

Azure DevOps Coding Challenge

Pratik Wani

Leverage the practises of CI-CD Using azure Data engineering and explain the architecture of the Azure synapse

- Continuous Integration/Continuous Deployment (CI/CD) practices in Azure Data Engineering involve automating the process of building, testing, and deploying data solutions.
- Continuous Integration: CI ensures that if we are adding new things then it is automatically tested and integrated with older things
- Continuous Deployment: CD ensures that the code which passed through the CI, immediately made live with so that user can use it.

CI/CD Practices in Azure Data Engineering

- **VCS (Version Control System)**
 - VCS like Git is used to store the code and configurations of data engineering solutions
 - VCS supports SQL scripts, Python notebooks as well
- **Automated Testing**
 - In CI-CD pipeline the newly added code is tested automatically, it validates the functionalities of code also checks the performance of the code under different aspects
 - It performs unit tests, integration tests, and performance tests as part of the CI/CD process to ensure the reliability of the solutions.
- **Automated Builds**
 - It provides a functionality of automated build pipelines and it triggers when we push new changes using VCS
 - It automatically runs the code, validate the code by performing different test

- **Monitoring**

- Automated monitoring system is present to track the performance, usage, and errors in the deployed data solutions.
- It keeps the members updated by sending alerts to all the team member for any issues or bugs comes in the process

Architecture of the Azure synapse

- Azure Synapse is a cloud-based analytics service that integrates enterprise data warehousing and Big Data analytics

- **SQL Pools**

- A dedicated SQL pool in Azure Synapse architecture is a fully managed, cloud-based, and optimized data warehouse.
- SQL Pools offer on-demand or provisioned resources for running T-SQL queries and analytics workloads against large datasets.

- **Spark Pools**

- It provides the platform to perform different big data processing tasks also helps to perform different analytics
- Users can run Spark jobs and notebooks to perform data transformations, machine learning model training, and interactive analytics.

- **Integration**

- It supports data integration through built-in connectors, Azure Synapse Pipelines
- We can ingest data from various sources, transform it using SQL or Spark, and load it into data warehouses or data lakes

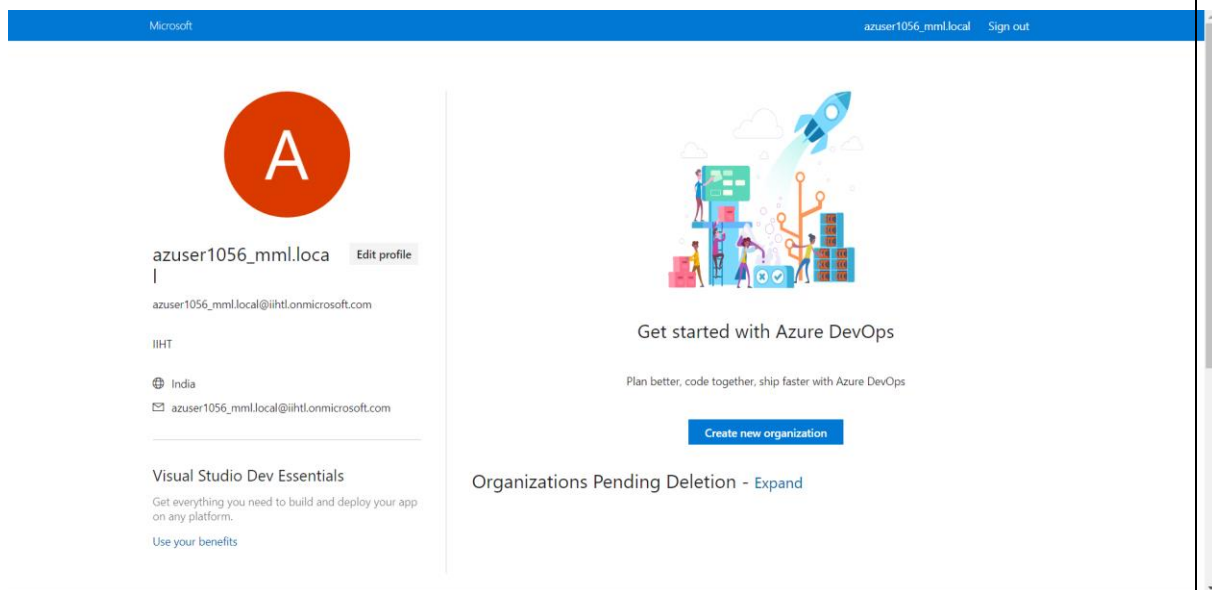
- **Integration with Other Azure Services**

- Azure Synapse integrates with other Azure services like Azure Data Lake Storage, Azure Blob Storage
- We can take benefits of other services with the functionalities of Azure Synapse

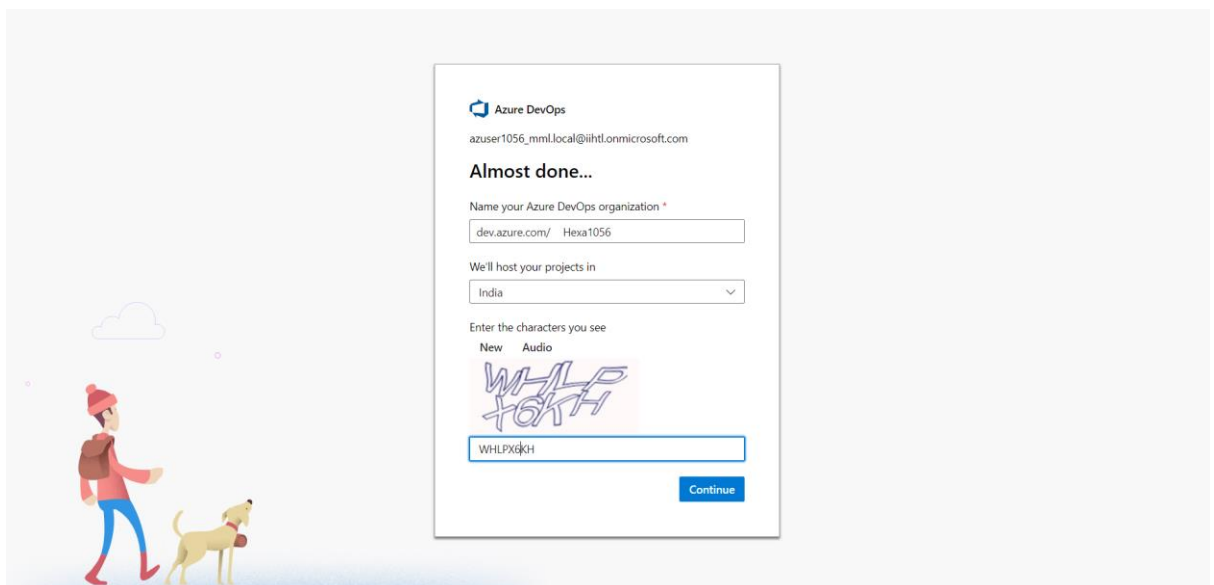
DevOps Environment and Clone, Push, Pull activities on Azure DevOps git and Local git

- **Creating DevOps Environment**

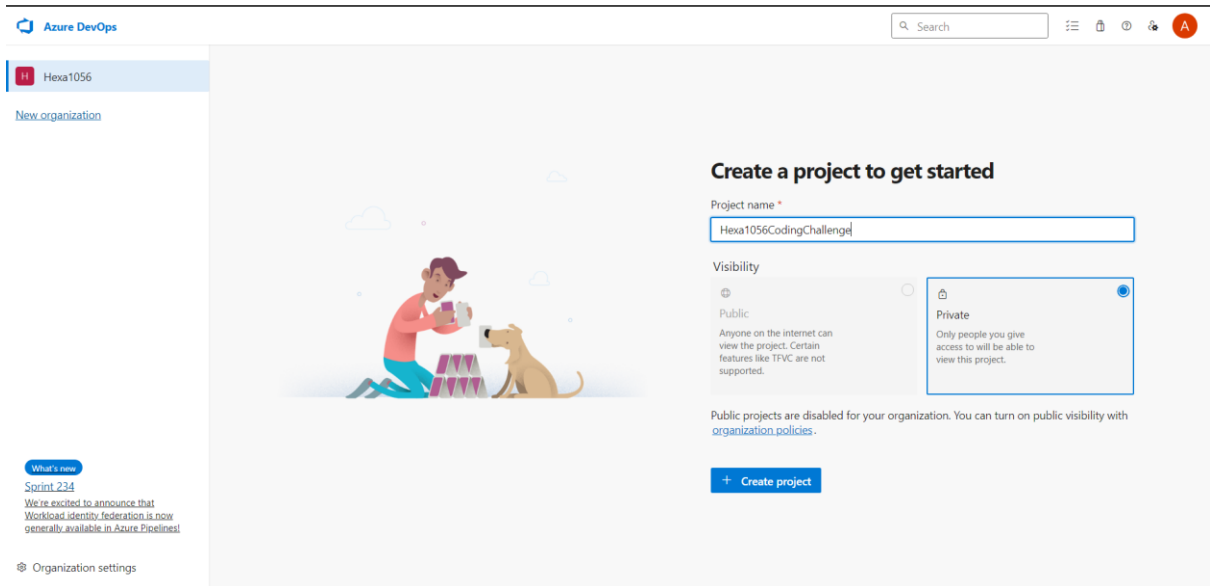
- Search for DevOps Environment and Click on my organizations



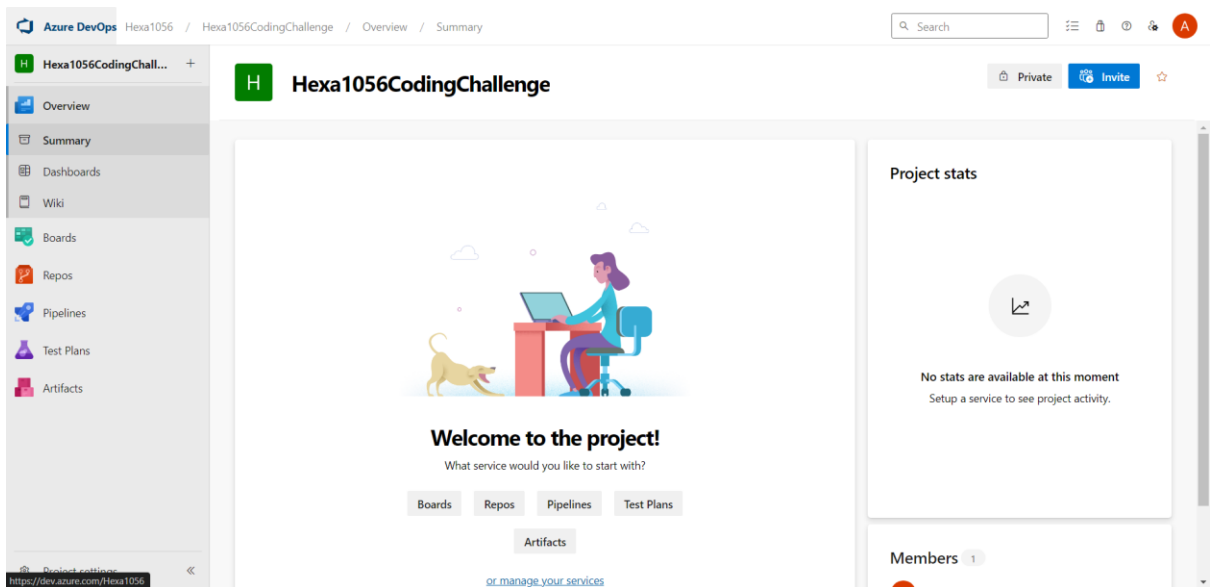
- Click on create new organization and log in



○ Create New Project

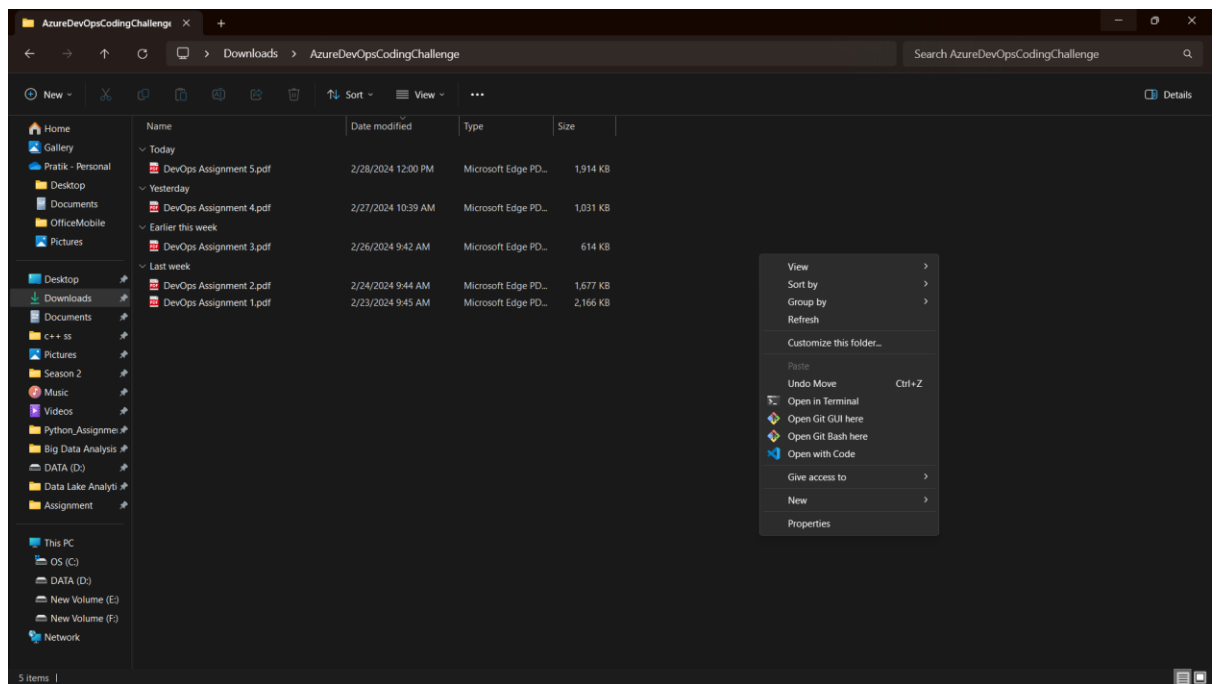


○ DevOps Environment

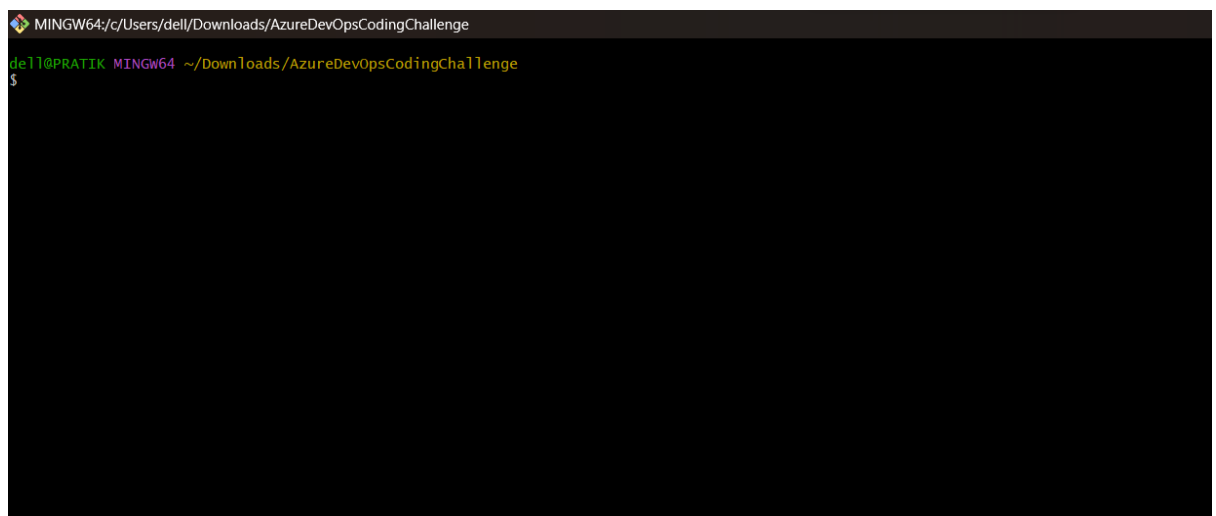


- **Git Clone**

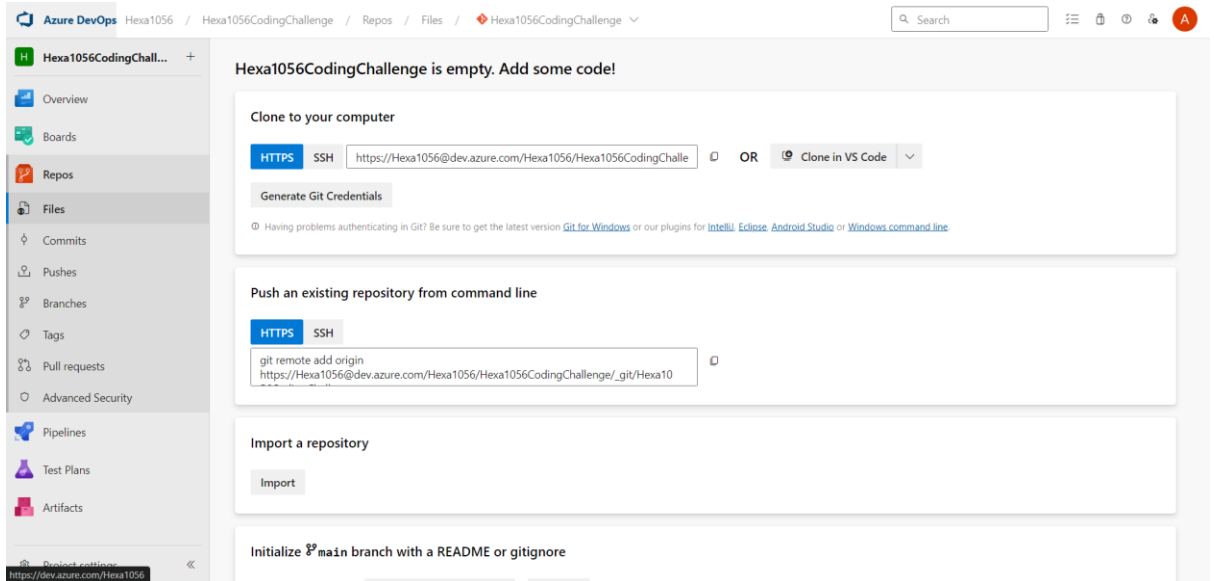
- Go to the folder in your local which we want to clone in our azure DevOps organization



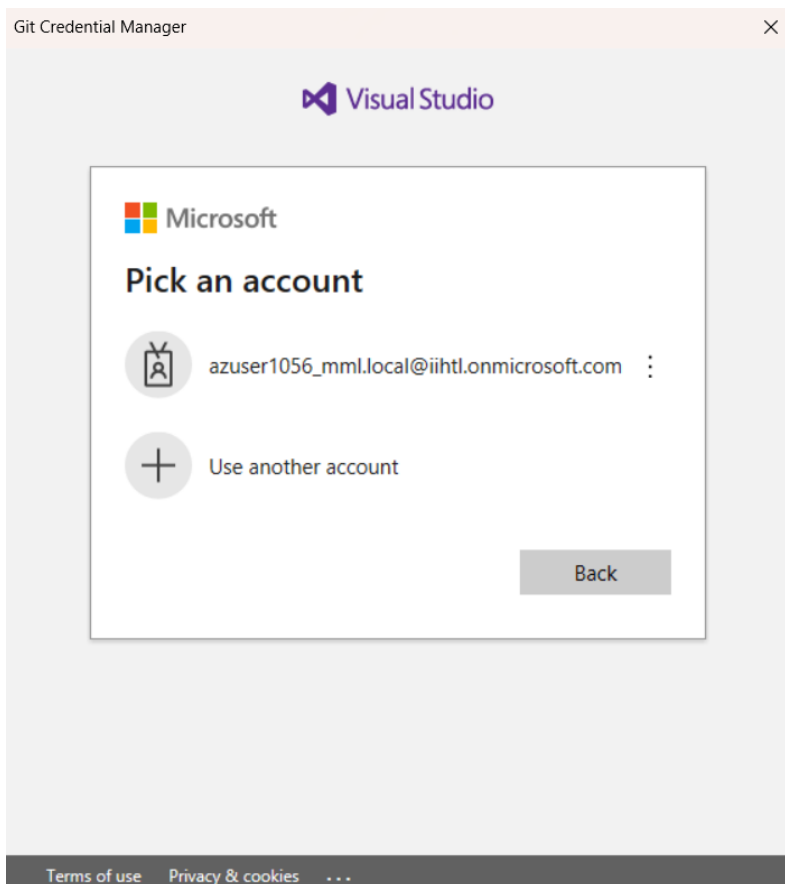
- Open Git bash



- Copy the HTTPS of our Azure Organization



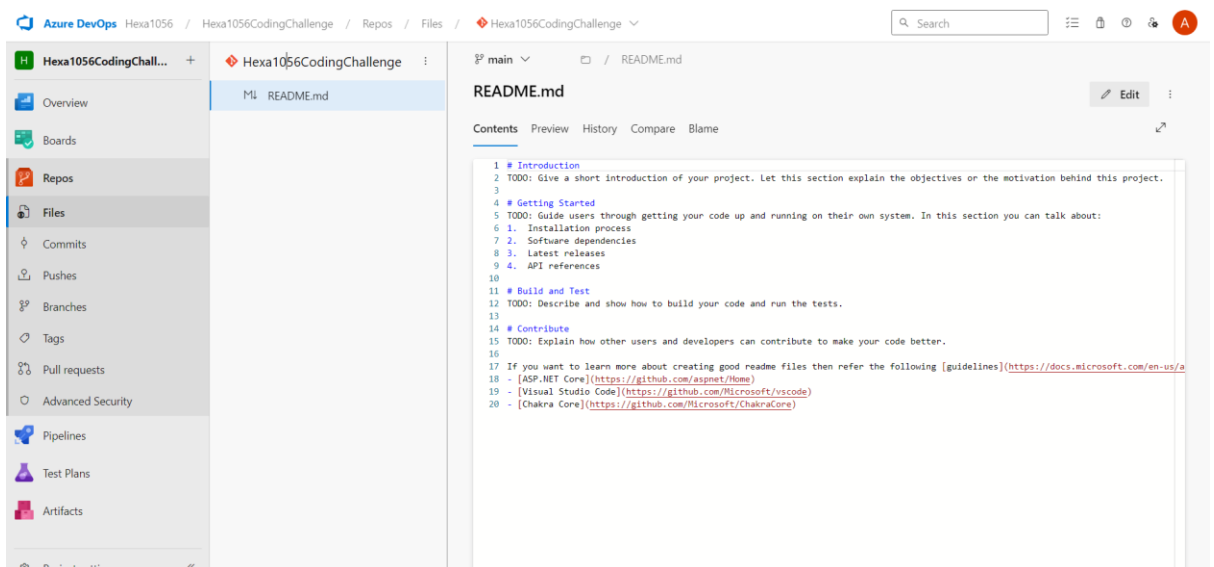
- Give command as `git clone <HTTPS>` and log in the git credential



- Project cloned successfully

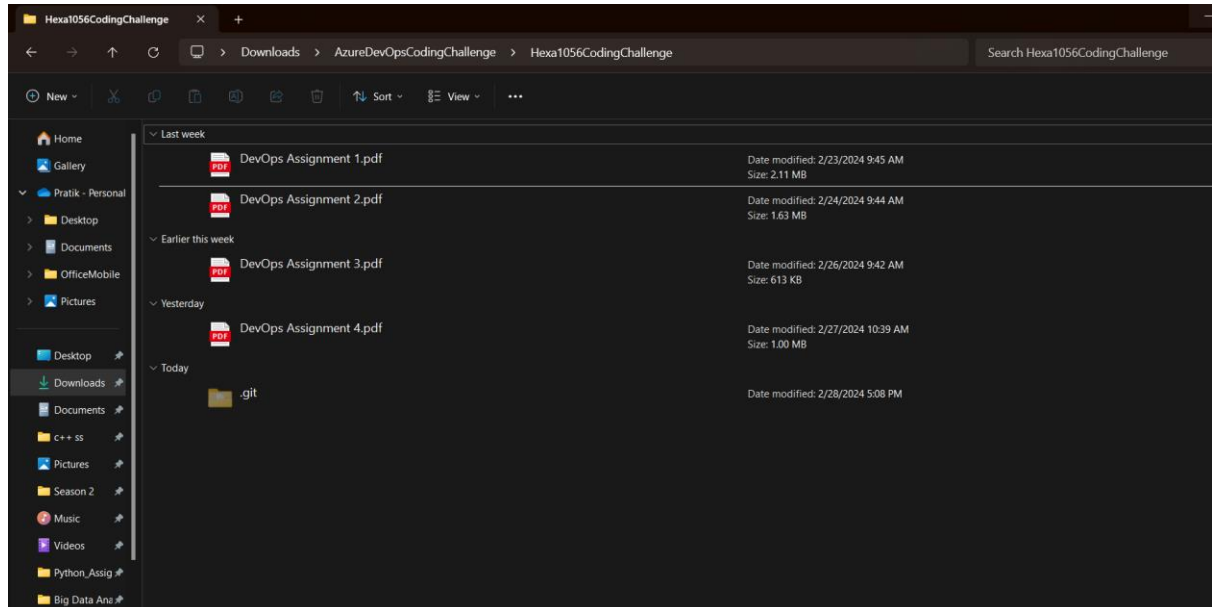
```
MINGW64~/c/Users/dell/Downloads/AzureDevOpsCodingChallenge
dell@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge
$ git clone https://Hexa1056@dev.azure.com/Hexa1056/Hexa1056CodingChallenge/_git/Hexa1056CodingChallenge
Cloning into 'Hexa1056CodingChallenge'...
warning: You appear to have cloned an empty repository.
dell@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge
$ |
```

- Cloned project in DevOps Organization



- **Git Push**

- Add some files in the cloned folder



- Now in git bash type ``git add .`` to add the files in stage area

```
dell@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge
$ cd Hexa1056CodingChallenge

dell@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge/Hexa1056CodingChallenge (master)
$ git add .

dell@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge/Hexa1056CodingChallenge (master)
$ |
```

- Now Add a commit message

```
de1l@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge/Hexa1056CodingChallenge (master)
$ git commit -m "CodingChallenge"
[master (root-commit) 2e02b33] CodingChallenge
4 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 DevOps Assignment 1.pdf
create mode 100644 DevOps Assignment 2.pdf
create mode 100644 DevOps Assignment 3.pdf
create mode 100644 DevOps Assignment 4.pdf

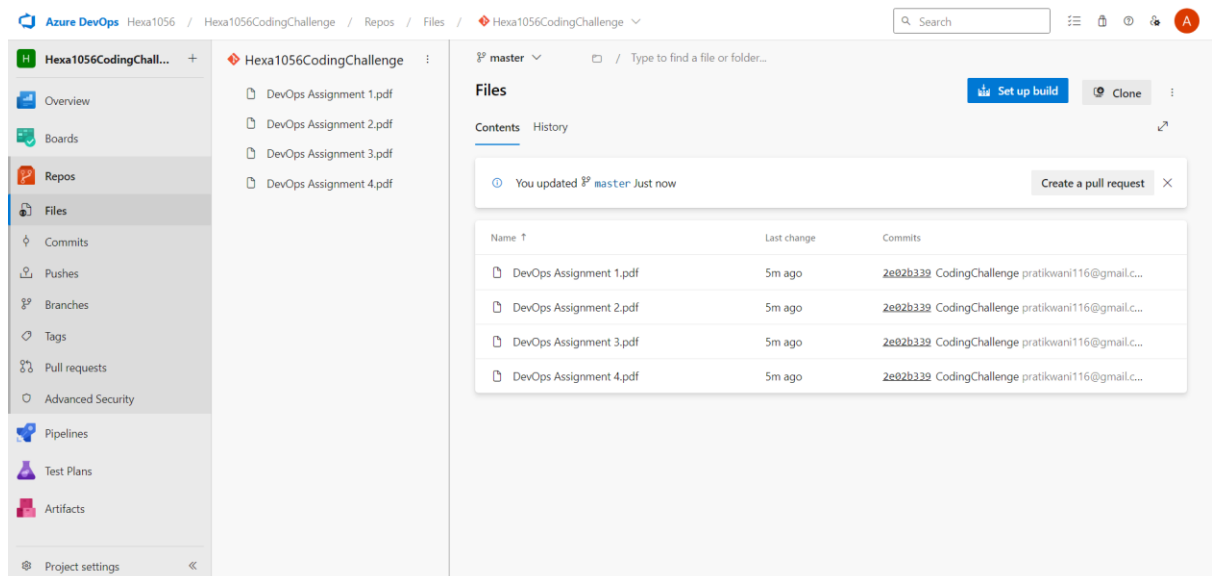
de1l@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge/Hexa1056CodingChallenge (master)
$
```

- Push the files into the repository

```
de1l@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge/Hexa1056CodingChallenge (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 4.73 MiB | 1.54 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Analyzing objects... (6/6) (2471 ms)
remote: Validating commits... (1/1) done (0 ms)
remote: Storing packfile... done (136 ms)
remote: Storing index... done (44 ms)
To https://dev.azure.com/Hexa1056/Hexa1056CodingChallenge/_git/Hexa1056CodingChallenge
 * [new branch]      master -> master

de1l@PRATIK MINGW64 ~/Downloads/AzureDevOpsCodingChallenge/Hexa1056CodingChallenge (master)
$
```

- Files successfully pushed in Master branch of DevOps organization



- **Git Pull**

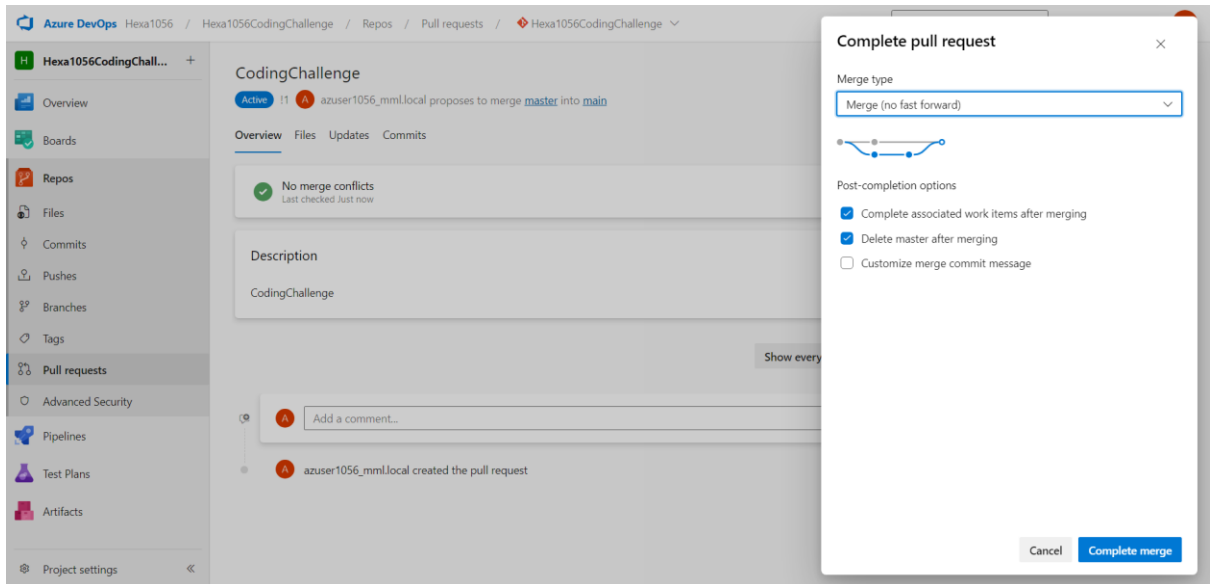
- Create new pull request to pull the files from master branch to main branch and click on create

The screenshot shows the 'New pull request' interface in Azure DevOps. The left sidebar contains navigation links: Overview, Boards, Repos, Files, Commits, Pushes, Branches, Tags, Pull requests (selected), Advanced Security, Pipelines, Test Plans, and Artifacts. The main area is titled 'New pull request' and shows a form for creating a pull request. The 'Title' field contains 'CodingChallenge'. The 'Description' field contains 'CodingChallenge'. The 'Reviewers' section has a search bar and a button 'Add required reviewers'. The 'Work items to link' section has a search bar. The 'Tags' section is empty. The 'Overview' tab is selected, and the 'Files' and 'Commits' tabs are also visible.

- Now click on complete

The screenshot shows the 'CodingChallenge' pull request in Azure DevOps. The left sidebar is the same as in the previous screenshot. The main area is titled 'CodingChallenge' and shows the pull request details. The 'Overview' tab is selected, and the 'Files', 'Updates', and 'Commits' tabs are also visible. The 'No merge conflicts' status is shown with a green checkmark. The 'Description' field contains 'CodingChallenge'. The 'Reviewers' section shows 'No required reviewers' and 'No optional reviewers'. The 'Tags' section shows 'No tags'. The 'Work items' section shows 'No work items'. The 'Add a comment...' field is visible. The 'Approve' and 'Complete' buttons are at the top right. The 'azuser1056_mml.local' user is shown as the creator of the pull request.

- Select Merge and complete the merge



- Files successfully pulled to main branch

