**Artifacts in Azure DevOps**

**Artifacts** in Azure DevOps are **packaged files or dependencies** (like .zip, .dll, .npm, or Docker images) or output produced during a build process in CI pipeline.

They act as a **storage hub** for reusable components, ensuring consistent deployments across environments.

**Why Use Artifacts?**

**1. Share Build Outputs**

* Store compiled code (e.g., .exe, .jar) or deployment packages (e.g., .war, .msi) for later use.
* Example: A CI pipeline builds an app and saves it as an artifact for the CD pipeline to deploy.

**2. Dependency Management**

* Host **NuGet, npm, Maven, or Python packages** in a private repository.
* Example: Share a custom .npm package across multiple projects.

**3. Reusability**

* Download artifacts in other pipelines (e.g., reuse a Docker image across dev/test/prod).
* Artifacts can be published in one pipeline and downloaded in another.
* Useful in multi-stage pipelines where, for example, a build artifact is created in one stage and deployed in another.

**4. Version Control & Traceability**

* Track versions of artifacts (e.g., myapp-v1.2.0.zip) to roll back if needed.
* You can trace which artifact was built from which pipeline run, which commit, and who triggered it.

**5. Faster Pipelines**

* Avoid rebuilding the same code repeatedly—just reuse the artifact.

**6. Package Sharing**

* You can store and share packages (like NuGet, npm, Maven, or Python packages) securely within your organization using **Azure Artifacts service**.

**Brief Explanation of Artifacts in Azure DevOps:**

When working with applications, different languages have their own central repositories to download packages:

* **Java applications** use [**Maven Central**](https://maven.org)
* **.NET applications** use [**NuGet.org**](https://nuget.org)

Similarly, in **Azure DevOps**, we use **Azure Artifacts** as a **central package manager** within our organization. It acts as a private, secure repository where we can store, manage, and share packages for our applications.

**Why Use Azure Artifacts?**

* While working on Java or .NET applications in Azure DevOps, we can easily fetch the required packages from **Azure Artifacts**.
* If a specific package isn’t available in Azure Artifacts, it has a special feature that allows it to automatically **download the package from external public repositories (like Maven Central or NuGet.org)** when needed.

## **What Can Azure Artifacts Store?**

Azure Artifacts isn’t limited to external packages. It can also store:

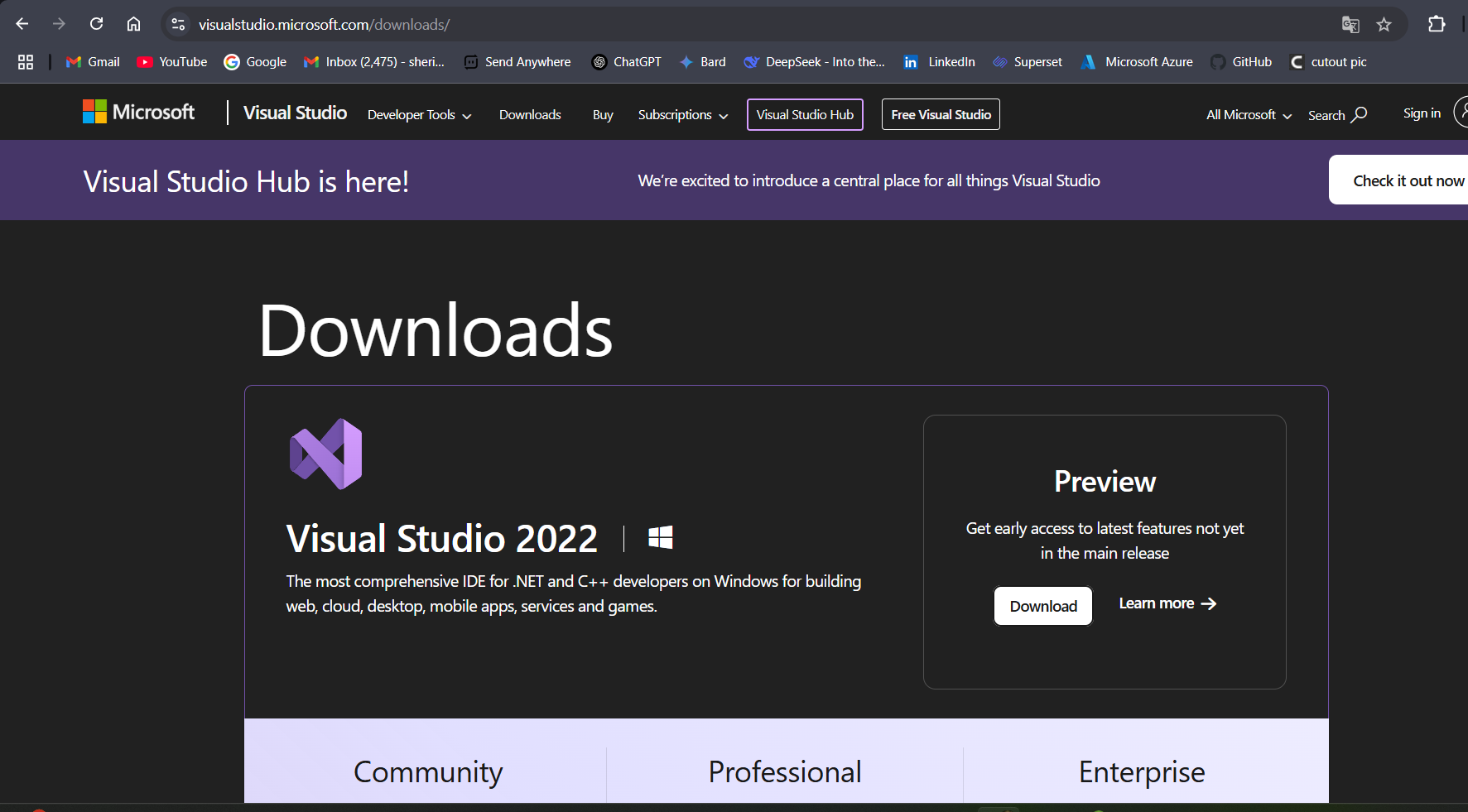
* **Build packages** like NuGet, Maven, npm, or Python packages
* **Published artifacts** — these are the output files generated from your **CI (Continuous Integration) pipelines**, which can later be used in deployment pipelines or shared with other projects.

**Summary:**

* Azure Artifacts acts as a **centralized package manager** within an organization using Azure DevOps.
* It can store both **downloaded packages** from public repositories and **custom packages** created within your organization.
* It also stores **published artifacts**, which are the results of your build pipelines, to be used in deployments or shared across projects.

Now let’s perform a task practically that, Creating of .NET application package manually and pushing it to the Azure Artifacts Using Visual community.

**Step1:** Download the Visual Community studio.



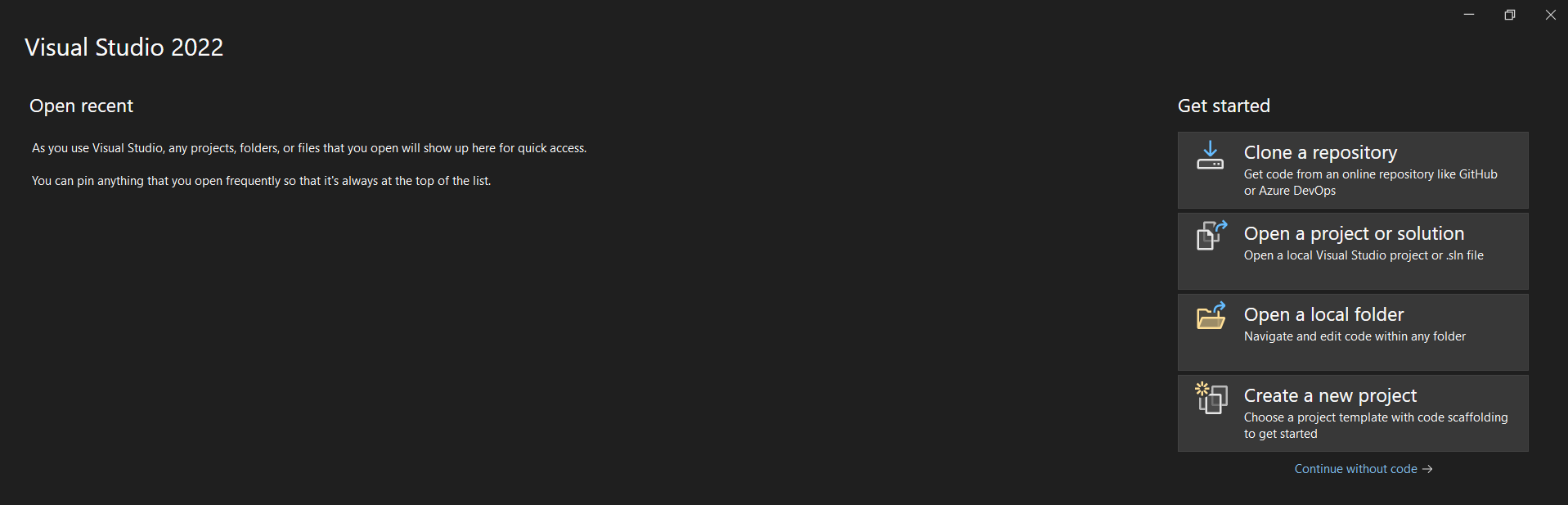


Fig: Continue without code.

**Step2:** Create a feed in azure Artifacts.

First create a project (project-01) within the project of Artifacts create a Feed

Organization🡪project🡪Artifacts🡪Create Feed.

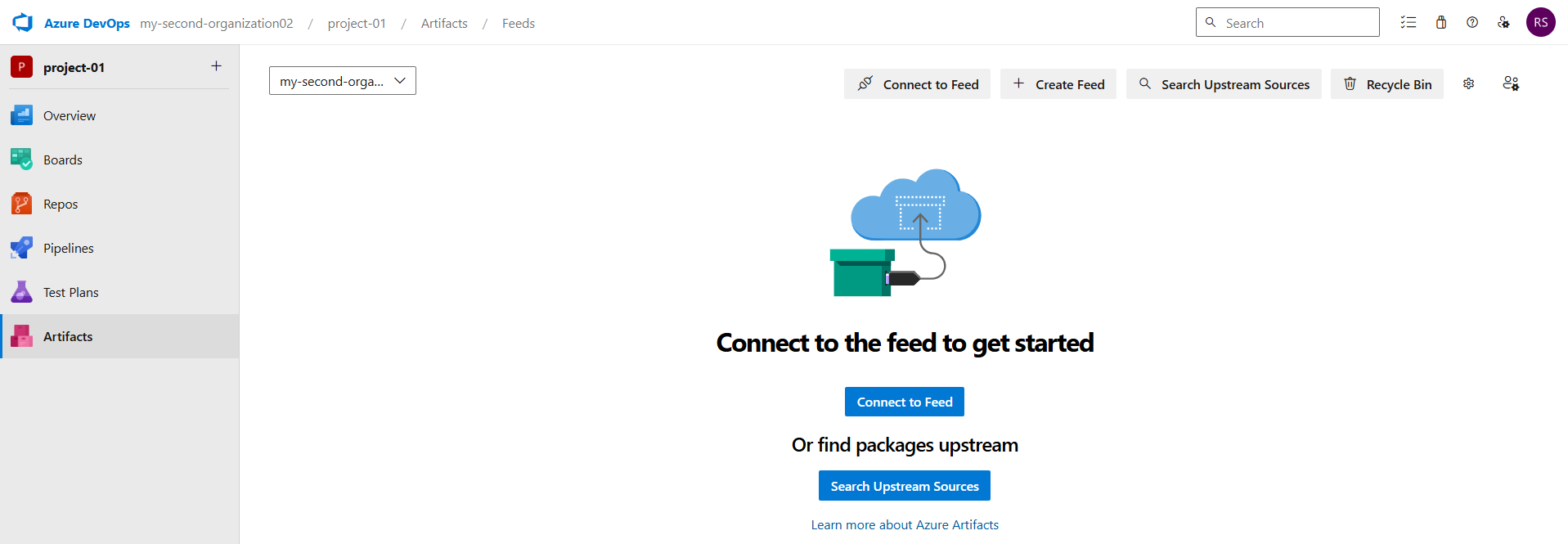


Fig: Create Feed.

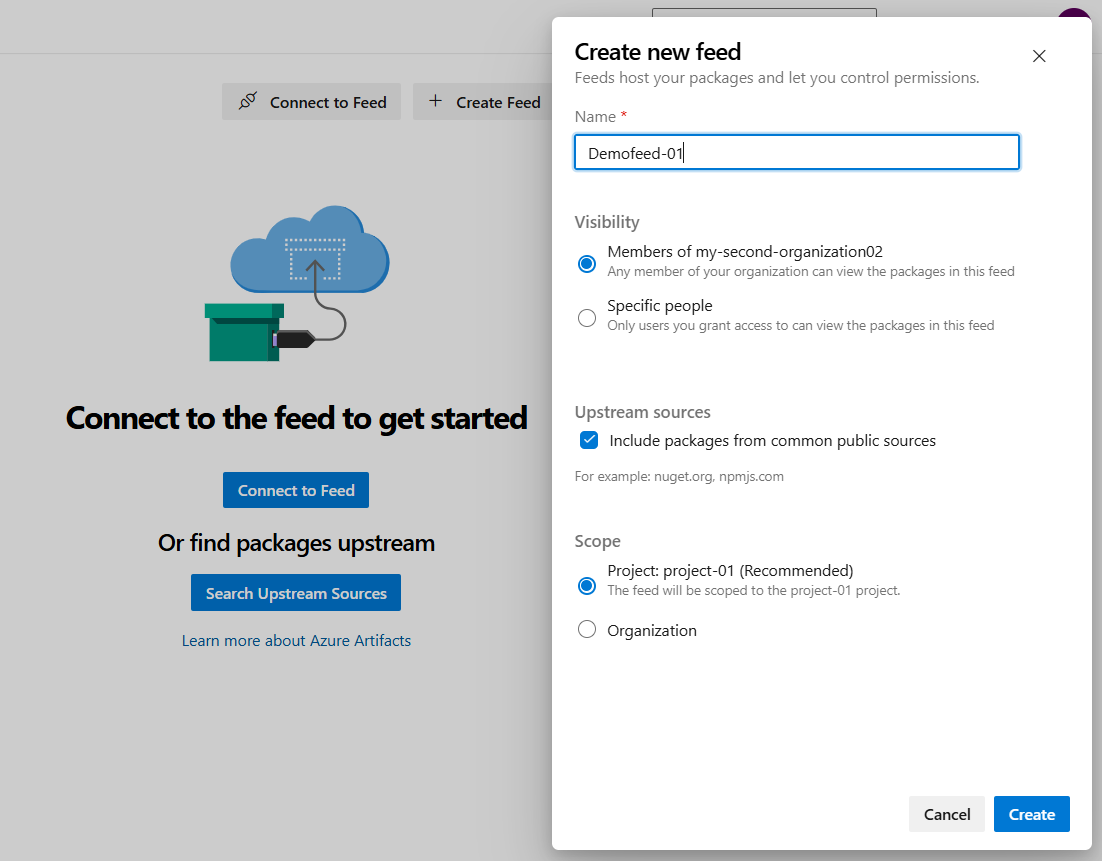


Fig: Configuring Feed creation in Artifacts.

**Note:** In general Artifacts Feed access is given at Organization level. So it will become a centralized package manager for a single or particular organization.

**Upstream Sources**: If the required package is not present in the local Azure Artifacts then it will enables us to connect to the public central package manager (maven.org & NuGet.org) to download required packages for our applications.

**Step3:** Create a project and convert it into package using Visual studio community and push it to Azure Artifacts.

**Case1:** Connect the visual studio community to the Azure DevOps of the Artifacts.

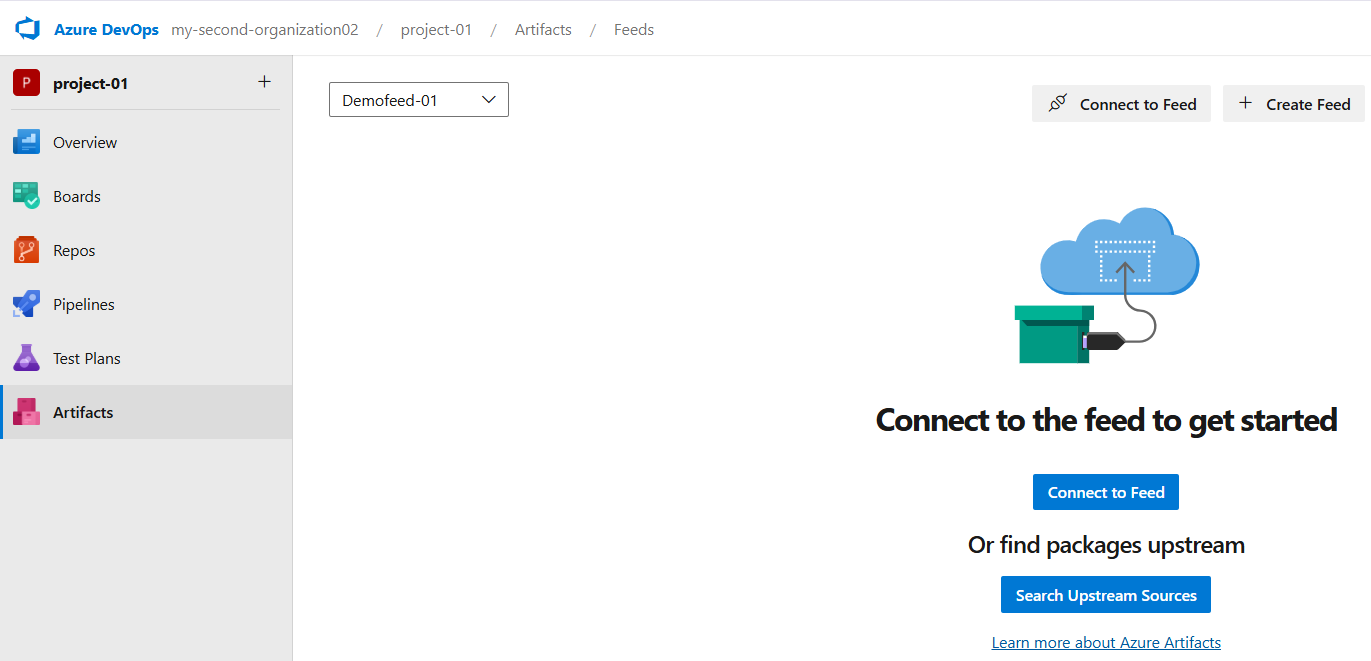


Fig: Connect the Feed to the Visual studio.

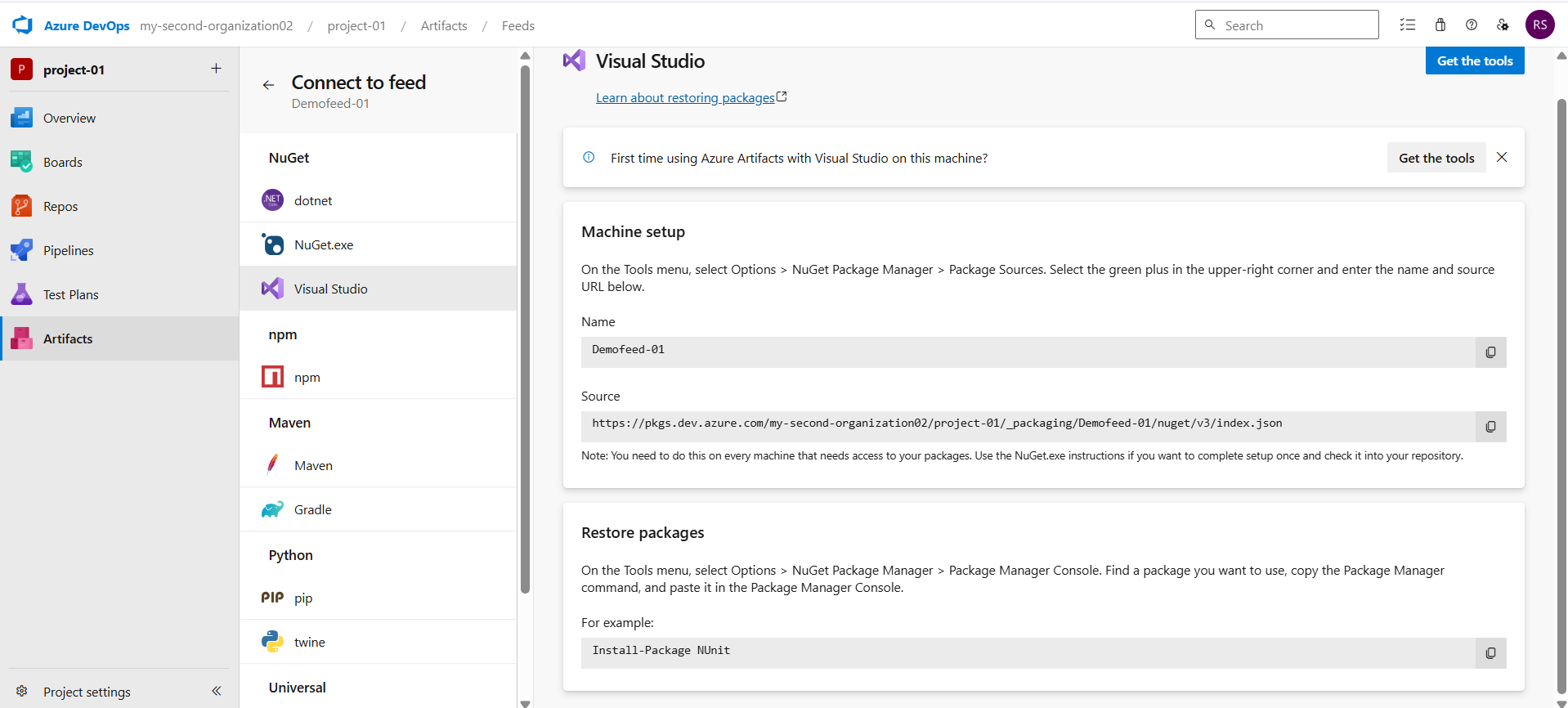


Fig: Select Visual studio.

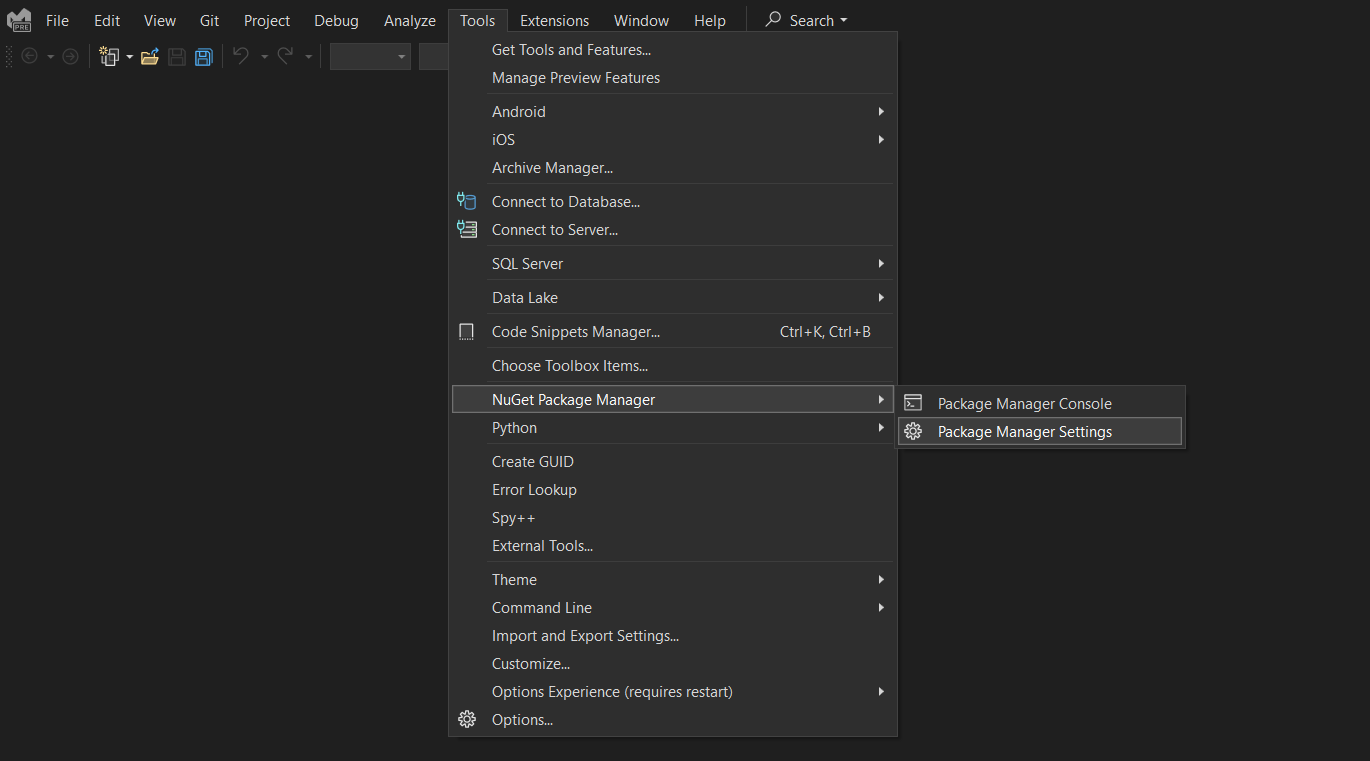


Fig: In Visual Studio select NuGet package manager.

In order to build the .NET package we use .NET package manager called NuGet.

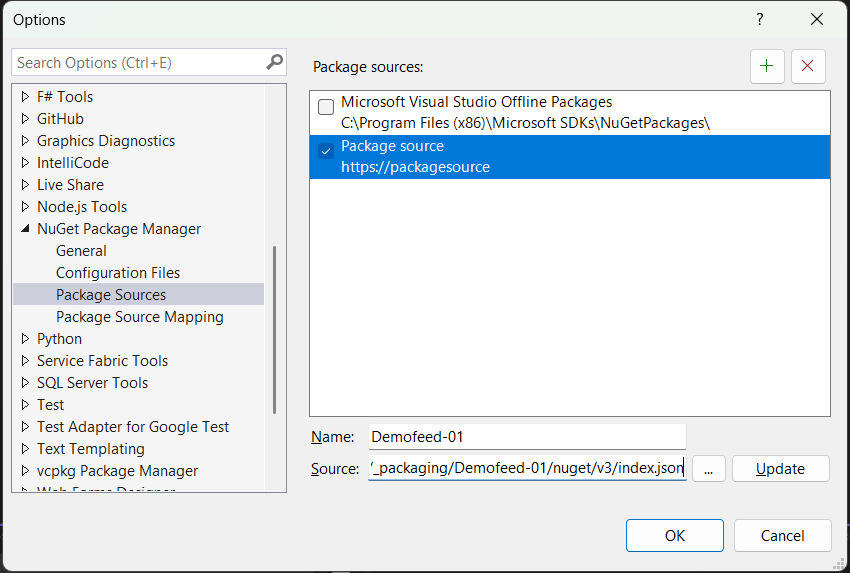


Fig: Configuring the source name and path in Visual studio to connect with Azure artifacts.

**Case2:** Create a new project in visual studio.

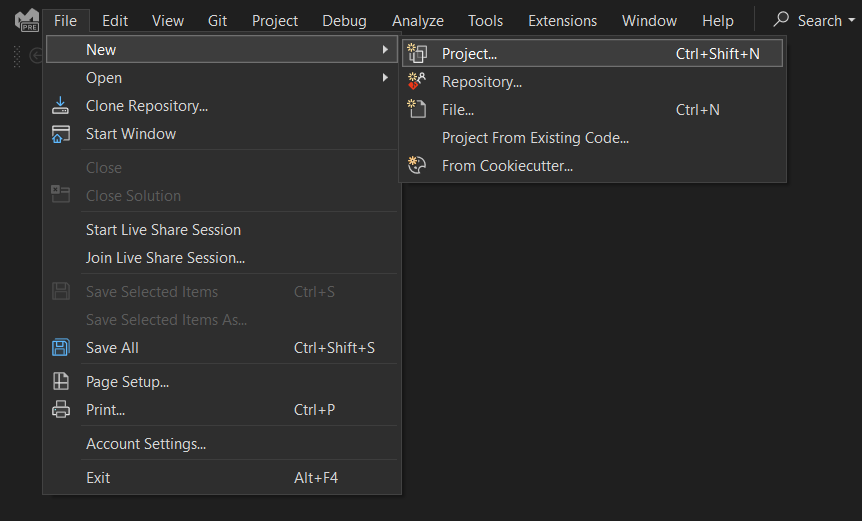
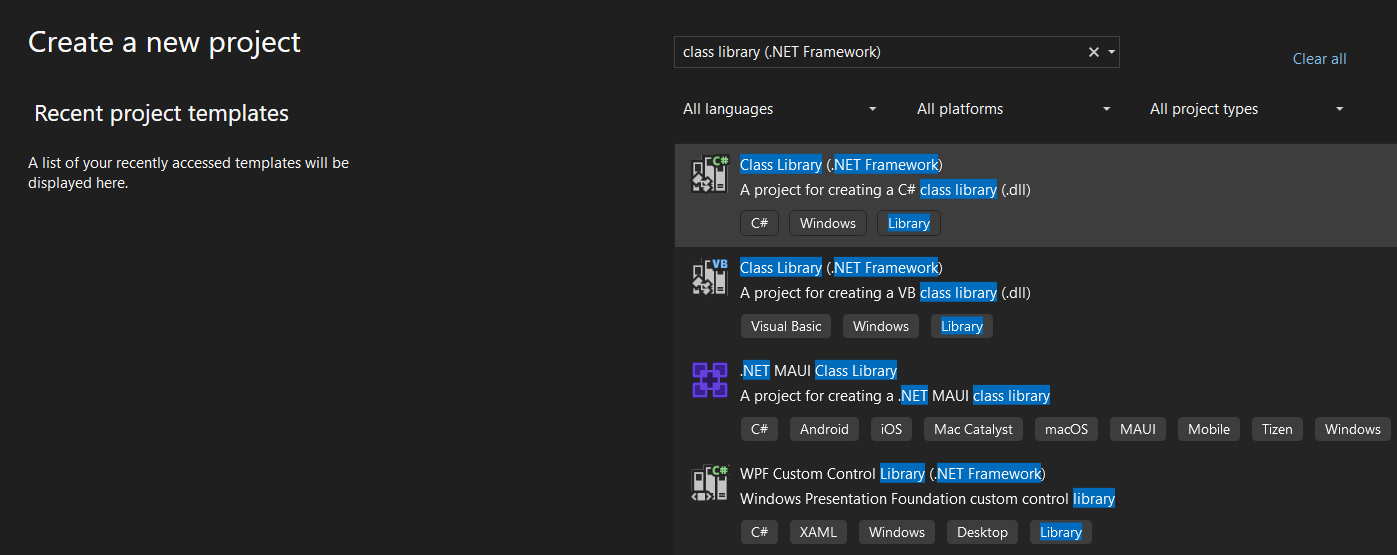


Fig: Creating of new project.



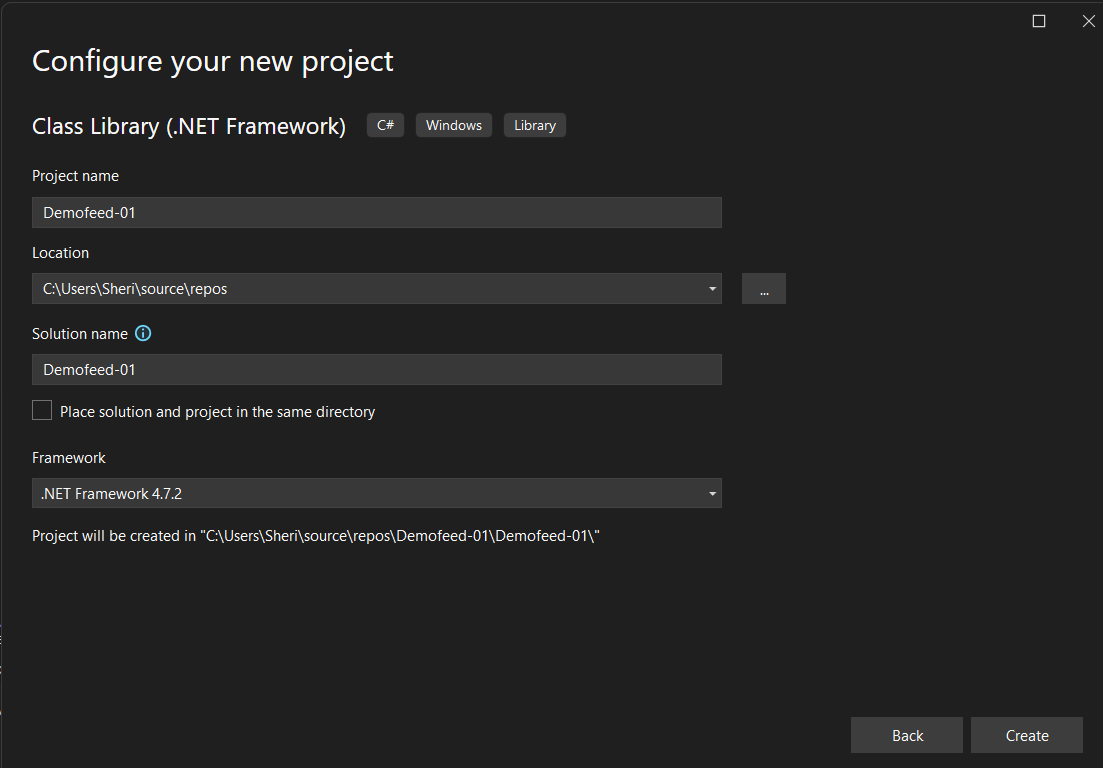


Fig: .NET project creation with name Demofeed-01.

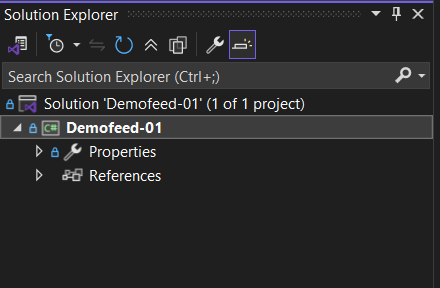
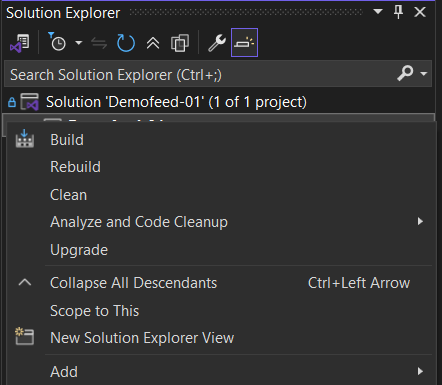


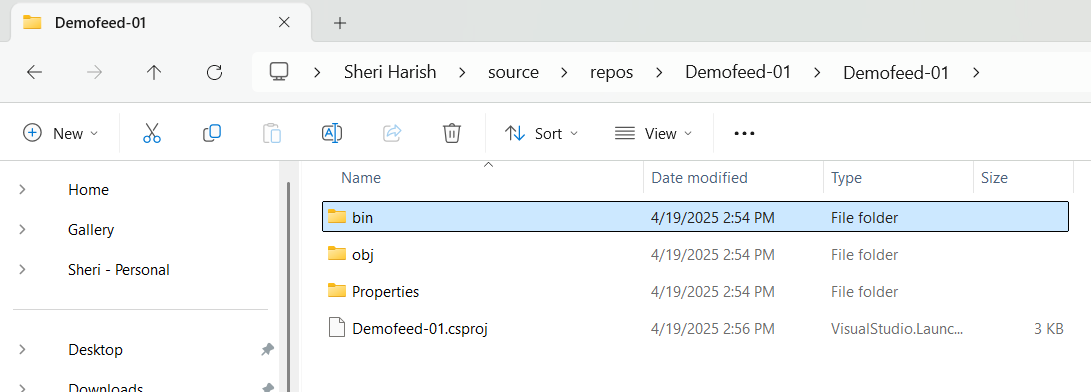
Fig: Project (Demofeed-01) is created successfully.

Now by **right clicking** on the project name and **press build option** then the project will be created.



The project will be created in our local machine at the below path which is displayed in the visual studio terminal after pressing the build button.

Demofeed-01 -> C:\Users\Sheri\source\repos\Demofeed-01\Demofeed-01\bin\Debug\Demofeed-01.dll



**Note:** .NET configuration and dependences are stored in .csproj, fsproj and vbproj.

Up to now the .NET project is built successfully.

**Case3:** Convert the above project (Demofeed-01) into the package by installing NuGet tool.

In order to cover the .NET project into package we have to download the NuGet tool, NuGet tool is a package manager for .NET projects.

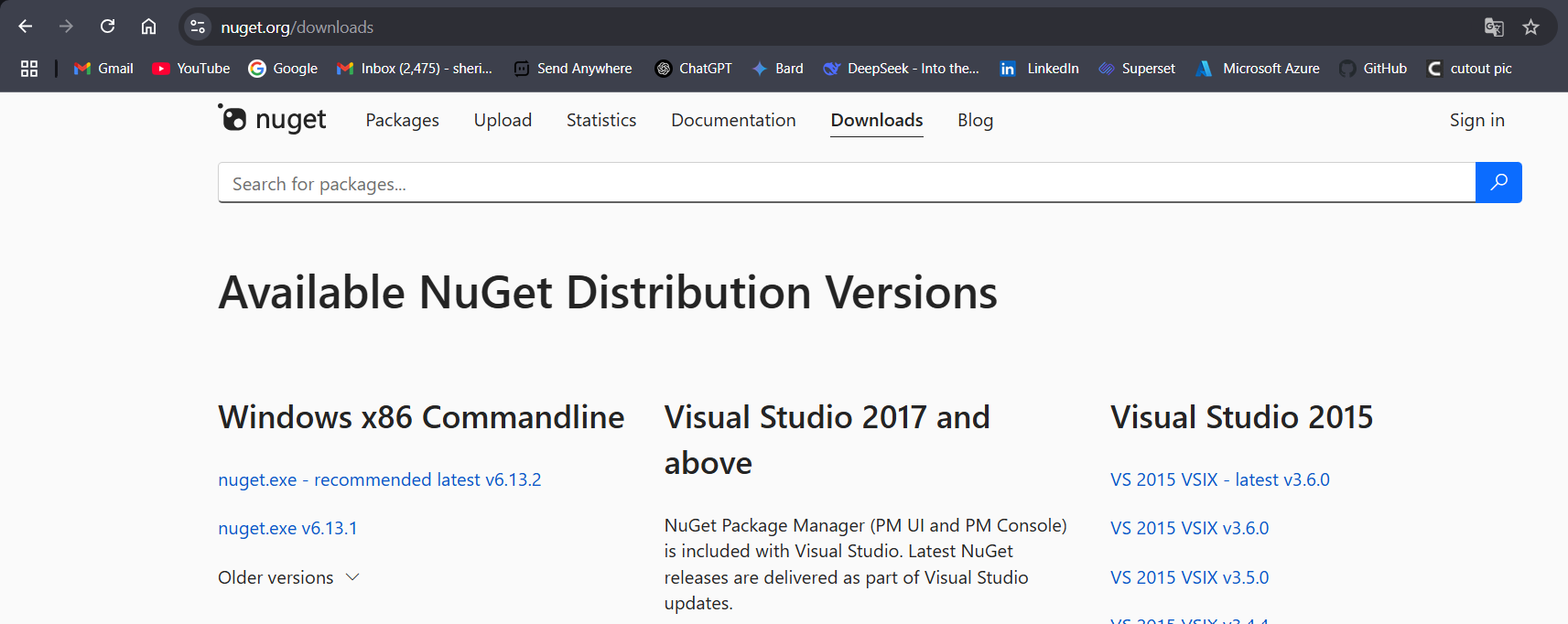


Fig: Downloading of NuGet tool.

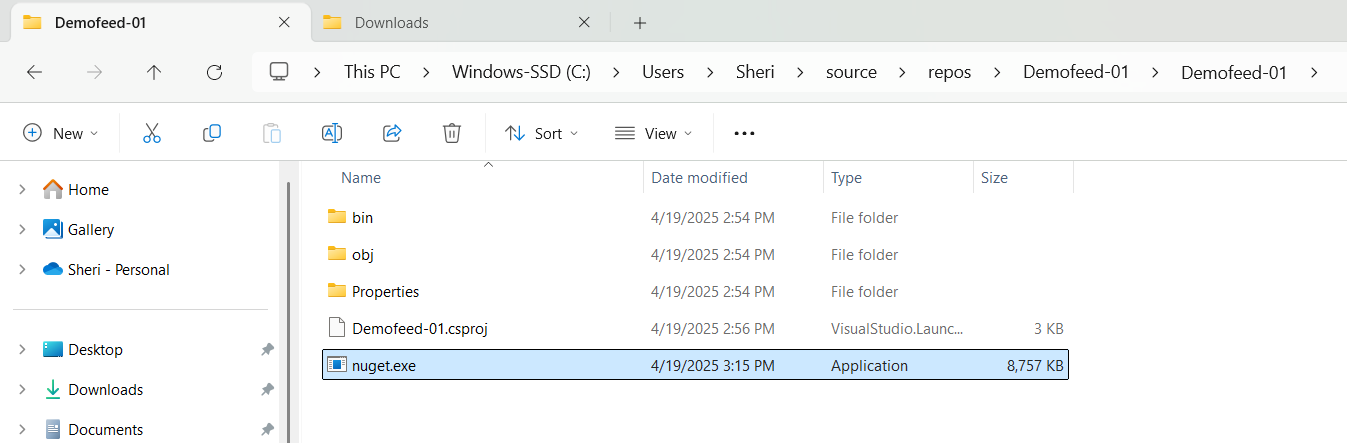


Fig: NuGet tool is placed in the project folder.

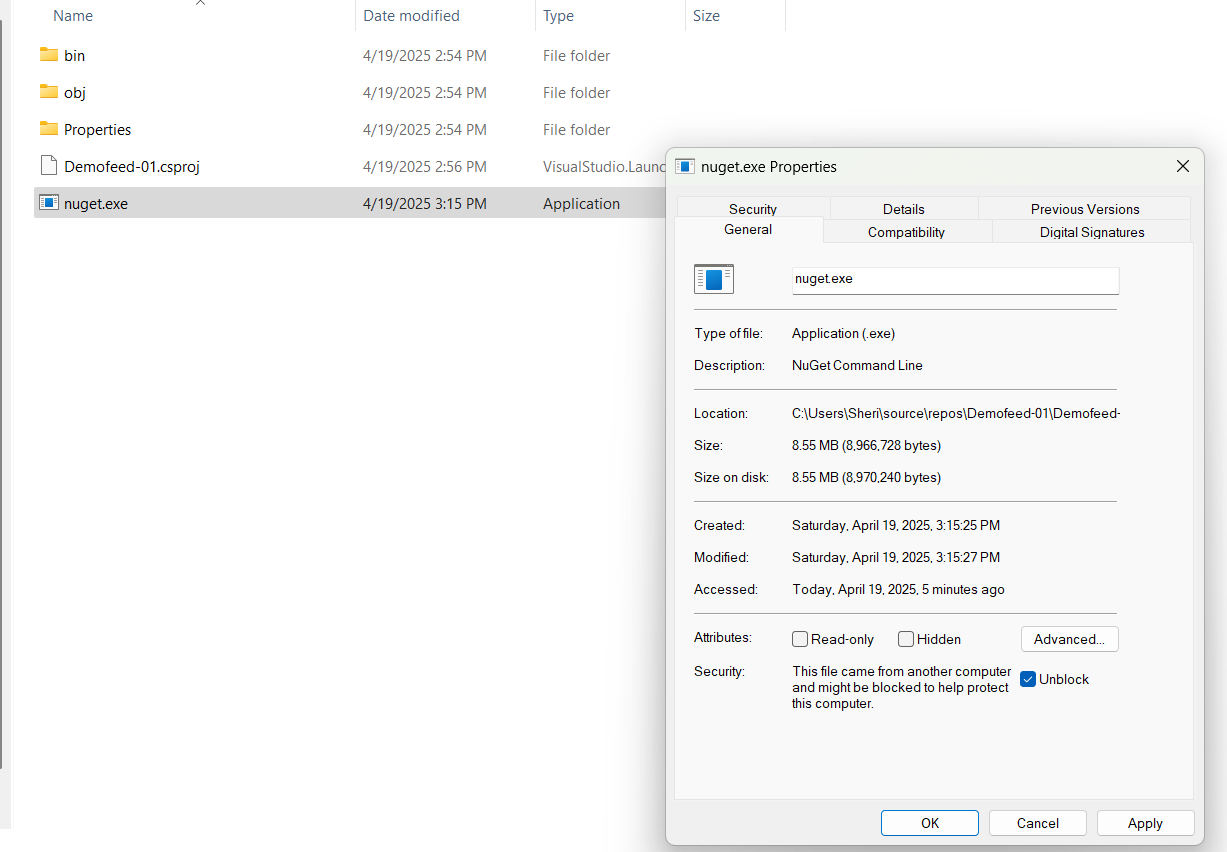


Fig: Enable unblock of NuGet tool.

Right click on NuGet tool🡪 show more options🡪properties🡪enable unblock.

By using the below command we can convert the project into Package:

**Command:** .\nuget.exe pack .\Demofeed.csproj.

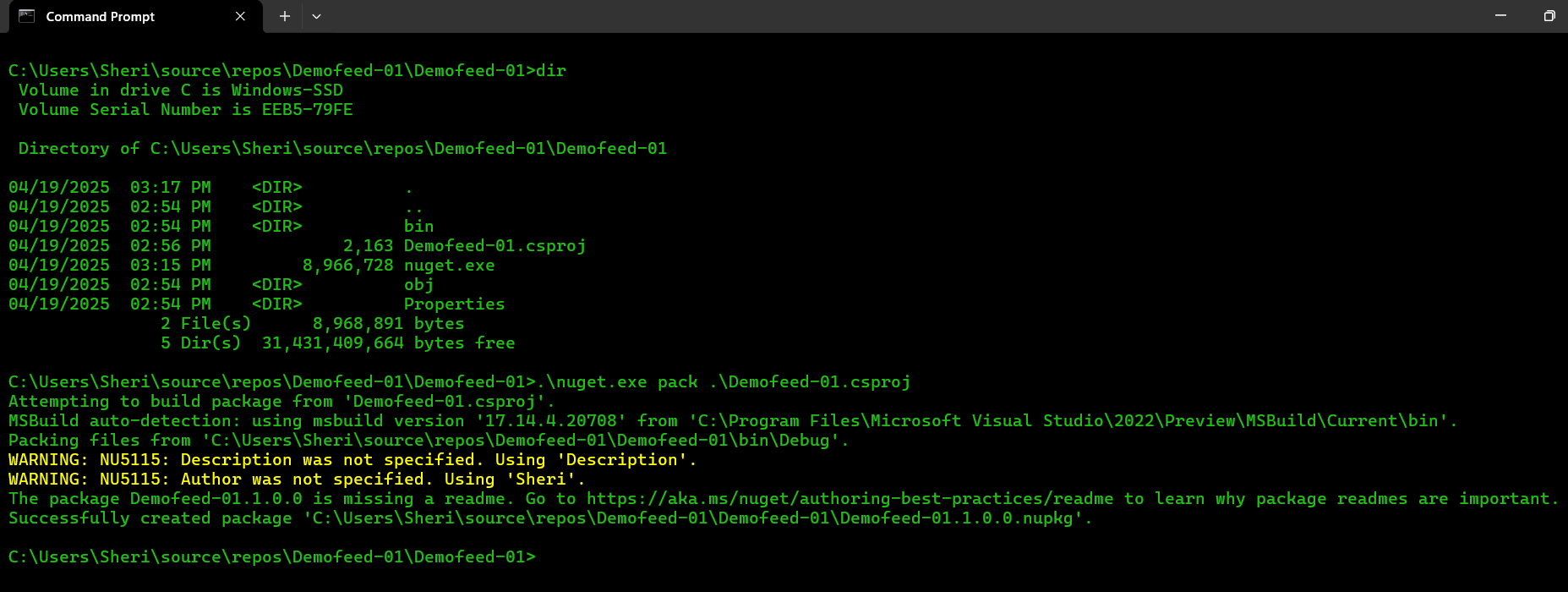
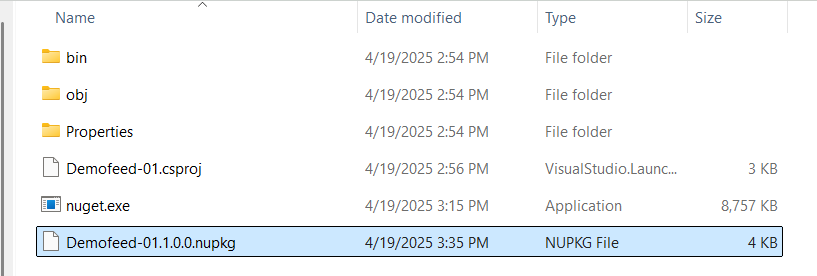


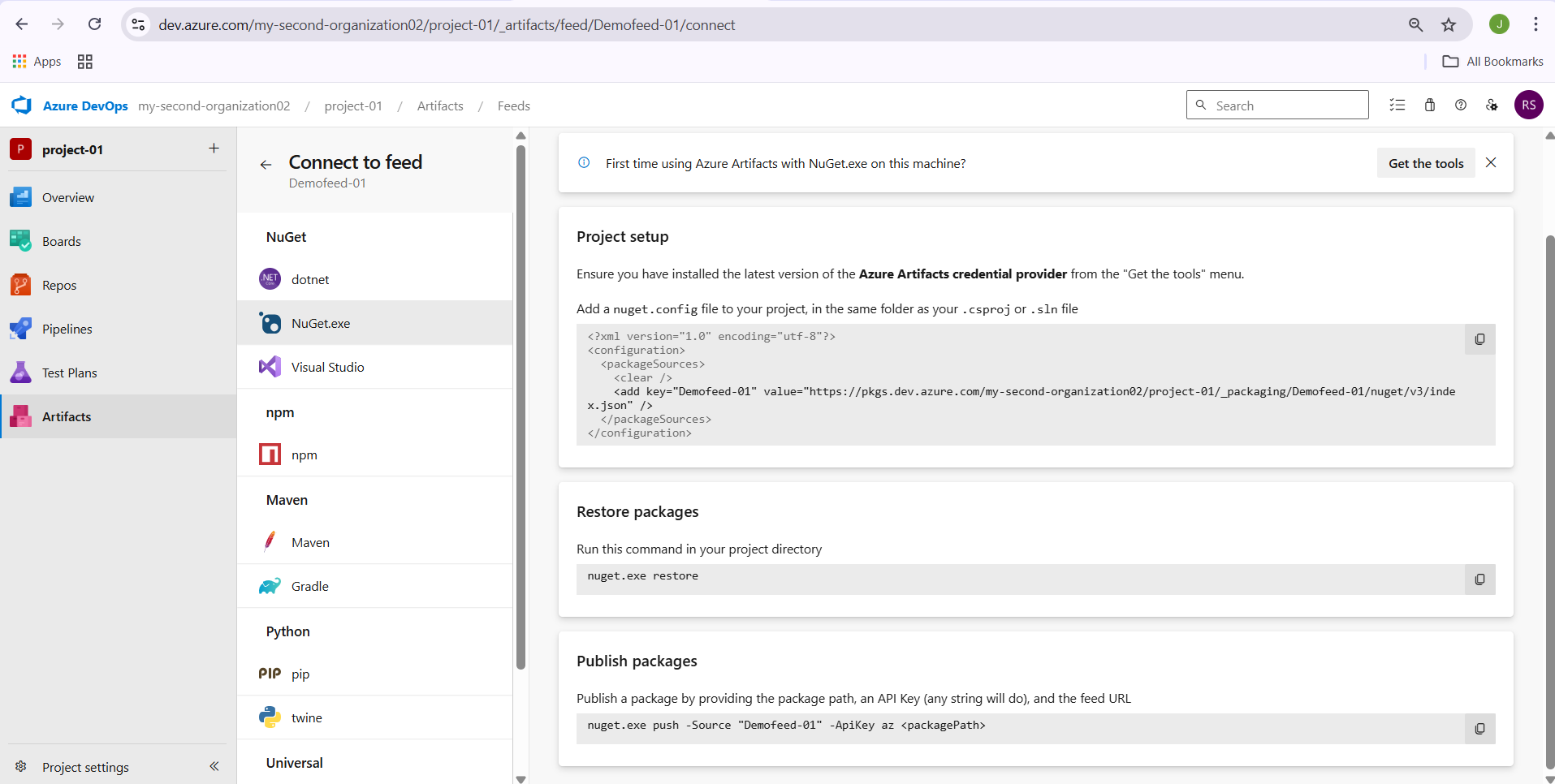
Fig: Package is built successfully.

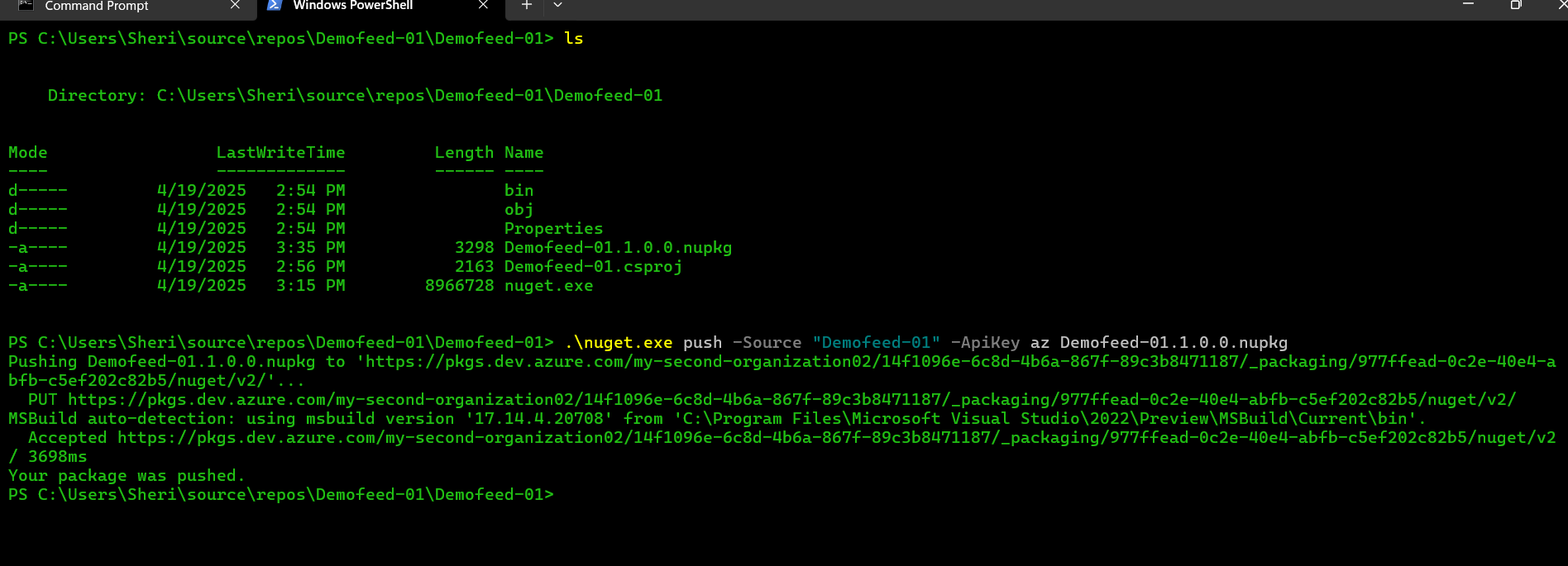
**Case4**: Now push the package into the Azure Artifacts.

The command to push the package into Azure Artifact is given below:

**Command:** .\nuget.exe push -Source "Demofeed-01" -ApiKey az Demofeed-01.1.0.0.nupkg

Fig: To push the package to the Azure artifacts.

Fig: To push the package to the Azure artifacts.



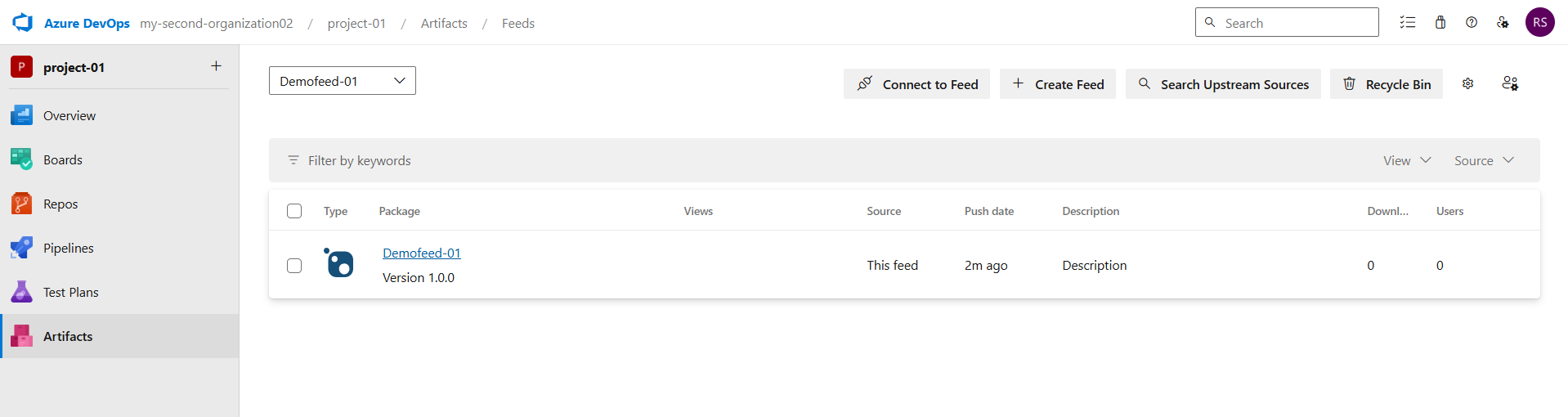
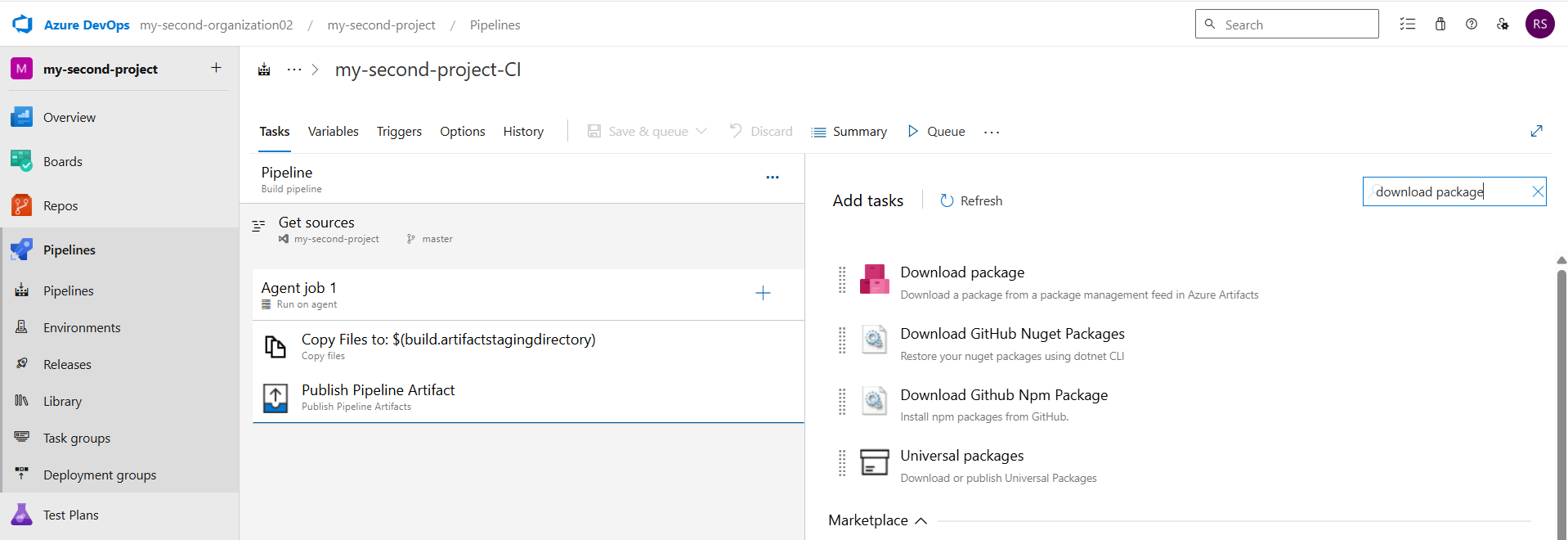


Fig: The package Demofeed-01 is successfully push to the Azure Artifacts.

This package can be used by many project team within the organization if they required.

They add this project at the pipeline stage as show below:

Project🡪pipeline🡪Add Task🡪Download Package🡪Add.



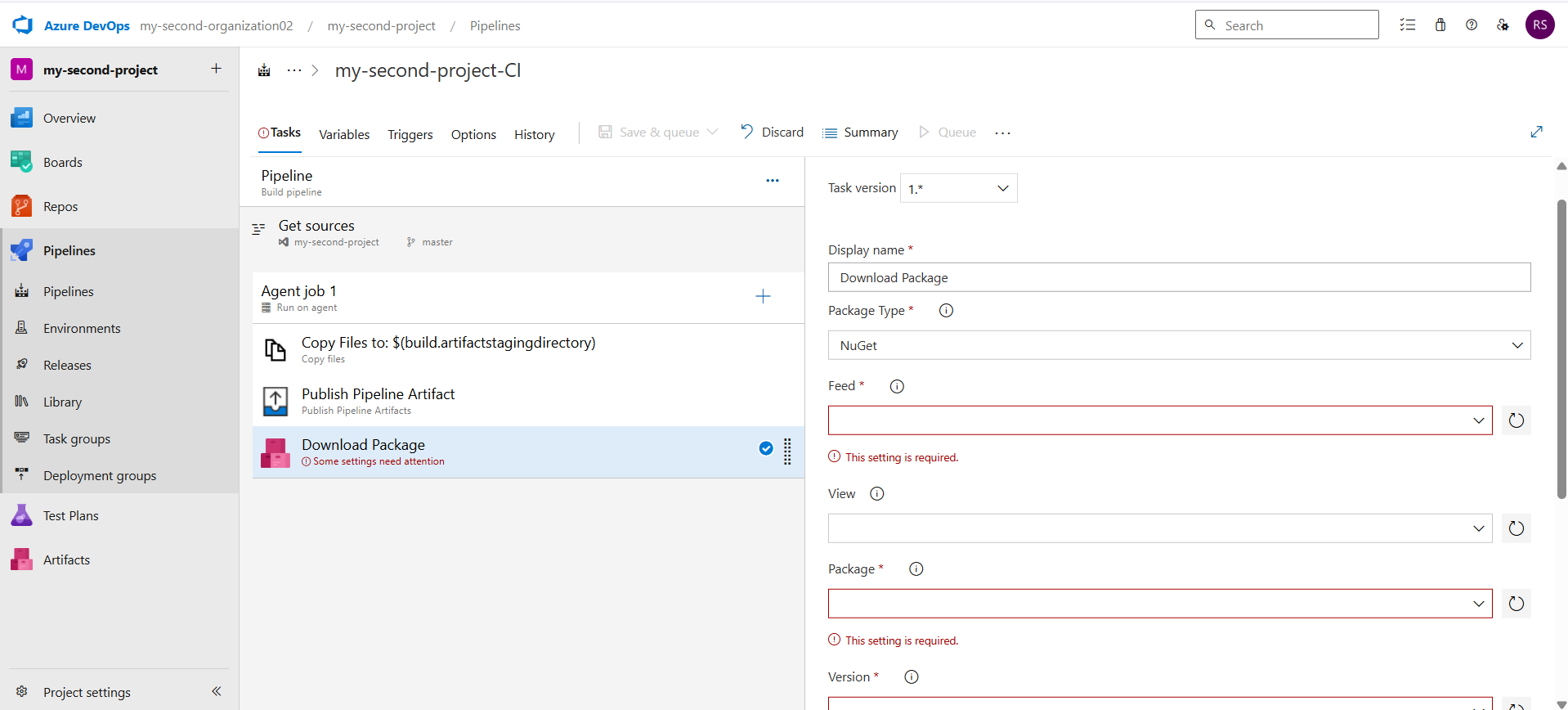


Fig: Configuration to add a specific package from the Azure artifacts to use.