



WorkshopPLUS Power Platform for Administrators

Application Lifecycle Management



Learning Units covered in this Module

- Understand Power Platform ALM
- Deploy Resources using a Package
- Deploy Resources using a Solution
- Power Platform Build Tools for Azure DevOps
- Power Platform + GitHub

Objectives

After completing this Learning, you will

- Get understanding about basic concepts about Application Lifecycle Management (ALM)
- Learn how to export/import Apps and Flows using packages
- Learn what Power Platform Solutions are
- Get understanding what Power Platform Build Tools for Azure DevOps are
- Get understanding what GitHub actions for Power Platform are



Understand Power Platform ALM

What is ALM?

- Lifecycle management of applications, which includes governance, development and maintenance
- Includes topics and tasks like:

Project, change and requirements management	Maintenance
Software architecture	Continuous Integration (CI)
Development	Continuous Deployment (CD)
Testing	Release management

- ALM tools provide a standardized system for communication and collaboration between software development teams and related departments, such as test and operations. These tools can also automate the process of software development and delivery.

Key areas of ALM

- **Governance** includes requirements management, resource management, data security, user access, change tracking, review, audit, deployment control, and rollback.
- **Application development** includes identifying current problems, and planning, design, building, and testing the application. This area includes traditional developer and app maker roles.
- **Maintenance** includes deployment of the app and maintenance of optional and dependent technologies.

Cyclical development process

The application lifecycle is the cyclical software development process that involves these areas.



ALM for Power Platform

Dataverse lets you securely store and manage data that's used by business applications.

To use the features and tools available for ALM, all environments that participate in ALM must include a Dataverse database.

The following concepts are important for understanding ALM using the Microsoft Power Platform:

- **Solutions** are the mechanism for implementing ALM which you use to distribute components across environments through export and import.
- **Dataverse** stores all the artifacts, including solutions.
- **Components** (inside solutions) such as apps, flows, tables, site maps etc. represents something that you can potentially customize
- **Source control** should be your source of truth for storing and collaborating on your components.

Types of environments used in ALM

- **Sandbox** environment is any non-production environment of Dataverse and is the place where you can safely develop and test application changes with low risk. Sandbox environments include capabilities that would be harmful in a production environment, such as reset, delete, and copy operations.
- **Production** is environment where apps and other software are put into operation for their intended use.
- **Development** (Developer Plan) a single-user environment primarily meant for learning purposes giving access to premium functionality of Power Apps, Power Automate, and Dataverse. Although you can't share assets from this environment, you can participate in the Azure DevOps pipeline.

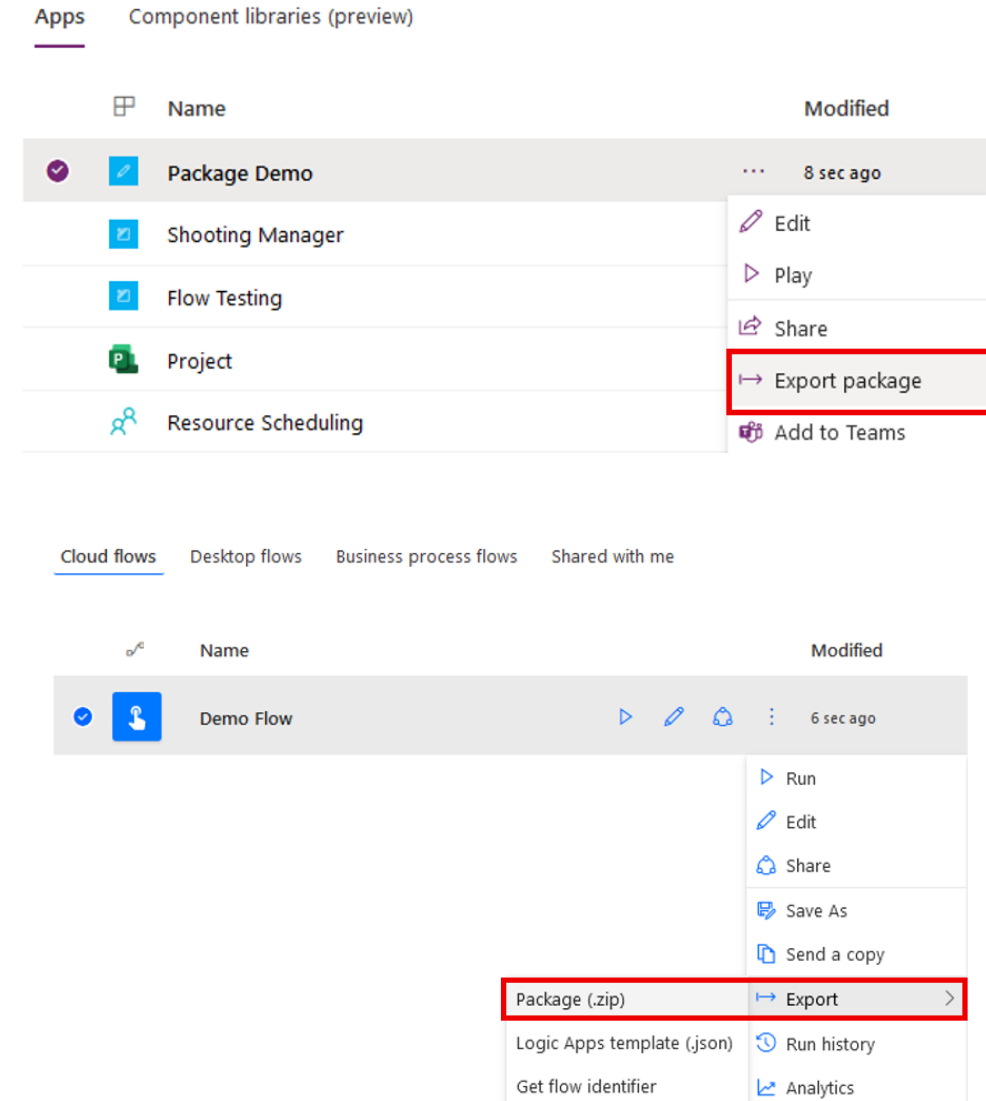
Who should have access?

Environment purpose	Roles that have access	Comments
Development	App makers and developers.	App users shouldn't have access. Developers require at least the Environment Maker security role to create resources.
Test	Admins and people who are testing.	App makers, developers, and production app users shouldn't have access. Test users should have just enough privileges to perform testing.
Production	Admins and app users. Users should have just enough access to perform their tasks for the apps they use.	App makers and developers shouldn't have access, or should only have user-level privileges.
Default	By default, every user in your tenant can create and edit apps in a Dataverse default environment that has a database.	We strongly recommend that you create environments for a specific purpose, and grant the appropriate roles and privileges only to those people who need them.

Deploy Resources using a Package

Power Platform Packages

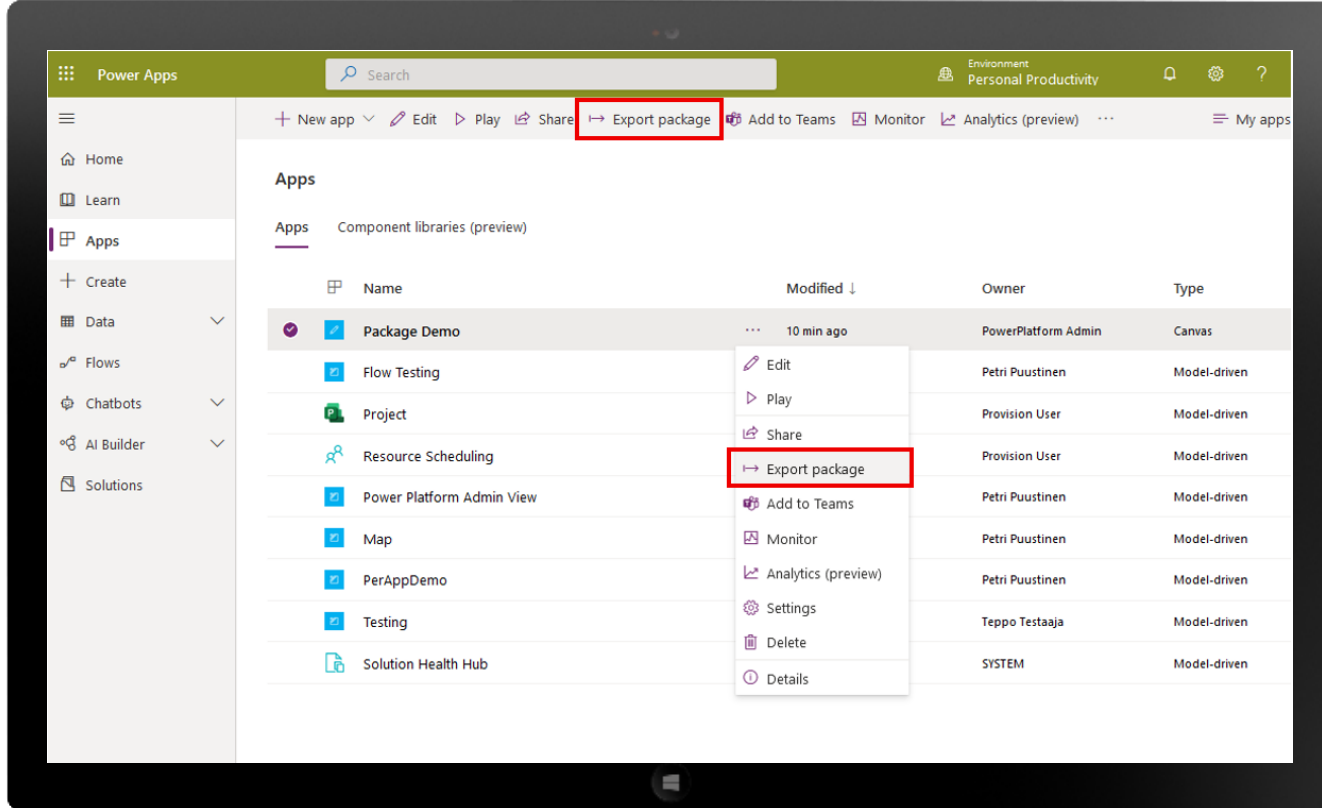
- You can deploy (export/import) canvas apps and flows by using packages.
- This feature allows you to export an app from one environment and import it to another.
- Only the **Owners** of an app or flow can export a package.
- To import an app or flow, the **Environment Maker** permission is required on the destination environment.



Which resources can be deployed using a package

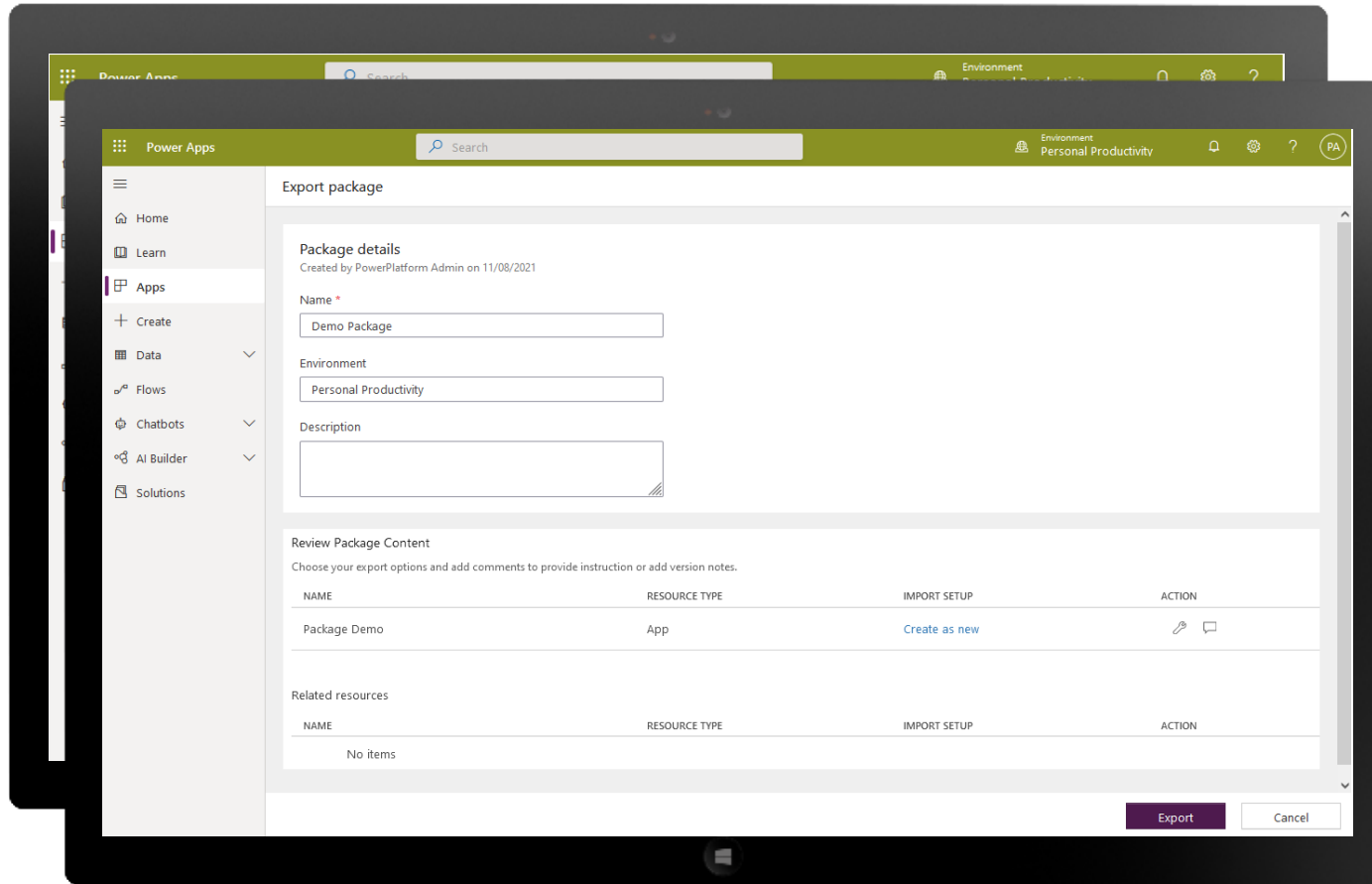
Resource type	Supported	Import options
App	Yes (canvas apps)	There are two options to import an app into an environment: Create new and Update
Power Automate	Yes	There are two options to import a flow into an environment: Create new and Update Note: All resources that the flow depends on will also be included within the app package that is exported and will need to be configured when the package is imported.
Custom Connectors	No	Not supported. You'll need to re-create the custom connector on the target environment.
Connections	No	Not supported. You'll need to re-create the connections on the target environment.
Dataverse Customizations	No	Exporting Dataverse customizations as a part of a canvas app / flow package isn't supported. You'll need to use Dataverse solutions instead.
Gateways	No	Not supported. You'll need to re-create the on-premise gateway on the target environment.

Package Export



Select Export package for an App you want to export

Package Export



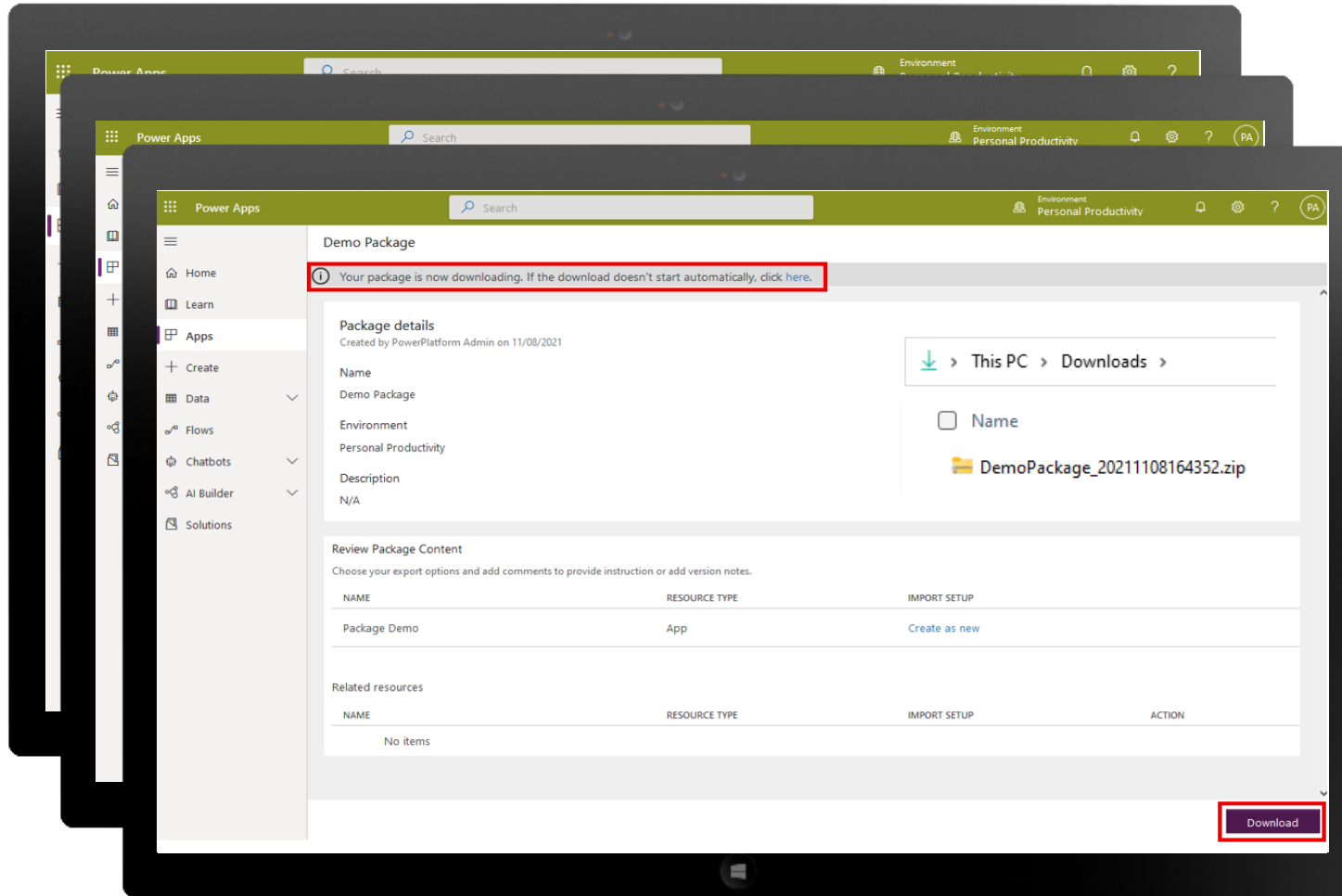
1

Select Export package for an App you want to export

2

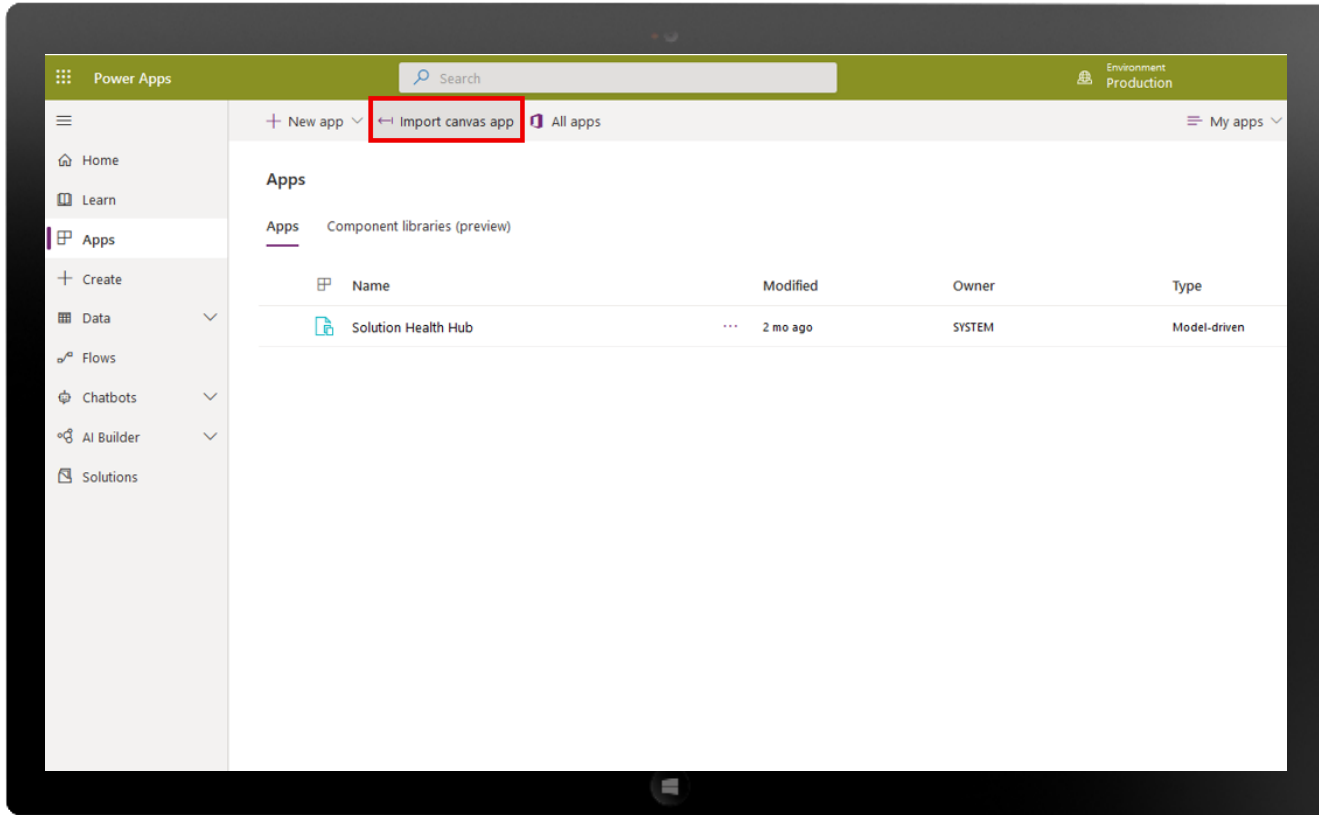
Define Name, Description and import setup action (create as new / update) for all resources and click Export

Package Export



- 1 Select Export package for an App you want to export
- 2 Define Name, Description and import setup action (create as new / update) for all resources and click Export
- 3 Package is downloaded a ZIP file

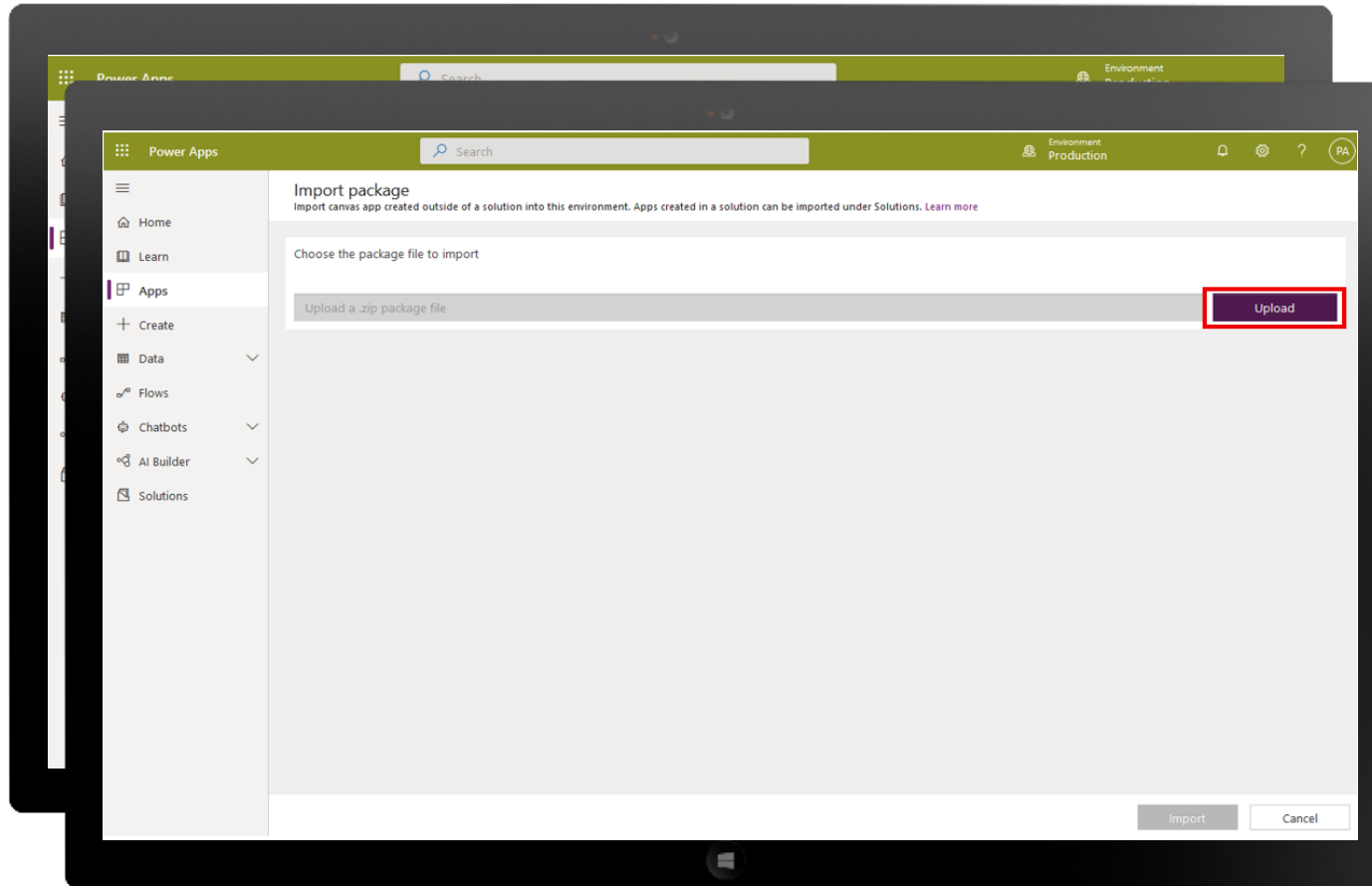
Package Import



1

Choose Import canvas app in make.powerapps.com or Import in aka.ms/flow

Package Import



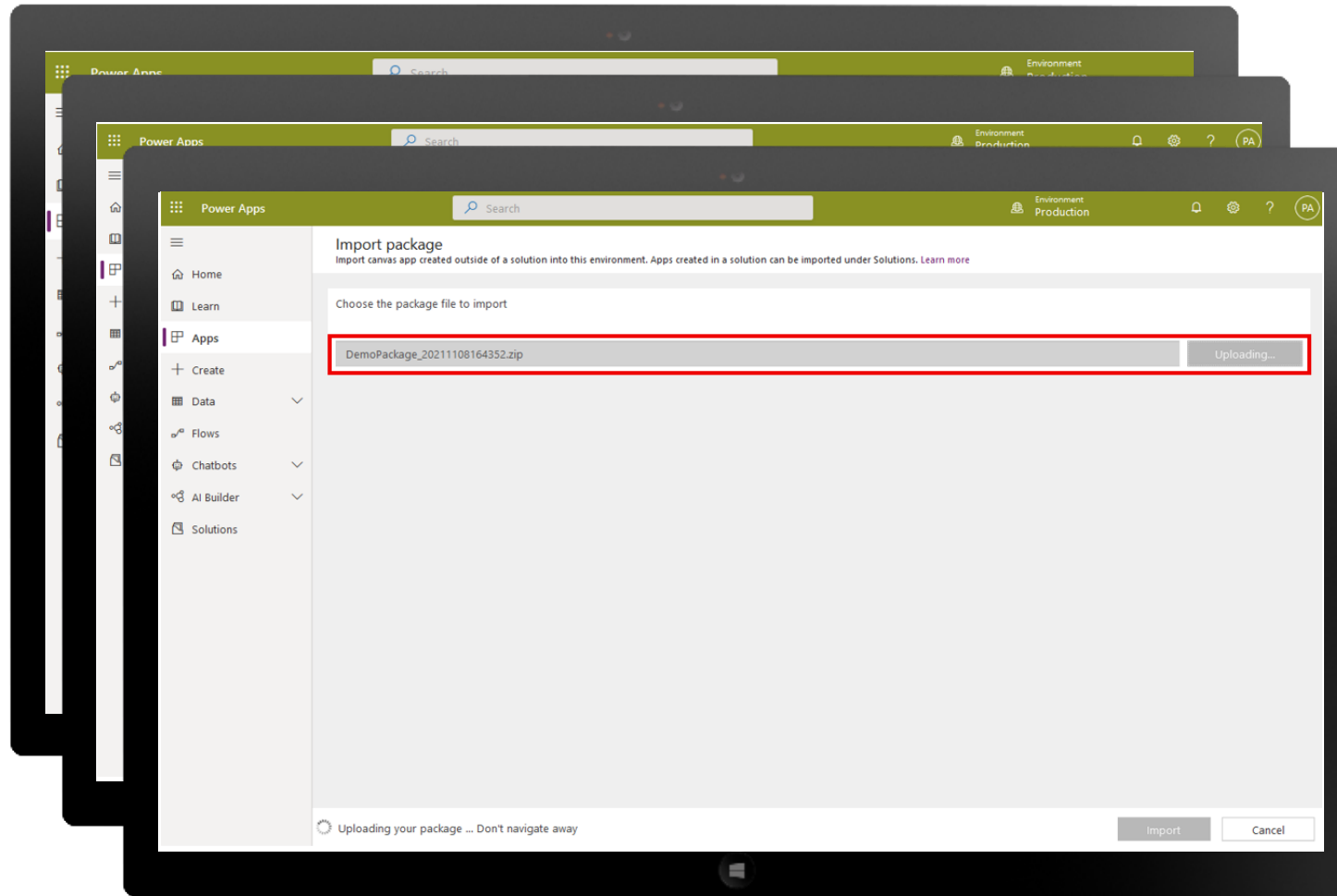
1

Choose Import canvas app in make.powerapps.com or Import in aka.ms/flow

2

Select exported package ZIP file

Package Import



1

Choose Import canvas app in make.powerapps.com or Import in aka.ms/flow

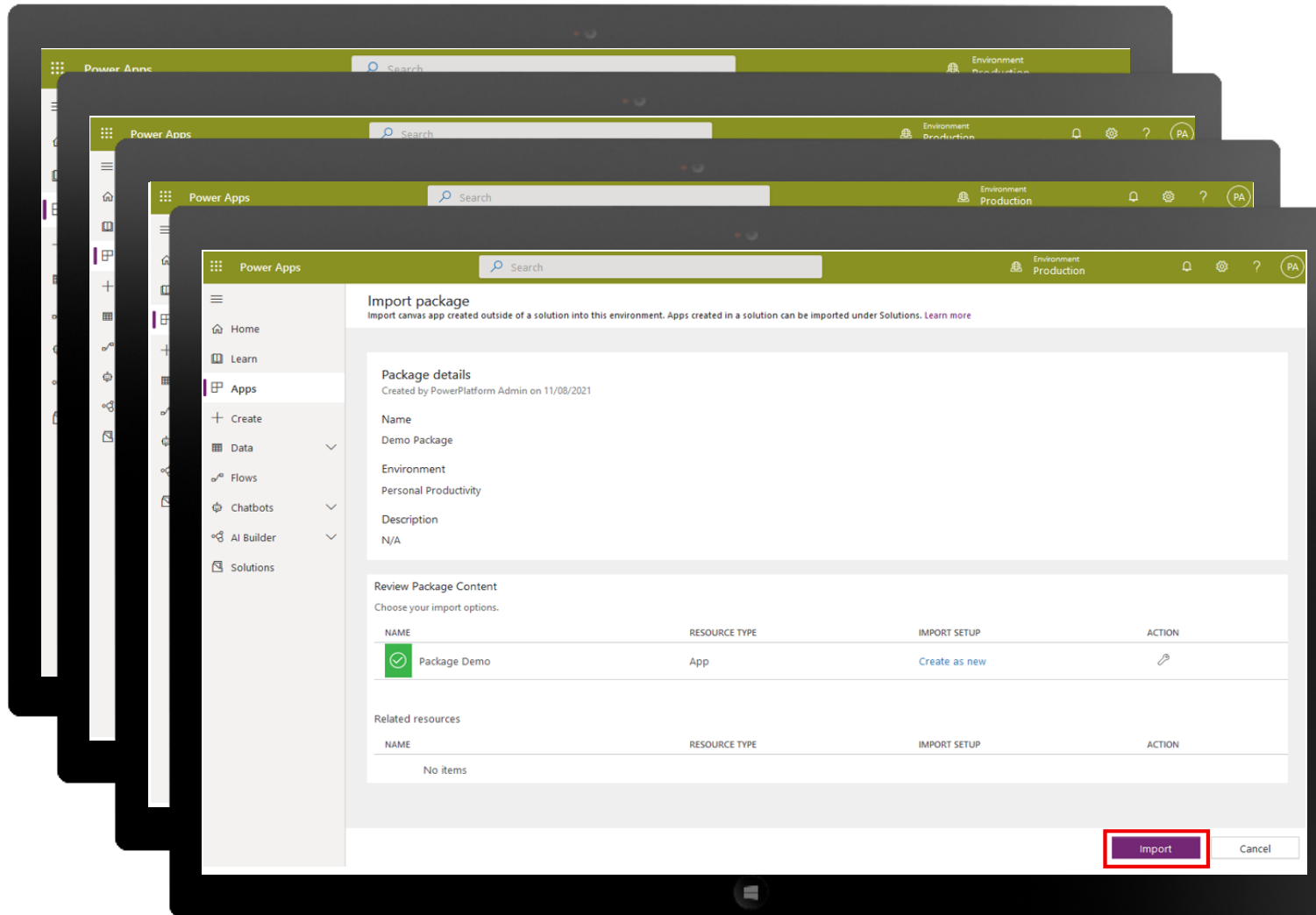
2

Select exported package ZIP file

3

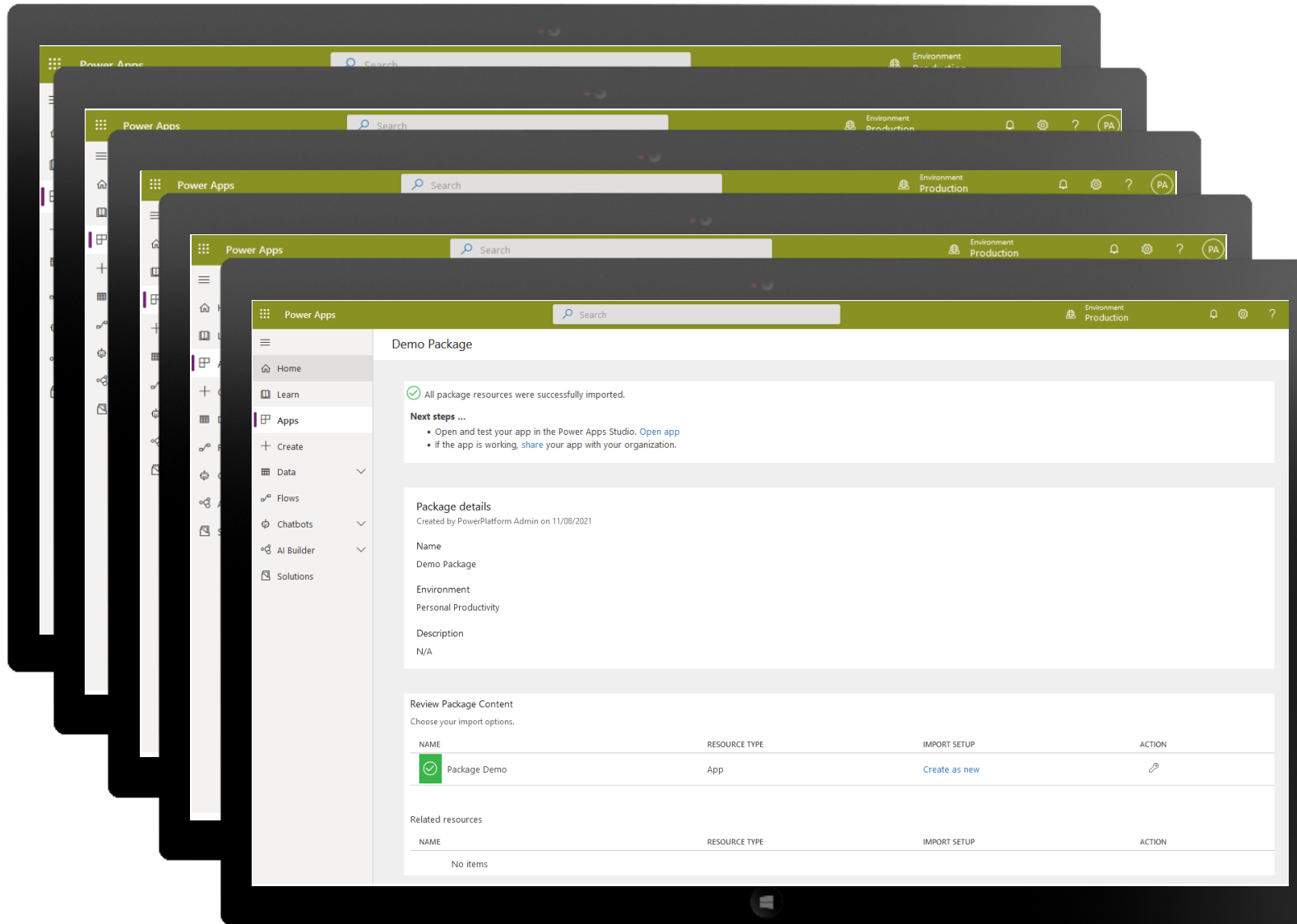
Package will be uploaded

Package Import



- 1 Choose Import canvas app in make.powerapps.com or Import in aka.ms/flow
- 2 Select exported package ZIP file
- 3 Package will be uploaded
- 4 Set import setup actions and click Import

Package Import



1

Choose Import canvas app in make.powerapps.com or Import in aka.ms/flow

2

Select exported package ZIP file

3

Package will be uploaded

4

Set import setup actions and click Import

5

Import is done and you can open the app

Deploy Resources using a Solution

Solutions Concept

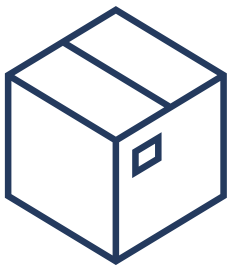
- Solutions are the mechanism for implementing ALM in Power Apps and Power Automate.
- With solutions we can transport Power Platform components from one environment to another or to apply customizations to existing apps.
- Can contain one or more apps and flows as well as other components such as site maps, entities, processes, web resources, option sets, and more
- Some components are nested within other components, for example, an Dataverse table contains forms, views, charts, fields, entity relationships, messages, and business rules.

Two types of Solutions



UNMANAGED

- Used in development environments while you make changes to your application
- No restrictions on what can be added, removed, or modified
- Can be exported as unmanaged or managed
- Should be checked into your source control system
- You can't delete its components by uninstalling the solution



MANAGED

- Complete solution that is intended to be distributed and installed
- Components can't be added or removed
- Cannot be exported
- Some changes can be done to components if explicitly allowed using managed properties
- When a managed solution is deleted (uninstalled), all the customizations and extensions included with it are removed.

Solution Lifecycle

Create

- Author and export unmanaged solutions

Update

- Create updates to a managed solution that are deployed to the parent managed solution
- You can't delete components with an update

Upgrade

- Upgrade an existing managed solution
- Involve rolling up (merging) all patches to the solution into a new version of the solution
- Will delete components that existed but are no longer included in the upgraded version
- You can choose to upgrade immediately or to stage the upgrade so that you can do some additional actions prior to completing the upgrade

Patch

- Contains only the changes such as adding or editing components and assets.
- Use patches when making small updates (similar to a hotfix)
- You can't delete components with a patch.

Solution Publisher

- The publisher of a solution where a component is created is considered the owner of that component.
- Owner controls what changes other publishers of solutions including that component are allowed to make or restricted from making
- Every solution has a publisher, you should create your own publisher rather than use the default. Publisher is specified when you create a solution.
- It is possible to move the ownership of a component from one solution to another within the same publisher but not across publishers.
- A solution publisher includes a prefix which is a mechanism to help avoid naming collisions.

The screenshot displays the Microsoft Dynamics 365 solution management interface. At the top, a search bar and navigation tabs are visible. Below the search bar, a table lists existing solutions. The 'Demo' solution is selected, and its details are shown in the 'New solution' form below.

Display name	Name	Type
Demo	contoso_demo	Table
Demo App	contoso_demo_ec7f4	Canvas App

New solution

Display name *

Name *

Publisher *

Select a Publisher

+ New publisher

Version *

1.0.0.0

More options ▾

New publisher

Publishers indicate who developed associated so

Properties Contact

Display name *

Contoso

Name *

contoso

Description

Prefix *

contoso

Choice value prefix *

33065

Preview of new object name

contoso_Object

Solution Dependencies

- Because of the way that managed solutions are layered, some managed solutions can be dependent on solution components in other managed solutions.
- Solution publishers can take advantage of this to build solutions that are modular.
- You may need to install a “base” managed solution first, and then you can install a second managed solution that will further customize the components in the base managed solution.
- The second managed solution depends on solution components that are part of the first solution.
- The system tracks these dependencies between solutions. If you try to install a solution that requires a base solution that isn’t installed, you won’t be able to install the solution.

Demonstration

Create Solution with Power
App, Flow and Table

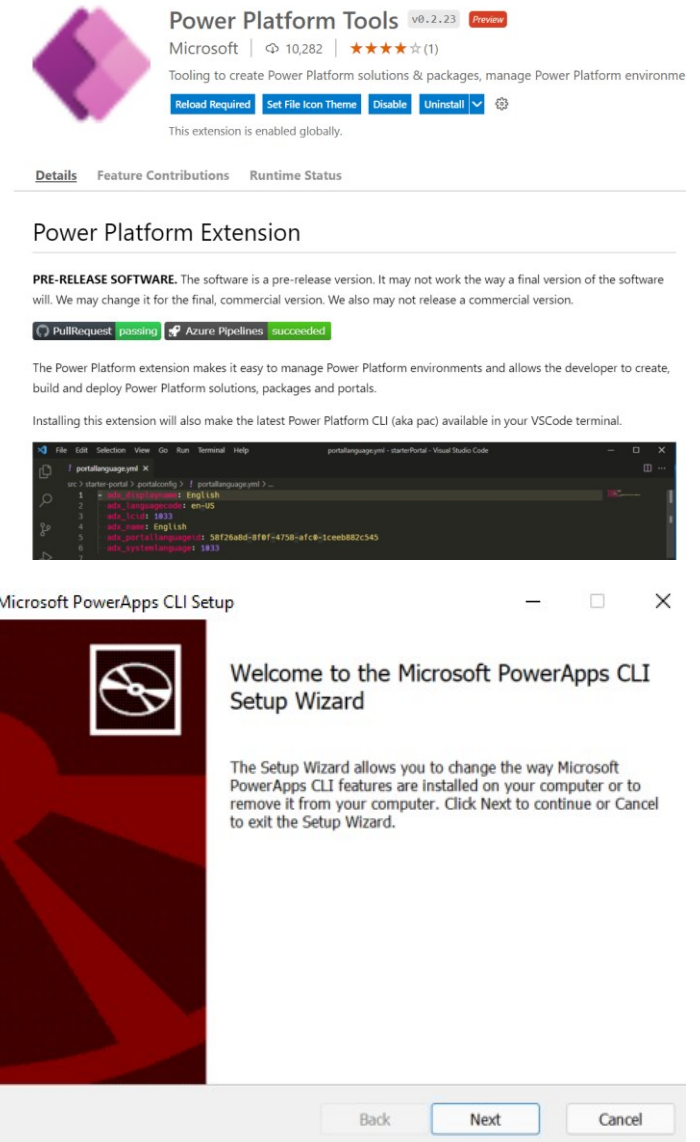


Power Platform CLI

Tools you can use to perform various operations in Microsoft Power Platform related to environment lifecycle, authentication, and work with Microsoft Dataverse environments, **solution packages**, portals, code components, and so on.

You can use either of the following ways to install Microsoft Power Platform CLI:

- Visual Studio Code extension
- Standalone MSI installation



Power Platform CLI – Common Commands

Command	Description
Admin	Commands for environment lifecycle features
Auth	Commands to authenticate to Dataverse
Canvas	Commands for working with canvas app source files
Org	Commands for working with Dataverse environments
Package	Commands for working with solution packages
Paportal	Commands for working with Power Apps portals (Preview)
PCF	Commands for working with Power Apps component framework
Plugin	Command to create a plug-in project
Solution	Commands for working with Dataverse solution projects
Telemetry	Manages the telemetry settings

Demonstration

Install Power Platform CLI
(MSI) and export the Solution



Power Platform Build Tools for Azure DevOps

Power Platform Build Tools for Azure DevOps

Is a collection of Power Platform–specific build tasks that eliminate the need to manually download custom tooling and scripts to manage the application lifecycle of apps built on Power Platform



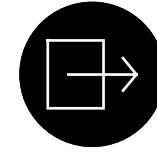
Initiate

Getting started, faster



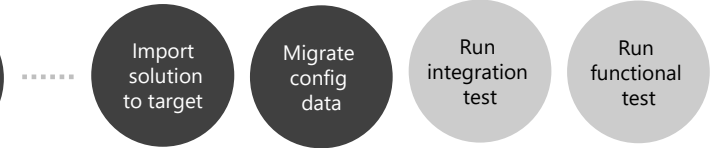
Build

Build and walk away



Release

Automated, predictive, repeatable



Initial pipelines

- Instantiates pristine development environments
- Check solutions into source control

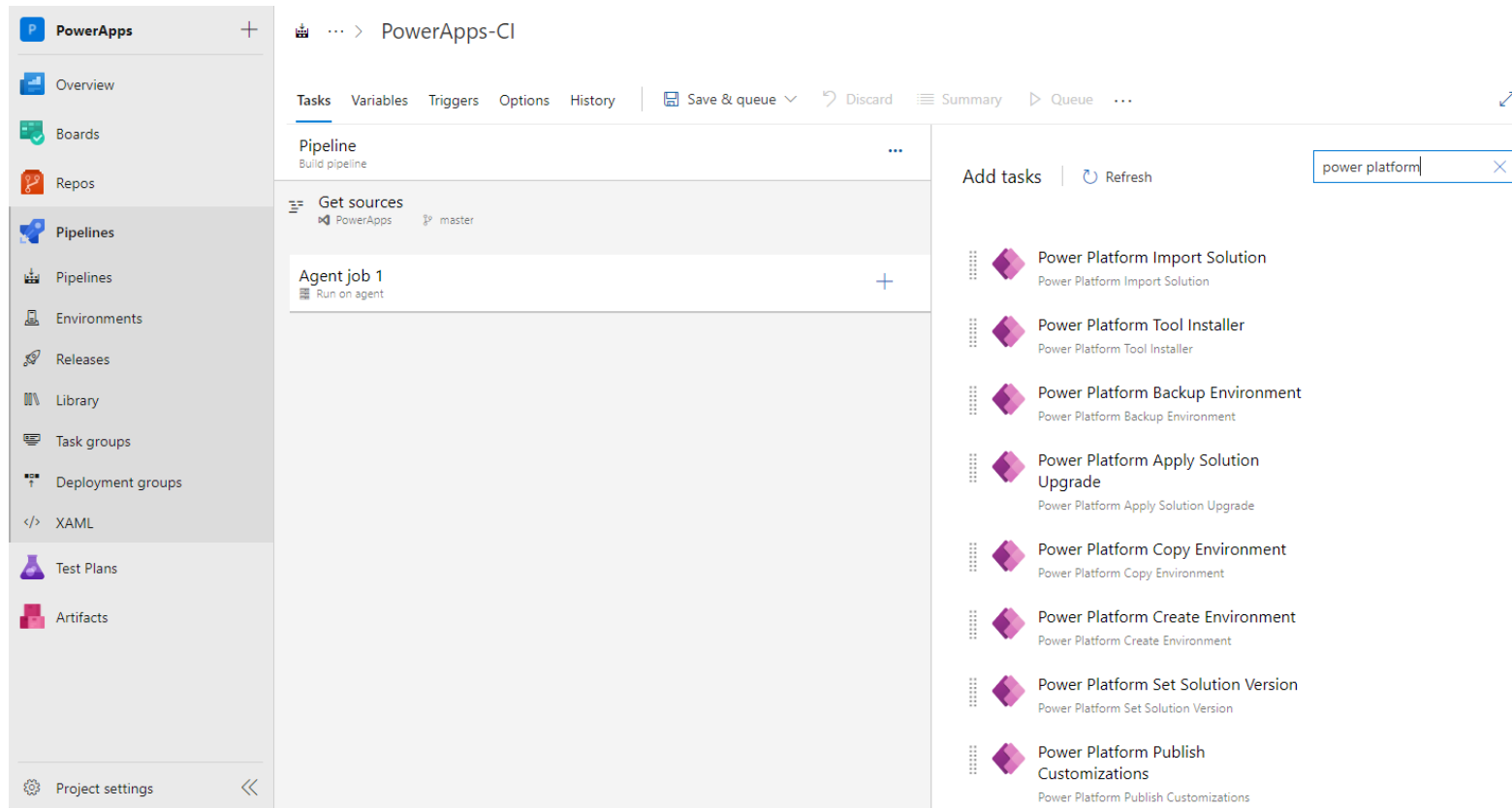
Build pipeline automates manual steps. Removes the need to manually export/import solutions and config data.

Automated release pipeline removes manual steps. Faster deployment becomes the new standard

Get Power Platform Build Tools

Power Platform Build Tools can be installed into your Azure DevOps organization from Azure Marketplace

After installation, all tasks included will be available to add into any new or existing pipeline. You can find them by searching for "Power Platform."

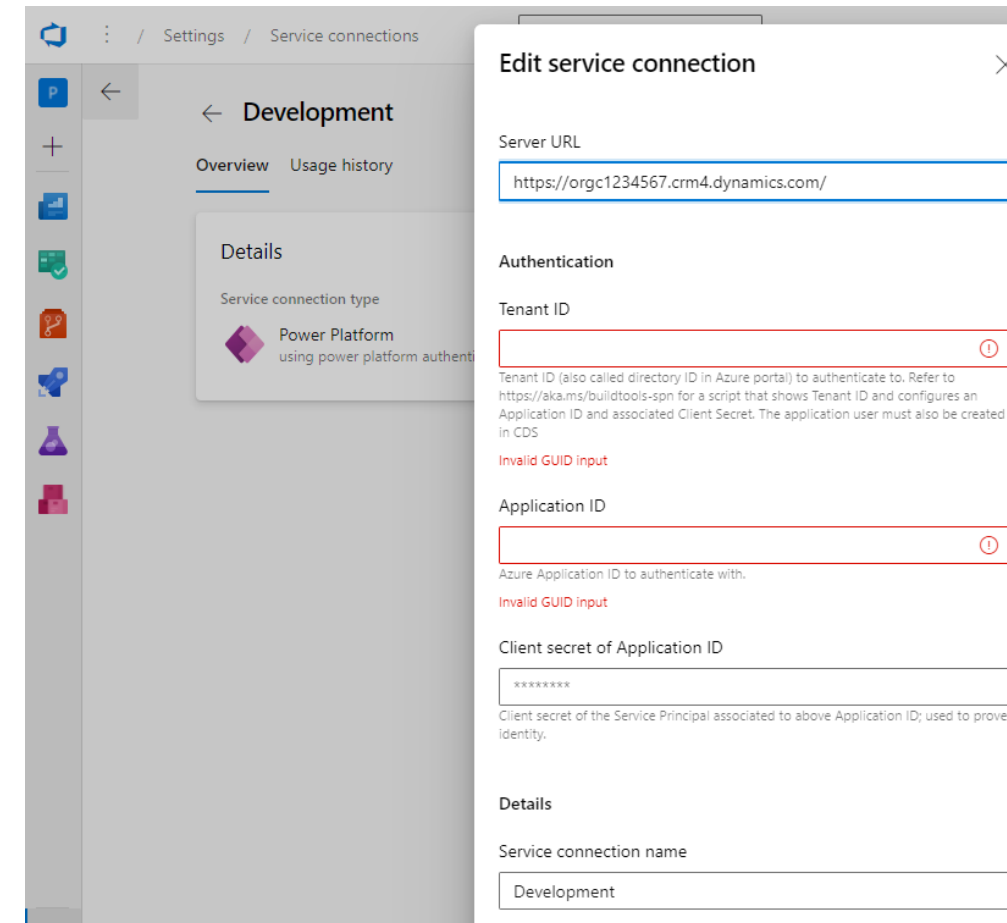


Connection to Power Platform environments

To interact with the Power Platform environment, a connection must be established that enables the various build tool tasks to perform the required actions.

Two types of connections are available:

- **Username/password:** Configured as a generic service connection with username and password. Note that username/password does not support multi-factor authentication.
- **Service principal and client secret:** (recommended) This connection type uses service principal based authentication and supports multi-factor authentication.



The screenshot shows the 'Edit service connection' dialog box in the Power Platform environment. The dialog is titled 'Edit service connection' and has a close button (X) in the top right corner. It contains the following fields and sections:

- Server URL:** A text input field containing the URL 'https://orgc1234567.crm4.dynamics.com/'.
- Authentication:** A section containing:
 - Tenant ID:** A text input field with a red border and an information icon (i) on the right. Below it, a note states: 'Tenant ID (also called directory ID in Azure portal) to authenticate to. Refer to https://aka.ms/buildtools-spn for a script that shows Tenant ID and configures an Application ID and associated Client Secret. The application user must also be created in CDS.' Below the note, a red error message reads 'Invalid GUID input'.
 - Application ID:** A text input field with a red border and an information icon (i) on the right. Below it, a note states: 'Azure Application ID to authenticate with.' Below the note, a red error message reads 'Invalid GUID input'.
 - Client secret of Application ID:** A text input field containing '*****'. Below it, a note states: 'Client secret of the Service Principal associated to above Application ID; used to prove identity.'
- Details:** A section containing:
 - Service connection type:** A dropdown menu showing 'Power Platform using power platform authentication'.
 - Service connection name:** A text input field containing 'Development'.

To configure a connection using service principal, you must first create an application registration in Azure Active Directory (AAD) with the required permissions and then create the associated Application User in the Microsoft Power Platform environment you want to connect to

Azure Pipelines

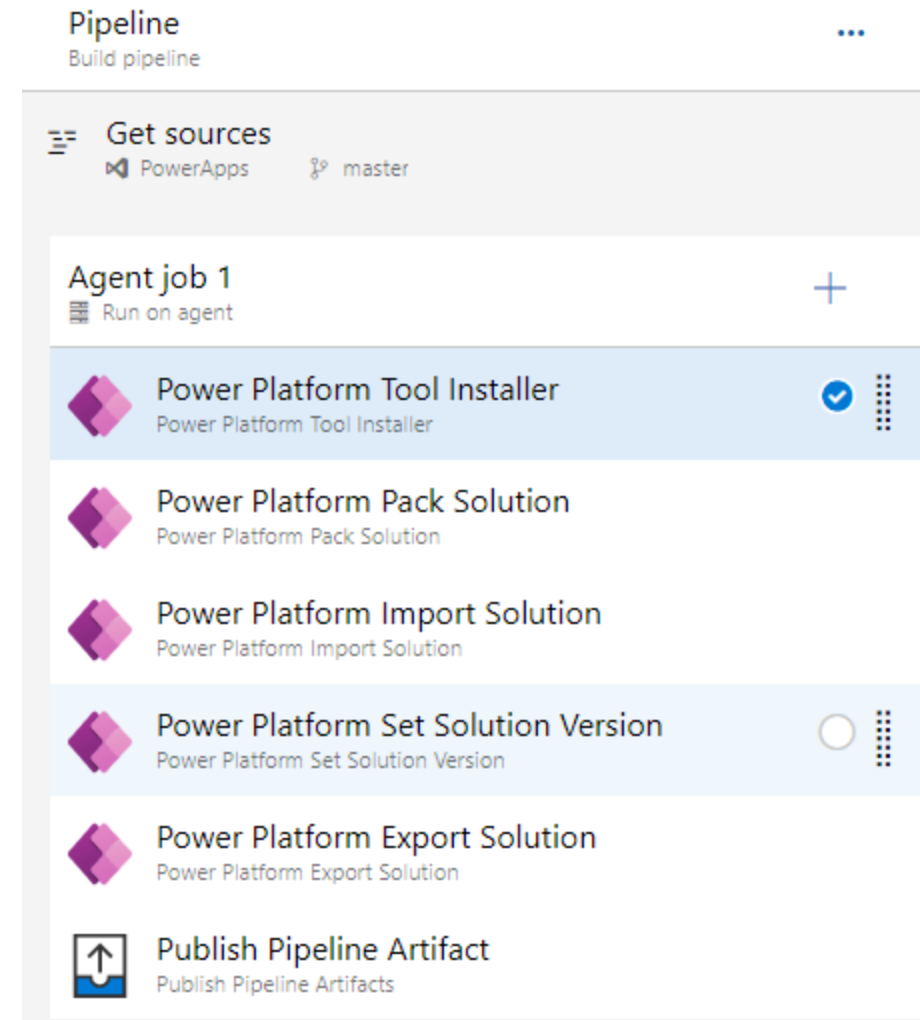
Azure Pipelines automatically builds and tests code projects to make them available to others.

Works with just about any language or project type

Combines Continuous Integration (**CI**) and Continuous Delivery (**CD**) to test and build your code and ship it to any target

- **CI** is the practice used by development teams of automating merging and testing code
- **CD** is a process by which code is built, tested, and deployed to one or more test and production environments

The starting point for configuring CI and CD for your applications is to have your source code in a version control system.



Azure Repos

Azure Repos is a set of version control tools that you can use to manage your code.

Version control systems are software that help you track changes you make in your code over time.

Provides two types of version control:

- **Git:** distributed version control
- **Team Foundation Version Control (TFVC):** centralized version control (should not be used for new projects)

Repository type	Azure Pipelines (YAML)	Azure Pipelines (classic editor)
Azure Repos Git	Yes	Yes
Azure Repos TFVC	No	Yes
GitHub	Yes	Yes
GitHub Enterprise Server	Yes	Yes
Bitbucket Cloud	Yes	Yes
Bitbucket Server	No	Yes
Subversion	No	Yes

Power Platform + GitHub

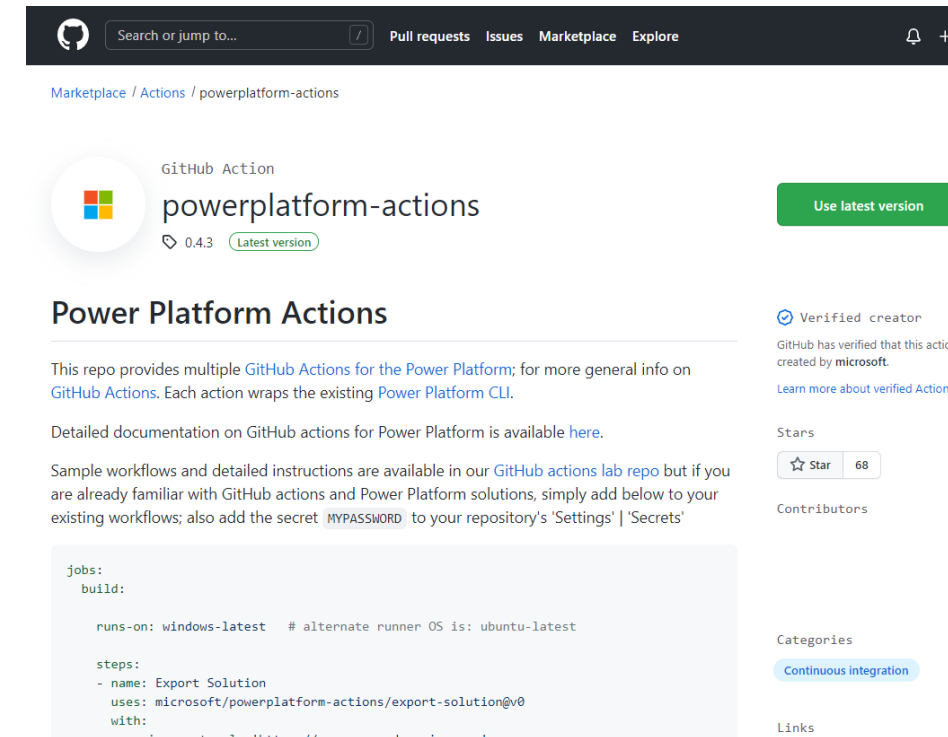
GitHub Actions

- Help you automate tasks within your software development life cycle
- Event-driven meaning that you can run a series of commands after a specified event has occurred
- Actions are defined in workflow definition file (.yml)

The screenshot displays the GitHub Actions interface for a repository. The top navigation bar includes links for Code, Issues, Pull requests, Actions (selected), Projects, Security, Insights, and Settings. On the left, the 'Workflows' section shows 'All workflows' with two listed: 'export-and-branch-solution' (highlighted in blue) and 'release-solution-to-prod'. The main area shows the 'export-and-branch-solution' workflow details, including a search bar with 'is:success' and a list of '3 workflow run results'. Each result shows a green checkmark, the workflow name, and a description like 'export-and-branch-solution #3: Manually run by petepuu'. A 'Run workflow' button is visible. A modal is open on the right, titled 'Use workflow from', with fields for 'Branch: main', 'name of the solution to worked on from Power Platform *' (filled with 'GitApp'), 'folder name for staging the exported solution *do not change* *' (filled with 'out/exported/'), 'staging the unpacked solution folder before check-in *do not change* *' (filled with 'out/solutions/'), and 'folder name to be created and checked in *do not change* *' (filled with 'solutions/'). A green 'Run workflow' button is at the bottom of the modal.

Power Platform + GitHub Actions

- Enable developers to build automated software development lifecycle workflows for Power Platform solutions
- Are supported only for environments with a database
- Similar tools available as in Azure DevOps Build Tools
- You can create workflows in your repository to build, test, package, release, and deploy apps; perform automation; and manage bots and other components built on Microsoft Power Platform
- GitHub Actions for Microsoft Power Platform can run on both Microsoft Windows agents and Linux agents



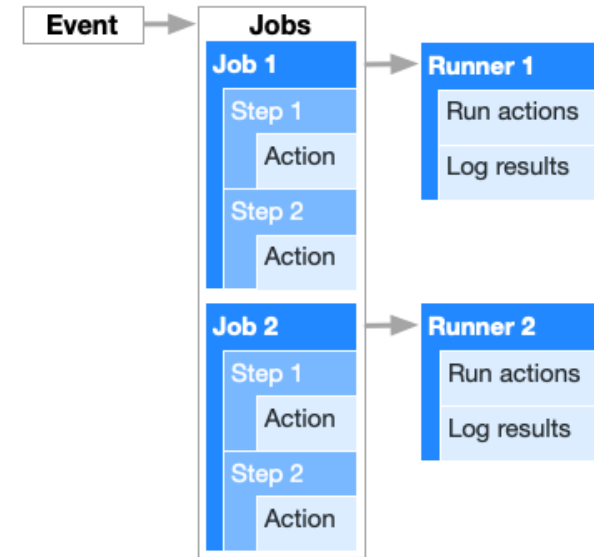
<https://github.com/marketplace/actions/powerplatform-actions>

Components of GitHub Actions

Workflows - Automated procedure which can be used to build, test, package, release, or deploy a project on GitHub. Made up of one or more jobs and can be scheduled or triggered by an event

Events - Is a specific activity that triggers a workflow. For example, when someone pushes a commit to a repository or when an issue or pull request is created

Jobs - Set of steps that execute on the same runner. By default, multiple jobs run in parallel but can also configure to run jobs sequentially. For example, a workflow can have two sequential jobs that build and test code where the test job is dependent on the status of the build job.



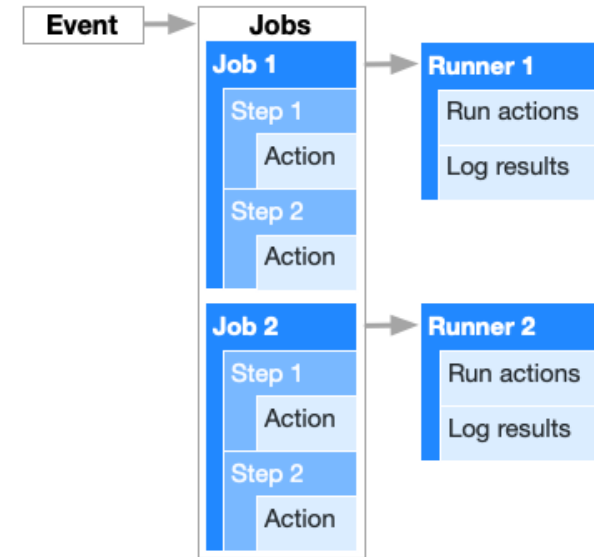
```
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
```

Components of GitHub Actions

Steps - An individual task that can run commands in a job. A step can be either an action or a shell command.

Actions - Standalone commands that are combined into steps to create a job. Actions are the smallest portable building block of a workflow. You can create your own actions or use actions created by the GitHub community.

Runners - Is a server that has the GitHub Actions runner application installed. You can use a runner hosted by GitHub, or you can host your own. A runner listens for available jobs, runs one job at a time, and reports the progress, logs, and results back to GitHub.

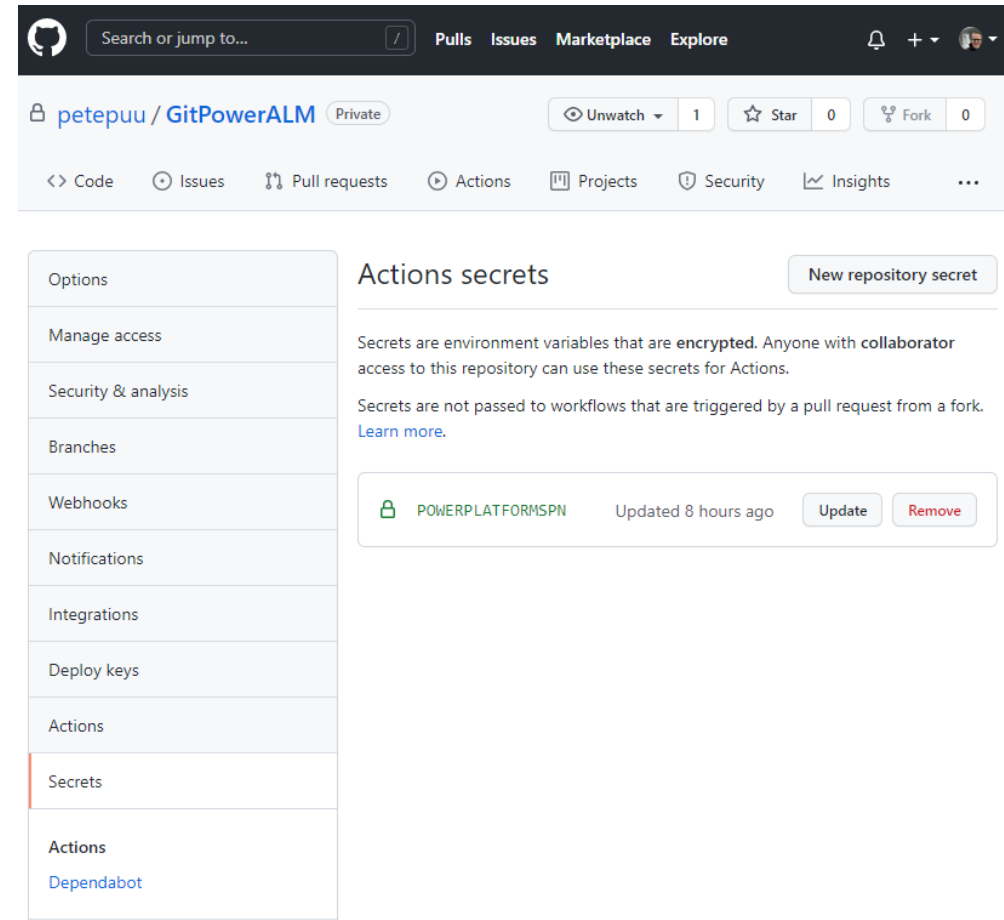


```
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
```

Connection to Power Platform environments

Similar as in Azure DevOps Build Tools you have two options to connect Power Platform environments:

- **Username/password:** Configured as a generic service connection with username and password. Note that username/password does not support multi-factor authentication
- **Service principal and client secret:** (recommended) This connection type uses service principal based authentication and supports multi-factor authentication.



Questions?



**Deploy Resources Between
Environments Using a Package**

**Deploy Resources Between
Environments Using a Solution**



