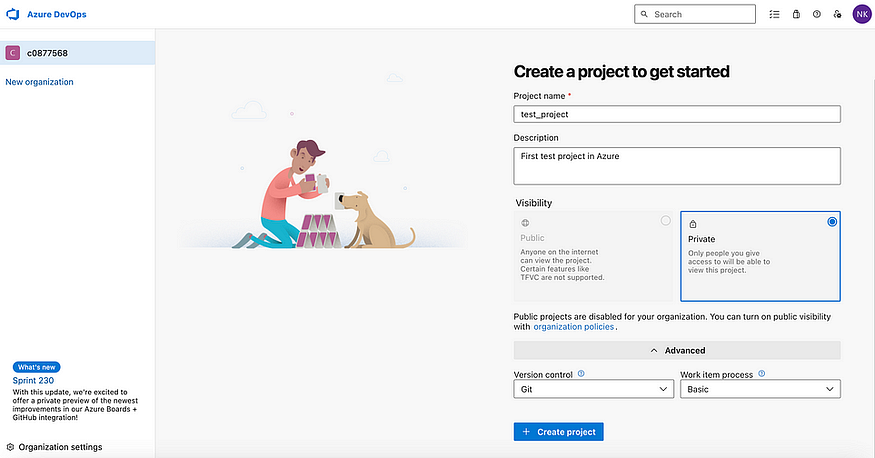
**Automating a Data Engineering Pipeline using DevOps**

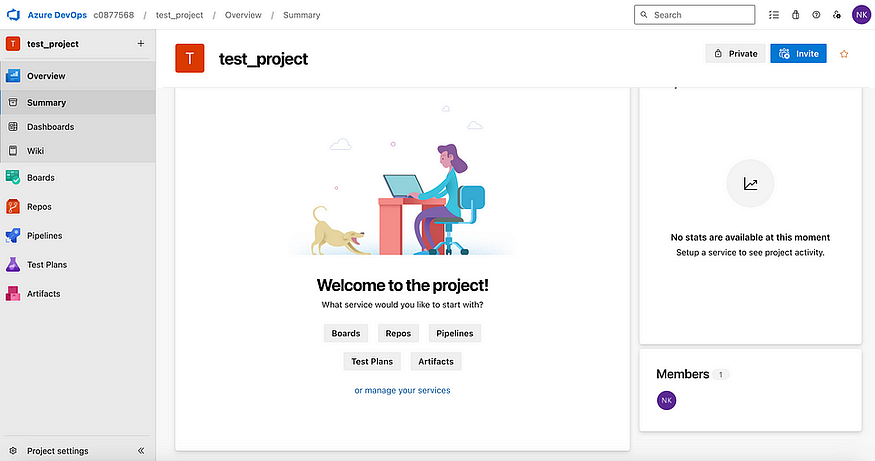
**Setting Up Your First Azure DevOps Project**

* Create an **Azure DevOps account** to access tools like Repos (version control), Pipelines (build/release), Boards, etc.
* [**https://r.search.yahoo.com/\_ylt=AwrPqmm7HaxoIwIAH6S7HAx.;\_ylu=Y29sbwNzZzMEcG9zAzEEdnRpZAMEc2VjA3Ny/RV=2/RE=1757319867/RO=10/RU=https%3a%2f%2fazure.microsoft.com%2fen-in%2fproducts%2fdevops%2f/RK=2/RS=rBd7jRNn8fFkamTOAUdB3FdJo7**](https://r.search.yahoo.com/_ylt=AwrPqmm7HaxoIwIAH6S7HAx.;_ylu=Y29sbwNzZzMEcG9zAzEEdnRpZAMEc2VjA3Ny/RV=2/RE=1757319867/RO=10/RU=https%3a%2f%2fazure.microsoft.com%2fen-in%2fproducts%2fdevops%2f/RK=2/RS=rBd7jRNn8fFkamTOAUdB3FdJo7)
* **https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops-**

**Step 2: Creating a New Project**

* Navigate to Azure DevOps portal → **New Project**.
* Provide project name, choose version control (Git/TFVC), set visibility (Public/Private).





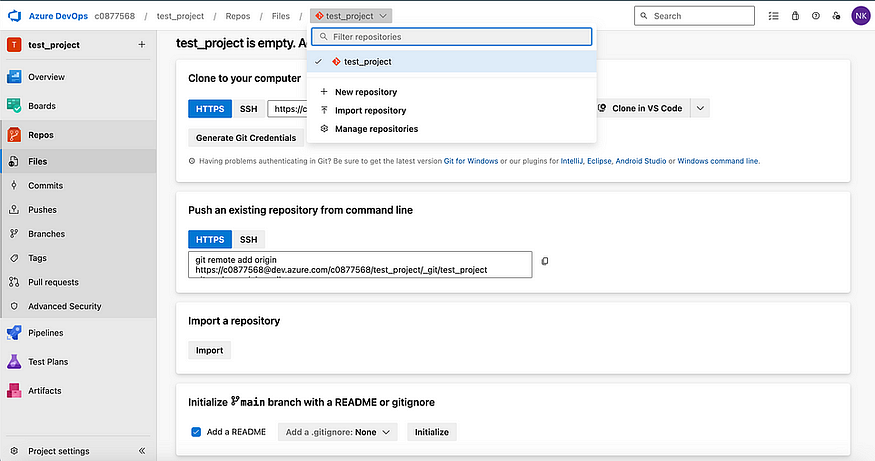
**Step 3: Setting Up Repositories**

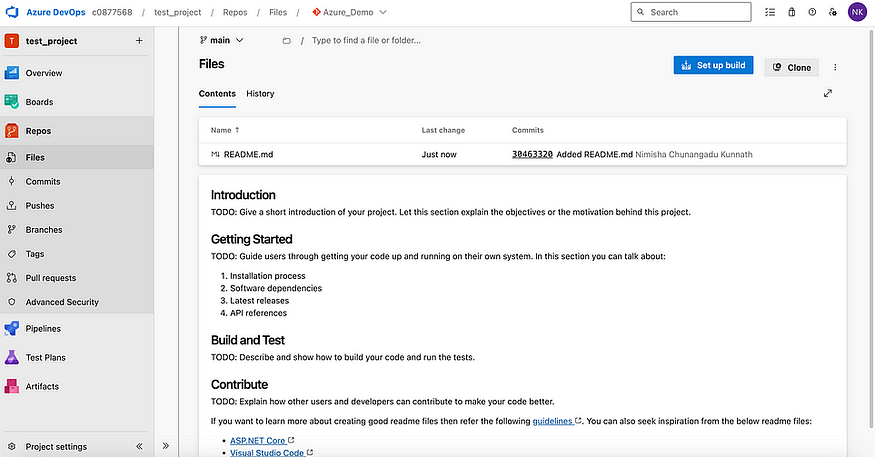
**3.1 Initialize Repository**

* Go to **Repos** → create a new repository.
* Add name/description, initialize with a README.

**3.2 Clone Repository**

* Copy repo URL → clone to local IDE (e.g., VS Code, IntelliJ).





**3.3 Integration with IntelliJ IDEA**

* Use Git integration in IntelliJ to clone, configure, and work with repo.
* Create/import project, enable version control, commit & push changes.

**3.4 Explore Azure Repos Features**

* Files: edit & track changes.
* Commits: view history & contributors.
* Branches: manage development workflows.
* Pull Requests: code reviews before merging.
* Tags & Security: snapshot releases, access control.

**Step 4: Configuring Build Pipelines**

**4.1 Select Repository as Source**

* Go to **Pipelines** → create new pipeline → choose repo as source.
* Configure pipeline (YAML or classic editor).

**4.2 Add Build Tasks**

* Restore dependencies, compile, test, package artifacts.
* Integrate external services (e.g., code analysis tools).

**4.3 Enable Triggers**

* Turn on **Continuous Integration (CI)** to auto-build on push.
* Configure batching, scheduled builds, or manual triggers.

**4.4 Verify & Improve**

* Run pipeline, monitor logs, fix failures.
* Continuously optimize pipeline settings.

**Step 5: Enabling CI/CD**

**5.1 Link Build & Release Pipelines**

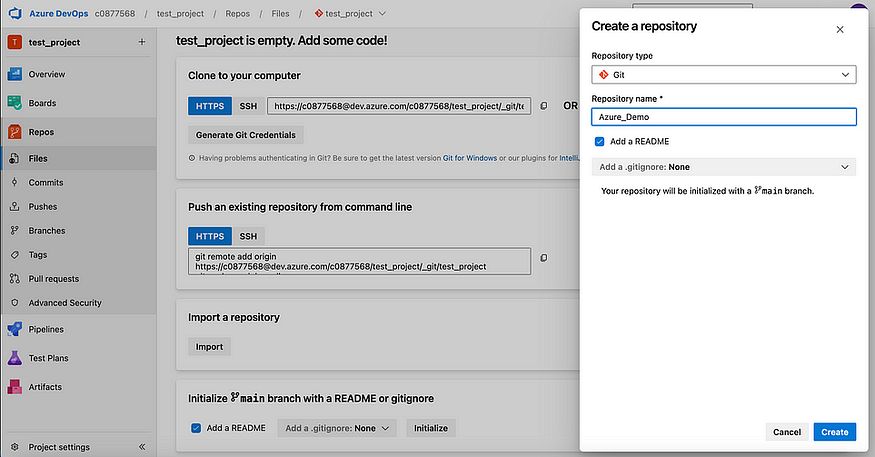
* Connect **build artifacts** to release pipeline.

**5.2 Configure Deployment Triggers**

* Enable **Continuous Deployment (CD)** → auto-deploy after successful build.
* Fine-tune triggers (conditional deployments, approvals).

**5.3 Validate CI/CD**

* Commit code → verify build auto-triggers.
* Check deployment logs for smooth releases.



**Final Outcome**

* Fully automated **CI/CD pipeline** for data engineering projects.
* Ensures:
  + Faster delivery with fewer manual steps.
  + Reliable, traceable deployments.
  + Continuous integration, testing, and deployment.