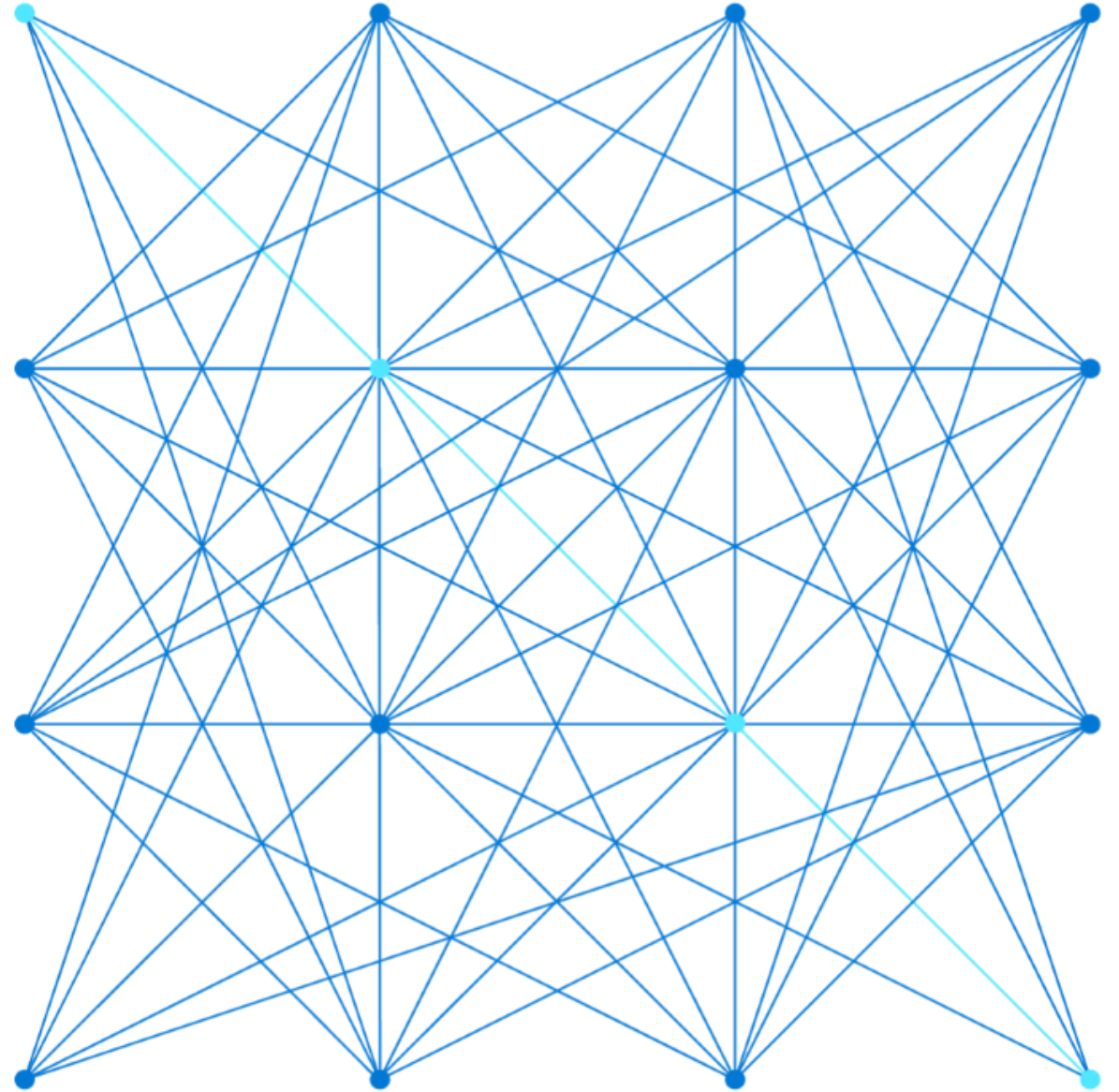
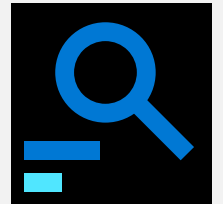


AZ-400.00

Module 9: Designing and Implementing a Dependency Management Strategy



Lesson 01: Module overview



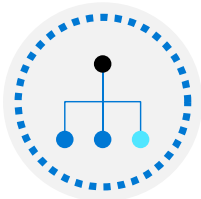
Module overview



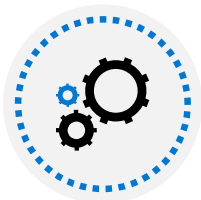
Lesson 1: Module overview



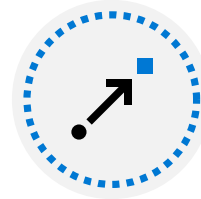
Lesson 2: Packaging dependencies



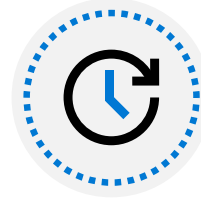
Lesson 3: Package management



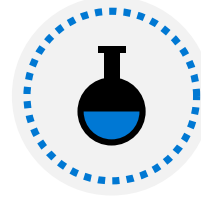
Lesson 4: Migrating and consolidating artifacts



Lesson 5: Package security



Lesson 6: Implement a versioning strategy



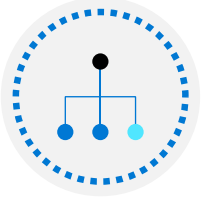
Lesson 7: Lab



Lesson 8: Module review and takeaways

Learning objectives

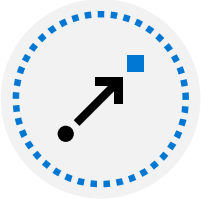
After completing this module, students will be able to:



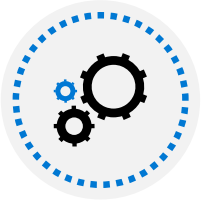
Recommend artifact management tools and practices



Abstract common packages to enable sharing and reuse



Migrate and consolidate artifacts

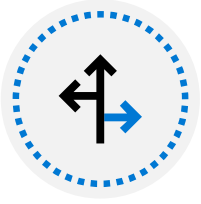


Migrate and integrate source control measures

Lesson 02: Packaging dependencies



What is dependency management?



Modern software is complex



Component based development is common

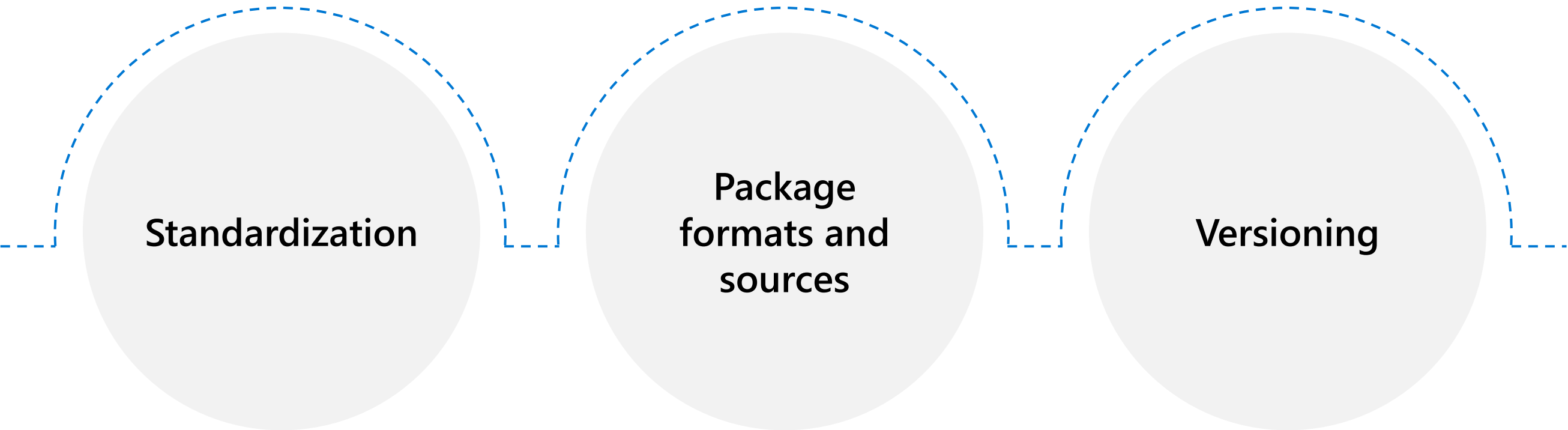


Not all software is written by a single team



Dependencies on components created by other teams or persons

Elements of a dependency management strategy



#mw: [How one programmer broke the internet by deleting a tiny piece of code — Quartz \(qz.com\)](https://quartz.cc/2017/01/20/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/)

Source and package componentization

Source componentization:

Split out components

Related projects in different solutions

Package componentization:

Composing your solution to use packages

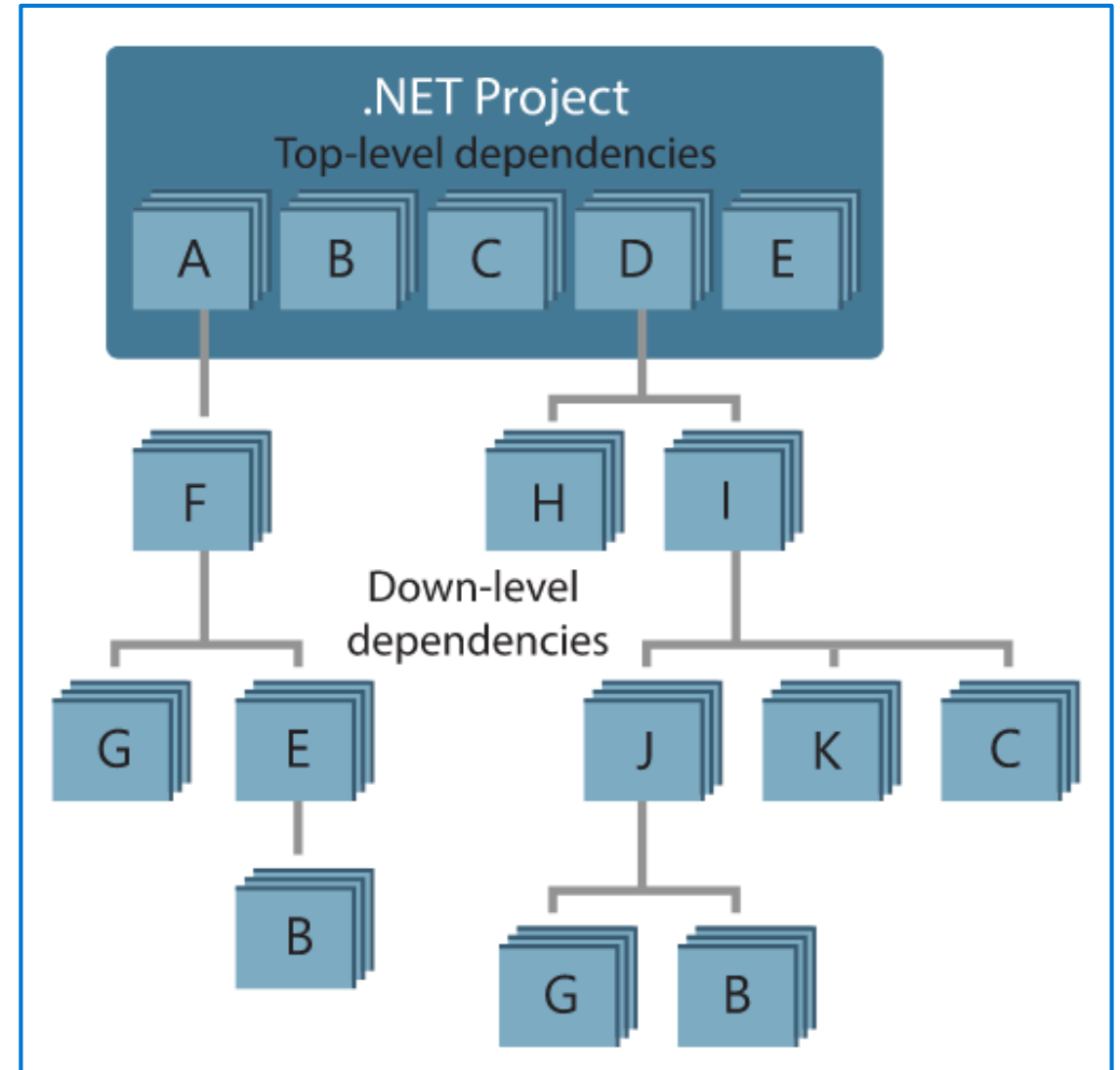
Decompose your system

Approach:

1. Draw a dependency graph
2. Group components in sets of related components

Ask:

- Will my teams often make spanning check-ins across the sets I've created?
- Is a single team responsible for the entire set?
- For a single set, is there a shared release cadence?



(#mw) Microsoft recommend (roughly)

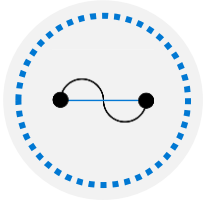
- Use source composition for related projects that are worked on by a single team
- Use binary composition & package composition for externals (components from faraway or isolated teams), and isolated shared components.

Scanning your codebase for dependencies

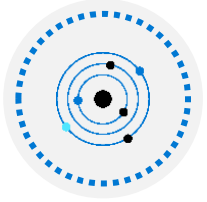
#mw-Ask: How can I best organize the components of my codebase? Should I separate some code into packages? What am I looking for when choosing candidates?



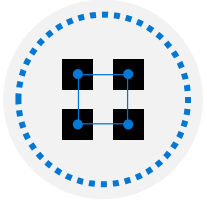
Duplicate code



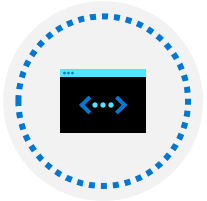
High cohesion and low coupling



Individual lifecycle

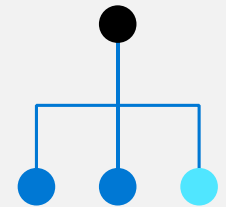


Stable parts



Independent code and components

Lesson 03: Package management (#mw now called *artifacts* in AzDevOps)



Packages

A package is a formalized way of creating a distributable unit of software artifacts that can be consumed from another software solution.



Microsoft platform and
.NET artifacts



Node.js modules



Python scripts



Universal packages

maven

Java packages



Docker images

Package feeds

Centralized storage of package artifacts:

- Public or privately available
- Offer secure access for private feeds
- Versioned storage of packages
- Managed by tooling

Package feeds are also known as:

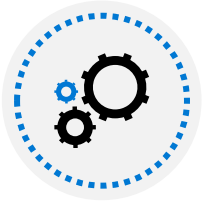
- Package repositories
- Package registries

Package types:

Public: NuGet.org, Npmjs.org, PyPi.org, Docker Hub

Private: MyGet, Azure Container Registry, Azure Artifacts, Self-hosted solutions

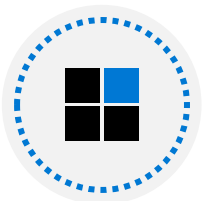
Package feed managers (#mw i.e what should a package feed allow me to do?)



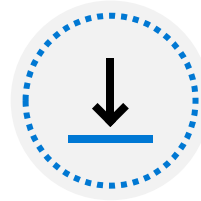
Manage feeds



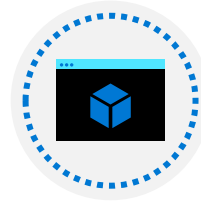
Search and list packages
from feed



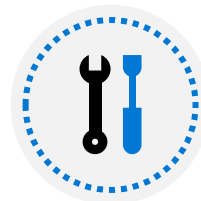
Consume packages



Maintain local installation cache



Publish packages



Choose tooling:

Command-line tooling

Integrated in build and release pipelines

Common public package sources

NuGet Gallery

<https://nuget.org>

NPMjs

<https://npmjs.org>

Maven

<https://search.maven.org>

Docker Hub

<https://hub.docker.com>

Python Package Index

<https://pypi.org>

Self-hosted and SaaS based package sources

Package type	Self-hosted private feed	SaaS private feed
NuGet	NuGet server	Azure Artifacts, MyGet
NPM	Sinopia, cnpmjs.org , Verdaccio	NPMjs.org , MyGet, Azure Artifacts
Maven	Nexus, Artifactory, Archiva	Azure Artifacts, Bintray, JitPack
Docker	Portus, Quay, Harbor	Docker Hub, Azure Container Registry, Amazon Elastic Container Registry
Python	PyPI Server	Azure Artifacts, Gemfury

Consuming packages

- 1 Identify a required dependency in your codebase
- 2 Find a component that satisfies the requirements for the project
- 3 Search the package sources for a package offering a correct version of the component
- 4 Install the package into the codebase and development machine
- 5 Create the software implementation that uses the new components from the package

Azure Artifacts



**Azure
Artifacts**

Create private and public **package feeds** for package types:

1. NuGet
2. NPM
3. Maven
4. Universal
5. Python

Publishing packages

From Azure DevOps portal

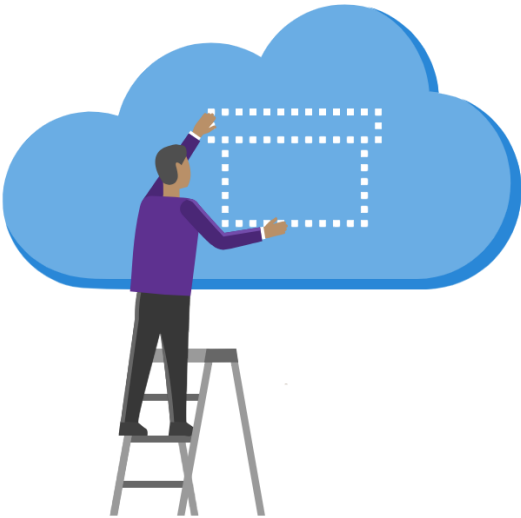
Feeds are centralized

Specify:

Name

Visibility

Public sources as upstream



Create new feed

Feeds host and control permissions for your packages.

Name *

Team project - [\(what's this?\)](#)

Visibility - Who can use your feed

- ☒  People in xpirit - Members of your organization can view the packages in your feed
- ☐  Specific people - Only people you give access to will be able to view this feed

Packages from public sources (nuget.org, npmjs.com)

- ☒ Use packages from public sources through this feed
- ☐ Only use packages published to this feed

Create

Cancel

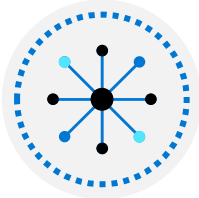
Lesson 04: Migrating and consolidating artifacts



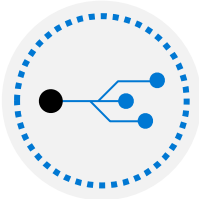
Identifying existing artifact repositories



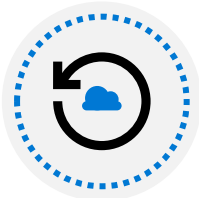
An artifact is a deployable component of your application.



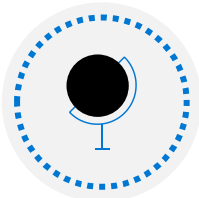
Azure Pipelines can work with a wide variety of artifact sources and repositories.



Each release can specify which version of the artifacts are required.



Azure Artifacts can eliminate the need to manage file shares or to host private package servers.



Azure Artifacts provides universal artifact management for Maven, npm and NuGet.

Migrating and integrating artifact repositories

[Get started with NuGet packages in Azure DevOps Services and TFS](#)

[Use npm to store JavaScript packages in Azure DevOps Services or TFS](#)

[Get started with Maven packages in Azure DevOps Services and TFS](#)

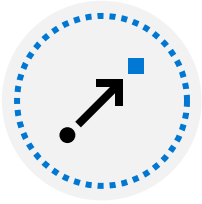
[Get started with Python packages in Azure Artifacts](#)

[Publish and then download a Universal Package](#)

Lesson 05: Package security



Securing access to package feeds



Feeds must be secured:

Private feeds

Not allow access by unauthorized users for publishing

Restricted access for consumption:

Whenever a package feed and its packages should only be consumed by a certain audience, it is required to restrict access to it. Only those allowed access will be able to consume the packages from the feed.

Restricted access for publishing:

Secure access is required to restrict who can publish, so feeds and their packages cannot be modified by unauthorized or untrusted persons and accounts.

Roles

Available roles in Azure Artifacts:

Reader: Can list and restore (or install) packages from the feed

Collaborator: Is able to save packages from upstream sources

Contributor: Can push and unlist packages in the feed

Owner: Has all available permissions for a package feed

Project Collection Build Service is contributor by default

DevOpsCertificationFeed > Feed settings	
Feed details	Permissions
Views	Upstream sources
+ Add users/groups	
Delete ...	
Filter by User/Group	
User/Group	Role
Alex Thissen	Owner
[xpirit]\Project Collection Administrators	Owner
[DevOpsCertification-Course-MS]\Project Administrators	Owner
Project Collection Build Service (xpirit)	Contributor
[DevOpsCertification-Course-MS]\Contributors	Contributor

Permissions

Roles have certain permissions

Permission	Reader	Collaborator	Contributor	Owner
List and restore/install packages	✓	✓	✓	✓
Save packages from upstream sources		✓	✓	✓
Push packages			✓	✓
Unlist/deprecate packages			✓	✓
Delete/unpublish package				✓
Edit feed permission				✓
Rename and delete feed				✓

(#mw: Two things going on here, a pipeline is accessing Azure Artifacts and a pipeline is accessing an external feed)

Authentication

Authentication is required for Azure Artifacts

Transparently taken care of when logged into portal or in build tasks

External package sources may require credentials:

Create a service connection

×

Add npm service connection

☒ Username and Password

☐ Authentication Token

Connection name

Registry URL

i

Username

i

Password

i

Learn More

☒ Allow all pipelines to use this connection.

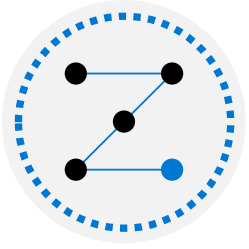
OK

Close

Lesson 06: Implement a versioning strategy

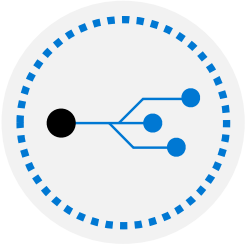


Introduction to versioning



Packages need to be versioned

- Identification
- Maintainability
- Each package has its own lifecycle and rate of change



Packages are immutable

- Once published a package cannot be changed
- Replacing or updating a package is not allowed
- Any change requires a new version

Versioning of artifacts

Way to express version technically,
varies per package type

Versioning requires a scheme

Typical Scheme:

The diagram illustrates a typical semantic versioning scheme using the example version 2.1.15. The version is displayed in large black font. Below it, three blue brackets are positioned under the numbers 2, 1, and 15 respectively. Each bracket has a vertical line pointing down to a label: 'Major' under the first bracket, 'Minor' under the second, and 'Patch' under the third.

2.1.15

Major Minor Patch

Semantic versioning

Express nature and risk of change

1.2.3-beta2

The diagram shows the semantic versioning string "1.2.3-beta2". The "1.2.3" part is in blue and is bracketed with a blue line underneath it, labeled "Nature of change". The "-beta2" part is in dark grey and is bracketed with a dark grey line underneath it, labeled "Quality of change".

See also: <https://semver.org>

Promoting packages

Promote packages from
@local view to other release
views.

Upstream sources will only be evaluated from
@local view:
Only visible in other release views after being promoted

The screenshot displays the PartsUnlimited web interface for the package 'PartsUnlimited.Security 1.0.1'. The 'Overview' tab is selected. In the top navigation bar, the 'Promote' button (represented by an upward arrow icon) is highlighted with a red rectangular box. Below this bar, the 'Get this package' section shows a 'Connect to feed' button followed by the command `PM> Install-Package PartsUnlimited.Security -version 1.0.1`. To the right, a modal dialog titled 'Promote this package' is open. It features a 'View' dropdown menu currently set to 'PartsUnlimited@Prerelease'. At the bottom of the modal are two buttons: 'Promote' (in blue) and 'Cancel' (in light gray).

Release views

Views help in defining quality without changing version numbers

Three default views:

1

Local

2

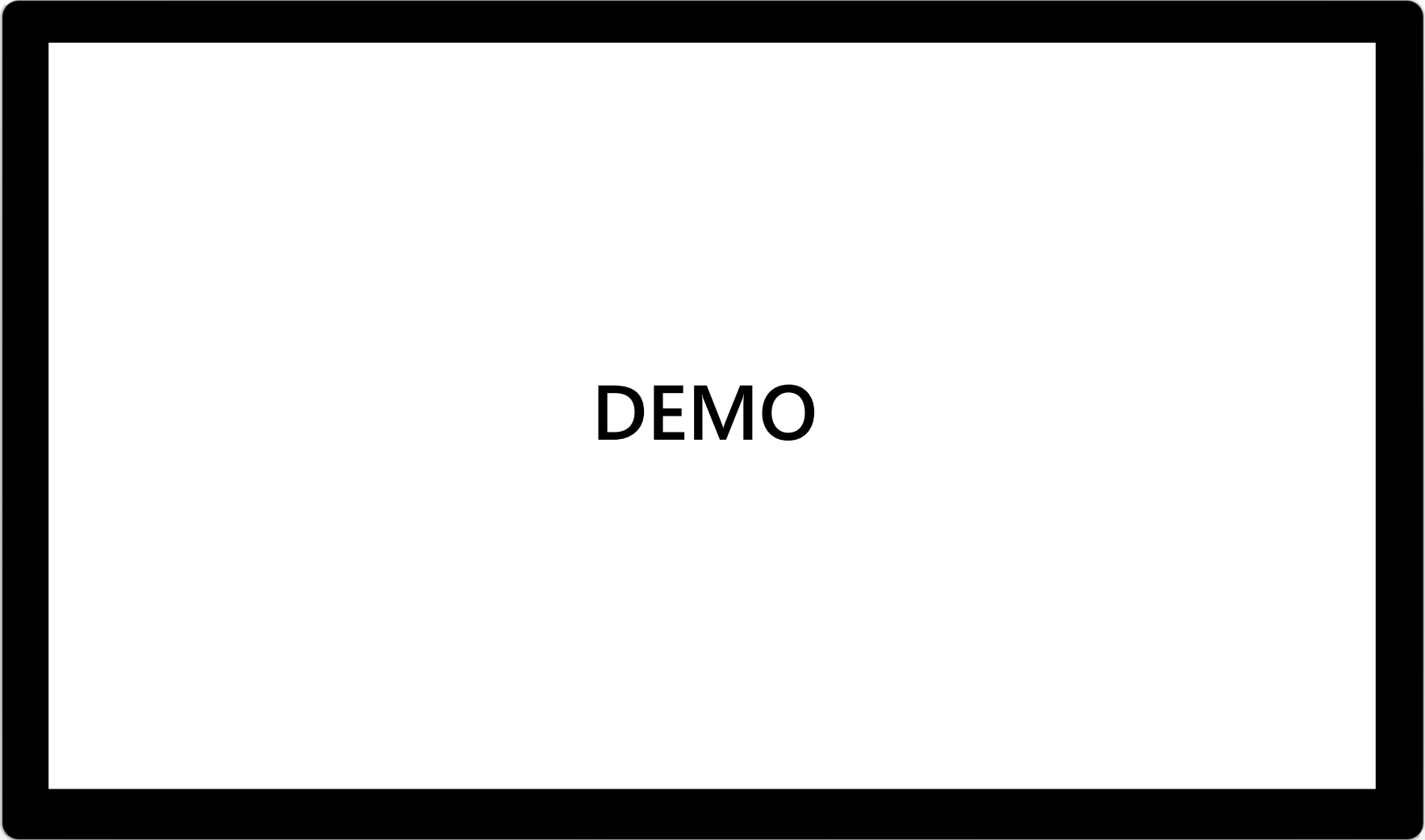
Release

3

Prerelease

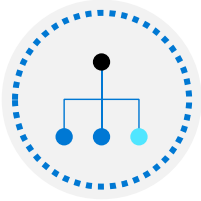
```
https://pkgs.dev.azure.com/{yourteamproject}/_packaging/{feedname}  
@{Viewname}/nuget/v3/index.json
```

Demonstration: promoting a package

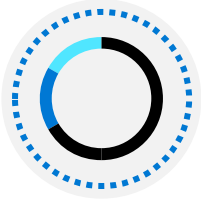


DEMO

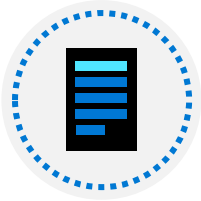
Best Practices for versioning



Have a documented versioning strategy



Adopt [SemVer 2.0](#) for your versioning scheme

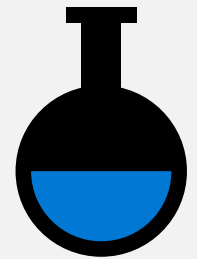


Each repository should only **push** to one feed



On package creation, automatically publish packages back to the feed

Lesson 07: Lab



Package management with Azure Artifacts

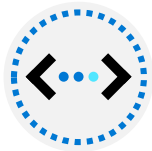


Lab overview:

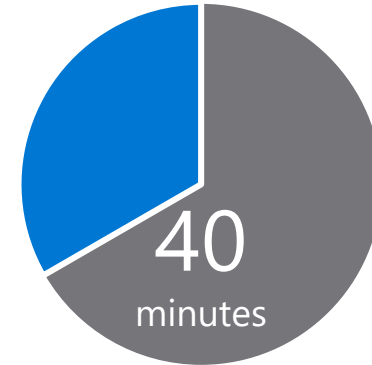
In this lab, you will learn how to work with Azure Artifacts.

Objectives:

- Create an Azure Active Directory (Azure AD) service principal
- Create an Azure key vault
- Track pull requests through the Azure DevOps pipeline



Duration:



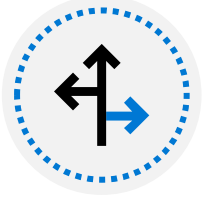
Lesson 08: Module review and takeaways



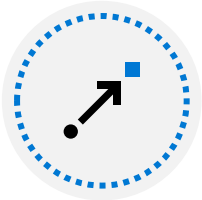
What did you learn?



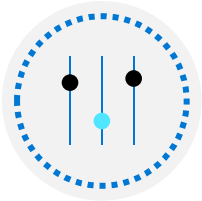
Recommend artifact management tools and practices



Abstract common packages to enable sharing and reuse



Migrate and consolidate artifacts



Migrate and integrate source control measures

Module review questions

1

If you are creating a feed that will allow yourself and those that you invite to publish, what visibility should you choose?

2

Can you create a package feed for Maven in Azure Artifacts?

3

What type of package should you use for Machine learning training data & models?

4

If an existing package is found to be broken or buggy, how should it be fixed?

5

What is meant by saying that a package should be immutable?