**Madiha Aimon Tappal**

[madihaaimon@gmail.com](mailto:madihaaimon@gmail.com)

**Data Engineering Batch – 1**

**Day – 24 Assignment**

**Azure Databricks**

**Azure Repos:**

* **The secure home for your code and project assets.**
* **Unlimited private Git repositories:** Host your codebase and collaborate with your team using branching, pull requests, and code reviews.
* **Version control:** Track changes, revert to previous versions if needed, and maintain a clear history of your project's evolution.
* **Integrations:** Connect with Pipelines for automated builds and deployments based on code changes.

Azure Repos, formerly known as Visual Studio Team Services (VSTS) Git repositories, offers a robust platform for version control and code collaboration. To answer your question about the "full theory" of Azure Repos, I'll break it down into key aspects:

**Concept and Functionality:**

* **Git-based:** Azure Repos leverages Git, a popular distributed version control system (DVCS). This means each developer has a complete copy of the codebase, enabling offline work and efficient branching and merging.
* **Version Control:** Track changes, revert to previous versions, and identify who made modifications with detailed history.
* **Code Collaboration:** Share code, create pull requests for review, and discuss changes through comments and threads.
* **Security and Access Control:** Define granular permissions for different users and teams, ensuring code security and access control.
* **Integrations:** Seamlessly integrate with other Azure DevOps services like Azure Pipelines for automated builds, testing, and deployments, and Azure Boards for project management and task tracking.

**Benefits and Features:**

* **Free private repositories:** Host unlimited private Git repositories for free with basic features.
* **Enhanced security:** Benefit from robust security features like role-based access control, two-factor authentication, and encryption.
* **Scalability:** Accommodate large projects and teams with ease.
* **Rich integrations:** Streamline your development workflow with seamless integration across Azure DevOps services.
* **Advanced features:** Utilize features like branch policies, code search, and continuous integration/continuous delivery (CI/CD) pipelines for efficient development.

**Learning Resources:**

* **Microsoft Documentation:** <https://learn.microsoft.com/en-us/azure/devops/repos/?view=azure-devops>
* **Azure Repos Tutorial:** <https://www.techtarget.com/searchcloudcomputing/tutorial/An-Azure-Repos-tutorial-to-build-a-container-image>
* **Azure DevOps Website:** <https://learn.microsoft.com/en-us/azure/devops/repos/?view=azure-devops>

**Beyond the Basics:**

While the above covers the core functionalities, Azure Repos offers deeper capabilities for advanced users and specific workflows:

* **Pull request reviews:** Utilize code review tools and workflows for in-depth code analysis and feedback.
* **Branching strategies:** Implement efficient branching strategies like Gitflow or feature branching to manage complex development cycles.
* **Continuous integration/continuous delivery (CI/CD):** Automate build, test, and deployment processes with Azure Pipelines for faster and more reliable software delivery.
* **Large file support:** Manage large files efficiently with Git LFS integration.

**Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018**

Azure Repos offers free unlimited private Git repositories, making it easy to try. Git is the most commonly used version control system today and is quickly becoming the standard for version control. It is a distributed version control system, meaning that your local copy of code is a complete version control repository. These fully functional local repositories make it easy to work offline or remotely. You can commit your work locally, and then sync your copy of the repository with the copy on the server.

To get started, see [Sign up for Azure Repos](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/sign-up-invite-teammates?view=azure-devops), [Code with Git](https://learn.microsoft.com/en-us/azure/devops/user-guide/code-with-git?view=azure-devops&preserve-view=true), and [What is Azure Repos?](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops).

**Key concepts**

* [Branches & branch policies](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies-overview?view=azure-devops)
* [Branch organization](https://learn.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops)
* [Forks](https://learn.microsoft.com/en-us/azure/devops/repos/git/forks?view=azure-devops)
* [History](https://learn.microsoft.com/en-us/azure/devops/repos/git/history?view=azure-devops)
* [Pull requests](https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops)

**Git**

Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning that your local copy of code is a complete version control repository. These fully functional local repositories make it easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server.

Git in Azure Repos is standard Git. You can use the clients and tools of your choice, such as Git for Windows, Mac, partners' Git services, and tools such as Visual Studio and Visual Studio Code.

* [Connect your favorite development environment](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops#connect-your-favorite-development-environment)
* [Review code with pull requests](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops#review-code-with-pull-requests)
* [Protect branches with policies](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops#protect-branches-with-policies)
* [Extend pull request workflows with pull request status](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops#extend-pull-request-workflows-with-pull-request-status)
* [Isolate code with forks](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops#isolate-code-with-forks)

**Connect your favorite development environment**

Connect your favorite development environment to Azure Repos to access your repos and manage your work. Share your code using:

* [Command-line](https://learn.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-cmdline?view=azure-devops)
* [Visual Studio Code](https://marketplace.visualstudio.com/vscode)
* [Visual Studio](https://learn.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-vs?view=azure-devops)
* [Xcode](https://learn.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-xcode?view=azure-devops)
* [Eclipse](https://learn.microsoft.com/en-us/previous-versions/azure/devops/all/java/download-eclipse-plug-in)
* [IntelliJ](https://learn.microsoft.com/en-us/previous-versions/azure/devops/all/java/download-intellij-plug-in)

**Review code with pull requests**

Review code with your team and make sure that changes build and pass tests before it gets merged.

* [Create a pull request](https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops)
* [Link work items to pull requests](https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops#link-work-items)
* [Set up branch policies](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#build-validation)
* [Squash merge pull requests](https://learn.microsoft.com/en-us/azure/devops/repos/git/merging-with-squash?view=azure-devops)
* [Git branch and pull request workflows](https://learn.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops)
* [Leave comments or vote on changes](https://learn.microsoft.com/en-us/azure/devops/repos/git/review-pull-requests?view=azure-devops)

**Extend pull request workflows with pull request status**

Pull requests and branch policies enable teams to enforce many best practices related to reviewing code and running automated builds. But many teams have other requirements and validations to perform on code. To cover these individual and custom needs, Azure Repos offers pull request statuses.

Pull request statuses integrate into the PR workflow. They allow external services to programmatically sign off on a code change by associating simple success/failure information with a pull request.

* [Pull request status overview](https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-request-status?view=azure-devops)
* [Create a PR status server with Node.js](https://learn.microsoft.com/en-us/azure/devops/repos/git/create-pr-status-server?view=azure-devops)
* [Use Azure Functions to create custom branch policies](https://learn.microsoft.com/en-us/azure/devops/repos/git/create-pr-status-server-with-azure-functions?view=azure-devops)
* [Configure a branch policy for an external service](https://learn.microsoft.com/en-us/azure/devops/repos/git/pr-status-policy?view=azure-devops)

1. **Install Git command-line tools**

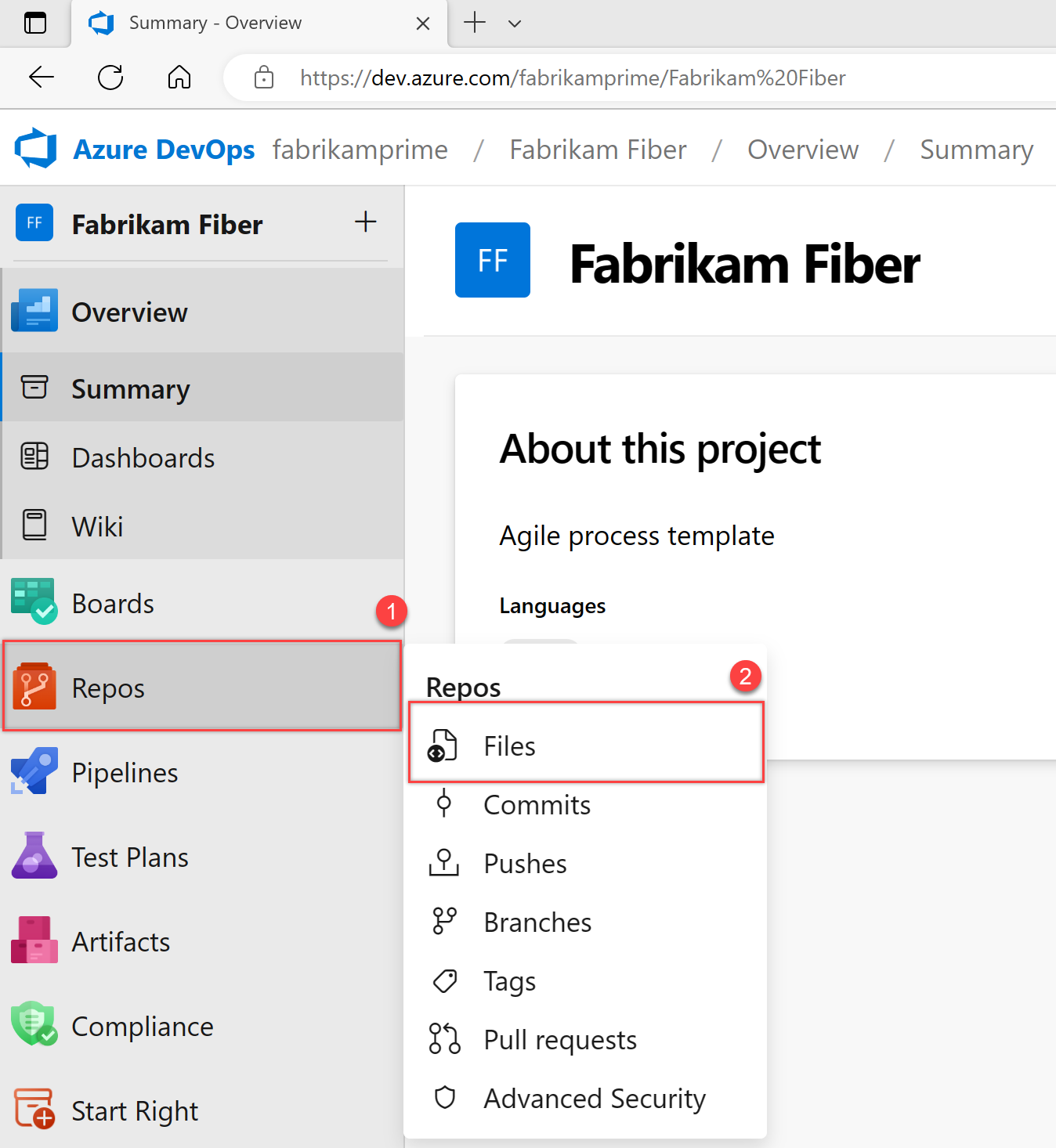
Install one of the following Git command-line tools:

* [Git for Windows and Git Credential Manager](https://learn.microsoft.com/en-us/azure/devops/repos/git/set-up-credential-managers?view=azure-devops).
* To install on macOS or Linux, check out the [Installing Git](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) chapter in the open-source *Pro Git* book. For macOS and Linux, we recommend that you [configure SSH authentication](https://learn.microsoft.com/en-us/azure/devops/repos/git/use-ssh-keys-to-authenticate?view=azure-devops).

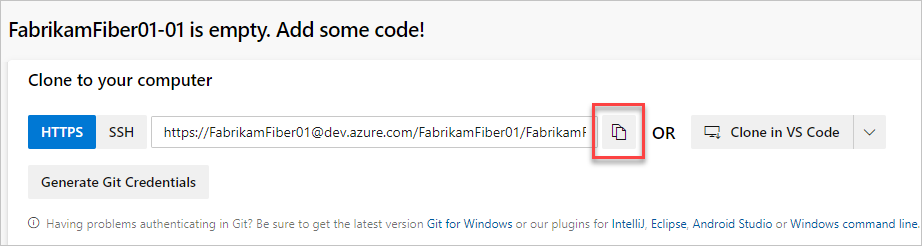
**2. Clone the repo to your computer**

To work with a Git repo, clone it to your computer, which creates a complete local copy of the repo. Your code might be in one of several places.

1. Complete the following step that's applicable to your scenario:
   * If **You don't have any code yet**, first [Create a new Git repo in your project](https://learn.microsoft.com/en-us/azure/devops/repos/git/create-new-repo?view=azure-devops#create-a-repo-using-the-web-portal), and then complete the next step.
   * If **the code is in another Git repo**, such as a GitHub repo or a different Azure Repo instance, [import it into a new or existing empty Git repo](https://learn.microsoft.com/en-us/azure/devops/repos/git/import-git-repository?view=azure-devops), and then complete the next step.
   * If **the code is on your local computer and not yet in version control**, either [create a new Git repo in your project](https://learn.microsoft.com/en-us/azure/devops/repos/git/create-new-repo?view=azure-devops#create-a-repo-using-the-web-portal) or add your code to an existing repository.
2. From your web browser, open the team project for your organization and select **Repos** > **Files**.



1. Select **Clone** in the upper-right corner of the **Code** window and copy the URL.



1. Open the Git command window (Git Bash on Git for Windows). Go to the folder where you want the code from the repo stored on your computer, and run git clone, followed by the path copied from **Clone URL** in the previous step. See the following example:

Copy

git clone https://FabrikamFiber01@dev.azure.com/FabrikamFiber01/FabrikamFiber01-01/\_git/FabrikamFiber01-01

Git downloads a copy of the code, including all [commits](https://learn.microsoft.com/en-us/azure/devops/repos/git/commits?view=azure-devops), and [branches](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies-overview?view=azure-devops) from the repo, into a new folder for you to work with.

1. Switch your directory to the repository that you cloned.

Copy

cd fabrikam-web

Keep this command window open to work in a branch.

**3. Work in a branch**

Git [branches](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies-overview?view=azure-devops) isolate your changes from other work being done in the project. We recommend using the [Git workflow](https://learn.microsoft.com/en-us/azure/devops/repos/git/gitworkflow?view=azure-devops), which uses a new branch for every feature or fix that you work on. For our examples, we use the branch, users/jamal/feature1.

1. Create a branch with the branch command.

Copy

git branch users/jamal/feature1

This command creates a reference in Git for the new branch. It also creates a pointer back to the parent commit so Git can keep a history of changes as you add commits to the branch.

If you're working with a previously cloned repository, ensure that you've checked out the right branch (git checkout main) and that it's up to date (git pull origin main) before you create your new branch.

1. Use checkout to switch to that branch.

Copy

git checkout users/jamal/feature1

Git changes the files on your computer to match the latest commit on the checked-out branch.

**Tip**

When you create a branch from the command line, the branch is based on the currently checked-out branch. When you clone the repository, the default branch (typically main) gets checked out. Because you cloned, your local copy of main has the latest changes.

Copy

git checkout main

git pull origin main

git branch users/jamal/feature1

git checkout users/jamal/feature1

You can replace the first three commands in the previous example with the following command, which creates a new branch named users/jamal/feature1 based on the latest main branch.

Copy

git pull origin main:users/jamal/feature1

Switch back to the Git Bash window that you used in the previous section. Run the following commands to create and check out a new branch based on the main branch.

Copy

git pull origin main:users/jamal/feature1

git checkout feature1

**4. Work with the code**

In the following steps, we make a change to the files on your computer, commit the changes locally, and then push the commit to the repo stored on the server.

1. Browse to the folder on your computer where you cloned the repo, open the README.md file in your editor of choice, and make some changes. Then, **Save** and close the file.
2. In the Git command window, go to the contoso-demo directory by entering the following command:

Copy

cd contoso-demo

1. Commit your changes by entering the following commands in the Git command window:

Copy

git add .

git commit -m "My first commit"

The git add . command stages any new or changed files, and git commit -m creates a commit with the specified commit message.

Check which branch you're working on before you commit, so that you don't commit changes to the wrong branch. Git always adds new commits to the current local branch.

1. Push your changes to the Git repo on the server. Enter the following command into the Git command window:

Copy

git push origin users/jamal/feature1

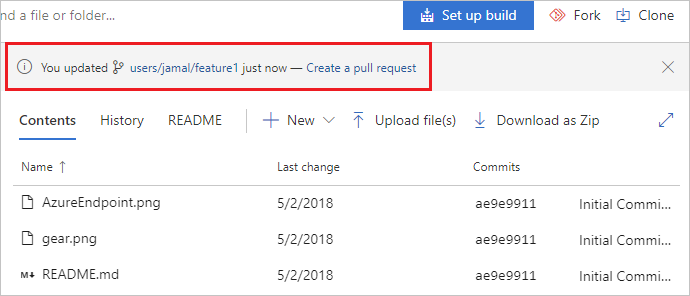
Your code is now shared to the remote repository, in a branch named users/jamal/feature1. To merge the code from your working branch into the main branch, use a pull request.

**5. Merge your changes with a pull request**

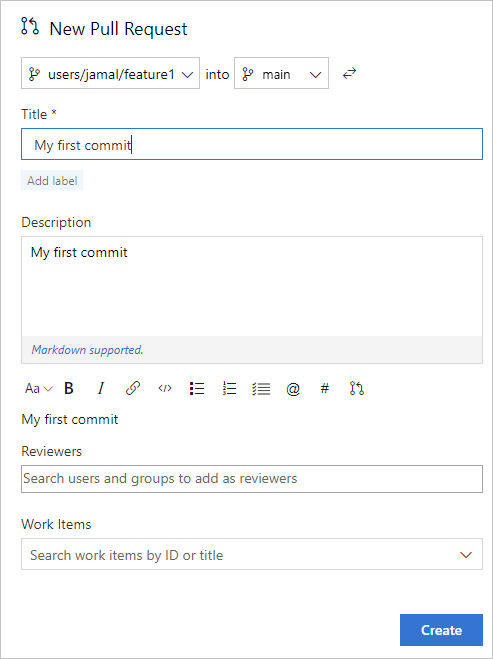
Pull requests combine the review and merge of your code into a single collaborative process. After you’re done fixing a bug or new feature in a branch, create a new pull request. Add the members of the team to the pull request so they can review and vote on your changes. Use pull requests to review works in progress and get early feedback on changes. There’s no commitment to merge the changes because you can abandon the pull request at any time.

The following example shows the basic steps of creating and completing a pull request.

1. Open the team project for your organization in your web browser and select **Repos** > **Files**. If you kept your browser open after getting the clone URL, you can just switch back to it.
2. Select **Create a pull request** in the upper-right corner of the **Files** window. If you don't see a message like **You updated users/jamal/feature1 just now**, refresh your browser.

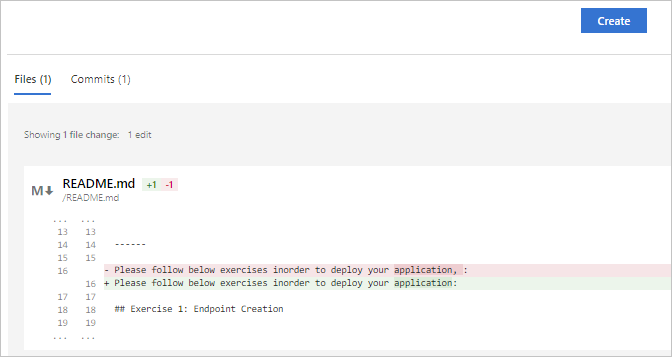


New pull requests are configured to merge your branch into the default branch, which in this example is main. The title and description are prepopulated with your commit message.



You can [add reviewers](https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops#add-and-remove-reviewers) and [link work items](https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops#link-work-items) to your pull request.

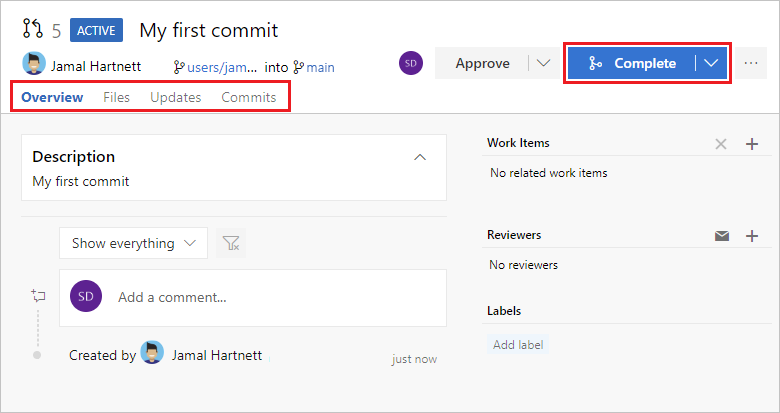
You can review the files included in the pull request at the bottom of the **New Pull Request** window.



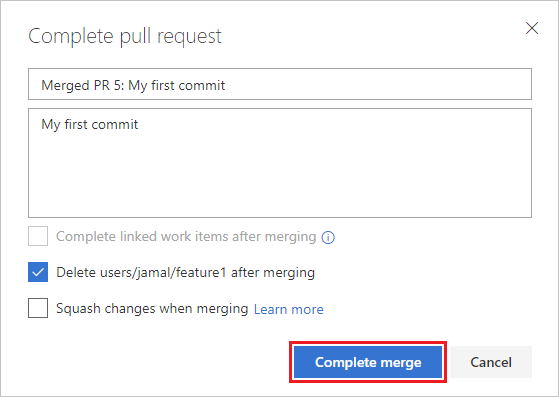
1. Select **Create**.

View the details of your pull request from the **Overview** tab. You can also view the changed files, updates, and commits in your pull request from the other tabs.

1. Select **Complete** to begin the process of completing the pull request.



1. Select **Complete merge** to complete the pull request and merge your code into the main branch.



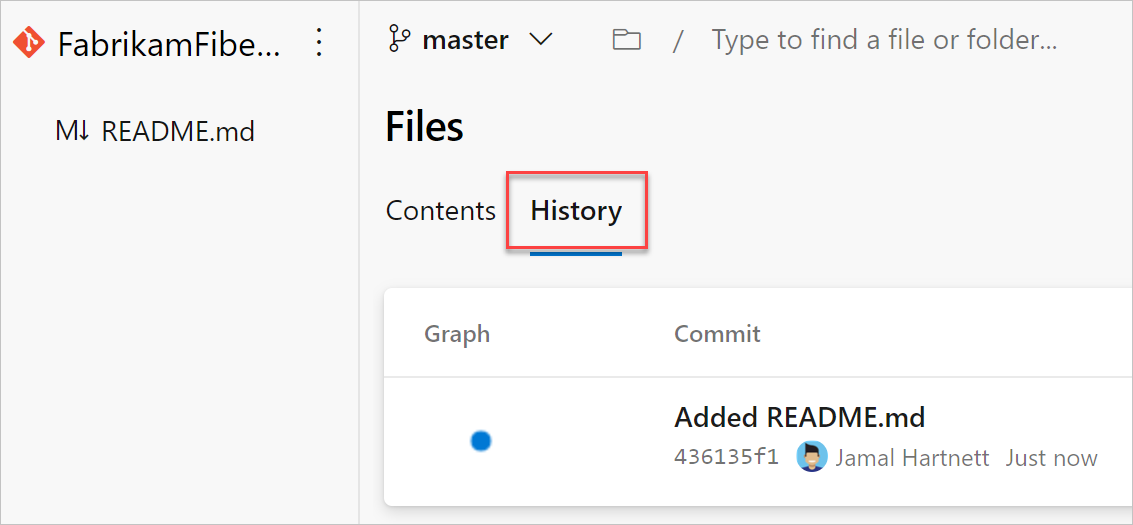
**Note**

This example shows the basic steps of creating and completing a pull request. For more information, see [**Create, view, and manage pull requests**](https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops).

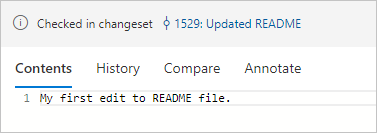
Your changes are now merged into the main branch, and your users/jamal/feature1 branch is deleted on the remote repository.

**View history**

1. Switch back to the web portal and select **History** from the **Code** page to view your new commit.



1. Switch to the **Files** tab, and select the README file to view your changes.



**Clean up**

Switch back to your Git Bash command prompt and run the following command to delete your local copy of the branch.

Copy

git checkout main

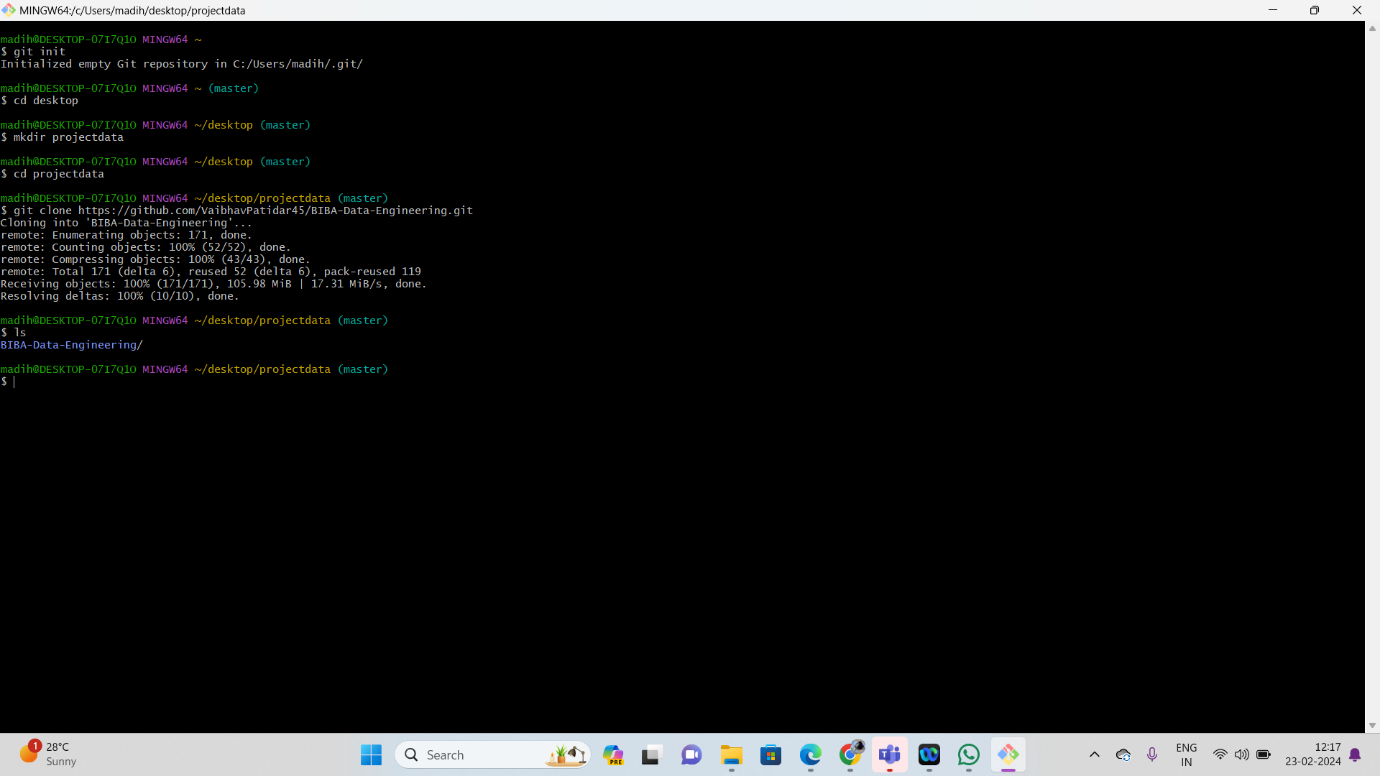
git pull origin main

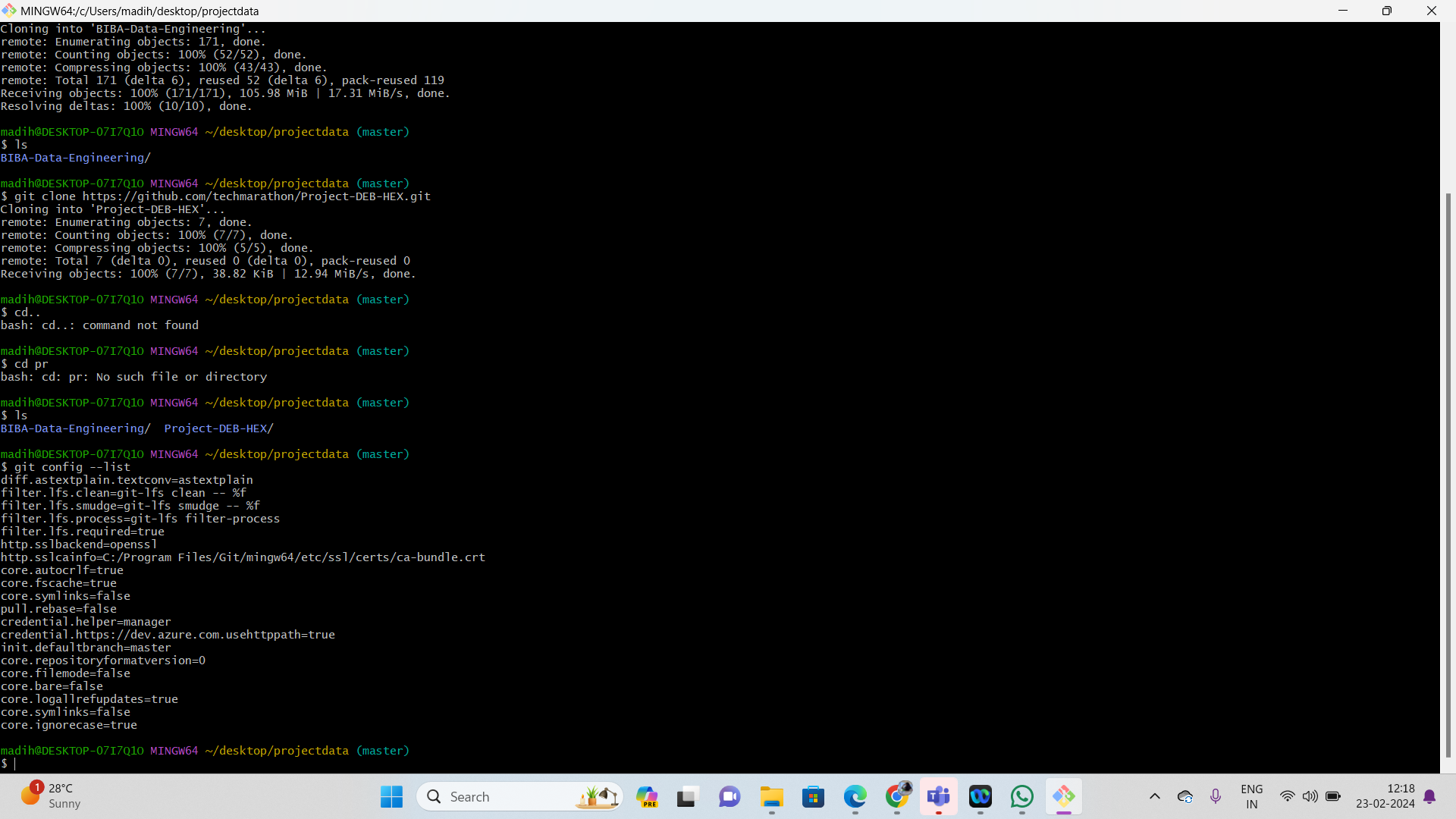
git branch -d users/jamal/feature1

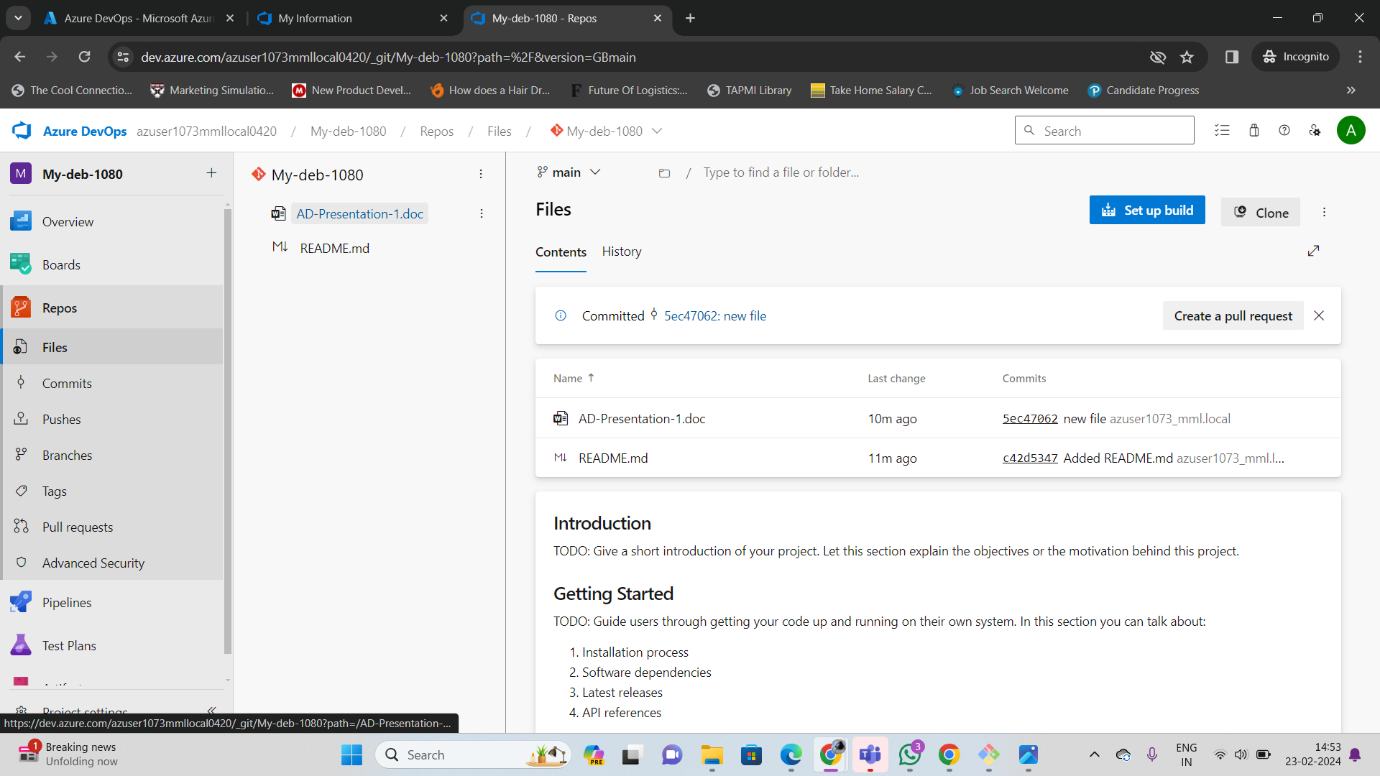
This action completes the following tasks:

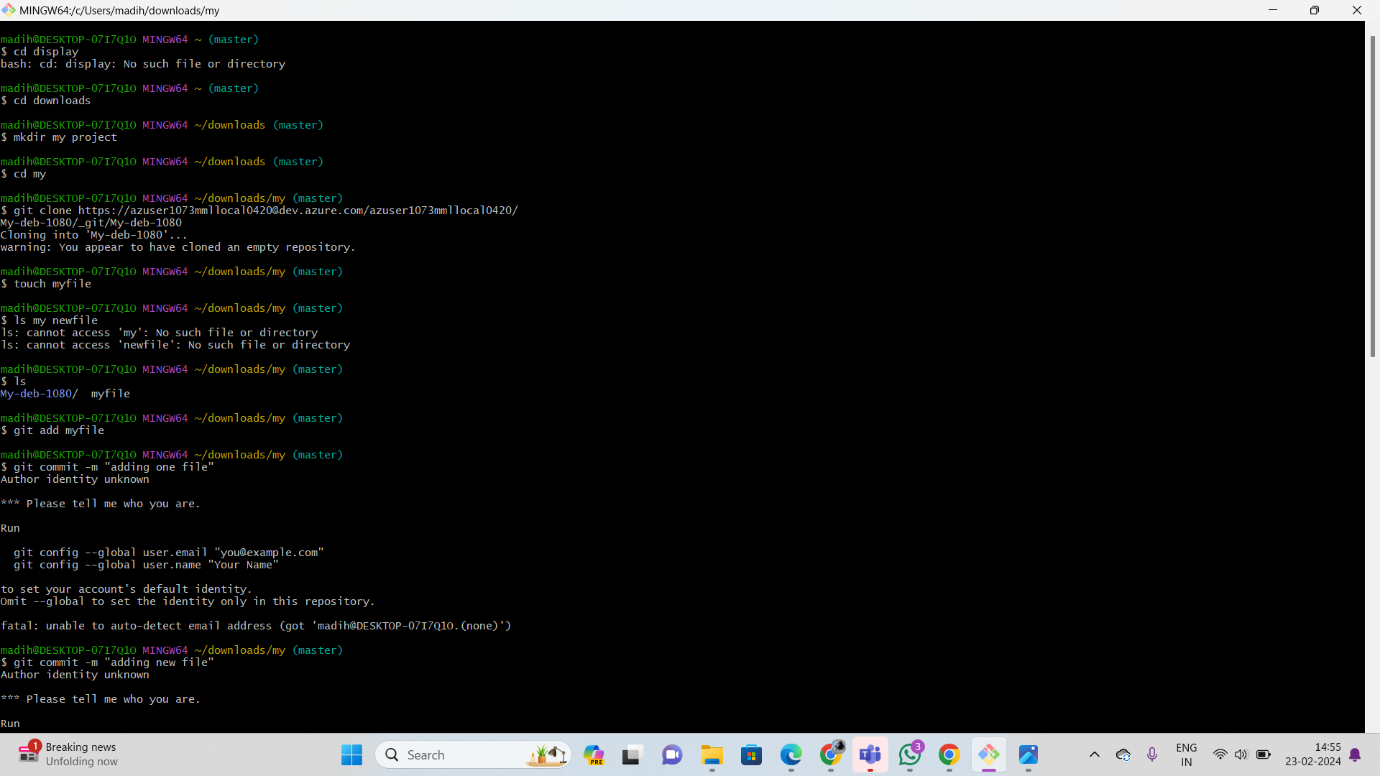
* The git checkout main command switches you to the main branch.
* The git pull origin main command pulls down the latest version of the code in the main branch, including your changes and the fact that users/jamal/feature1 was merged.
* The git branch -d users/jamal/feature1 command deletes your local copy of that branch.

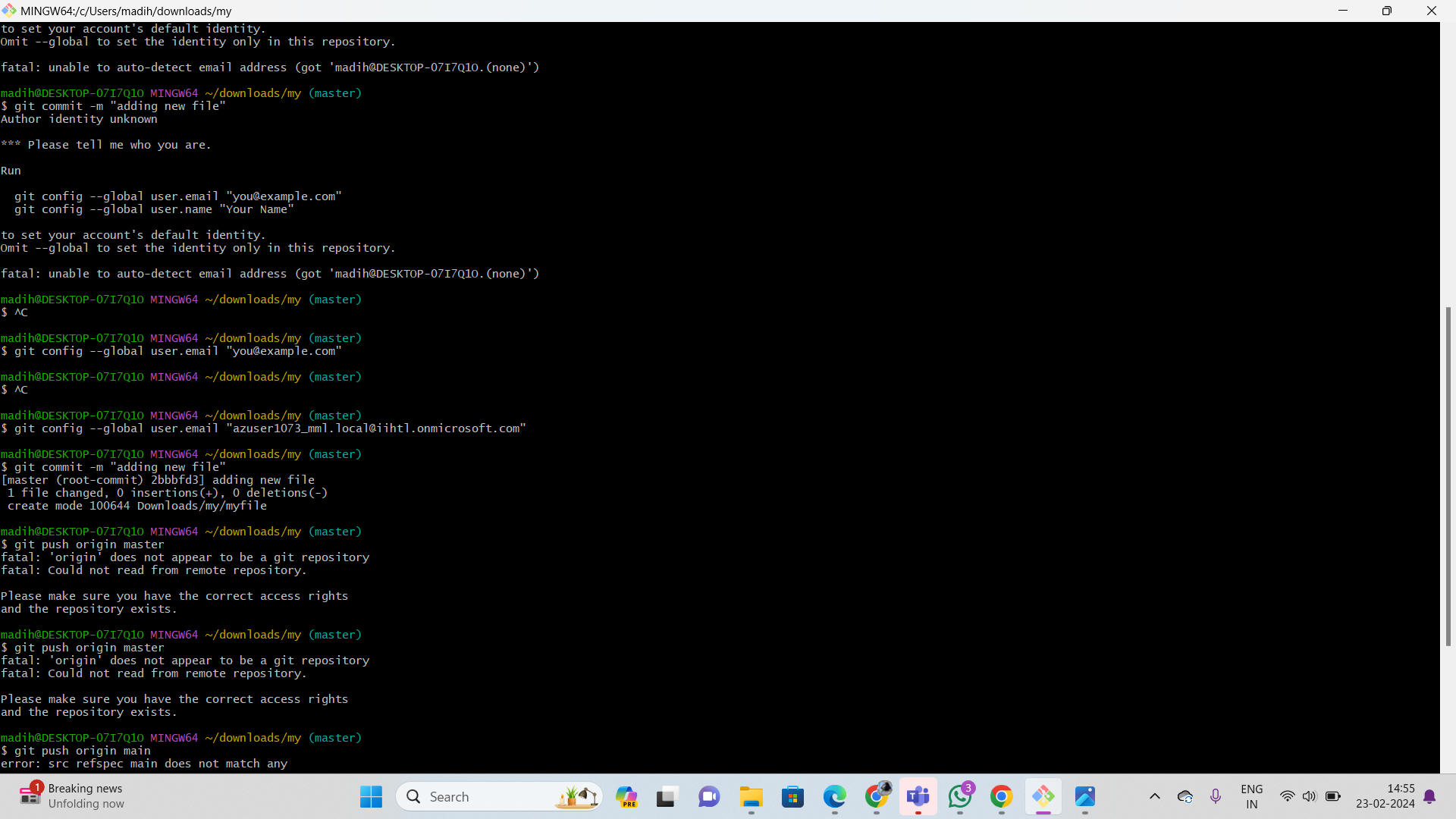
**HANDS ON:**

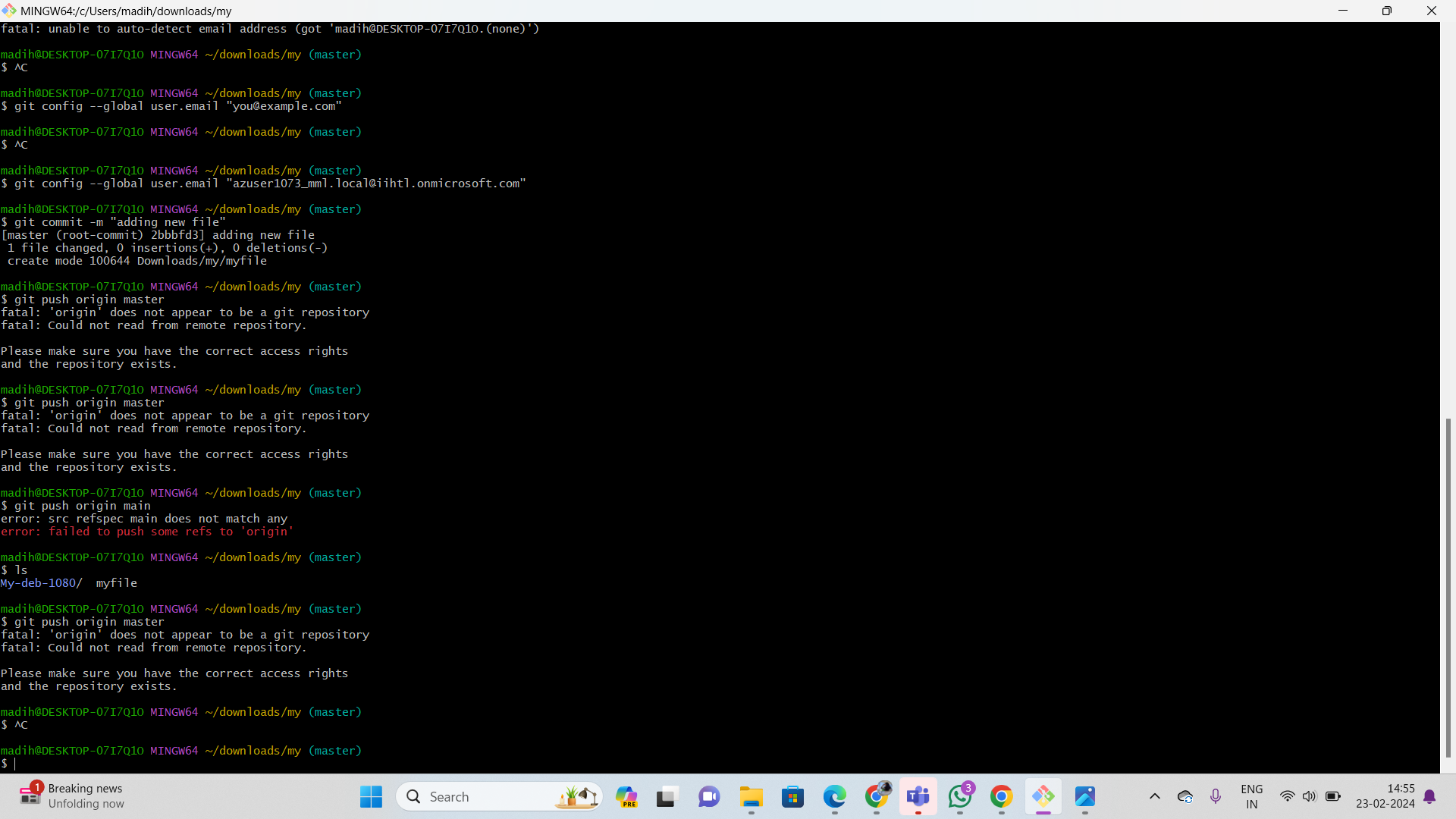
****

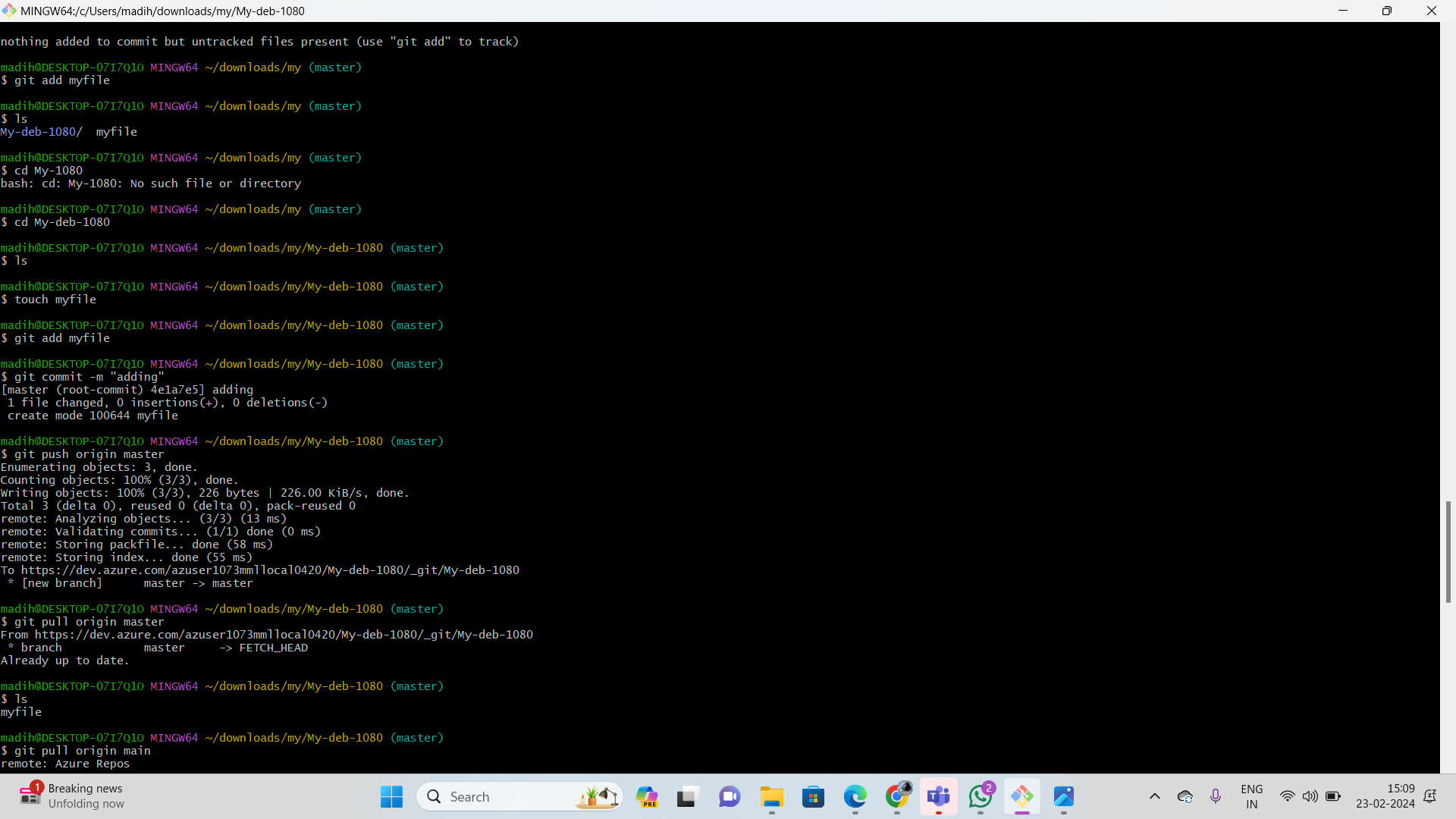
****

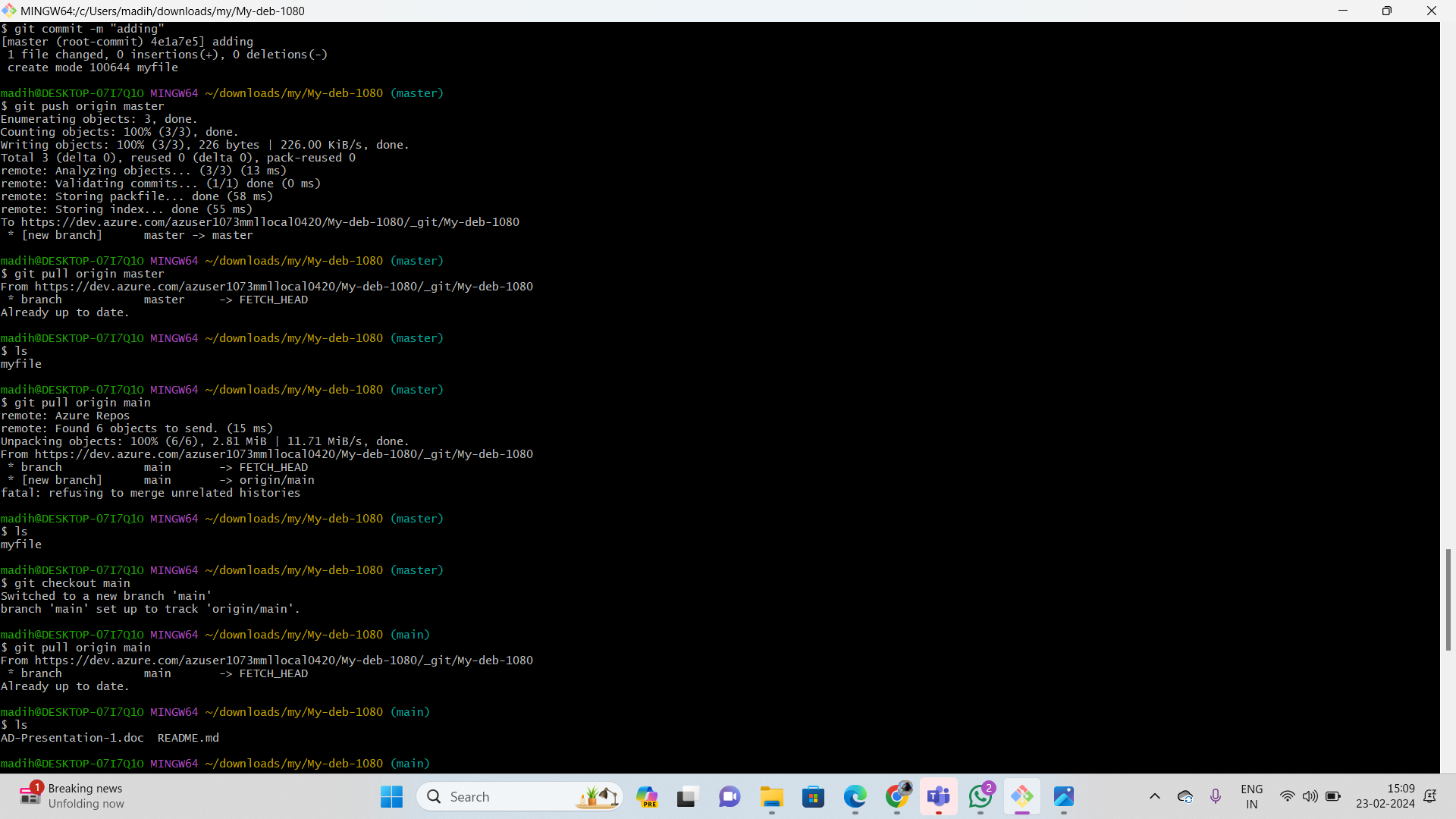
****

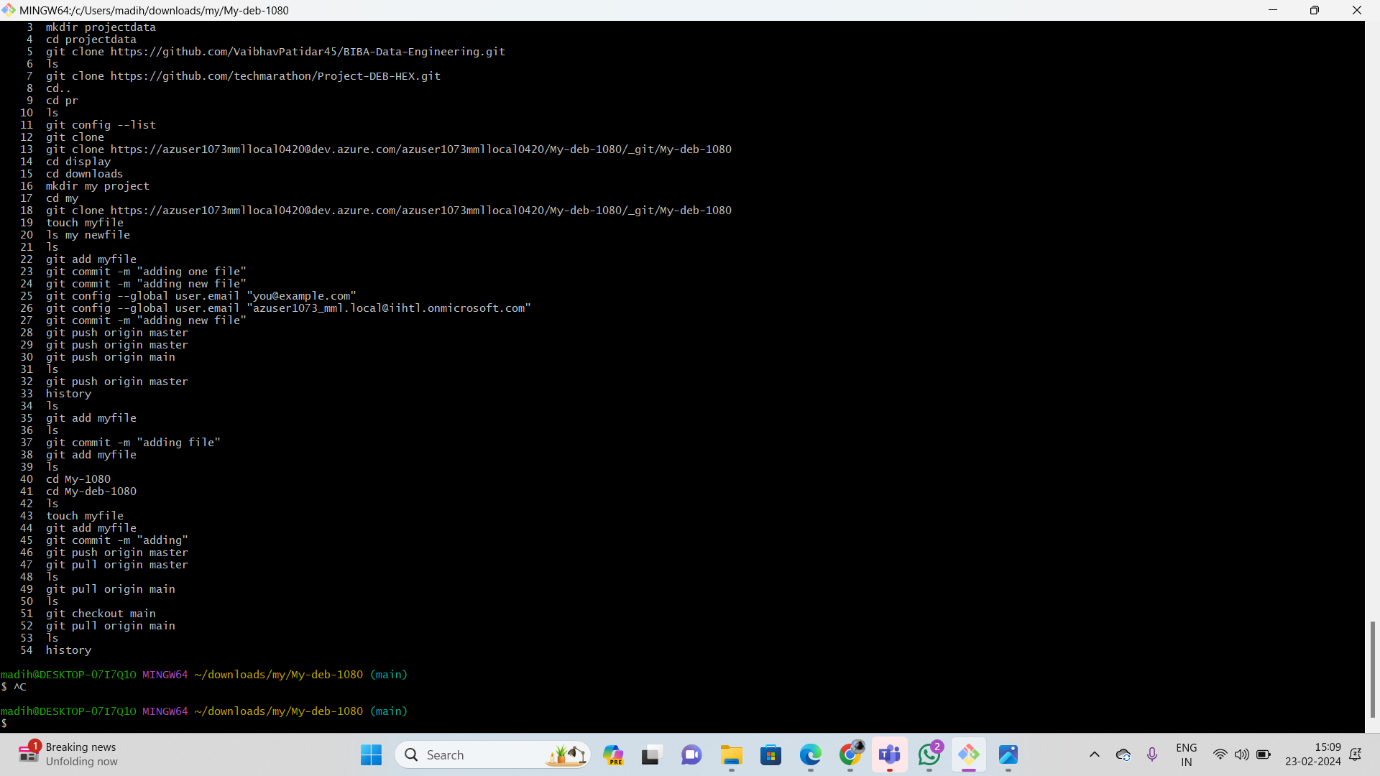
****

****

****

****

****

****