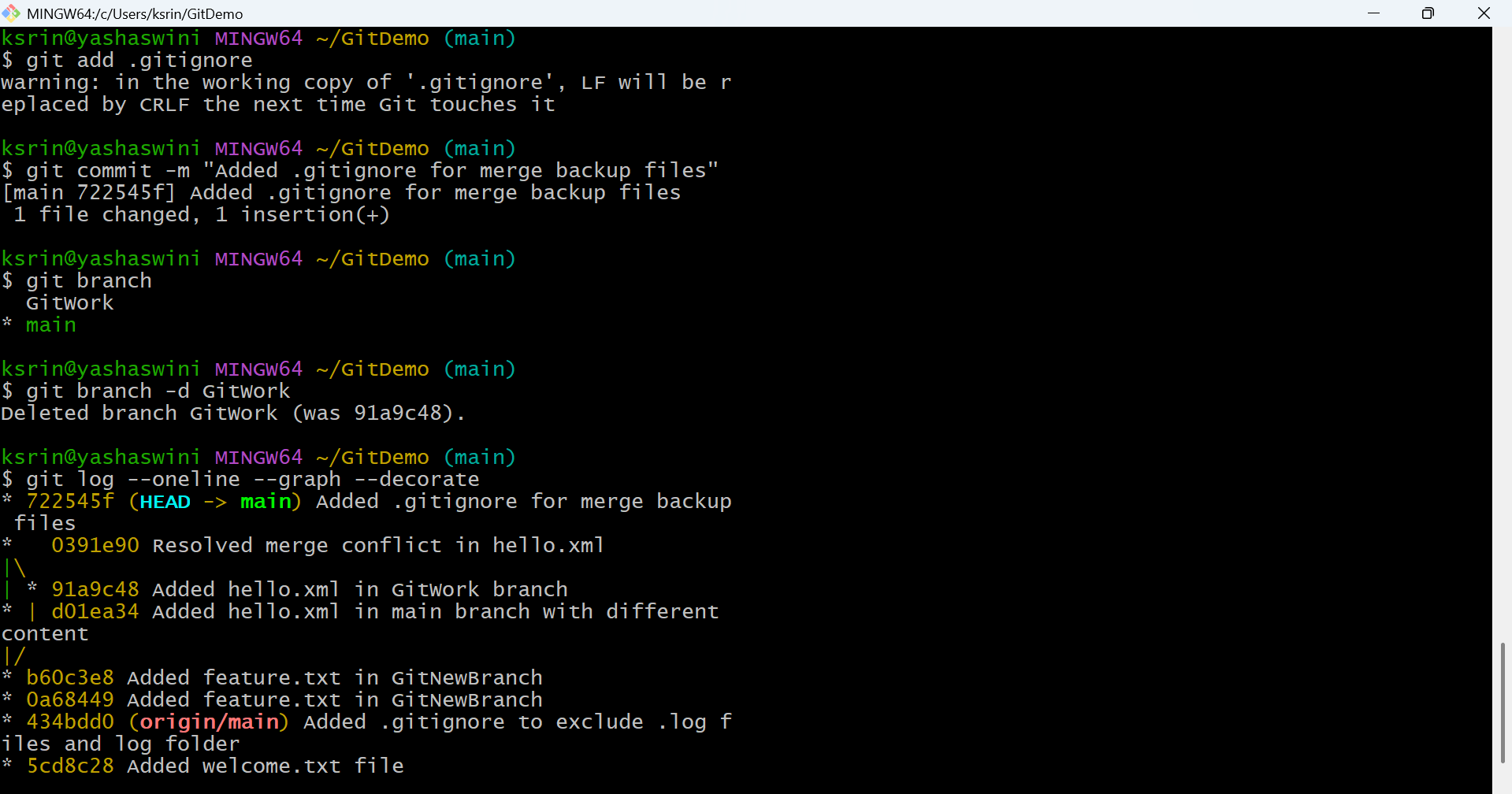








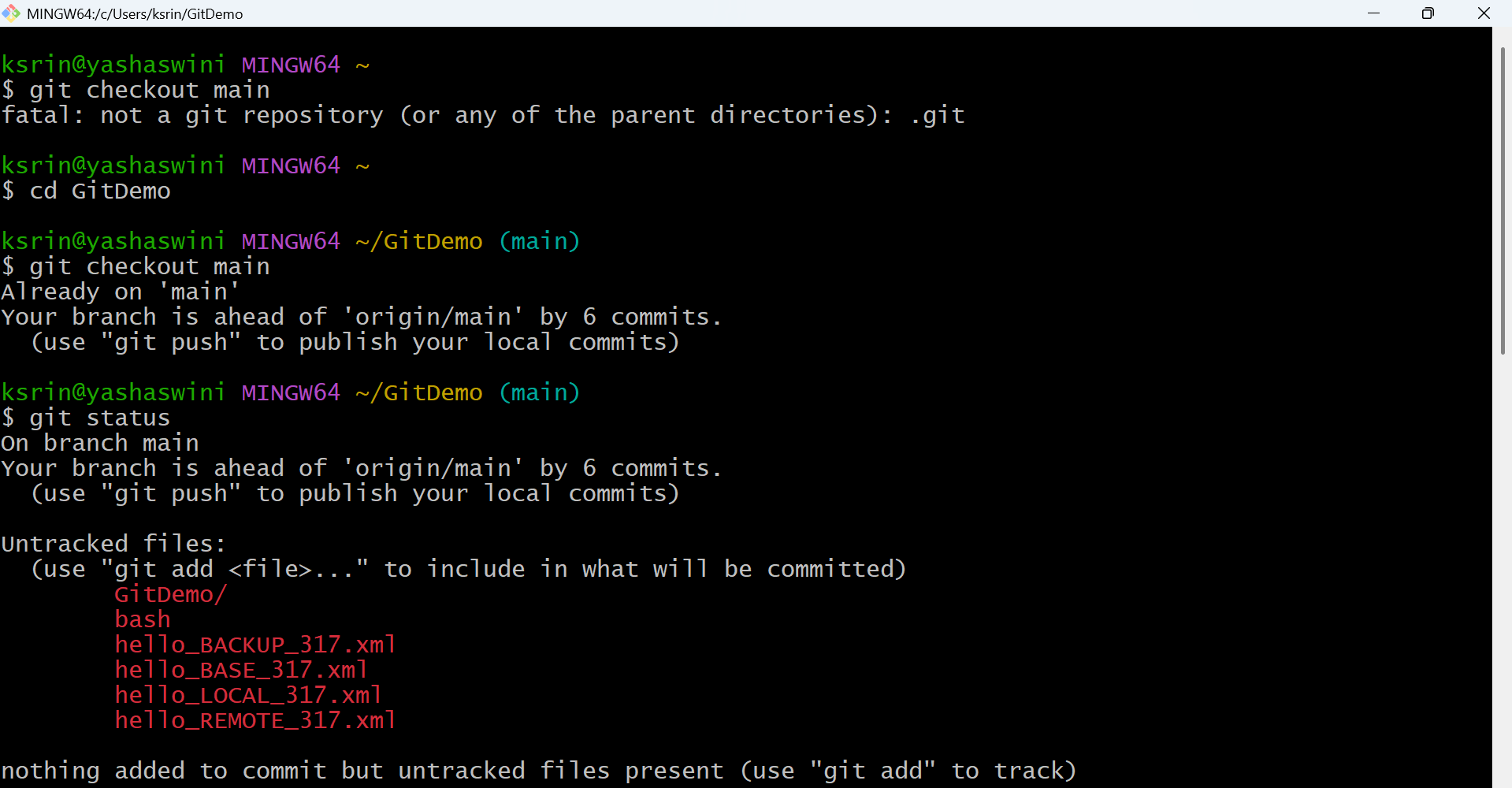


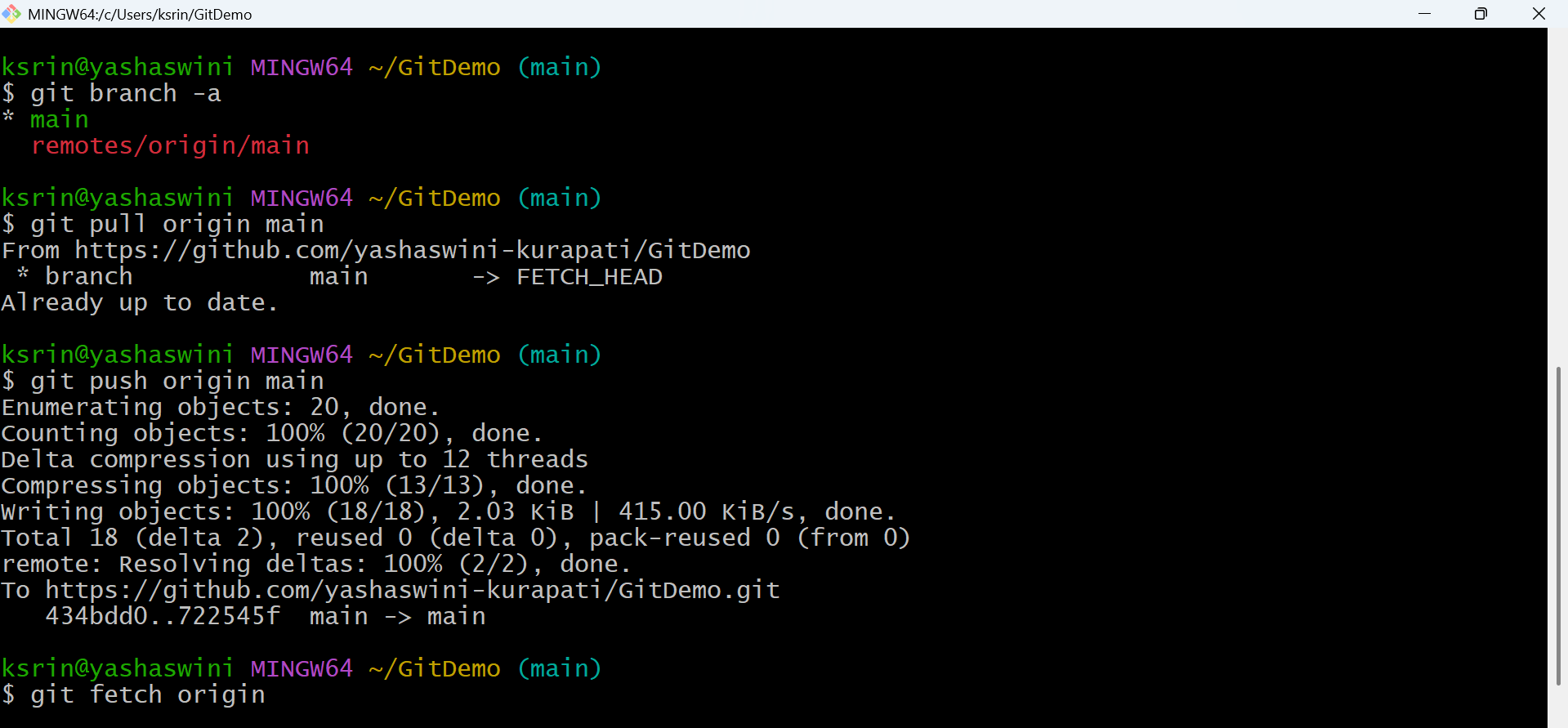


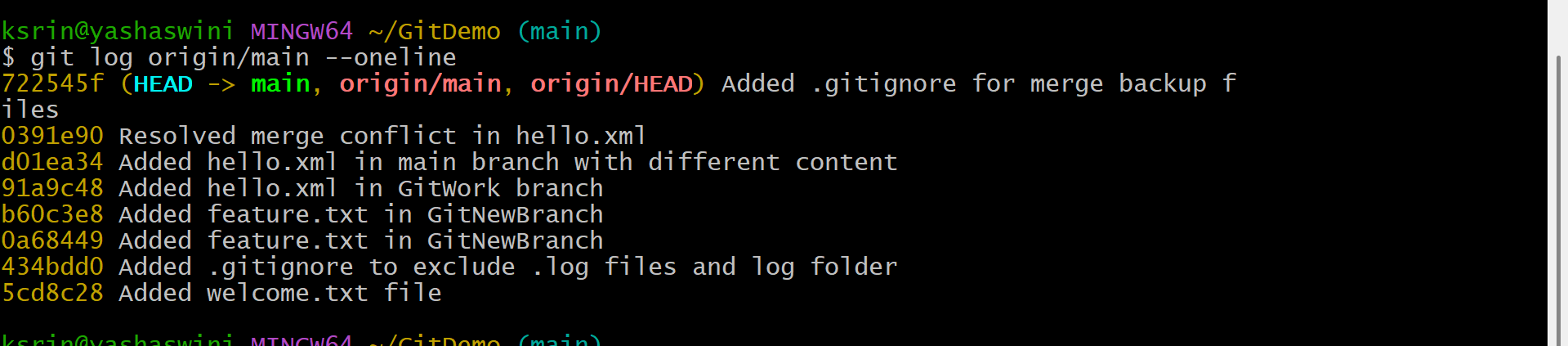
5. **Git-HOL**

**Explain how to clean up and push back to remote Git**

* To clean up and push back to remote Git:
* Ensure your branch is clean (git status), pull the latest changes from the remote (git pull origin main), resolve any conflicts if needed, then push your local commits to the remote using git push origin main.

****

****

****