

edureka!



Microsoft Azure DevOps Solution Certification (AZ-400)

COURSE OUTLINE



Azure AZ-400

MODULE 1: Introduction to Azure DevOps

MODULE 2: Implementing Continuous Integration

MODULE 3: Build Containers with Azure DevOps

MODULE 4: Designing a Dependency Management Strategy and Managing Artifact Versioning

MODULE 5: Setting up Release Management Workflow

MODULE 6: Implementing Deployment Models and Services

MODULE 7: Implement and Optimize Continuous Feedback Mechanism

MODULE 8: Azure Tools: Infrastructure and Configuration, and Third-Party Tools

MODULE 9: Implementing Compliance and Security

MODULE 10: Azure Case Studies

edureka!

Designing a Dependency Management Strategy and Managing Artifact Versioning

Topics

Following are the topics covered in this module:

- Package dependencies
- Package management
- Migrating and consolidating artifacts
- Package security
- Open-source software
- License and vulnerability scan integration

Objectives

After completing this module, you should be able to:

- Migrate and consolidate artifacts
- Implement a versioning strategy
- Analyze open-source software or package security
- Update packages
- Use WhiteSource to manage open-source security and license



Bing Data Security

Bing Data Management

- It is no surprise what Bing deals with; a lot of public and private data
- Most of the non-critical data are publicly made available by Bing



Threat to the Private Data



- Private data that Bing deals with is vulnerable to many cyber attacks
- And due to its importance, it becomes a real issue if the data is lost or made publically available

To secure and manage this data, the company has hired Harald as an Azure DevOps Engineer

Managing the Data



Harald
Azure DevOps Engineer

To address this issue, I have decided to use Azure DevOps tools and WhiteSource

Okay. What exactly is your plan?

- We must scan the development code and update the packages
- Scan the application for vulnerabilities and loopholes using WhiteSource to manage the open-source security and license



CTO
Bing

The Solution



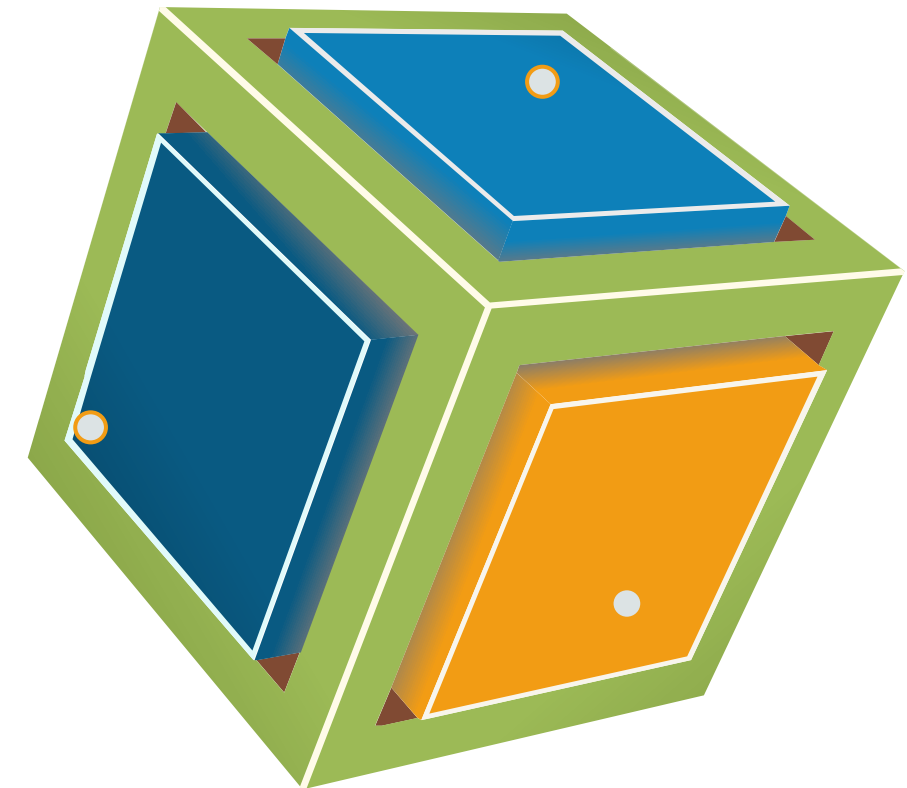
- WhiteSource is an open-source security and license compliance management platform
- WhiteSource Bolt is used which is a free extension for Azure DevOps that can scan all of your projects for any vulnerabilities
- WhiteSource automates the entire process of open-source component selection, approval, and management



Package Dependencies and Package Management

Need of Package Dependencies

- A dependency is a package that is required for another package to execute
- Sometimes developers want to create their package to be used across many applications
- This will reduce the development effort as existing work can be used as part of new development



Examples of Package Dependencies

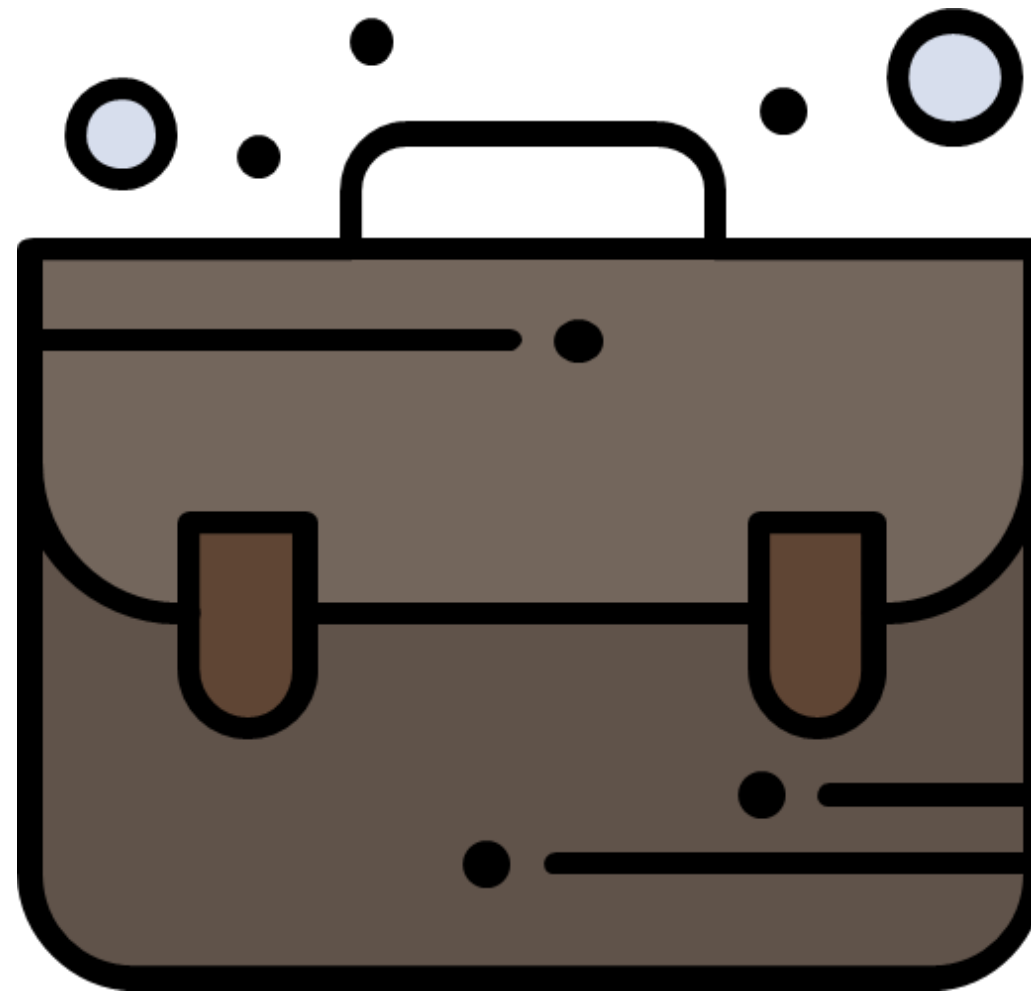
In case of a .NET application creation, the user needs to do JSON related operation, and for such case, JSON.NET can be used, which is an open-source package.

So, the creation of one package depends on another package.
Such a scenario is referred to as package dependencies.



What is Package Management?

A package contains a reusable code that other developers can use for their further application development.

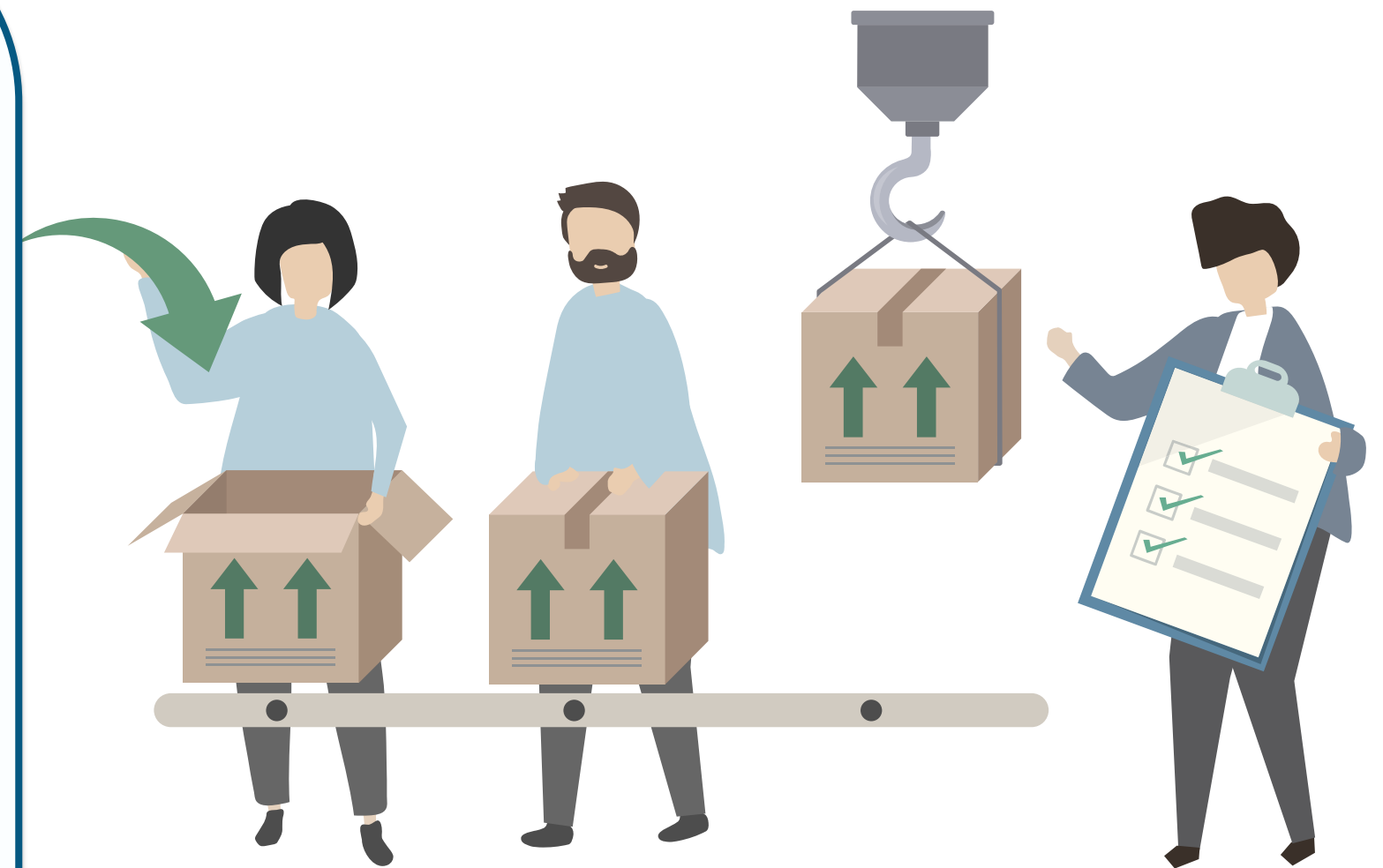


- The package can be compiled binary code
 - Example: .dll for .NET, .class file for Java
- For languages like JavaScript or Python, the package may include source code

- Packages are generally compressed as .zip file or similar format
- Packages usually have .nupkg or .jar extension

Why Package Management?

- The package is created to reduce the code duplication and save the effort
- Generally, in code duplication, the code can be modified to suit the requirements, thus creating drift
- So, after many code copy, significant drift (configuration change or code change) can happen which is undesired



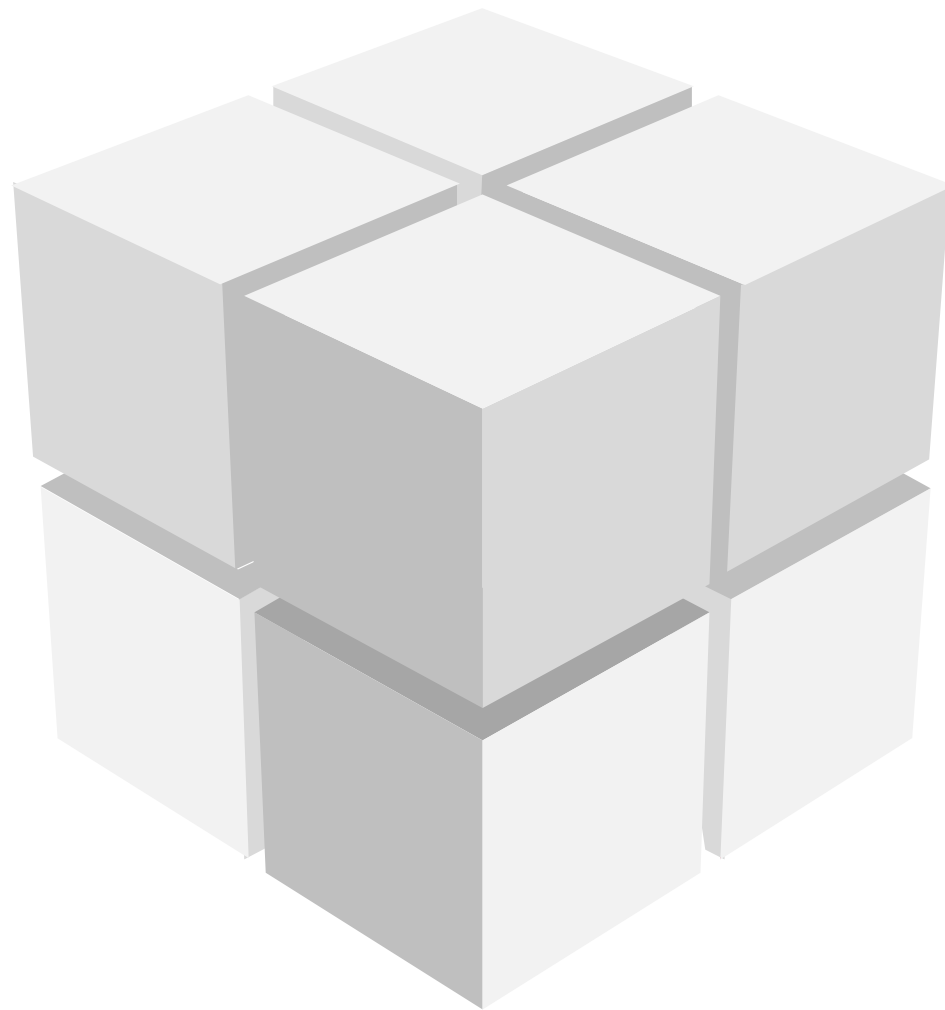
Why Package Management? (Cont.)








To deal with such a scenario, the package is created, which is immutable and prevents code duplication. JSON.NET is a popular package for .NET, which enables the developers to work with JSON files.

Package Management: Hosting Services

The package is hosted in hosting services, and for different languages, the hosting services are different. For instance,



-  NuGet: Packages .NET Libraries
-  Maven: Packages JavaScript Libraries
-  npm: Packages JavaScript Libraries
-  Chocolatey: Packages Windows Applications
-  RubyGems: Packages Ruby Libraries



Migrating and Consolidating Artifacts

What is Artifact?

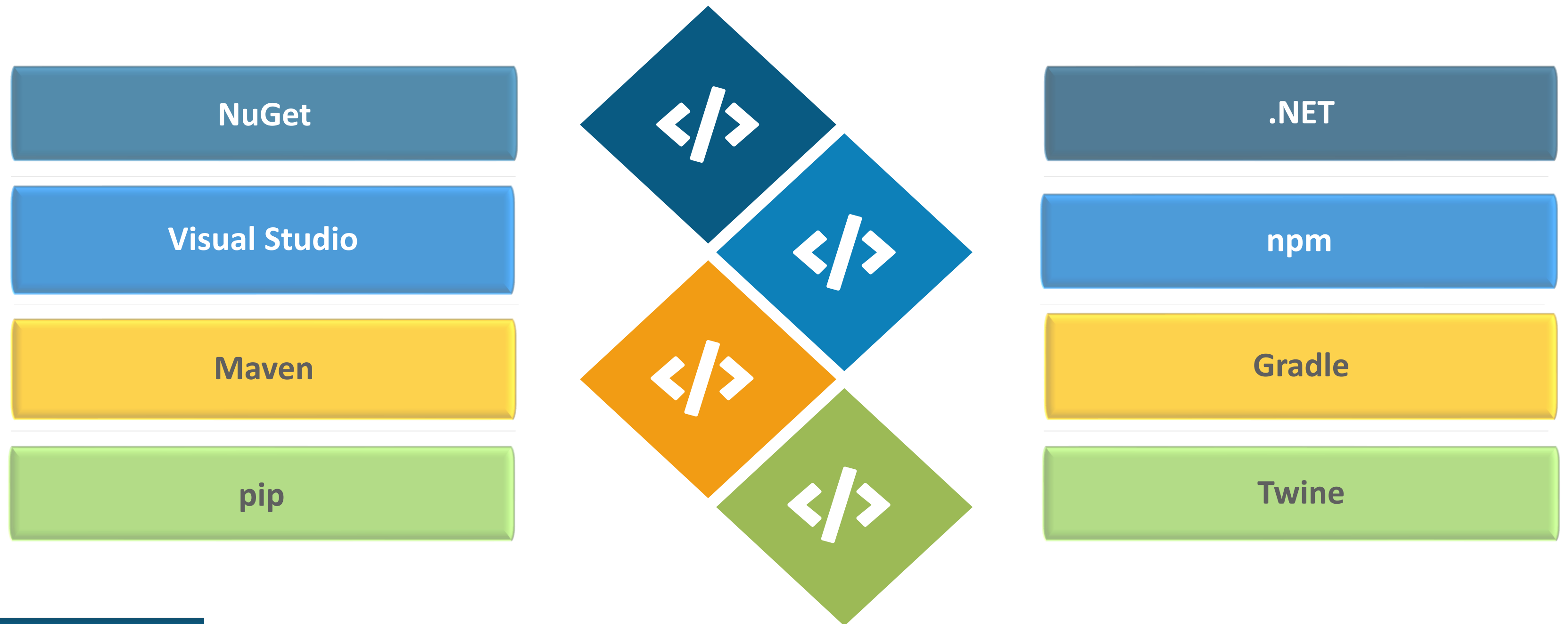
An artifact is a place where any custom package is stored for reuse across many applications from many developers.



The advantage is no code duplication, and lots of efforts are saved because of the use of existing development in the form of package.

Sources for Migrating Artifacts

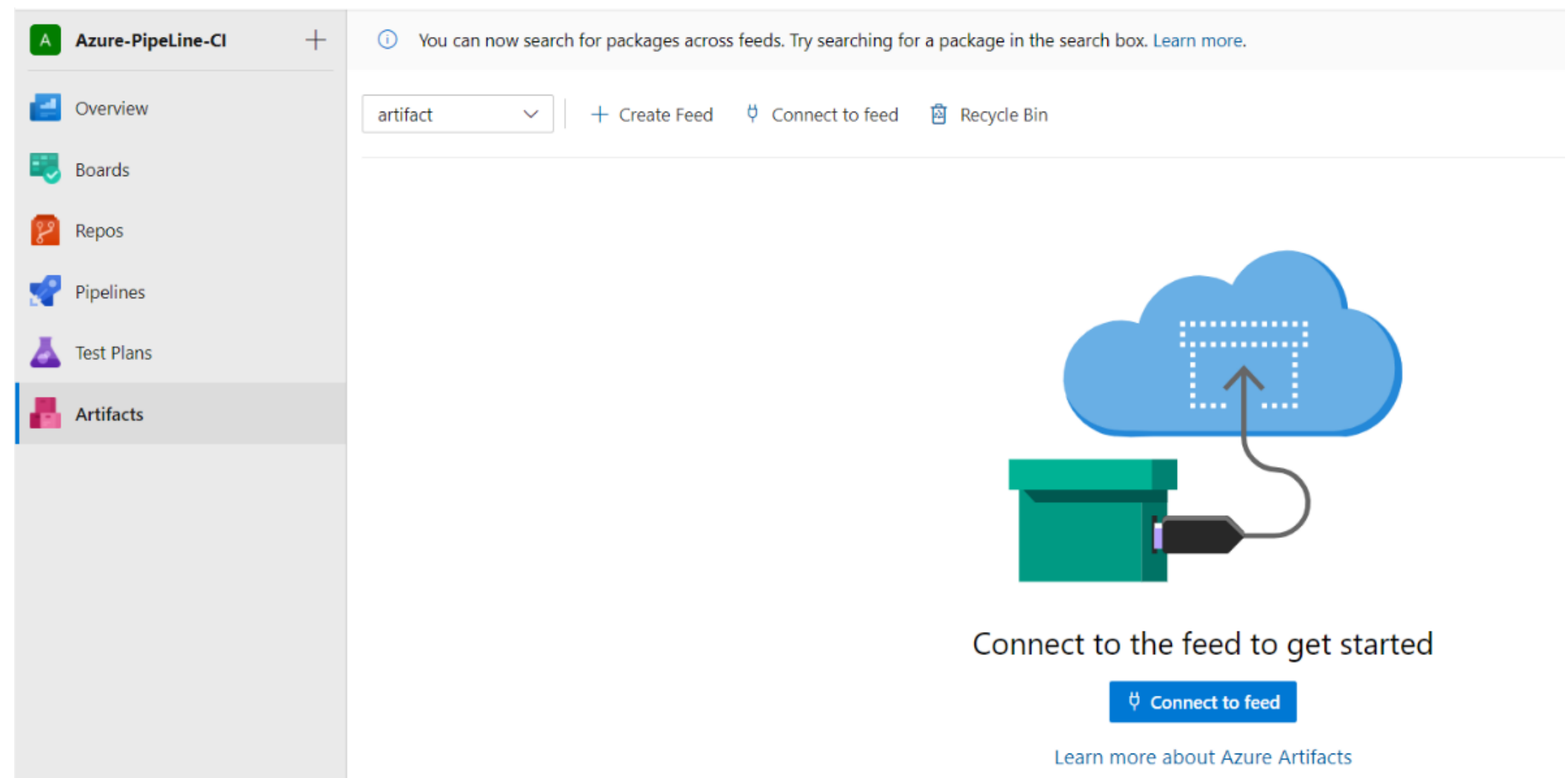
It specifies the feed details to migrate the artifact to Azure DevOps. A feed can be from many sources such as:



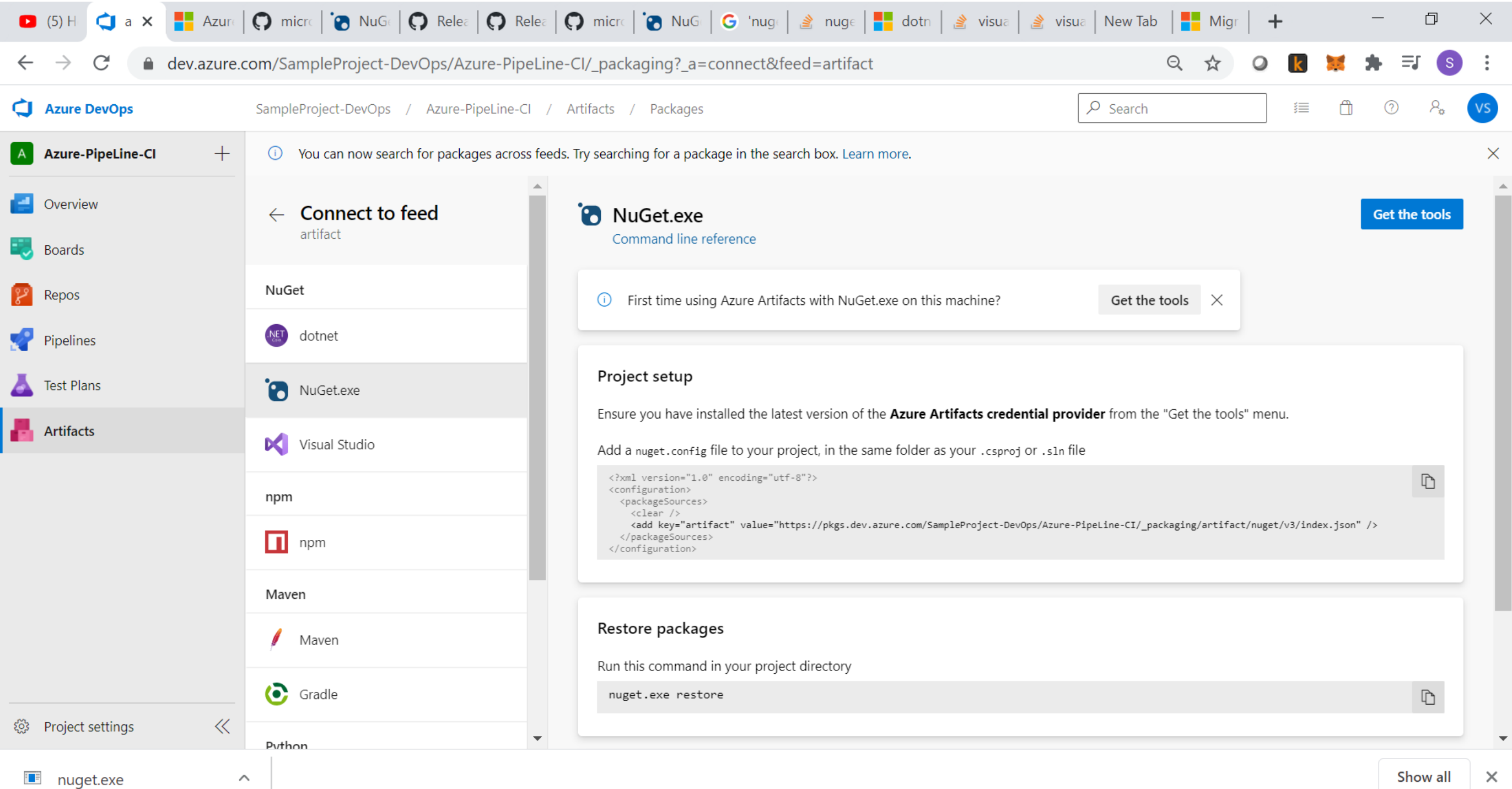
Connecting to Feed

An artifact created locally using visual studio can be migrated to Azure Artifact.

For the given project, go to the Artifact section and connect to feed as shown.



Migrating and Consolidating Artifacts

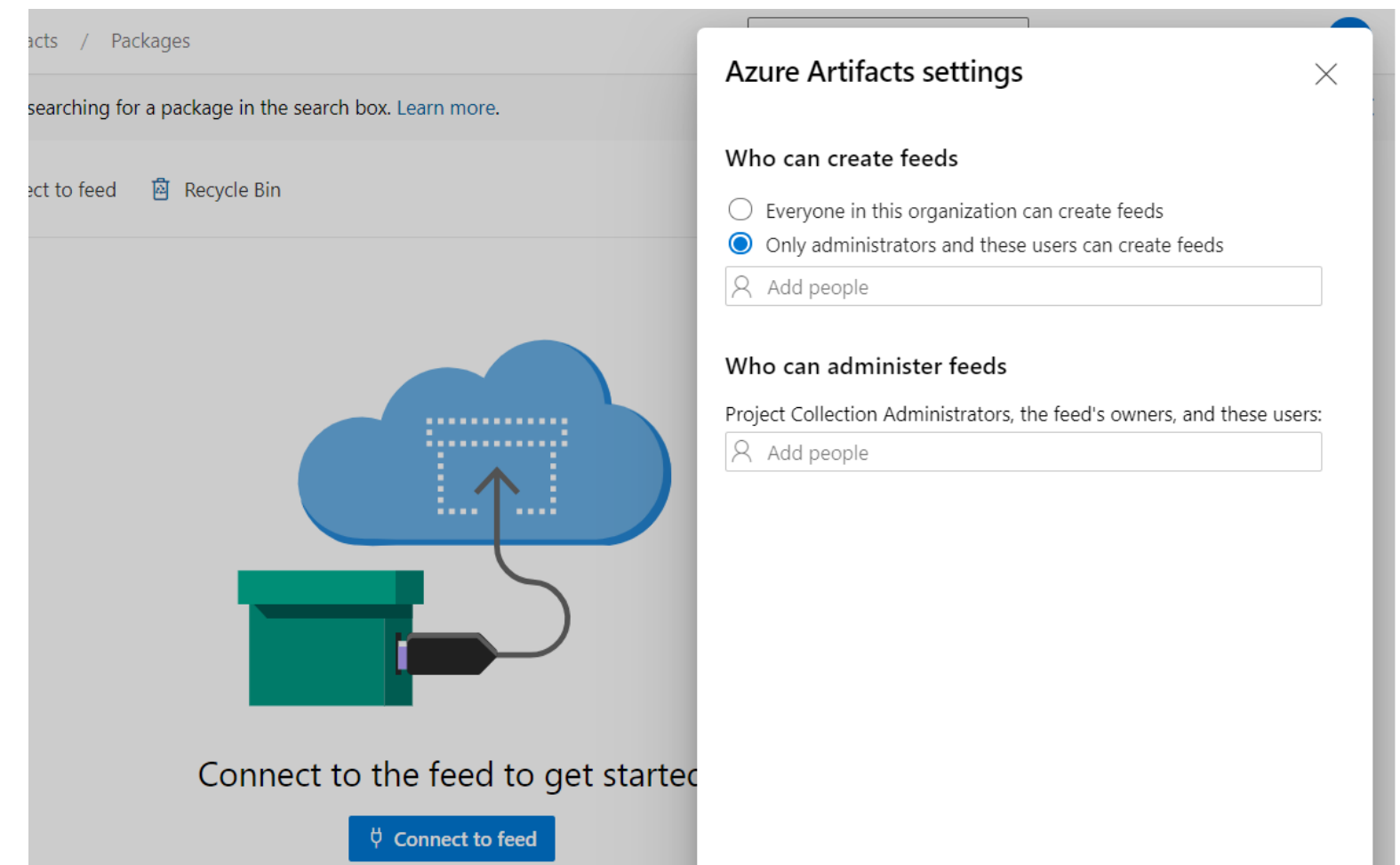


Package Security

Why Package Security?



Package security is required to control who can create the feed in the artifact. This is required to make sure that the package created in Azure Artifact is from a reliable source.

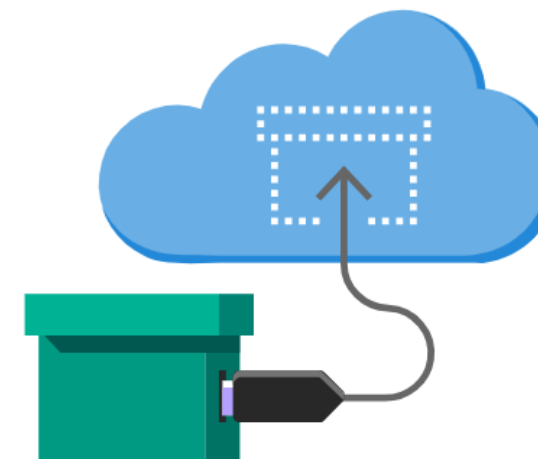
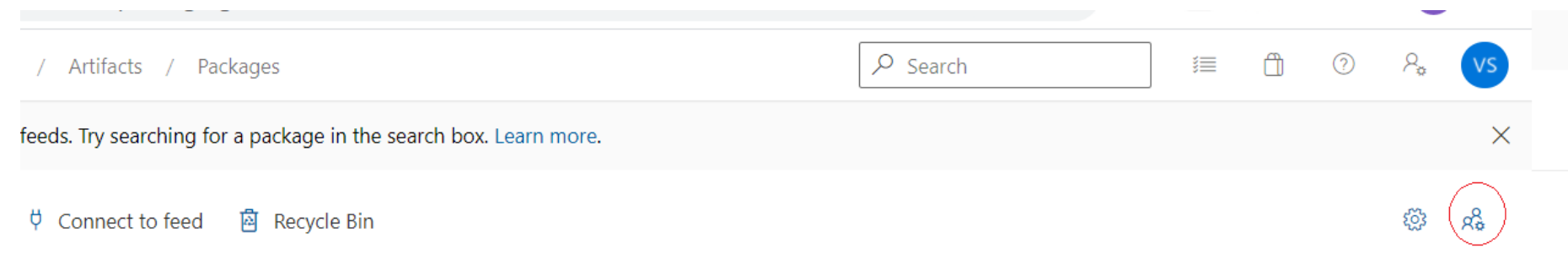


Specify the access as per the above options. This way, the package is secured.

Package Security

The package hosted in Azure Artifact can be granted permissions on who can view or who can not. This can be set from the Azure Artifact screen, as shown below. A package created can't be modified, but a new version of the same package can be created.

Click on the encircled red area.



Connect to the feed to get started

[Connect to feed](#)

Open-Source Software

Open-source software, packages, and libraries are available publicly for application development. For example, JSON.NET is open-source software used for JSON document processing.



- It is common to use open-source software or package to use in application development
- This can be vulnerable to application security
- Developers must take additional care, ensuring the open-source software or package is secure

License and Vulnerability Scans Integration

01

There are many open-source software or packages which are used for application development across many projects. They can also have a vulnerability, or some versions of these projects can be vulnerable in terms of security.

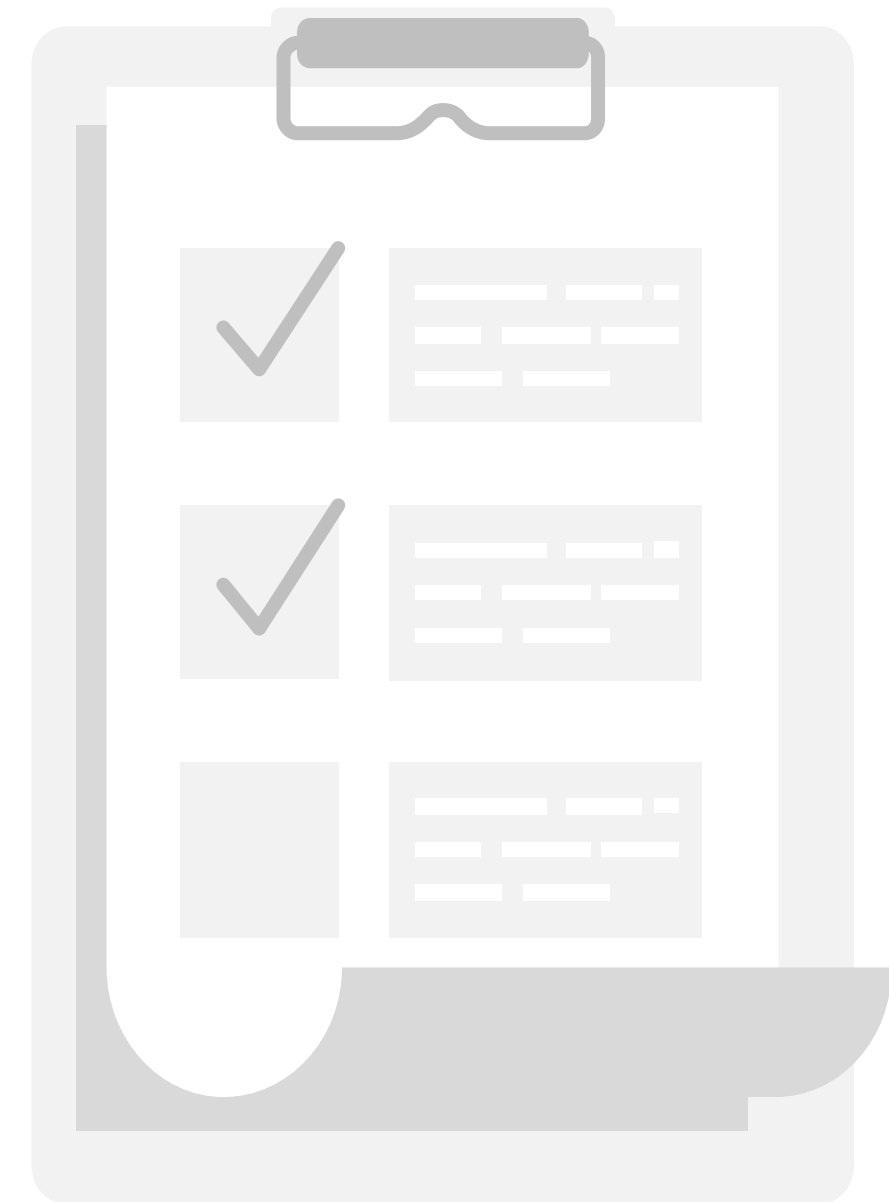
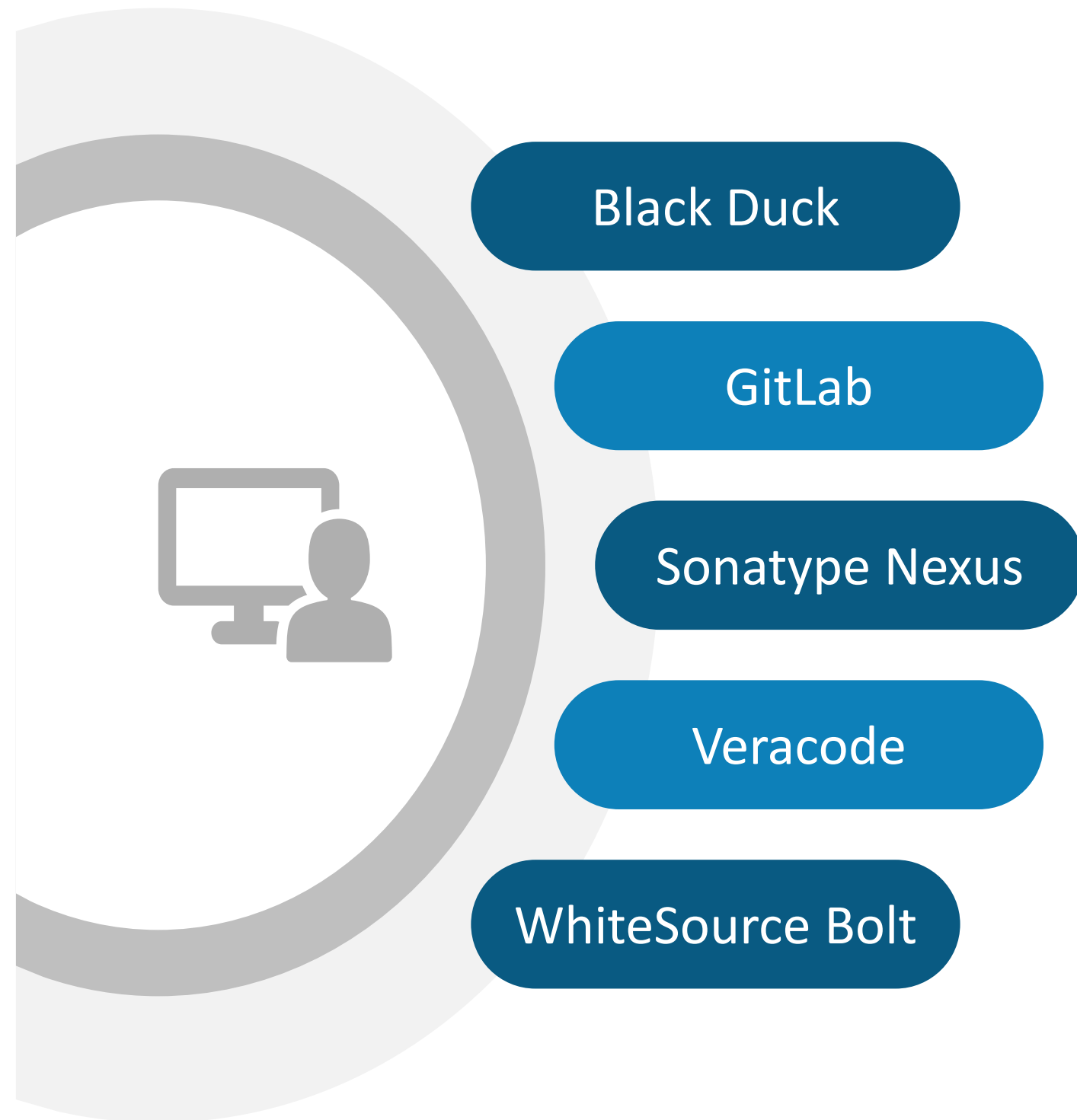
It is important to evaluate these packages or software. There are several tools available to verify the security aspects of these software.

02

03

To check the software license and whether it is still valid, a scan is required. Also, to check whether the open-source package is safe for application development.

Tools to Check Vulnerabilities



Implement a Versioning Strategy

Packages are managed through versioning strategy. This helps keep track of the new version of packages.

1.2.3 – beta2



nature of
change

quality of
change

Package Versioning





Demo: Updating Packages




Demo: Using WhiteSource to Manage Open-Source Security and License

Summary

What is Package Management?

A package contains a reusable code that other developers can use for their further application development.



- The package can be compiled binary code
 - Example: .dll for .NET, .class file for Java
- For languages like JavaScript or Python, the package may include source code

- Packages are generally compressed as .zip file or similar format
- Packages usually have .nupkg or .jar extension

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Why Package Security?



Package security is required to control who can create the feed in the artifact. This is required to make sure that the package created in Azure Artifact is from a reliable source.



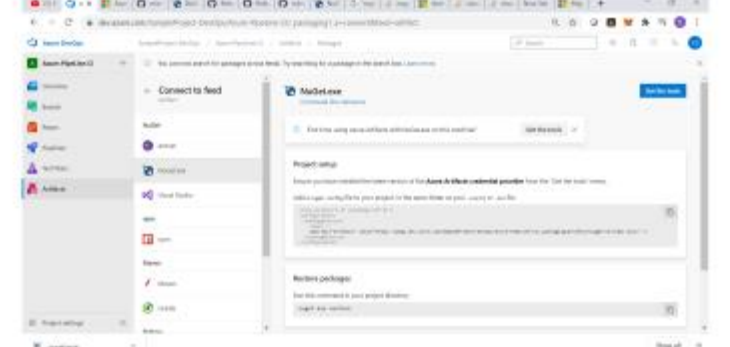


Specify the access as per the above options. This way, the package is secured.

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Migrating and Consolidating Artifacts



edureka!


Copyright © edureka and/or its affiliates. All rights reserved.

Package Versioning

Major: Major version represents the new package or some major breaking change of the existing package

Patch: Any bug fixes, but no additional feature

Package are versioned like:



Minor: Minor version represents the package having a new feature but backward compatible

Suffix: It is optional and identifies the package to some pre-release version. E.g., 1.10.2.beta1

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Demo: Updating Packages

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Demo: Using WhiteSource to Manage Open-Source Security and License

edureka!

Copyright © edureka and/or its affiliates. All rights reserved.

Questions



FEEDBACK



Thank You



For more information please visit our website
www.edureka.co