Module 6 - Artifacts & Package Feeds

1. Azure Artifacts Overview

Azure Artifacts is a **package management service** in Azure DevOps that helps teams:

- Host and share **NuGet**, **npm**, **Maven**, **Python**, and **Universal Packages**
- Control versions and dependencies centrally
- Apply retention, permissions, and governance
- Integrate package restore and publishing into CI/CD pipelines

1.1 Why use Azure Artifacts?

- **Centralized package repository** for your organization
- **Private, secure feeds** instead of relying on public registries
- **Upstream sources** to cache external packages (NuGet.org, npmjs.com, Maven Central)
- **Improved build reliability** (no random public registry outages)
- **Compliance & control** (you control which versions are used internally)

1.2 Key Concepts

- **Feed** → Logical container where packages are stored
- **Package** → NuGet/NPM/Maven/Python/Universal artifact
- **Scope** → Feed can be scoped to:
  - A single **project**
  - The entire **organization**
- **Upstream sources** → External registries or other internal feeds from which packages can be proxied and cached

---

2. Managing NuGet / NPM Feeds

You can create **separate feeds** for different teams, projects, or environments.

2.1 Creating a Feed

1. In Azure DevOps, go to **Artifacts**
2. Click **New feed**
3. Provide:
   - **Name** (e.g., MyCompany-NuGet, Web-Frontend-npm)

   - **Scope**:
     - Project (limited to current project)
     - Organization (visible to selected projects)
   - **Visibility** and permissions:
     - Who can **Read**
     - Who can **Contribute**
4. Configure **Upstream sources** (optional but recommended):
   - For NuGet: nuget.org
   - For npm: npmjs.com

2.2 NuGet Feeds (for .NET)

2.2.1 Connect from Visual Studio / dotnet CLI

Azure DevOps gives you the nuget.config or dotnet command under **Artifacts → Connect to Feed**.

Typical nuget.config entry:

```
<configuration>
  <packageSources>
    <add key="MyCompany-NuGet"
value="https://pkgs.dev.azure.com/<org>/<project>/_packaging/MyCompany-NuGet/nuget/v3/index.json" />
    <add key="nuget.org" value="https://api.nuget.org/v3/index.json" />
  </packageSources>
</configuration>
```

2.2.2 Publishing NuGet Package from CLI

```
dotnet pack src/MyLib/MyLib.csproj -c Release -o ./artifacts


dotnet nuget push ./artifacts/MyLib.1.0.0.nupkg \
  --api-key <AZURE_ARTIFACTS_PAT> \
  --source "MyCompany-NuGet"
```

In Azure Pipelines, you typically use NuGetCommand@2 or DotNetCoreCLI@2 tasks instead of raw CLI.

2.3 npm Feeds (for Node.js)

2.3.1 Configuring .npmrc

Azure Artifacts uses a special scoped registry like @mycompany:registry.

Example .npmrc:

```
registry=https://registry.npmjs.org/


@mycompany:registry=https://pkgs.dev.azure.com/<org>/<project>/_packaging/Web-Frontend-npm/npm/registry/

//pkgs.dev.azure.com/<org>/<project>/_packaging/Web-Frontend-npm/npm/registry/:username=AzureDevOps

//pkgs.dev.azure.com/<org>/<project>/_packaging/Web-Frontend-npm/npm/registry/:_password=<BASE64_PAT>

//pkgs.dev.azure.com/<org>/<project>/_packaging/Web-Frontend-npm/npm/registry/:email=build@local

always-auth=true
```

Packages published will have a **scope**:

```
{
  "name": "@mycompany/web-ui",
  "version": "1.0.0",
  "main": "index.js"
}
```

2.3.2 Publishing npm Package

```
npm login --registry=https://pkgs.dev.azure.com/<org>/<project>/_packaging/Web-Frontend-npm/npm/registry/

npm publish
```

In pipelines, authentication is often done via npm task with service connection or feed auth.

---

3. Versioning and Artifact Retention Policies

3.1 Versioning Strategy

Package versioning should follow a consistent rule, often **Semantic Versioning (SemVer)**:

MAJOR.MINOR.PATCH
Example: 2.3.1

- **MAJOR** – breaking changes
- **MINOR** – new features, backward compatible

- **PATCH** – bug fixes only

For pre-release versions:

- 1.0.0-alpha
- 1.0.0-beta
- 1.0.0-rc.1

3.2 Azure Artifacts Versioning Behavior

- Each published package version is stored separately
- You can **promote** or **deprecate** package versions (depending on the client and policy)
- Builds/pipelines can pin to **specific version** or use ranges like 1.* or ^1.2.0 (NuGet/npm semantics)

3.3 Retention Policies

Retention policies help control storage growth.

What you can configure:

- **Number of versions to keep per package**
- **Age-based cleanup** (delete versions older than X days)
- Optionally preserve:
  - Latest version
  - Versions used by specific builds/releases

Steps to configure retention:

1. Go to **Artifacts → Your Feed → Settings**
2. Select **Retention policies**
3. Define:
   - Minimum number of versions to keep
   - Days to retain unreferenced versions
4. Save changes

This ensures:

- Old, unused package versions are automatically removed
- Storage costs stay under control
- Feeds remain clean and manageable

---

4. Lab: Publish Package to Feed and Consume It in a Pipeline

Scenario

- You have a simple **.NET class library** project MyCompany.Logging
- You will:
    1. Pack and publish it as a **NuGet package** to Azure Artifacts
    2. Consume it from another application in a **YAML pipeline**

4.1 Pre-requisites

- Azure DevOps organization + project
- Azure Artifacts enabled
- A feed called MyCompany-NuGet
- A Personal Access Token (PAT) with **Packaging (Read & Write)**

4.2 Step 1 – Create and Pack a .NET Library

Example project:

```
src/
  MyCompany.Logging/
    MyCompany.Logging.csproj
```

Sample MyCompany.Logging.csproj (key parts only):

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <GeneratePackageOnBuild>true</GeneratePackageOnBuild>
    <Version>1.0.0</Version>
    <Authors>MyCompany</Authors>
    <PackageId>MyCompany.Logging</PackageId>
  </PropertyGroup>
</Project>
```

Build and pack locally (optional check):

```
dotnet pack src/MyCompany.Logging/MyCompany.Logging.csproj -c Release -o ./artifacts
```

4.3 Step 2 – Create a YAML Pipeline to Publish Package

Create azure-pipelines-publish-logging.yml at repo root:

```
trigger:
  branches:
    include:
      - main

pool:
  vmImage: 'windows-latest'

variables:
  buildConfiguration: 'Release'
  feedName: 'MyCompany-NuGet'

stages:
- stage: PackAndPublish
  displayName: 'Pack and Publish NuGet Package'
  jobs:
  - job: BuildAndPublish
    displayName: 'Build, Pack, and Publish'
    steps:
    - task: UseDotNet@2
      displayName: 'Install .NET SDK'
      inputs:
        version: '8.0.x'
        packageType: 'sdk'

    - task: DotNetCoreCLI@2
      displayName: 'Restore'
      inputs:
        command: 'restore'
        projects: 'src/MyCompany.Logging/MyCompany.Logging.csproj'

    - task: DotNetCoreCLI@2
      displayName: 'Build'
      inputs:
        command: 'build'
        projects: 'src/MyCompany.Logging/MyCompany.Logging.csproj'
        arguments: '--configuration $(buildConfiguration)'

    - task: DotNetCoreCLI@2
      displayName: 'Pack'
      inputs:
        command: 'pack'
        packagesToPack: 'src/MyCompany.Logging/MyCompany.Logging.csproj'
        configuration: '$(buildConfiguration)'
        outputDir: '$(Build.ArtifactStagingDirectory)/nuget'

    - task: NuGetCommand@2
      displayName: 'Publish to Azure Artifacts'
      inputs:
        command: 'push'
        publishVstsFeed: '$(feedName)'
        allowPackageConflicts: true
        packagesToPush: '$(Build.ArtifactStagingDirectory)/nuget/*.nupkg'
```

Steps:

1. In Azure DevOps → Pipelines → New Pipeline
2. Choose the repo
3. Select **Existing YAML file** → azure-pipelines-publish-logging.yml
4. Run pipeline
5. After success, go to **Artifacts → MyCompany-NuGet** – you should see MyCompany.Logging version 1.0.0.

4.4 Step 3 – Consume the Package in Another Project

Assume another app project:

```
src/
  WebApp/
    WebApp.csproj
```

In WebApp.csproj, add package reference:

```
<ItemGroup>
  <PackageReference Include="MyCompany.Logging" Version="1.0.0" />
</ItemGroup>
```

Configure feed connection via Azure DevOps “Connect to feed” → generate nuget.config & commit it to your repo, e.g.:

```
<configuration>
  <packageSources>
    <add key="MyCompany-NuGet"
value="https://pkgs.dev.azure.com/<org>/<project>/_packaging/MyCompany-NuGet/nuget/v3/index.json" />
    <add key="nuget.org" value="https://api.nuget.org/v3/index.json" />
  </packageSources>
</configuration>
```

Ensure this nuget.config is in the repo root or referenced in pipeline tasks.

4.5 Step 4 – YAML Pipeline to Restore from Feed and Build

Create azure-pipelines-webapp.yml:

```
trigger:
  branches:
    include:
      - main


pool:
  vmImage: 'windows-latest'
```

```
variables:
  buildConfiguration: 'Release'

stages:
- stage: RestoreAndBuild
  displayName: 'Restore from Feed and Build WebApp'
  jobs:
  - job: BuildJob
    displayName: 'Restore & Build'
    steps:
    - task: UseDotNet@2
      displayName: 'Install .NET SDK'
      inputs:
        version: '8.0.x'
        packageType: 'sdk'

    - task: NuGetToolInstaller@1
      displayName: 'Install NuGet'

    - task: NuGetCommand@2
      displayName: 'NuGet restore using Azure Artifacts feed'
      inputs:
        command: 'restore'
        restoreSolution: 'src/WebApp/WebApp.csproj'
        feedsToUse: 'select'
        vstsFeed: 'MyCompany-NuGet'

    - task: DotNetCoreCLI@2
      displayName: 'Build WebApp'
      inputs:
        command: 'build'
        projects: 'src/WebApp/WebApp.csproj'
        arguments: '--configuration $(buildConfiguration)'
```

Run this pipeline – it should:

- Restore packages from **MyCompany-NuGet** (including MyCompany.Logging)
- Build WebApp successfully