**Akilesh K**

**k.akilesh123@gmail.com**

**Data engineering - Batch 1**

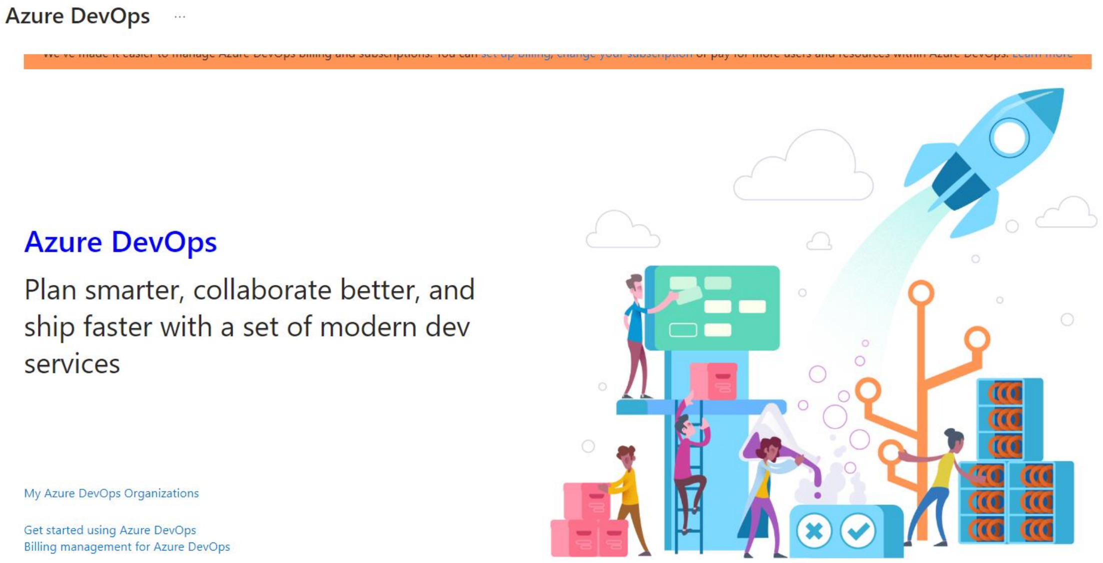
**Date: 28-02-24**

# CODING ASSESSMENT – AZURE DEVOPS

## TASK 1

**Create Azure DevOps Environment and configuring Azure DevOps Git Repository, configure on your local git to implement this upload few test files on same.**

- **Launch azure devOps**



Azure DevOps ···

We've made it easier to manage Azure DevOps billing and subscriptions. You can set up billing, change your subscription or pay for more users and resources within Azure DevOps. Learn more

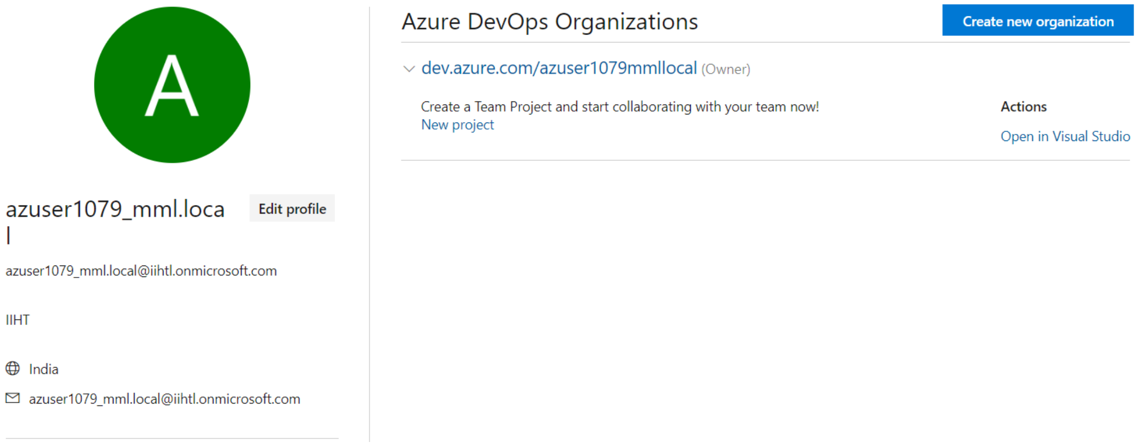
Azure DevOps

Plan smarter, collaborate better, and ship faster with a set of modern dev services

My Azure DevOps Organizations

Get started using Azure DevOps
Billing management for Azure DevOps

- **Create a new organization**



A

azuser1079_mml.loca
l

Edit profile

azuser1079_mml.local@iihtl.onmicrosoft.com

IIHT

India

azuser1079_mml.local@iihtl.onmicrosoft.com

Azure DevOps Organizations

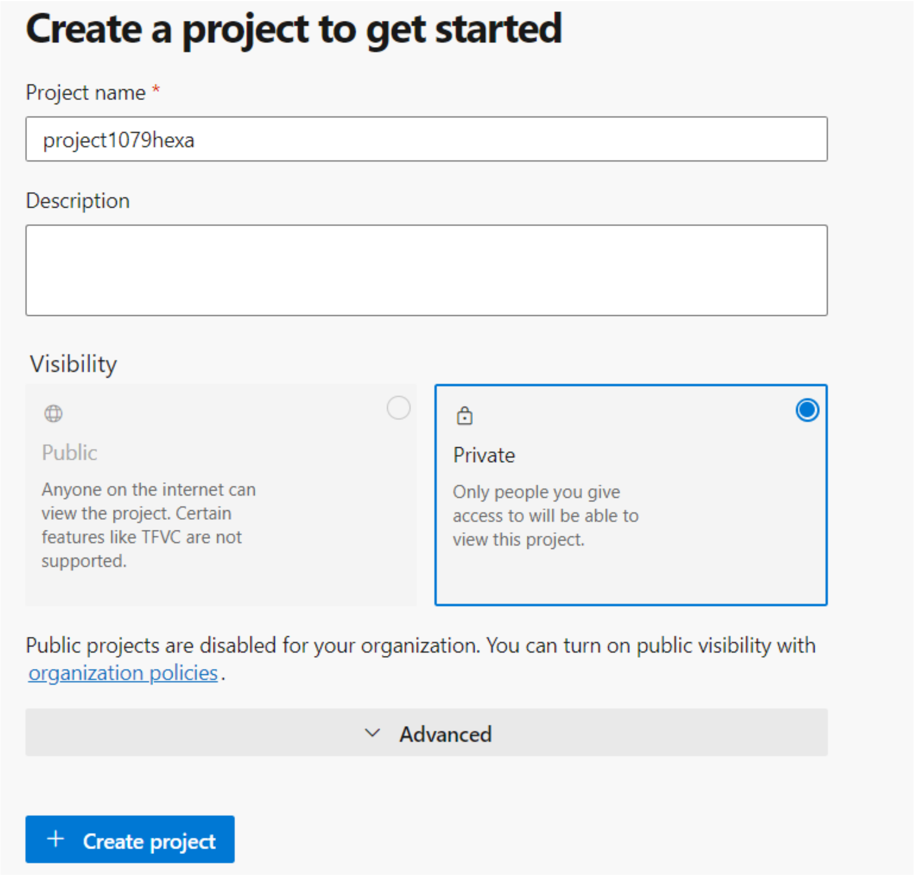
Create new organization

dev.azure.com/azuser1079mmllocal (Owner)

Create a Team Project and start collaborating with your team now!
New project

Actions
Open in Visual Studio

- **Create a new project**



## Create a project to get started

Project name *

project1079hexa

Description

Visibility

○ Public

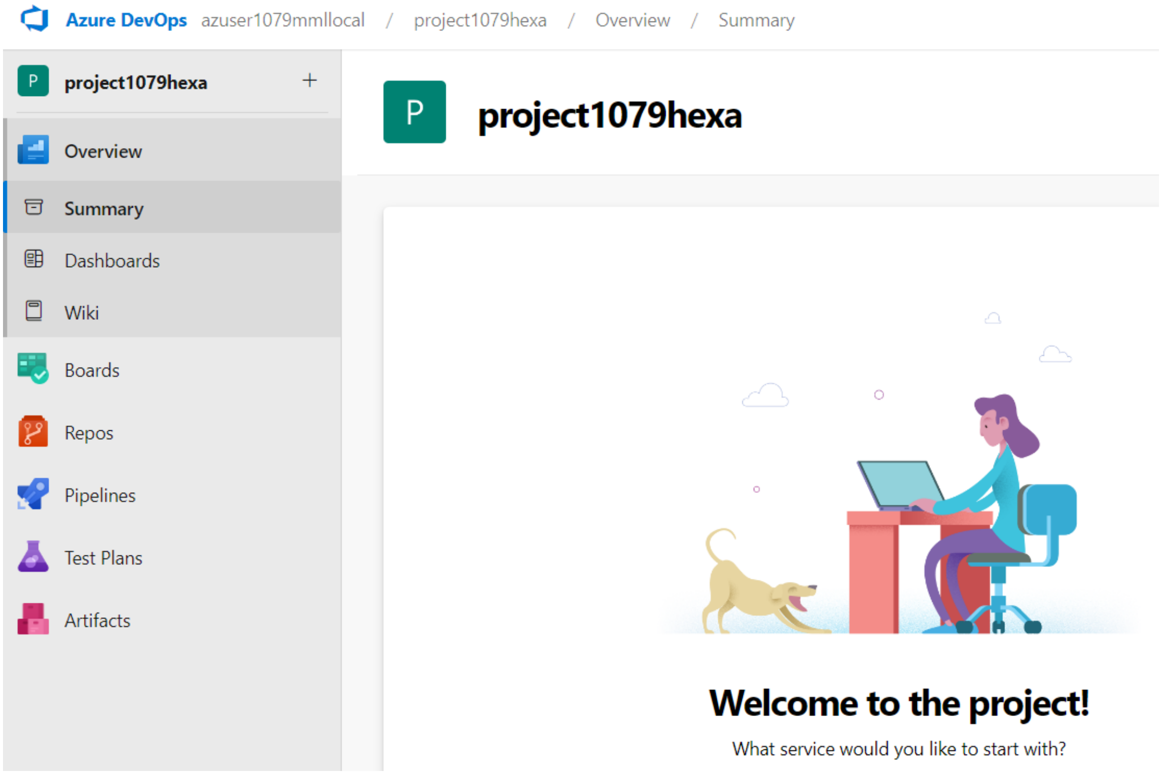
Anyone on the internet can view the project. Certain features like TFVC are not supported.

● Private

Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with organization policies.

- **Summary of azure DevOps**

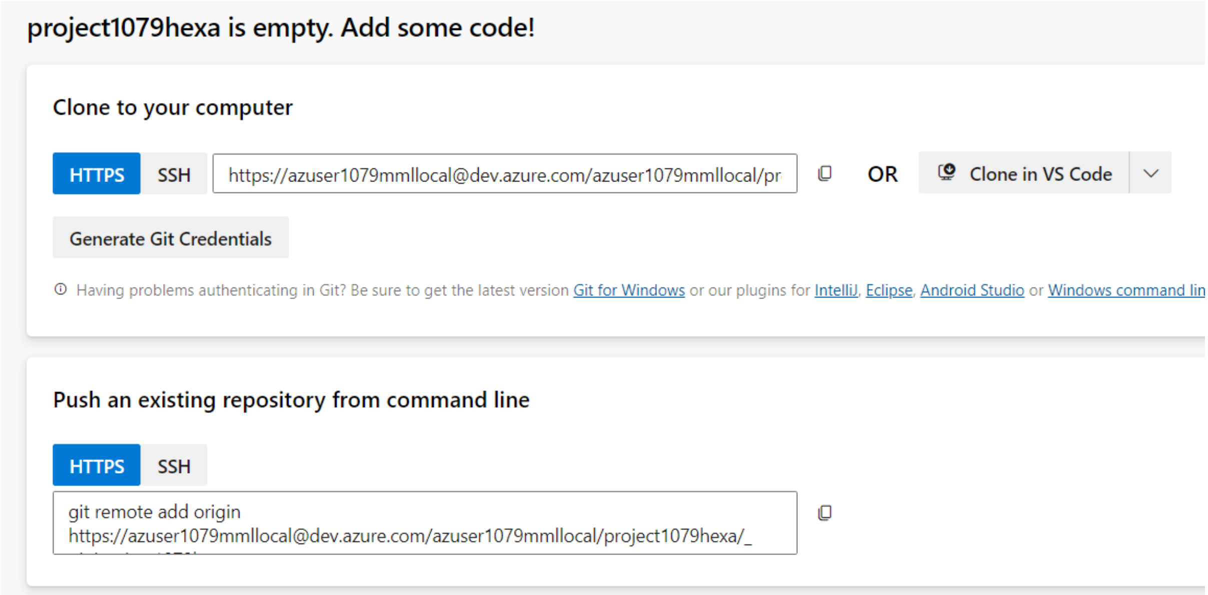


Azure DevOps azuser1079mmllocal / project1079hexa / Overview / Summary

Welcome to the project!

What service would you like to start with?

- **Go to repos and copy HTTPS link**



project1079hexa is empty. Add some code!

Clone to your computer

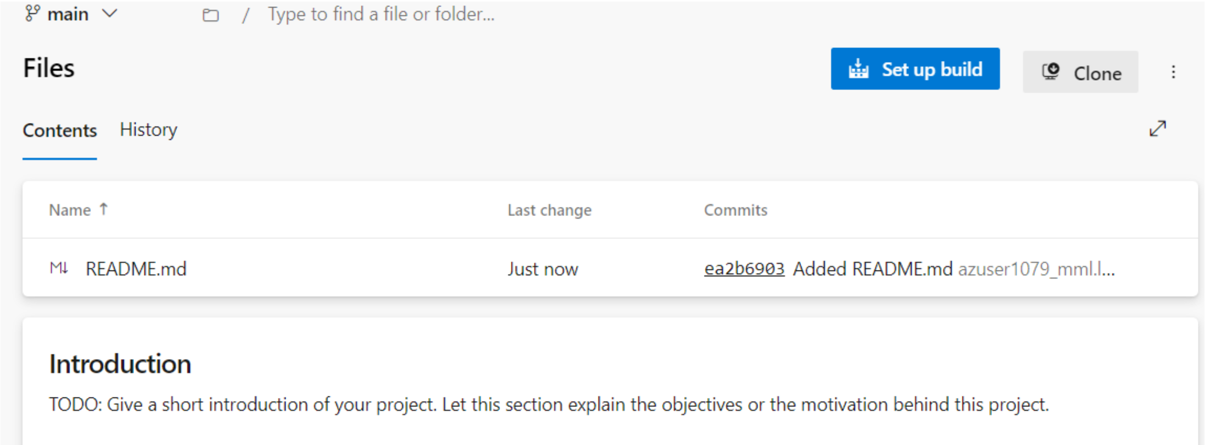
https://azuser1079mmllocal@dev.azure.com/azuser1079mmllocal/pr

ⓘ Having problems authenticating in Git? Be sure to get the latest version Git for Windows or our plugins for IntelliJ, Eclipse, Android Studio or Windows command lin

Push an existing repository from command line

```
git remote add origin
https://azuser1079mmllocal@dev.azure.com/azuser1079mmllocal/project1079hexa/_
```

- **Set it up by adding a readme file**

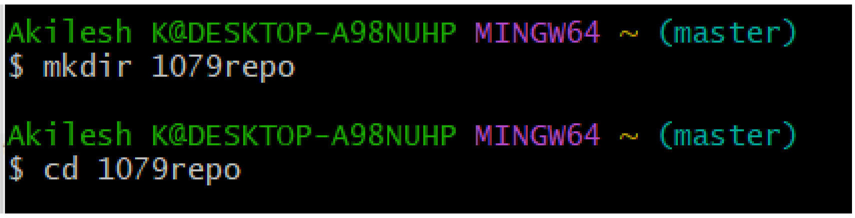


| Name ↑ | Last change | Commits |
|---|---|---|
| README.md | Just now | ea2b6903 Added README.md azuser1079_mml.l... |

Introduction

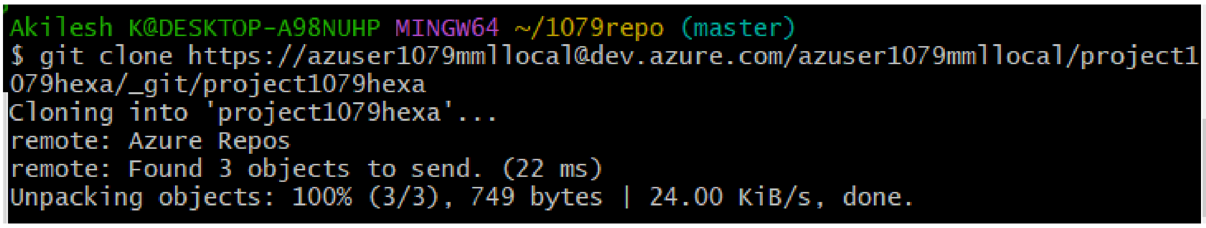
TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.

- **Open Git bash and make a new directory**



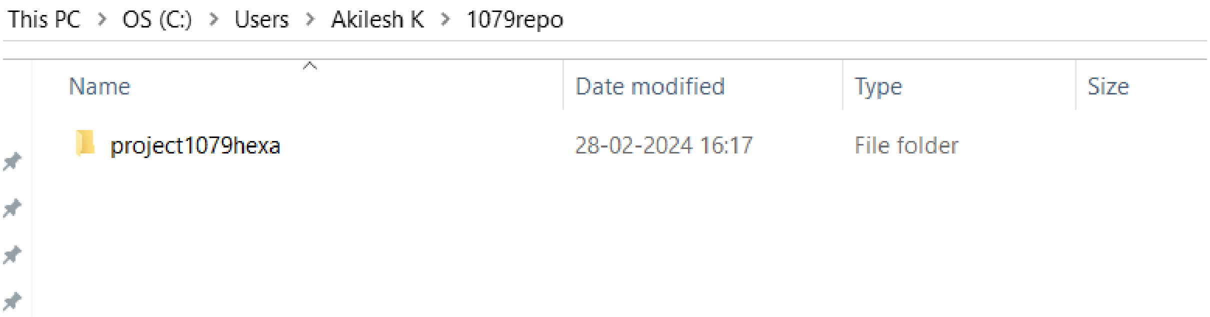
```
Akilesh K@DESKTOP-A98NUHP MINGW64 ~ (master)
$ mkdir 1079repo

Akilesh K@DESKTOP-A98NUHP MINGW64 ~ (master)
$ cd 1079repo
```

- **Using Git clone, copy the project file to our local repository**



```
Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo (master)
$ git clone https://azuser1079mmllocal@dev.azure.com/azuser1079mmllocal/project1
079hexa/_git/project1079hexa
Cloning into 'project1079hexa'...
remote: Azure Repos
remote: Found 3 objects to send. (22 ms)
Unpacking objects: 100% (3/3), 749 bytes | 24.00 KiB/s, done.
```

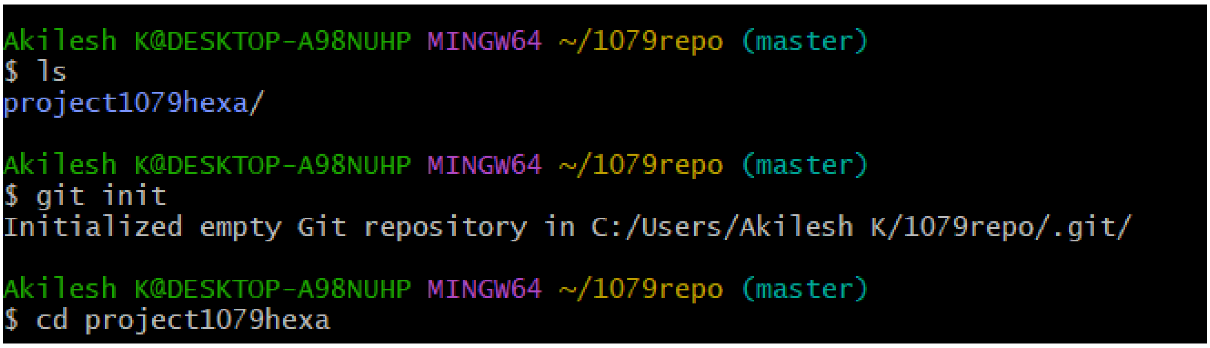
- **It gets cloned in our desired location**



This PC › OS (C:) › Users › Akilesh K › 1079repo

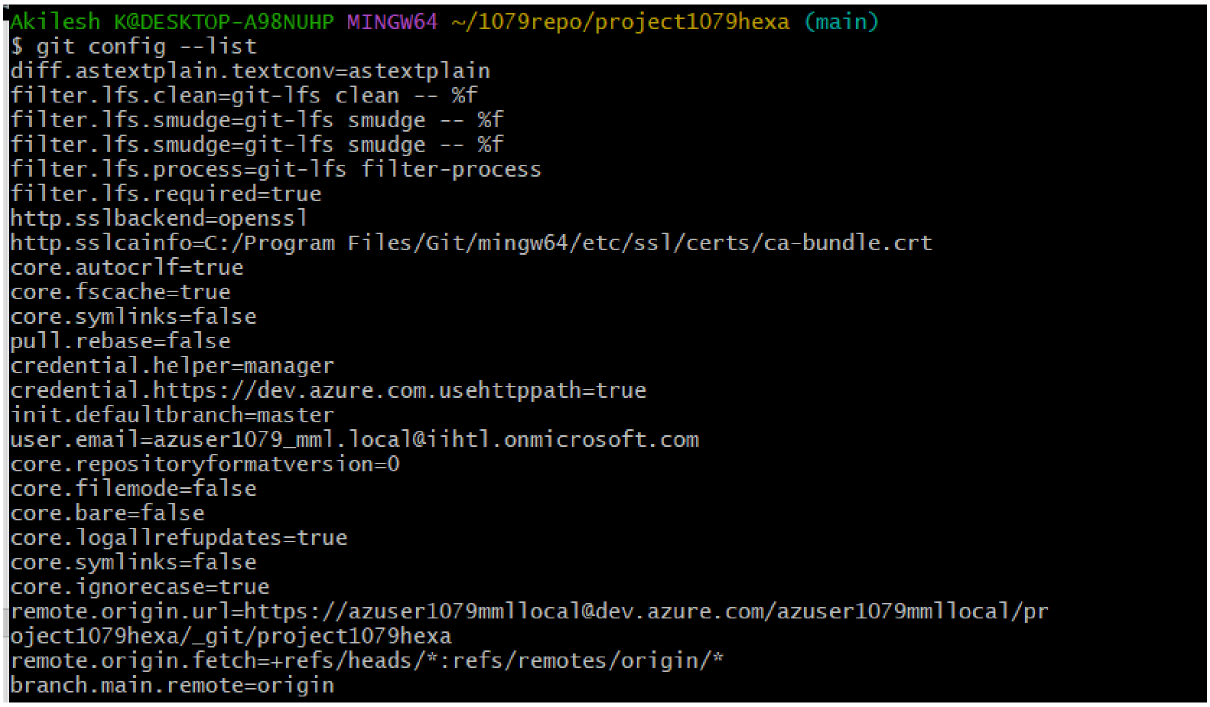
| Name | Date modified | Type | Size |
|---|---|---|---|
| project1079hexa | 28-02-2024 16:17 | File folder | |

- **Go to project folder and initialize git**



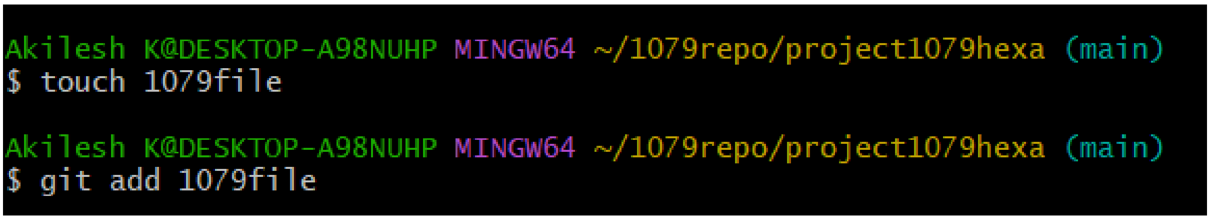
```
Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo (master)
$ ls
project1079hexa/

Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo (master)
$ git init
Initialized empty Git repository in C:/Users/Akilesh K/1079repo/.git/

Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo (master)
$ cd project1079hexa
```

- **Configurate Git**



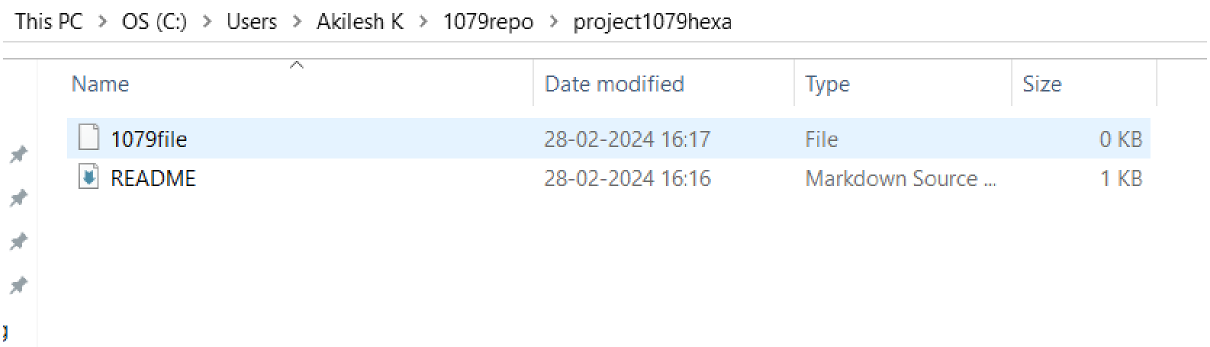
```
Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo/project1079hexa (main)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=azuser1079_mml.local@iihtl.onmicrosoft.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=https://azuser1079mmllocal@dev.azure.com/azuser1079mmllocal/pr
oject1079hexa/_git/project1079hexa
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
```

- **Using touch command create a new file**



```
Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo/project1079hexa (main)
$ touch 1079file

Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo/project1079hexa (main)
$ git add 1079file
```

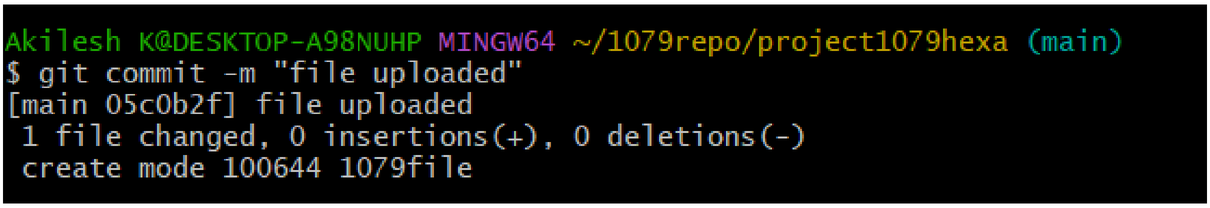
- **The file is created**



This PC > OS (C:) > Users > Akilesh K > 1079repo > project1079hexa

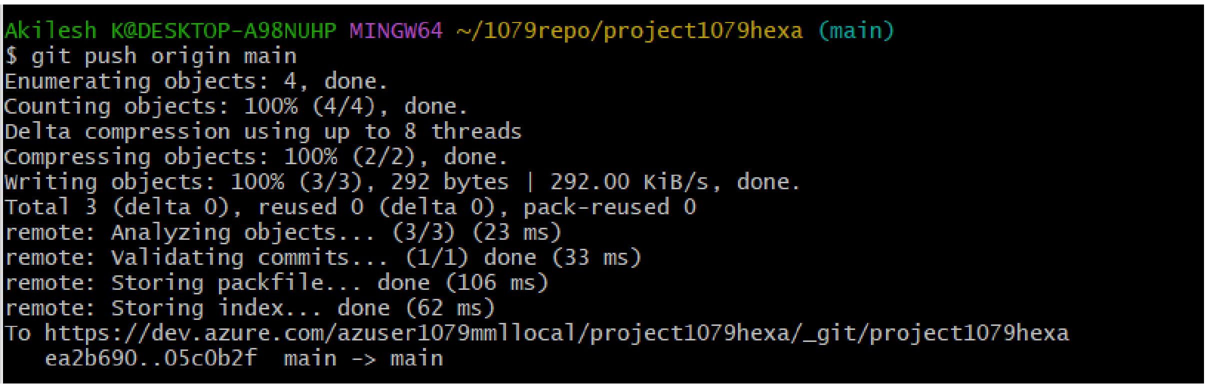
| Name | Date modified | Type | Size |
|---|---|---|---|
| 1079file | 28-02-2024 16:17 | File | 0 KB |
| README | 28-02-2024 16:16 | Markdown Source ... | 1 KB |

- **Write git commit to mention the changed and save this file**



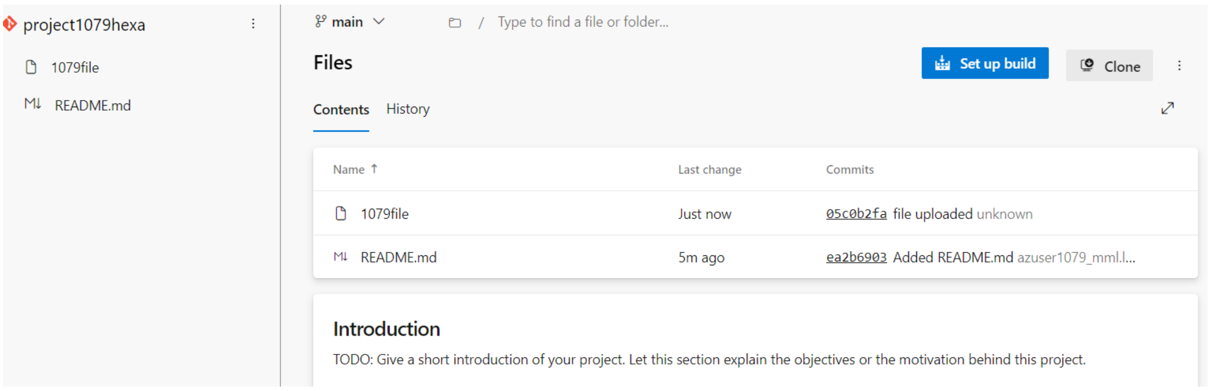
```
Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo/project1079hexa (main)
$ git commit -m "file uploaded"
[main 05c0b2f] file uploaded
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1079file
```

- **Using Git push , upload the changed file to azure repository**



```
Akilesh K@DESKTOP-A98NUHP MINGW64 ~/1079repo/project1079hexa (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 292 bytes | 292.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Analyzing objects... (3/3) (23 ms)
remote: Validating commits... (1/1) done (33 ms)
remote: Storing packfile... done (106 ms)
remote: Storing index... done (62 ms)
To https://dev.azure.com/azuser1079mmllocal/project1079hexa/_git/project1079hexa
   ea2b690..05c0b2f  main -> main
```

- **The file is pushed in the repositary**

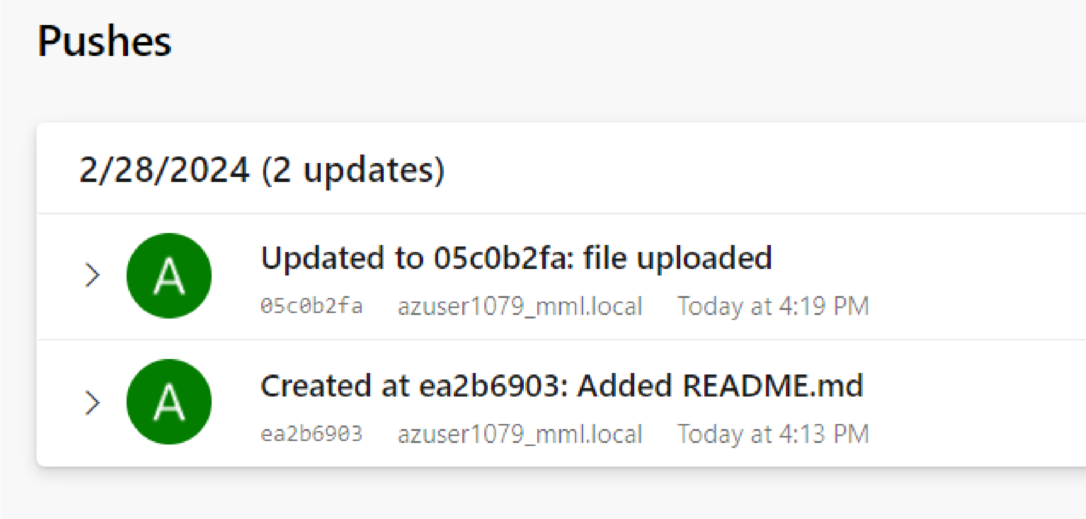


| Name ↑ | Last change | Commits |
|---|---|---|
| 1079file | Just now | 05c0b2fa file uploaded unknown |
| README.md | 5m ago | ea2b6903 Added README.md azuser1079_mml.l... |

Introduction

TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.

- **We can view the git pushed history**



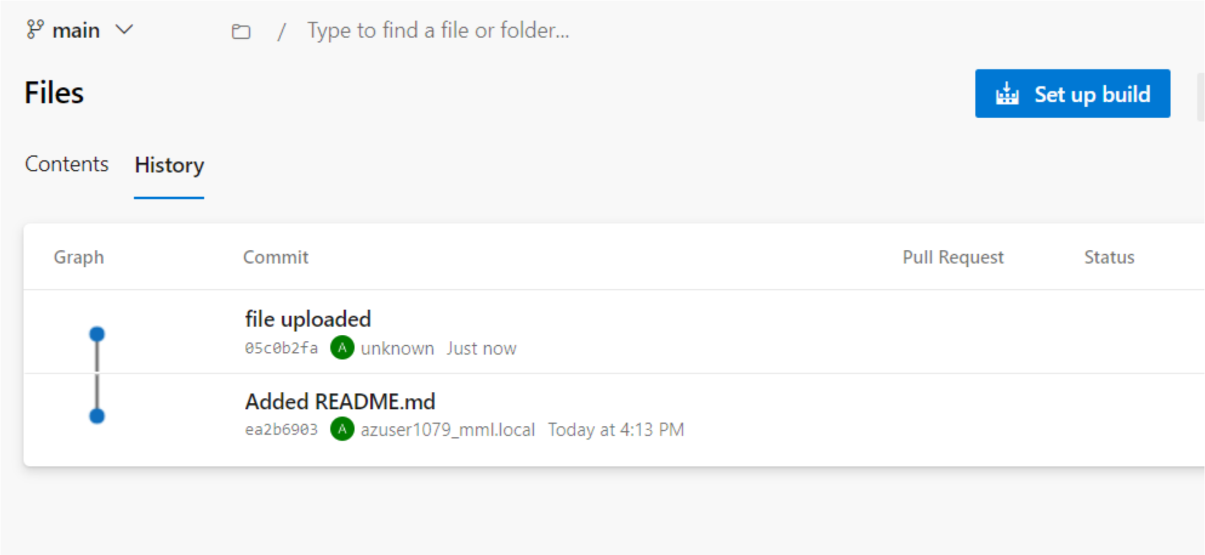
Pushes

2/28/2024 (2 updates)

Updated to 05c0b2fa: file uploaded
05c0b2fa azuser1079_mml.local Today at 4:19 PM

Created at ea2b6903: Added README.md
ea2b6903 azuser1079_mml.local Today at 4:13 PM

- **The commit message is displayed**



| Graph | Commit | Pull Request | Status |
|---|---|---|---|
| | file uploaded<br>05c0b2fa A unknown Just now | | |
| | Added README.md<br>ea2b6903 A azuser1079_mml.local Today at 4:13 PM | | |

## TASK 2

## Leverage the practises of CI CD Using azure Data engineering

Implementing Continuous Integration and Continuous Deployment (CI/CD) practices in Azure Data Engineering involves automating the process of building, testing, and deploying data solutions.

**Leverage CI/CD using Azure Data Engineering**

- **Version Control with Git**: Begin by storing your code, scripts, and configurations in a version control system like Git. This allows multiple team members to collaborate on the same codebase and track changes over time.

- **Automated Testing:** Set up automated testing to validate the correctness and quality of your data pipelines and transformations. This ensures that any changes introduced do not break existing functionality. Azure provides services like Azure Data Factory and Azure Databricks, which offer testing capabilities for data workflows and transformations.

- **Continuous Integration (CI**): Automate the process of integrating code changes into a shared repository multiple times a day. In Azure Data Engineering, you can use Azure DevOps pipelines to trigger CI builds whenever code changes are pushed to the repository. These builds can include tasks to compile code, run tests, and package artifacts.

- **Continuous Deployment (CD):** Automate the deployment of data solutions to different environments (e.g., development, staging, production) after passing CI tests. Azure Data Factory allows you to create release pipelines that automatically deploy data pipelines and configurations to various environments based on predefined triggers, such as successful CI builds.

- **Infrastructure as Code (IaC):** Define your data infrastructure, including data stores, compute resources, and networking, as code using Azure Resource Manager (ARM) templates or Azure Bicep. This allows you to provision and manage infrastructure programmatically, ensuring consistency and repeatability across environments.

- **Monitoring and Logging:** Implement monitoring and logging solutions to track the performance, availability, and health of your data solutions in real-time. Azure Monitor provides monitoring capabilities for Azure services, allowing you to set up alerts, visualize metrics, and diagnose issues proactively.

- **Security and Compliance:** Incorporate security best practices and compliance requirements into your CI/CD pipelines. Azure offers built-in security features and compliance certifications for data services, such as encryption at rest and in transit, identity and access management (IAM), and regulatory compliance controls.

**CI/CD and Ephemeral Environments:** CI/CD tools automate the provisioning of ephemeral dev/test environments in response to branching events in Git. These environments are temporary and are spun up for development, testing, or experimentation purposes. They closely mirror production environments and are automatically torn down after testing is completed to avoid unnecessary costs and resource usage.

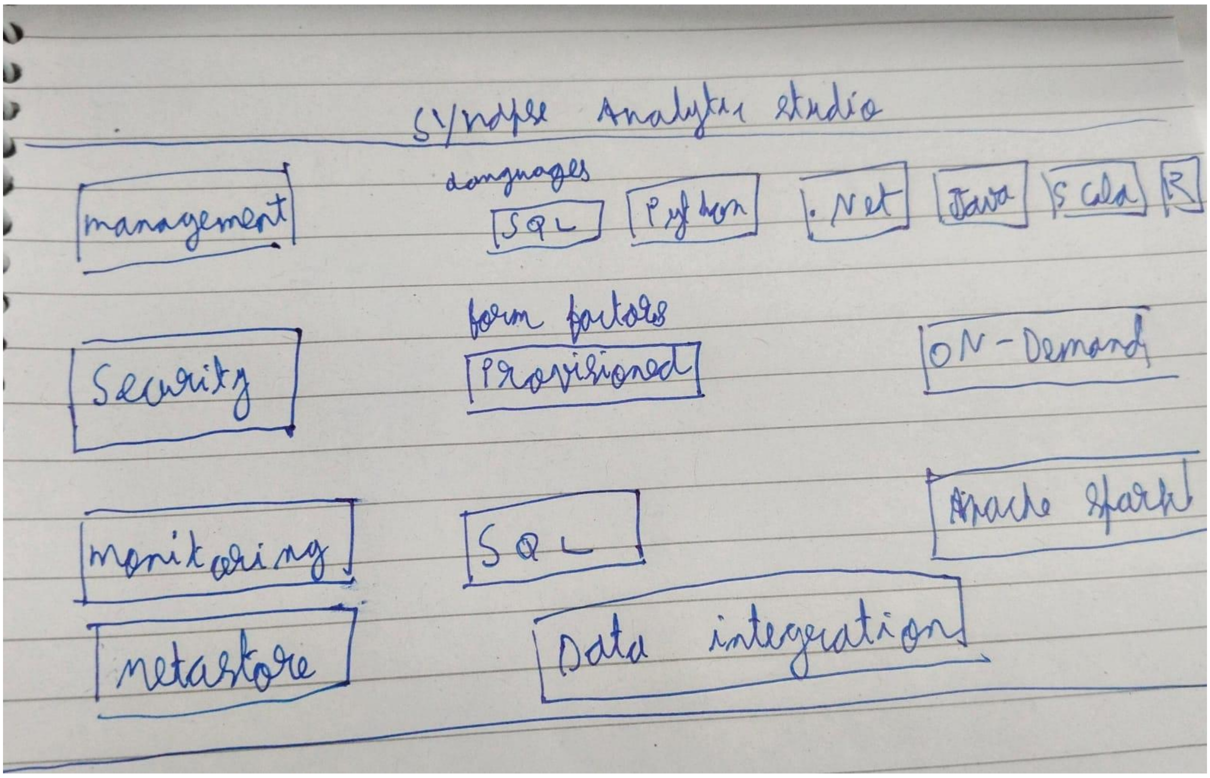
### Best Practices

- **Handling Large Data Files:** Avoid storing large datasets in Git repositories. Instead, version cloud storage buckets or use dedicated data versioning tools.
- **Pull Requests:** Utilize pull requests for proposing changes, gathering feedback, and ensuring code quality before merging.
- **Code Reviews:** Foster a culture of code review to catch errors, share knowledge, and maintain coding standards.
- **Commit Often:** Make frequent, smaller commits rather than large, infrequent ones to facilitate easier merging and tracking of changes.
- **Clear Commit Messages:** Write clear and concise commit messages that explain the rationale behind the changes, not just the changes themselves.
- **Branch Deployments:** Implement branch deployments to create staging or temporary environments for testing and validation of changes before merging into the main or production branch.

By adopting CI/CD practices in Azure Data Engineering, organizations can accelerate the development lifecycle, improve collaboration between teams, and ensure the reliability and scalability of their data solutions

## TASK 3

## Explain the architecture of the Azure synpase



Azure Synapse Analytics is a cloud-based analytics service that brings together enterprise data warehousing and big data analytics. Its architecture is designed to handle diverse workloads and support both traditional relational data and big data analytics seamlessly.

**Integration Layer:**

- At the core of Azure Synapse is an integration layer that seamlessly combines both big data and relational data processing capabilities.
- It integrates with various data sources and services within the Azure ecosystem, as well as external data sources.
- Designed for analytics workloads at any scale
- SaaS developer experiences for code free and code first

- Multiple languages suited to different analytics workloads
- Integrated analytics runtimes available provisioned and serverless on demand
- Integrated platform services for, management, security, monitoring, and metastore

**SQL Analytics**

- Azure Synapse provides a SQL pool, which is a distributed, parallel, and fully managed data warehouse.
- SQL pool enables you to run complex analytical queries on large datasets with high performance and scalability.
- It supports T-SQL queries and provides familiar SQL-based tools and interfaces for data analysts and developers.

**Spark**

- Azure Synapse includes a Spark pool that provides Apache Spark-based big data processing capabilities.
- Spark pool is ideal for processing and analyzing large volumes of semi-structured or unstructured data, including streaming data.
- It supports various Spark components and libraries for data manipulation, machine learning, and streaming analytics.
- Spark for big data processing with Python, Scala, R and .NET

**Data Integration:**

- Azure Synapse offers robust data integration capabilities for ingesting and processing data from various sources.
- It includes connectors for seamless integration with Azure Data Lake Storage, Azure Blob Storage, Azure SQL Database, Azure Cosmos DB, and more.
- Data integration features include data ingestion, data movement, data transformation, and data orchestration.

**Analytics Services:**

- Azure Synapse offers a range of analytics services and tools for data exploration, visualization, and advanced analytics.
- This includes integration with Power BI for interactive data visualization, Azure Machine Learning for building and deploying machine learning models, and Azure Data Lake Analytics for ad-hoc big data analytics.

**Integration with Azure Services:**

- Integrates seamlessly with various Azure services like Azure Data Lake Storage, Azure Blob Storage, and Power BI for comprehensive analytics solutions.
- This architecture provides a foundation for handling diverse workloads, from traditional data warehousing to big data analytics, in a scalable and efficient manner within the Azure cloud environment.

Overall, the architecture of Azure Synapse Analytics is designed to provide a unified and scalable platform for modern analytics and data-driven insights, enabling organizations to leverage both relational and big data analytics seamlessly in the cloud.