Q1. Create Azure Devops Environment and configure Azure Devops Git Repository ,configure on your local git to implement this upload few test files on the same.

Azure DevOps Environment:

Azure DevOps Environment is a cloud-based service that provides a set of development tools for software development teams to plan, develop, test, and deploy applications. It includes features for version control, continuous integration (CI), continuous delivery (CD), agile project management, and more.

An environment in Azure DevOps represents a collection of resources, such as virtual machines, databases, and web servers, where you deploy and run your applications. Environments can be used for various purposes, such as development, testing, staging, and production.

Configure Azure DevOps Git Repository:

To configure a Git repository in Azure DevOps, follow these steps:

1.Create a New Project: Sign in to Azure DevOps, navigate to the project where you want to create the repository, and create a new project if necessary.

2.Create a Git Repository: Within the project, navigate to the Repos tab and click on "New repository". Give your repository a name and optionally add a description.

3.Set Repository Options: Configure repository options such as version control system (Git), permissions, and branch policies as per your project requirements.

4.Configure Local Git Environment:
To configure your local Git environment to work with Azure DevOps, follow these steps:

5.Install Git: If you haven't already, download and install Git on your local machine from the official Git website: Git Downloads.

6.Clone Repository: Clone the Azure DevOps Git repository to your local machine using the git clone command. You can find the clone URL on the Azure DevOps repository page.

7.Configure Remote: Inside the cloned repository directory, set the Azure DevOps repository as the remote origin using the git remote add origin <repository_url> command.

8.Pull Latest Changes: Fetch the latest changes from the remote repository using git pull origin <branch_name>.

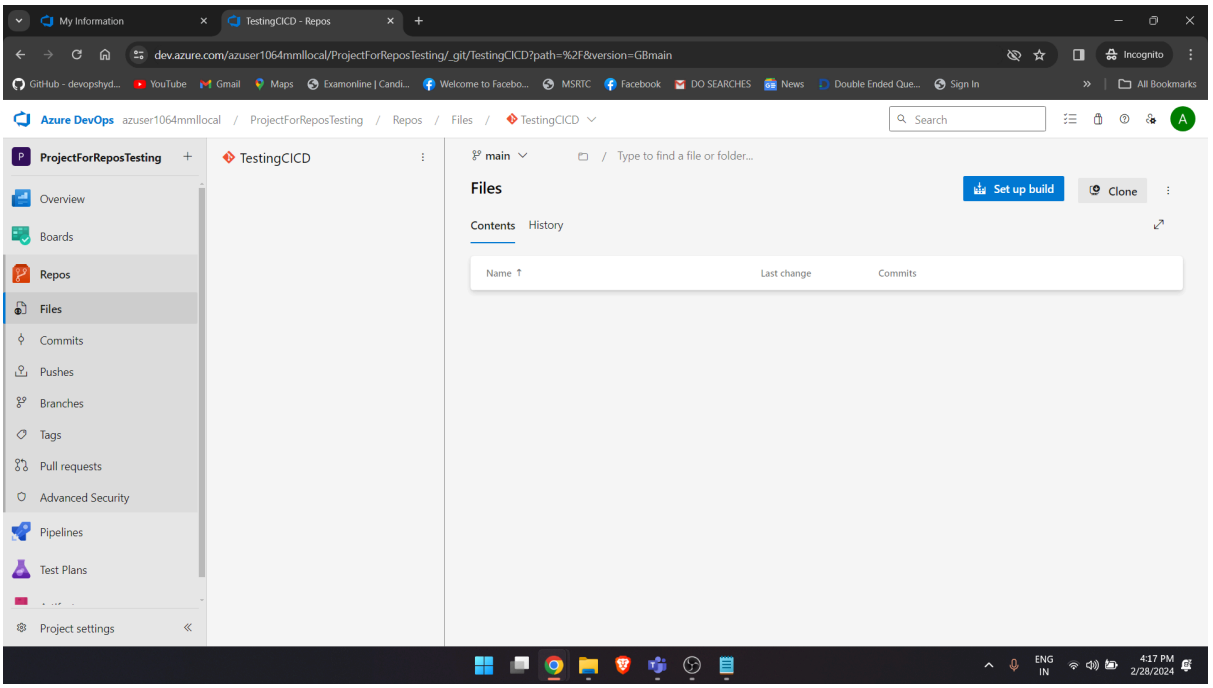
9.Add Test Files: Create and add your test files to the local repository using git add and git commit commands.

10.Push Changes: Push your changes to the Azure DevOps repository using git push origin <branch_name>.

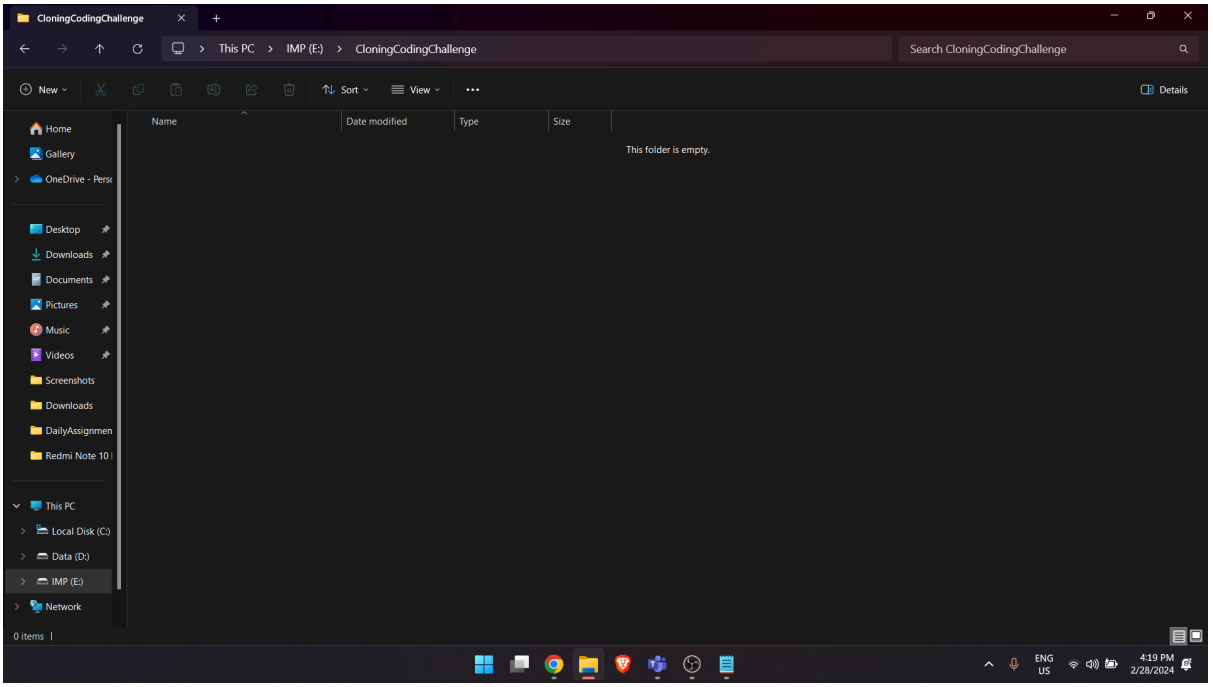
By following these steps, you can configure a Git repository in Azure DevOps and set up your local Git environment to work with it. This allows you to collaborate with your team, version control your code, and seamlessly integrate with Azure DevOps' CI/CD pipelines.

Implementation :

Creation of Repos :

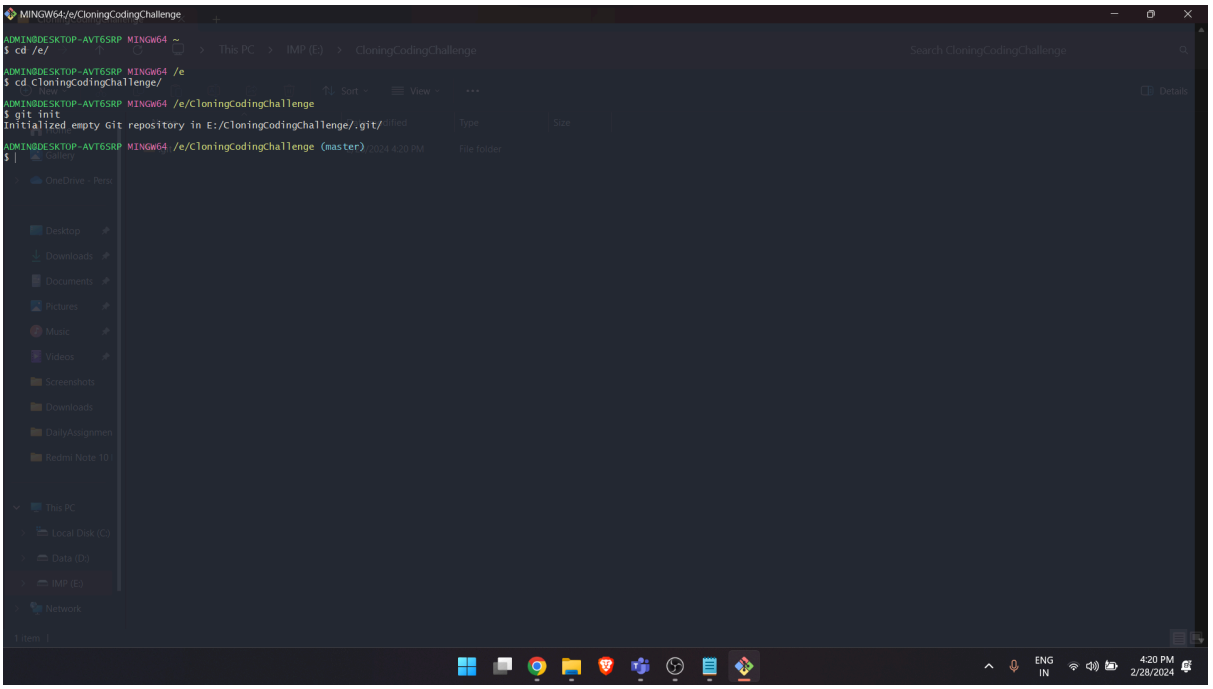


Creating a file in local :



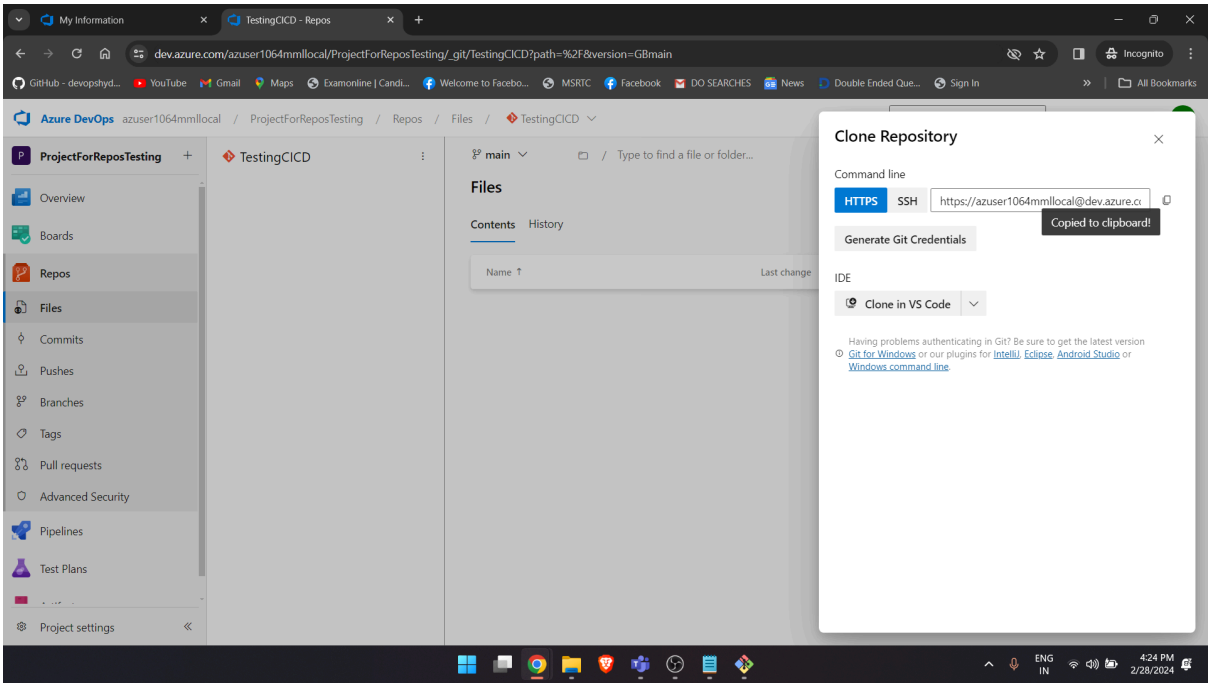
Using file explorer the directory is made.

Git init :



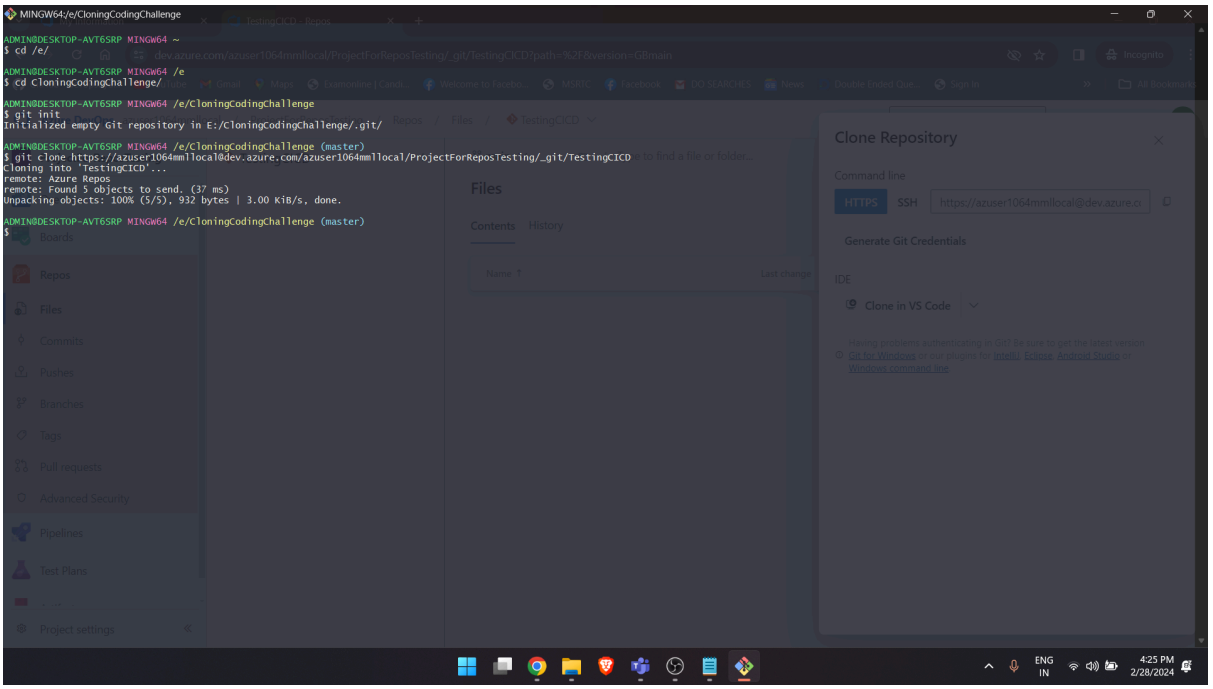
Using git bash we started git in the local system.

Cloning from Azure Devops :



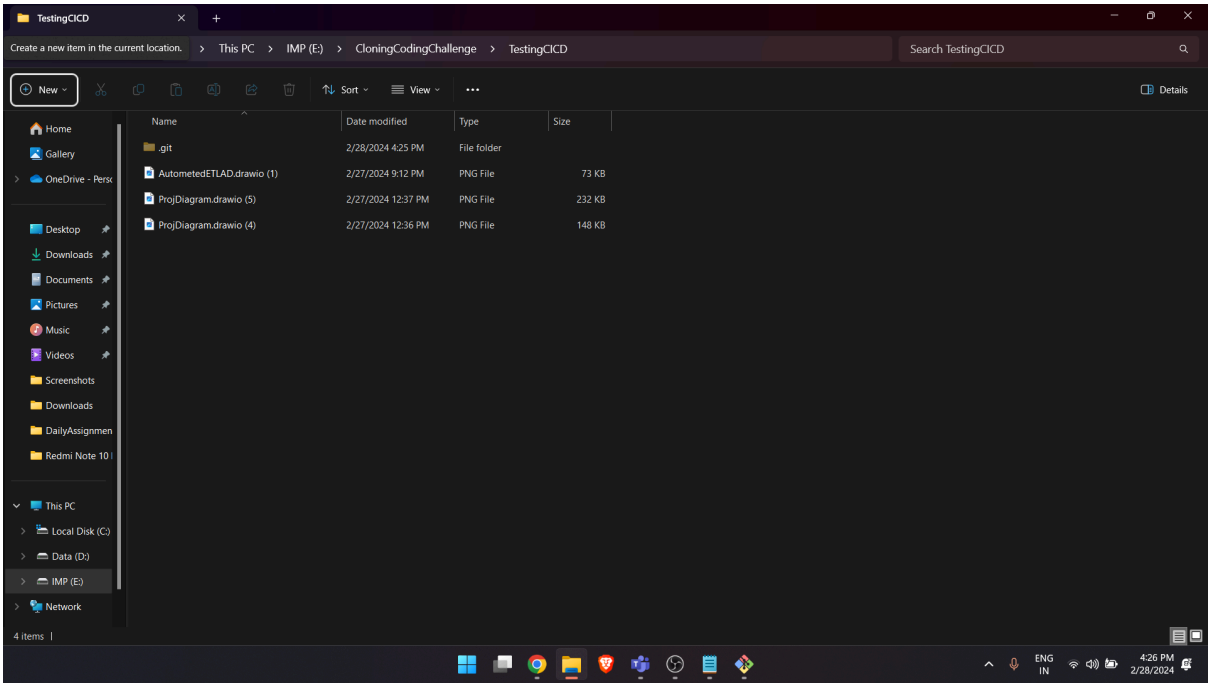
Copied the path.

Cloning using git bash:

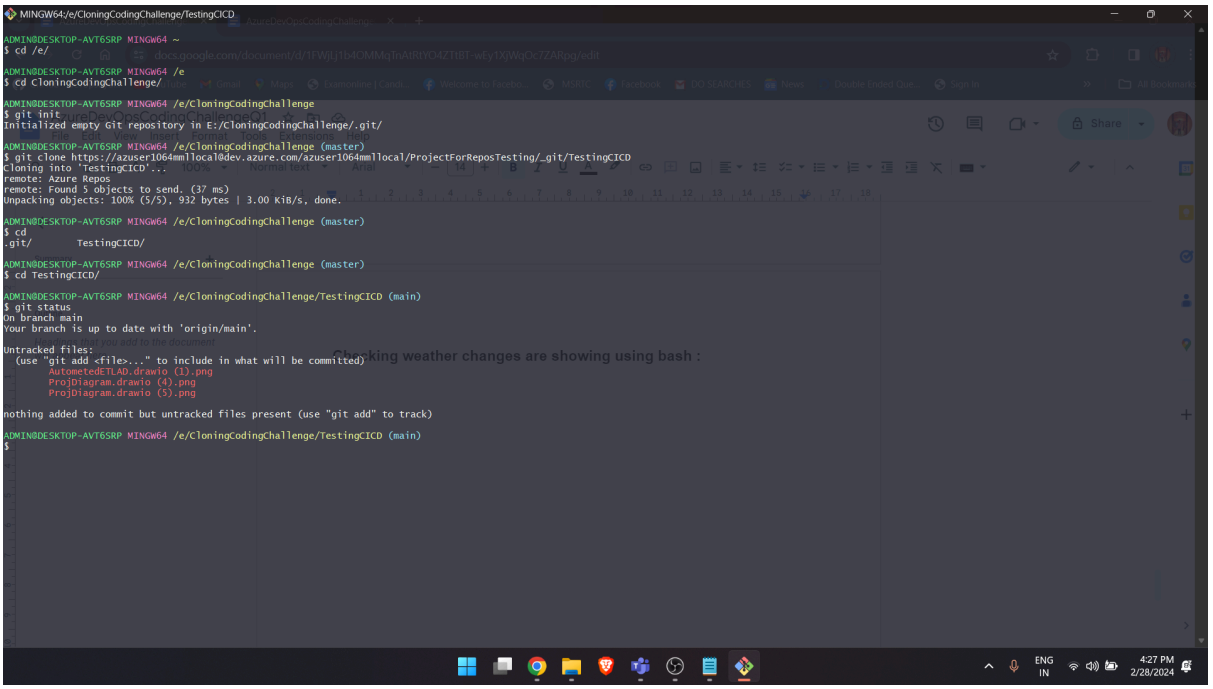


Successfully cloned the repository.

Adding Sample files to local :

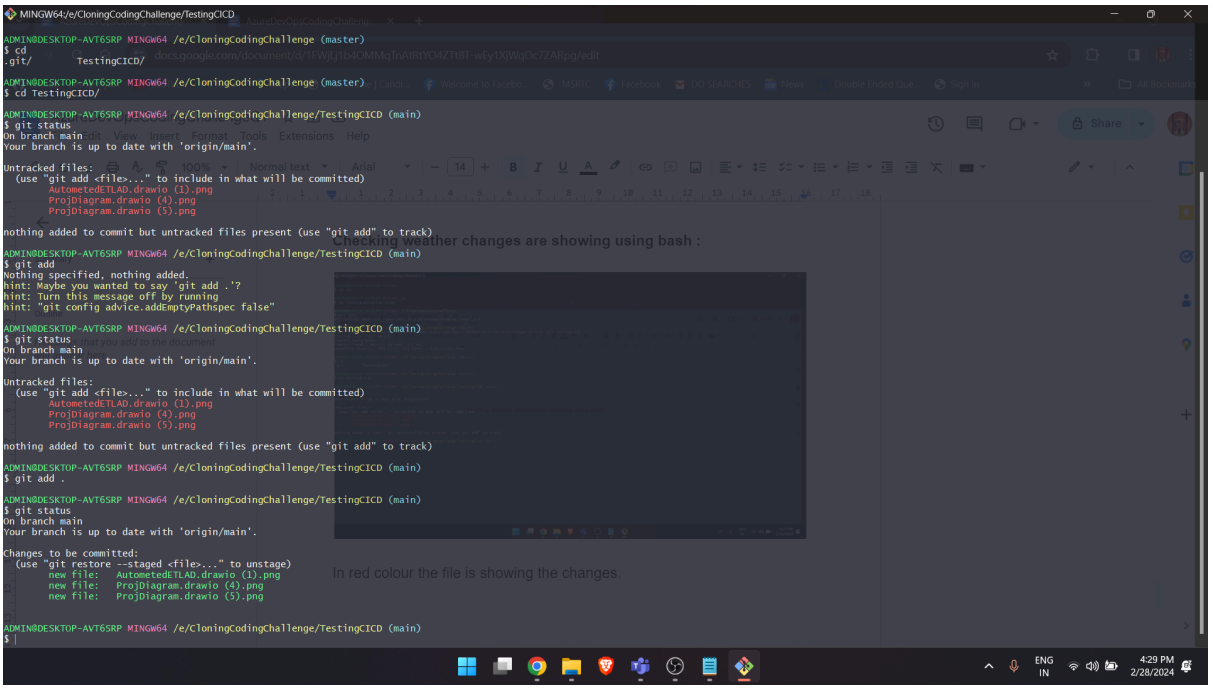


Checking weather changes are showing using bash :



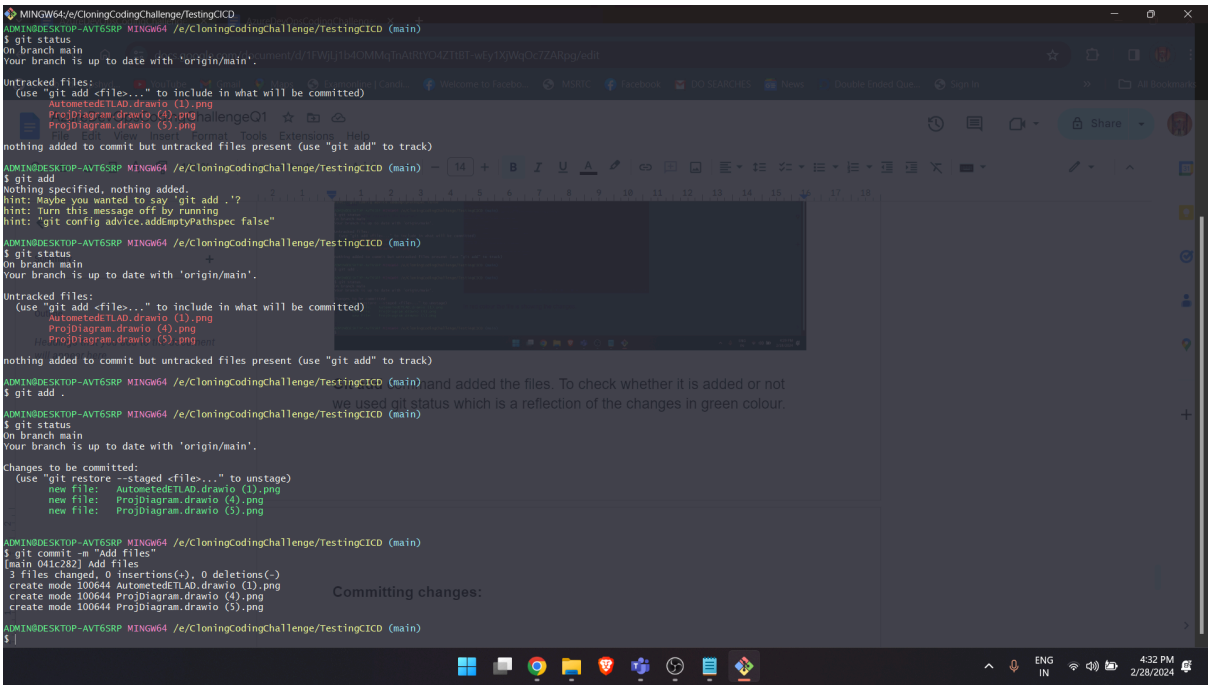
In red colour the file is showing the changes.

Staging the files using bash :



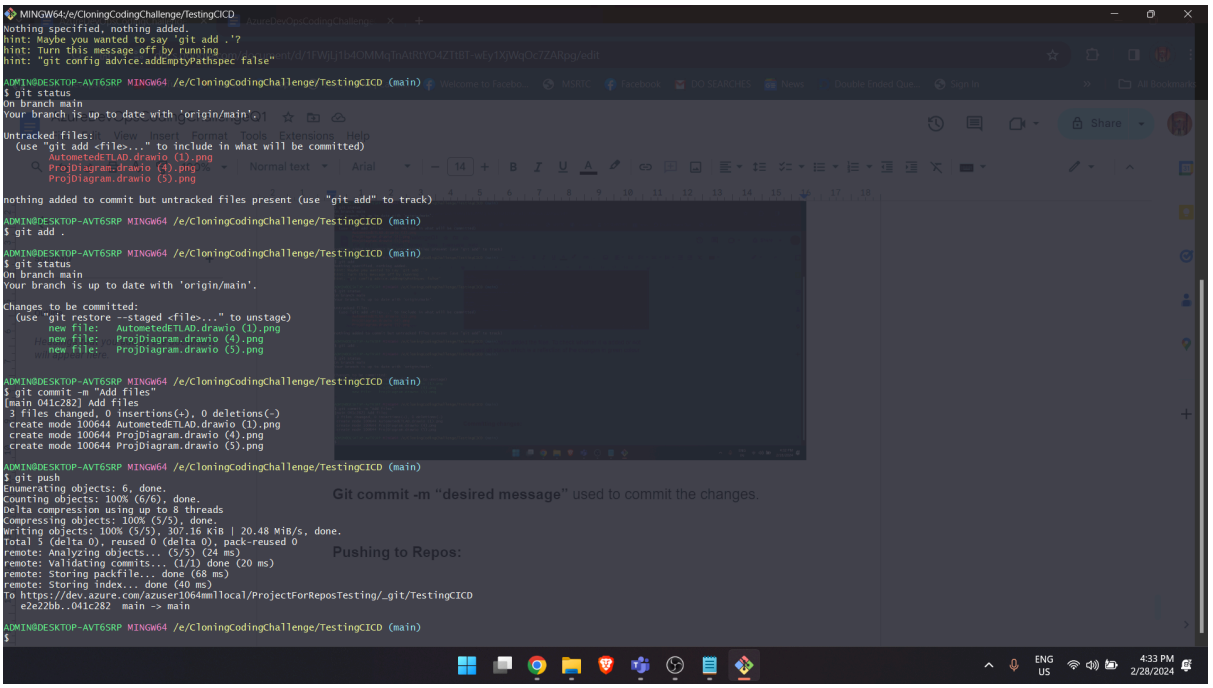
Git add command added the files. To check whether it is added or not we used git status which is a reflection of the changes in green colour.

Committing changes:

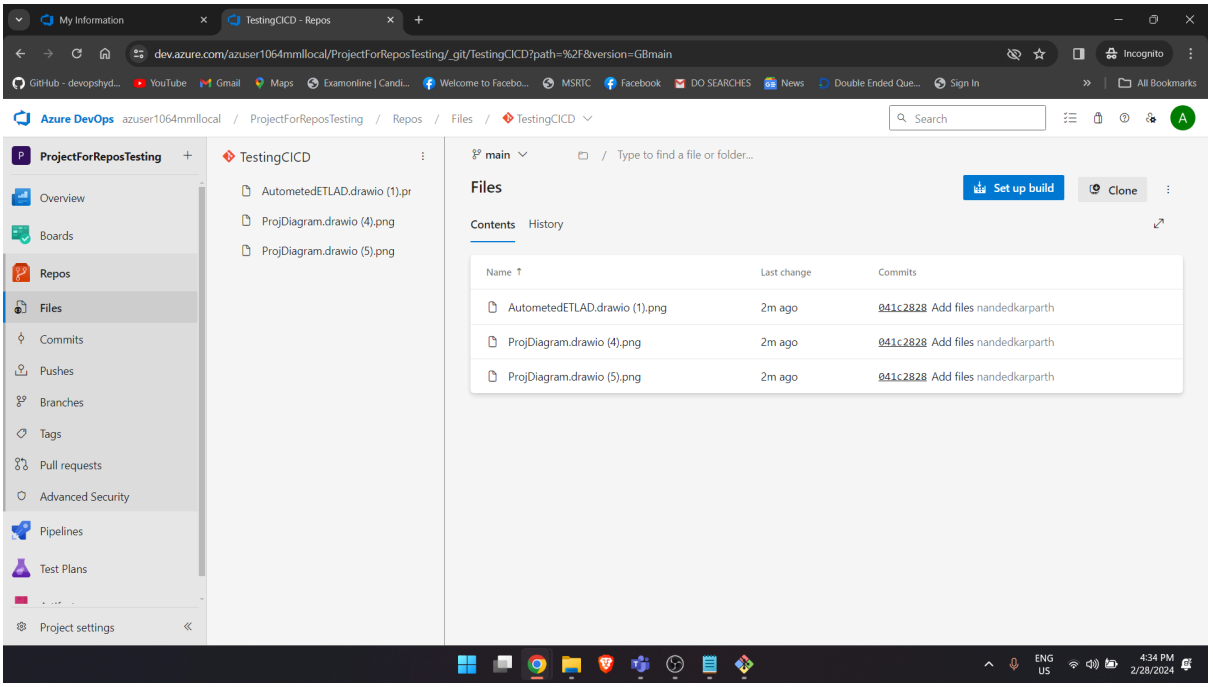


Git commit -m “desired message” used to commit the changes.

Pushing to Repos:

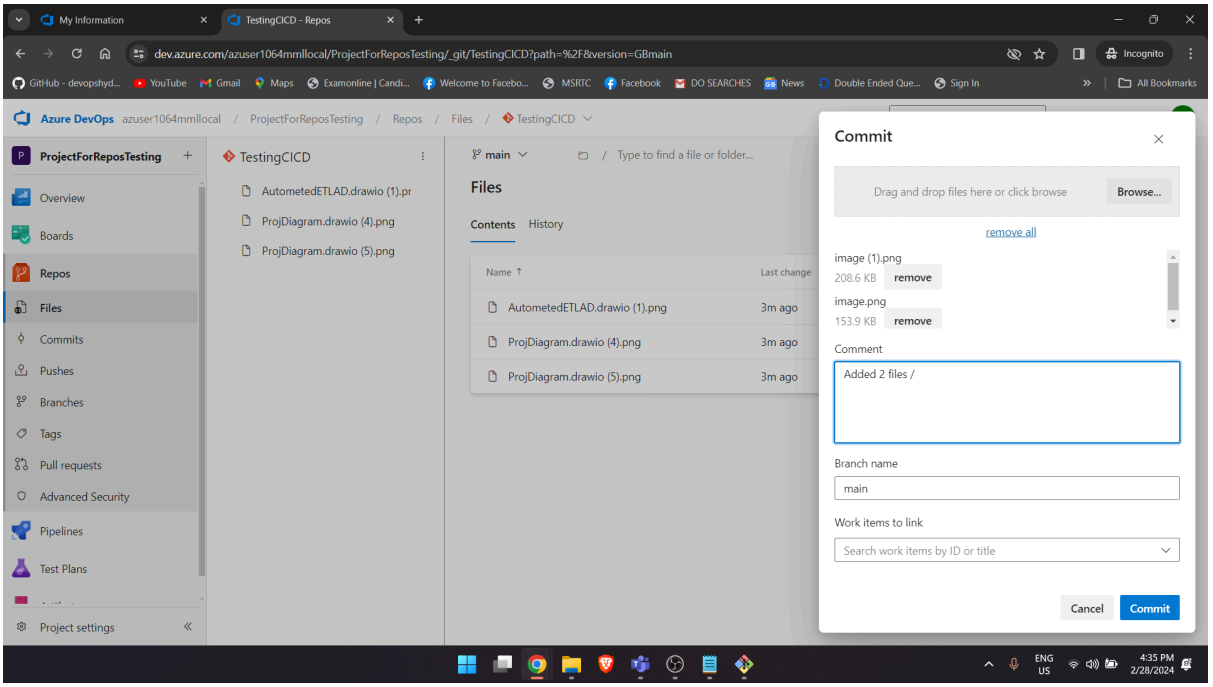


Git push command is used to reflect changes on azure repos.

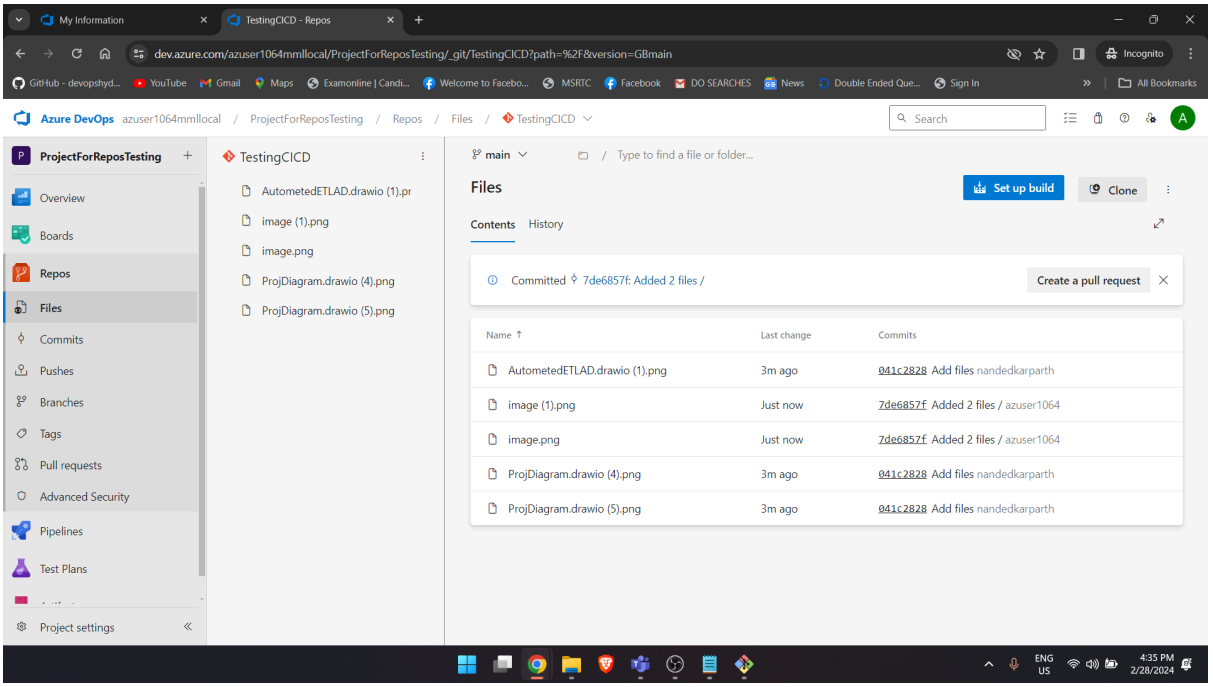


We can see on azure repos the files are added.

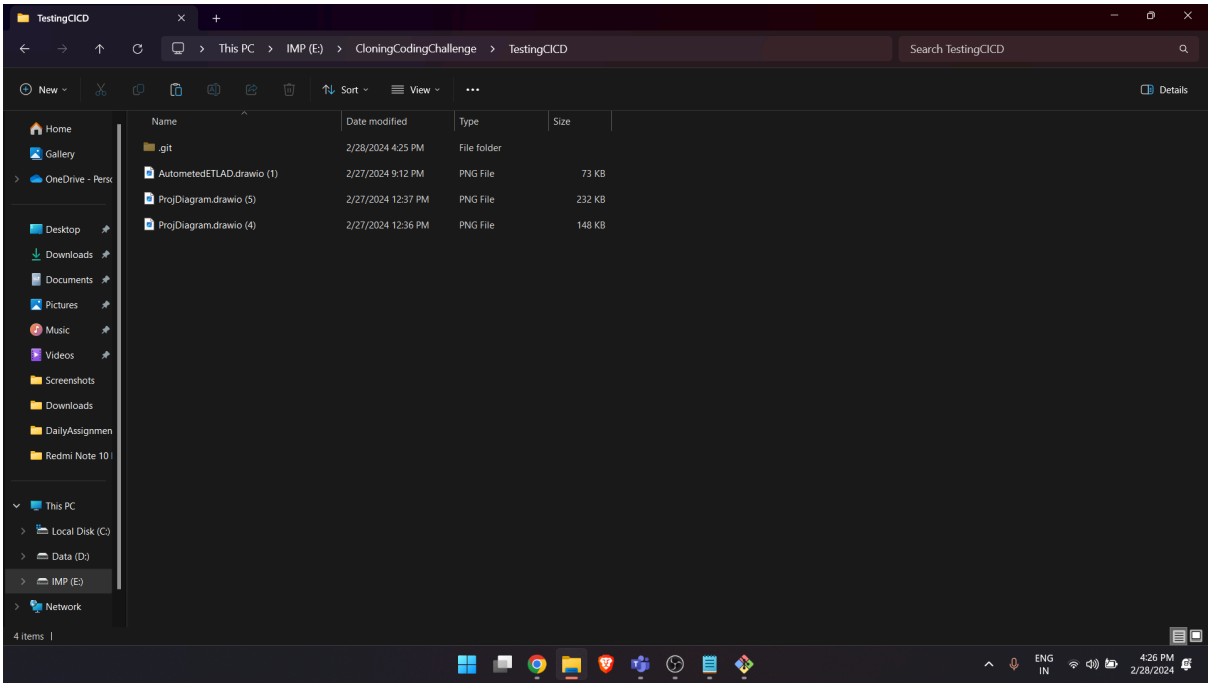
Adding files on Azure Repos :



Adding 2 files through repos.

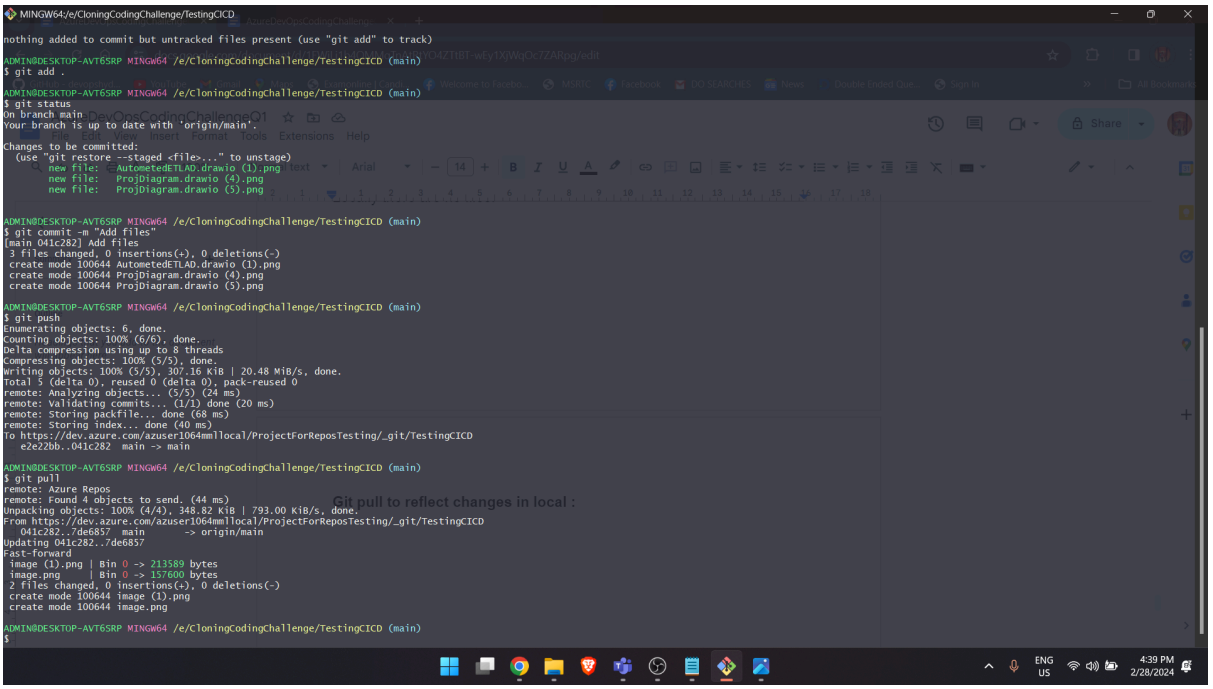


Files added successfully.



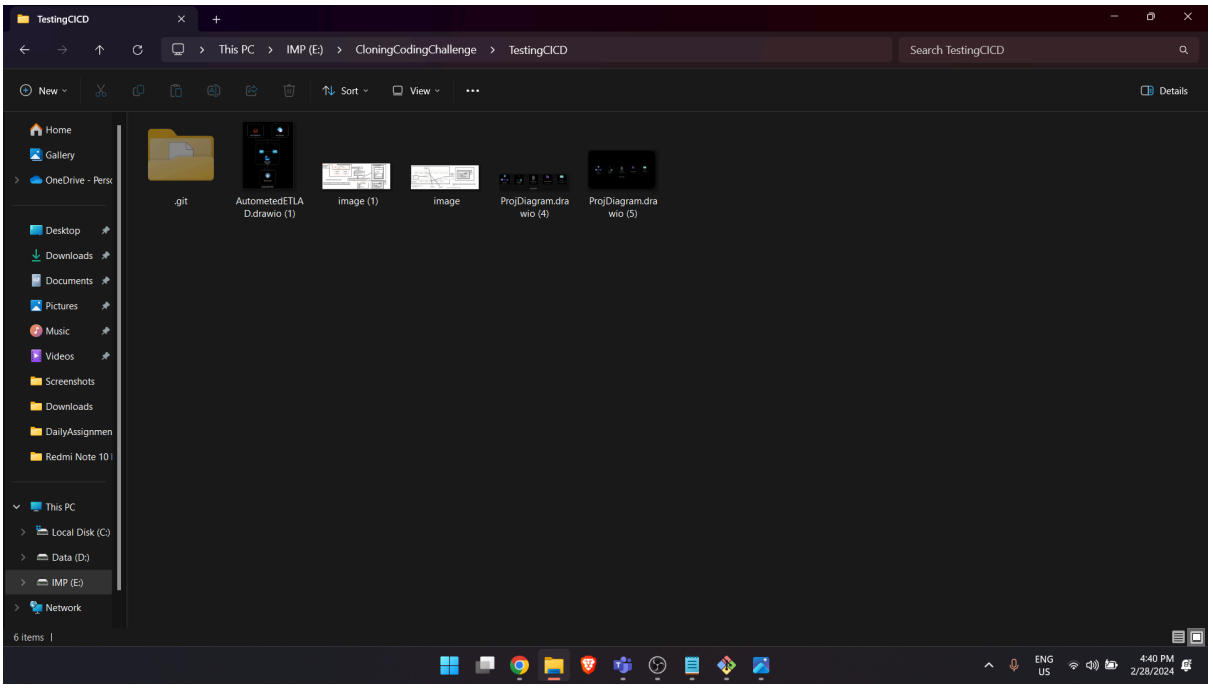
Locally there are no files.

Git pull to reflect changes in local :



Using the git pull command updated the local directory.

Reflected Changes :



Got the files locally.