

# Marketplace & Extensibility documentation

Discover, manage, develop extensions, and widgets for integration with Azure DevOps.

## Discover & manage Marketplace extensions

### CONCEPT

[Learn about Azure DevOps Marketplace](#)

### HOW-TO GUIDE

[Request extensions](#)

[Install extensions](#)

[Manage extension permissions](#)

## Develop extensions

### QUICKSTART

[Develop a web extension](#)

[Extension samples](#)

[Extensibility points](#)

### TUTORIAL

[Create a custom pipelines task](#)

## Integrate with Slack or Microsoft Teams

### HOW-TO GUIDE

[Use Azure Boards with Slack](#)

[Use Azure Pipelines with Slack](#)

[Use Azure Pipelines with Microsoft Teams](#)

## Integrate with Service hooks

---

### OVERVIEW

[Integrate with service hooks](#)

---

### TUTORIAL

[Create a service hook with Microsoft Teams](#)

[Create a service hook with WebHooks](#)

## Building applications on Azure DevOps

---

### OVERVIEW

[Learn about building applications](#)

# Extensions overview

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Extensions are add-ons that you can use to customize and extend your experience with Azure DevOps. They're written using standard technologies such as HTML, JavaScript, and CSS, and can be developed using your preferred development tools.

Extensions are published on the [Visual Studio Marketplace](#), where they can be kept private for you and your team or [shared publicly](#) with millions of developers currently using Azure DevOps.

Extensions use our [RESTful API Library](#) to easily interact with Azure DevOps and other applications/services.

## Understand parts of an extension



The following items make up an extension:

- [JSON manifest file](#): Contains basic info about the extension.
- Discovery assets: Markdown and images that make up the extension's overview and aesthetics in the Marketplace.
- Static files: Contain the logic of the extension, including HTML, JS, and CSS files. Static files are only applicable to contribution-based extensions.

These files and assets get bundled up to make a [VSIX file](#) that gets published to the Marketplace.

From the Marketplace, users can [install extensions](#) directly into their organization. If you don't have permissions to install an extension, but you're a project member, you can [request an](#)

[extension](#) instead.

## Use an extension

There are dozens of ways you can use an extension and places where you can add to the user interface, and we're adding more every sprint. Learn about all of the places where you can add a hub in the [Extensibility points](#).

- Provide new [Azure Pipelines tasks](#) that teams can use in their builds.
- Use [dashboard widgets](#) to get custom views within Azure DevOps.
- Extend the [work item form](#) with new tabs, sections, and actions.
- Create [your own hub](#) to embed new capabilities within our Agile, code, build, and test experiences.
- Develop [actions](#) that can be run on hubs, whether they're ours or ones you created.

## Evaluate a Marketplace extension

To evaluate a Marketplace extension, review the information and resources described in the following table. You can find this information in the extension information

### Information

### Usage

---

#### Top Publisher badge



The publisher demonstrates commitment to its customers and the Marketplace through excellent policies, quality, reliability, and support. For more information, see [Top Publisher](#).

---

### Q & A

The Q & A section of published extensions might answer questions you have. Also, they're a good mechanism to engage with the extension's publishers to have a meaningful dialogue to

make yourself comfortable. Use the Q & A information to understand the development, testing, and security practices the publisher follows. It also gives you a sense of the publisher's responsiveness.

---

## Ratings & reviews

Ratings and reviews indicate how others perceive the offering. For more information, see [Respond to customer feedback](#).

---

## Privacy, license, and support policies

See if the publisher provided them and if they meet your needs or concerns. For more information, go to [Safety information](#).

---

## Safety information

The Marketplace ensures the safety and integrity of extensions through the following measures:

- **Malware scan:** The Marketplace runs a virus scan on each new and updated extension package to ensure its safety. Until the scan is clear, the extension isn't published for public usage. If a concern surfaces, the Marketplace team can disable the extension immediately and notify its existing customers.
- **Content scan:** The Marketplace scans the content of every new and updated extension to avoid surfacing inappropriate or offensive content on the Marketplace pages.
- **Access to approved scopes only:** An extension can only operate within the granted scopes. For example, an extension with read-only permissions on work items can't modify your features and bugs. Azure DevOps web extensions run in a sandboxed browser iframe and can only access Azure DevOps data and APIs approved for the extension. During installation, admins are prompted to approve permissions and scopes. To protect yourself, carefully review the scopes the extension requests.

### Note

If the scopes change for an extension, you must approve the update before it can be applied to your organization or collection.

- **Third-party build and release tasks:** Tasks are implemented as code that executes on an agent machine. Tasks can only access secrets explicitly provided to them (see [variable secrets](#)), but generally have full access to the agent machine itself. To reduce risk, run builds on Microsoft-hosted agents, which are VMs isolated from other jobs and recycled after each job. Alternatively, limit file and network access on private hosted agent machines. Learn more about [build and release agents](#).
- **Third-party code execution on the server:** Extensions can't install or execute any code on Azure DevOps Server.

## Build an extension

Before you build an extension, familiarize yourself with the extension types already available within the Marketplace, [Extensions for Azure DevOps ↗](#). Learn how to build your first extension and check out our full set samples.

- [Develop your first extension](#)
- [Samples](#)

For more information about building extensions, see the following articles:

- [REST APIs](#)
- [Service Hooks](#)
- [Package, publish, and install your extension](#)
- [Package and publish your integration with an external app or service](#)
- [Share your work publicly with the entire community](#)

## Next steps

[Develop an extension](#)

## Related content

- [Visual Studio Marketplace ↗](#)
- [Extension Publisher Page ↗](#)
- [Visual Studio Partner Program ↗](#)
- [Extension manifest reference](#)

# Authentication methods for Azure DevOps

08/27/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

This article describes authentication methods for Azure DevOps integration and helps you choose the best option for your scenario. Modern authentication approaches like Microsoft Entra ID provide enhanced security and the best approach for new applications.

## Important

We recommend Microsoft Entra ID authentication for new applications that integrate with Azure DevOps Services. Use personal access tokens sparingly, and use them only when Microsoft Entra ID isn't available.

OAuth 2.0 and Microsoft Entra ID authentication are available for Azure DevOps Services only, not Azure DevOps Server.

For on-premises scenarios, use [.NET client libraries](#), Windows authentication, or [personal access tokens](#).

## Authentication methods by scenario

Choose the appropriate authentication method based on your application type and requirements.

 Expand table

Application type	Description	Example	Recommended method	Code samples
Web/desktop apps	Interactive applications using current frameworks	React app, .NET desktop app	<a href="#">Microsoft Entra OAuth with the Microsoft Authentication Library (MSAL)</a>	<a href="#">Managed client console app</a>
Service/background apps	Applications running without user interaction	Azure Functions, background services	<a href="#">Service principals and managed identities</a>	<a href="#">Service principals</a>
Legacy client apps	Existing applications using	Console apps with Azure	<a href="#">.NET client libraries with OAuth</a>	<a href="#">Client library console app</a>

Application type	Description	Example	Recommended method	Code samples
	client libraries	DevOps .NET libraries		
Headless/CLI apps	Noninteractive command-line tools	Build scripts, automation tools	Device authorization grant flow	Device profile <a href="#">↗</a>
Azure DevOps extensions	Extensions running within Azure DevOps	Custom dashboard widgets and work item forms	Azure DevOps web extension SDK <a href="#">↗</a>	Add a dashboard widget
Azure DevOps Server apps	On-premises Azure DevOps Server integrations	Custom server extensions	.NET client libraries or Windows Auth	Client library console app <a href="#">↗</a>
Personal/ad hoc scripts	Quick scripts for personal use	PowerShell scripts, curl commands	Personal access tokens	Get started with the REST APIs

## Suggestions for getting started

The following sections provide recommendations for getting started in different scenarios.

### New applications

- Build Azure DevOps integrations with Microsoft Entra OAuth apps for the best security and future compatibility.
- Use service principals or managed identities for service-to-service scenarios.
- Avoid personal access tokens in production applications.

### Existing applications

- Plan migration from personal access tokens to Microsoft Entra ID authentication.
- Consider the [authentication migration timeline](#) [↗](#) for Azure DevOps improvements and reducing the use of personal access tokens.
- Review your current authentication approach against security best practices.

### Azure DevOps Server

- Use .NET client libraries with Windows Authentication when possible.

- Use personal access tokens for Azure DevOps Server scenarios when they're acceptable.
- Plan for future Azure DevOps Services migration to take advantage of modern authentication.

## Answers to common questions

The following sections provide answers to frequently asked questions.

### Should I use Microsoft Entra ID OAuth or personal access tokens?

Use Microsoft Entra ID OAuth in the following scenarios:

- New applications and integrations
- Production workloads that require robust security
- Applications that need enterprise identity integration
- Long-term projects with compliance requirements

Only use personal access tokens in the following scenarios:

- Personal scripts and ad hoc tasks
- Legacy applications during migration planning
- Azure DevOps Server scenarios where modern authentication isn't available

### Should I use service principals or user delegation for authentication?

Use service principals or managed identities in the following scenarios:

- Build applications that operate independently (background services, automation).
- Create apps that don't require user interaction.
- Implement service-to-service communication.
- Build continuous integration and continuous delivery (CI/CD) pipelines or automated workflows.

Use user delegation (OAuth with user consent) in the following scenarios:

- Build applications that act for human users.
- Create interactive apps where users sign in with their own credentials.
- Implement features that require user-specific permissions.
- Build apps that respect users' individual access rights.

# How do I authenticate with both Azure DevOps Services and Azure DevOps Server?

The best practice is to create separate authentication paths:

- **Azure DevOps Services:** Use Microsoft Entra ID OAuth.
- **Azure DevOps Server:** Use .NET client libraries with Windows Authentication or personal access tokens.

Use the `requestContext` method to detect the service type, and apply the appropriate authentication method.

## Why can't my service account access Azure DevOps APIs?

Here are some common issues that can affect service account access:

- **Service account not "materialized":** Use the correct sign-in method. Service accounts need interactive sign-in permissions or proper Microsoft Entra ID registration.
- **Insufficient permissions:** Ensure that the service account has appropriate Azure DevOps permissions.
- **Authentication method:** Use service principals or managed identities instead of trying to authenticate as a service account.

## How do I migrate from personal access tokens to modern authentication?

Follow these steps:

1. Identify current personal access token usage in your applications.
2. Choose an alternate authentication method:
  - Microsoft Entra ID OAuth for user-delegated scenarios
  - Service principals for service-to-service scenarios
3. Update the authentication code by using the [Azure DevOps migration authentication samples](#).
4. Test the changes thoroughly before you remove any personal access token dependencies.
5. Monitor and validate the new authentication method.

## Implementation procedures

After you choose the authentication method for your scenario, finish the implementation:

- **New applications:** [Build Azure DevOps integrations with Microsoft Entra OAuth apps](#)
- **Service applications:** [Use service principals and managed identities in Azure DevOps](#)
- **Personal scripts:** [Use personal access tokens](#)

## Related content

- [OAuth 2.0 for Azure DevOps](#)
- [Azure DevOps Services REST API reference](#)
- [Security and identity in Azure DevOps](#)
- [Azure DevOps data protection overview](#)

ⓘ **Note:** The author created this article with assistance from AI. [Learn more](#)

# Install extensions

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Install, assign, disable, and uninstall extensions, which add new features and capabilities for Azure DevOps.

For more information about extensions, see the [developing](#) and [publishing](#) overviews.

## Prerequisites

Expand table

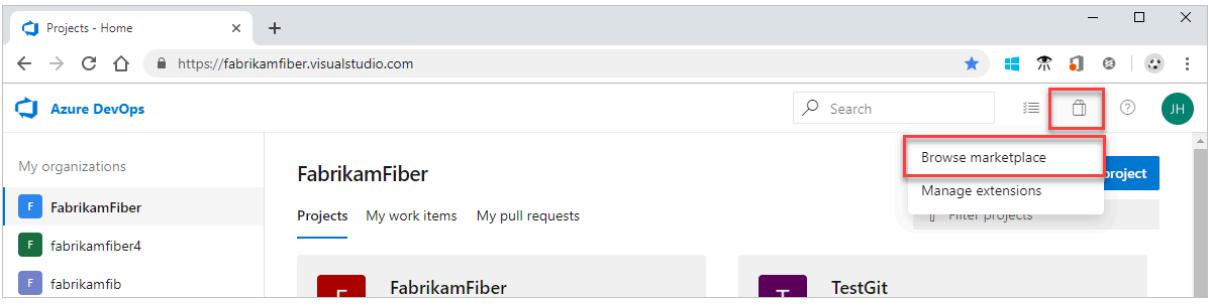
Category	Requirements
Permissions	Member of the Project Collection Administrators group. Organization owners are automatically members of this group. If you don't have permissions, you can <a href="#">request extensions</a> instead or <a href="#">look up a project collection administrator</a> .
Extension sharing	Private extensions <a href="#">shared with your organization</a> .

## Install an extension

Install an extension to your organization by doing the following steps.

Browser

1. Sign in to your organization ([https://dev.azure.com/{Your\\_Organization}](https://dev.azure.com/{Your_Organization})).
2. Select the shopping bag icon, and then select **Browse Marketplace**.



3. Find the extension that you want to install and select **Get it free**.

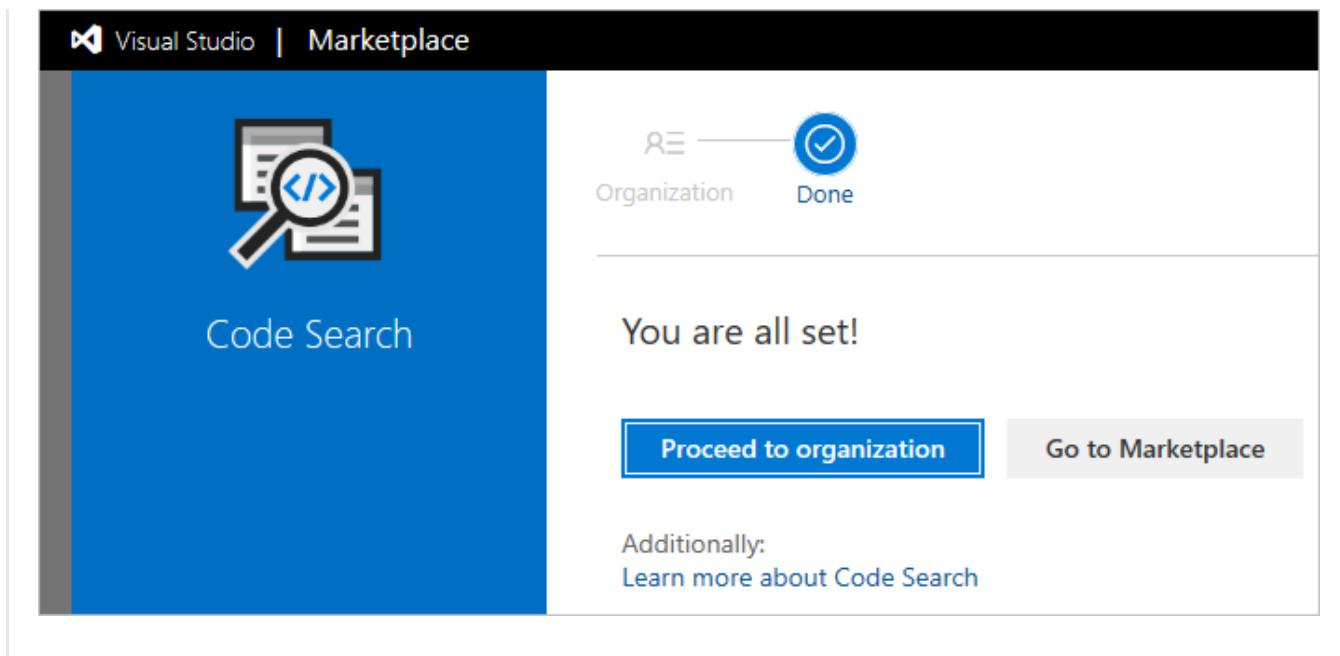
The screenshot shows the Visual Studio Marketplace page for the "Code Search" extension. At the top, there's a large icon of a magnifying glass over code snippets. To the right of the icon, the extension name "Code Search" is displayed in bold. Below it, the developer information "Microsoft" with a blue checkmark, the URL "microsoft.com", the download count "224,515 installs", a rating of "4.5 stars (102)", and the word "Free". A brief description follows: "Code Search provides fast, flexible and accurate search across all your code". A green "Get it free" button is at the bottom.

4. Select your organization from the dropdown menu, and then select **Install** to install the extension.

The screenshot shows the "Select an Azure DevOps organization" step in the Visual Studio Marketplace. On the left, there's a sidebar with the "Code Search" extension logo and name. The main area has a blue header bar with the "Organization" icon and a checked "Done" button. Below this, the text "Select an Azure DevOps organization" is centered. A dropdown menu is open, showing "FabrikamFiber" as the selected item. At the bottom, a blue "Install" button is highlighted with a red border.

- [Why don't I see any organizations?](#)
- [Why can't I install this extension?](#)

Your extension is now installed! You can now go to your organization to use your extension. Also, tell your team about this extension, so they can start using its capabilities.



## High privilege, pipeline decorators, and unpublished extensions

Extensions with high privilege scopes, pipeline decorators, or unpublished status can pose potential security risks if not properly vetted. High privilege scopes grant extensive access to your organization's resources, while pipeline decorators can modify all pipelines in your organization. Unpublished extensions might no longer be maintained by their publishers. For more information on managing these types of extensions, see [Manage high privilege scopes, pipeline decorators, and unpublished extensions](#).

## Uninstall or disable an extension

Browser

1. Sign in to your organization ([https://dev.azure.com/{Your\\_Organization}](https://dev.azure.com/{Your_Organization})).
2. Select **Organization settings**.

The screenshot shows the Azure DevOps interface. On the left, there's a sidebar with 'FabrikamFiber' selected. Below it are 'FabrikamFiber25', '14 more organizations', and 'New organization'. A red box highlights the 'Organization settings' link. The main area is titled 'FabrikamFiber' and shows a project card for 'Fabrikam Fiber'. At the bottom of the main area, there's a 'Delivery Plans' card with 'Uninstall' and 'Marketplace' buttons.

3. Select **Extensions**, and then select the extension that you want to uninstall or disable.

The screenshot shows the 'Organization Settings' page with the 'Extensions' option selected. In the 'Installed extensions' list, the 'Delivery Plans' extension is highlighted with a red box. The 'Delivery Plans' card on the right shows its details, including a 'Uninstall' button.

4. Select **Uninstall** or select the ellipses (...), and then select **Disable**.

The screenshot shows the 'Delivery Plans' extension details page. It includes a 'Uninstall' button and an ellipsis (...) button, both of which are highlighted with red boxes.

## Enable or list extensions through the command line

Enable an extension with the `az devops extension enable` command. To get started, see [Get started with Azure DevOps CLI](#).

#### Azure CLI

```
az devops extension enable --extension-name  
    --publisher-name  
    [--org]
```

## Parameters - enable extension

- **extension-name**: The name of the extension to enable.
- **publisher-name**: The name of the extension publisher.
- **org**: Azure DevOps organization URL. Configure the default organization with `az devops configure -d organization=ORG_URL`. Required if not configured as default or picked up using `git config`. Example: `--org https://dev.azure.com/MyOrganizationName/`.

## Example - enable extension

The following command enables the **Timetracker** extension and shows the result in table format.

#### Azure CLI

```
az devops extension enable --extension-name Timetracker --publisher-name 7pace --  
output table
```

Publisher Id	Extension Id	Name	Version	Last Updated	States
-	-	-	-	-	-
7pace	Timetracker	Timetracker	5.0.1.34507	2019-11-13	none

## List extensions

You can list the extensions that are installed in your organization with the `az devops extension list` command. To get started, see [Get started with Azure DevOps CLI](#).

#### Azure CLI

```
az devops extension list [--include-built-in {false, true}]  
    [--include-disabled {false, true}]  
    [--org]
```

## Optional parameters - list extensions

- **include-built-in**: Include the built-in extensions. Accepted values are *true* (default) and *false*.
- **include-disabled**: Include the disabled extensions. Accepted values are *true* (default) and *false*.
- **org**: Azure DevOps organization URL. You can configure the default organization using `az devops configure -d organization=ORG_URL`. Required if not configured as default or picked up using `git config`. Example: `--org https://dev.azure.com/MyOrganizationName/`.

## Example - list extensions

The following command lists extensions in your organization. It excludes the disabled and built-in extensions, and shows the results in table format.

Azure CLI

```
az devops extension list --include-built-in false --include-disabled false -output table
```

Publisher Id	Extension Id	Name	Version
Last Updated	States	Flags	
ms	vss-analytics	Analytics	
18.160.0.2130149925	2019-11-22	multiVersion, trustee...	trusted
ms	vss-code-search	Code Search	
18.160.0.1640944814	2019-11-22	multiVersion, trustee...	trusted
ms	vss-plans	Delivery Plans	
18.160.0.1266795967	2019-11-25	multiVersion, trustee...	trusted
ms-eswm	dependencytracker	Dependency Tracker	2.1910.12801
2019-10-28	none		
ms-devlabs	workitem-feature-tim...	Feature timeline and...	0.0.357
2019-10-14	none		
AgileParts	gantt	GANTT chart	1.0.79
2019-10-25	none		
gordon-bee...	github	GitHub Widget	0.10.0
2016-03-16	none		
ms-devlabs	vsts-extensions-mult...	Multivalue control	2.2.26
2019-11-15	none		
agile-exte...	product-vision	Product Vision	2.0.6
2019-06-04	none		
mohitbagra	related-workitems	Related Work items	2.0.4
2017-11-12	none		
YodLabs	TagsManager2	Tags Manager	0.9.31
2019-02-04	none		
ms-devlabs	team-calendar	Team Calendar	2.0.15
2019-11-01	none		
ms	vss-testmanager-web	Test Manager for TFS...	

18.160.0.2130893445	2019-11-25	multiVersion, trustee...	trusted
mmanela	vsts-workitem-recent...	Who recently viewed ...	1.0.4
2019-03-22	none		
ottostreif...	wiql-editor	Wiql Editor	2.0.90
2019-06-21	none		
mohitbagra	workitem-checklist	Work item checklist	3.2.4
2019-06-24	none		
mohitbagra	witoneclickactions	Work item form one c...	2.3.2
2018-04-03	none		
ms-devlabs	WorkItemVisualizatio...	Work Item Visualizat...	1.4.64
2018-04-03	none		

## List extension information

You can list the details about an extension with the `az devops extension show` command. To get started, see [Get started with Azure DevOps CLI](#).

### Azure CLI

```
az devops extension show --extension-name
                        --publisher-name
                        [--org]
```

## Parameters - list extension information

- **extension-name**: The name of the extension.
- **publisher-name**: The name of the extension publisher.
- **org**: Azure DevOps organization URL. You can configure the default organization using `az devops configure -d organization=ORG_URL`. Required if not configured as default or picked up using `git config`. Example: `--org https://dev.azure.com/MyOrganizationName/`.

## Example - list extension information

The following command shows information about the **Timetracker** extension in table format.

### Azure CLI

```
az devops extension show --extension-name Timetracker --publisher-name 7pace --
output table
```

Publisher Id	Extension Id	Name	Version	Last Updated	States
-	-	-	-	-	-

# Troubleshoot extension installation

To resolve common issues, follow these troubleshooting steps:

- **Extension fails to install:**
  - **Check permissions:** To install extensions, ensure you're a Project Collection Administrator or have the necessary permissions granted by an administrator.
  - **Verify extension compatibility:** Ensure the extension is compatible with your version of Azure DevOps. Check the extension's details page for compatibility information.
  - **Network issues:** Verify that your network connection is stable and that there are no firewall or proxy settings blocking the installation process.
- **Extension crashes or causes errors:**
  - **Collect diagnostic information:** If an extension crashes or causes errors, collect diagnostic information to help identify the issue. This information includes error messages, logs, and screenshots of the problem.
  - **Check extension logs:** Some extensions provide logs that can help diagnose issues. Check the extension's documentation for information on how to access and interpret these logs.
  - **Contact support:** If you can't resolve the issue, contact the extension's support team with the collected diagnostic information. Provide as much detail as possible to help them troubleshoot the problem.
- **Extension not visible after installation:**
  - **Refresh the page:** Sometimes, the extension might not appear immediately after installation.
  - **Check permissions:** Ensure you have the necessary permissions to view and use the extension. Some extensions might require specific permissions to be visible.
  - **Reinstall the extension:** If the extension still isn't visible, try uninstalling and reinstalling it.

## Next steps

[Manage extension permissions](#)

## Related content

- Request and approve extension requests
- Develop a web extension
- Review Azure Billing and Subscription FAQs ↗
- Access Azure billing support ↗
- Get Visual Studio subscriptions support ↗

ⓘ Note: The author created this article with assistance from AI. [Learn more](#)

# Request and approve extensions

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

If you don't have permissions to install extensions, you can request extensions instead. When another project member requests an extension, Project Collection Administrators receive an email notification. When a request gets approved, Azure DevOps automatically installs the extension.

## Prerequisites

[ ] [Expand table](#)

Category	Requirements
Permissions	<ul style="list-style-type: none"><li>- To request extensions: <b>Contributor</b> for your organization.</li><li>- To approve extensions: Member of the <b>Project Collection Administrators</b> group and <a href="#">Edit collection-level information permissions</a>.</li></ul>

## Request an extension

1. Sign in to your organization ([https://dev.azure.com/{Your\\_Organization}](https://dev.azure.com/{Your_Organization})).
2. Select  [Organization settings](#).

The screenshot shows the Azure DevOps interface for the 'FabrikamFiber' organization. On the left sidebar, there are links for 'FabrikamFiber', 'FabrikamFiber25', '14 more organizations', 'New organization', and 'Organization settings'. The 'Organization settings' link is highlighted with a red box. The main content area displays the organization's name 'FabrikamFiber' and a summary card for 'Fabrikam Fiber'.

### 3. Select Extensions > Browse marketplace.

The screenshot shows the 'Extensions' page within the organization settings. The left sidebar has a 'General' section with 'Extensions' selected. The main area shows a list of installed extensions, with a 'Browse marketplace' button highlighted by a red box.

Extension	Provider	Description
SARIF SAST Scans Tab	Microsoft DevLabs	Adds a 'Scans' tab to each Build Result and Work Item for viewing associated SARIF SAST logs.
1ES GPT	Microsoft	1ES PT helper extension for running 1ES PT executables. Please visit <a href="https://aka.ms/1esp">https://aka.ms/1esp</a> for more details.
Accessibility Insights for Azure DevOps (Microsoft-Internal)	Accessibility Insights	Scan accessibility issues in an Azure DevOps pipeline
Advanced Security Build Tasks	Microsoft	A set of build tasks for Advanced Security that analyze a repository by running in a pipeline.
Azure DevTest Labs Tasks	Microsoft	Collection of Azure Pipelines tasks to interact with Azure DevTest Labs.
Build Security Monitoring	1ES Security Monitoring	Enables Build Security Monitoring for macOS build agents to meet EO requirements. Opt in with variable: EOCompliance-mac=true. Support @ <a href="https://aka.ms/1esbot">https://aka.ms/1esbot</a>

### 4. Select an extension to install.

### 5. If you don't have permission to install the extension, you can request it.

Review your requests after the Marketplace sends the request to your Project Collection Administrator.

Your requests appear on the **Extensions** page, **Requested** tab.

The screenshot shows the 'Organization Settings' page in Azure DevOps. The top navigation bar includes the Azure DevOps logo, the organization name 'FabrikamFiber', and the current page path 'Organization Settings / Extensions'. On the left, a sidebar lists 'General' settings like Overview, Users, Billing, Auditing, Global notifications, Usage, and Extensions, with 'Extensions' highlighted by a red box. The main content area is titled 'Extensions' and contains three tabs: 'Installed', 'Requested' (which is underlined and has a red box around it), and 'Shared'. The 'Requested' tab is currently selected.

To approve extensions, have [edit collection-level information](#) permissions.

# Approve an extension request

1. Go to your Azure DevOps project,  
`https://dev.azure.com/{Your_Organization}/{Your_Project}.`
  2. Select the shopping bag icon, and then **Manage extensions**

A screenshot of the Azure DevOps interface. At the top, there's a browser-style header with the URL https://dev.azure.com/fabrikamfiber/\_projects. Below the header, the main navigation bar includes the Azure DevOps logo, a search icon, a filter icon, and a 'Manage extensions' button, which is highlighted with a red box. On the left, there's a sidebar with 'My organizations' and a selected 'Fabrikam Fiber' item. The main content area features the title 'Fabrikam Fiber'. At the bottom, there are links for 'Projects', 'My Work Items', and 'Filter projects'.

3. Review and approve your requested extensions.

After you approve extension requests, the extensions are automatically installed.

#### 4. [Assign the extensions](#) to users who need access.

Tell your team about your installed extensions, so they can start using their capabilities.

## Related content

- [Review FAQs](#)
- [Set up billing](#)
- [Explore Azure DevOps pricing ↗](#)
- [Assign access levels by group membership](#)

# Manage extension permissions

07/17/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

This article helps you understand how to grant and manage permissions for users or groups, enabling them to manage extensions effectively. By setting the appropriate permissions, you can maintain control over the extensions used within your organization and ensure compliance with your security policies.

## Prerequisites

 Expand table

Category	Requirements
Permissions	Member of the Project Collection Administrators group. Organization owners are automatically members of this group. If you don't have permissions, you can <a href="#">request extensions</a> instead or <a href="#">look up a project collection administrator</a> .
Extension sharing	Private extensions <a href="#">shared with your organization</a> .

## Manage permissions

1. Sign in to your organization ([https://dev.azure.com/{Your\\_Organization}](https://dev.azure.com/{Your_Organization})).
2. Select  [Organization settings](#).

The screenshot shows the Azure DevOps interface. At the top, there's a header bar with the title "Projects - Home" and a URL "dev.azure.com/FabrikamFiber01". Below the header is the "Azure DevOps" logo. On the left, a sidebar lists organizations: "FabrikamFiber01" (selected and highlighted in green), "fabrikamfiber02", and "fabrikamffiber". It also includes links for "14 more organizations" and "New organization". A "What's new" section highlights "Sprint 162 release notes" with a message about the sprint burndown being back. At the bottom of the sidebar, the "Organization settings" link is highlighted with a red box. The main content area is titled "FabrikamFiber01" and contains tabs for "Projects", "My work items", and "My pull requests". A card for the project "Fabrikam Fiber" is shown, featuring a blue square icon with "FF" and the project name.

3. Select **Extensions**.

Azure DevOps fabrikamprime / Settings / Extensions / Installed

Search

Organization Settings fabrikamprime

General

- Overview
- Projects
- Users
- Billing
- Auditing
- Global notifications
- Usage
- Extensions**
- Microsoft Entra

Extensions

Installed Requested Shared

- SARIF SAST Scans Tab by Microsoft DevLabs
- 1ES GPT by Microsoft
- Accessibility Insights for Azure DevOps (Microsoft-Internal) by Accessibility Insights
- Advanced Security Build Tasks by Microsoft
- Azure DevTest Labs Tasks by Microsoft
- Build Security Monitoring by 1ES Security Monitoring

#### 4. Select Security.

FabrikamFiber / Organization Settings / Extensions / Installed

Search

Extensions

Installed Requested Shared

- Analytics by Microsoft

Gain insights into the health and status of your Azure DevOps Server projects.

Not Supported: Team Foundation Server 2018 and prior versions.

#### 5. Add users or update permission settings.

Security

Add Inheritance

User	Role	Access
[FabrikamFiber]\Project Collection Administrat	Manager	Assigned

By following these steps, you can effectively manage extension permissions within your Azure DevOps organization. This ensures that only authorized users can install, update, or remove extensions, maintaining the security and integrity of your development environment.

## Related content

- [Install extensions](#)

- Request extensions
- Uninstall or disable extensions
- About permissions

# Manage high privilege scopes, pipeline decorators, and unpublished extensions

07/17/2025

Extensions in Azure DevOps enhance functionality and streamline workflows, but some extensions might pose security vulnerabilities due to their high privilege scopes or unpublished status. This article explains how to identify and manage high privilege, pipeline decorators, and unpublished extensions to protect your Azure DevOps organization from potential security vulnerabilities or unexpected behavior.

## What are high privilege scopes and high privilege extensions?

### High privilege scopes

Scopes determine in general which resources an extension can access and the operations permitted to perform on those resources. Extensions might use multiple scopes.

As for what is defined as a high privilege scope, it's a scope that is overly permissive.

For example a high privilege scope can:

- Read, update, and delete your source code
- Read, write, and manage your identities and groups
- Create, read, update, and delete your projects

For the full list of scopes, including the high privilege scopes, see the [Manifest reference](#).

### High privilege extensions

High privilege extensions make use of one or more high privilege scopes. As high privilege extensions can access sensitive resources and perform critical operations, it's essential to evaluate them carefully to ensure they align with your organization's security and operational standards.

When it comes to any extension, and even more a high privilege extension, consider the following elements:

- **Trusted publisher:** Install and use extensions only if you trust their code and publisher

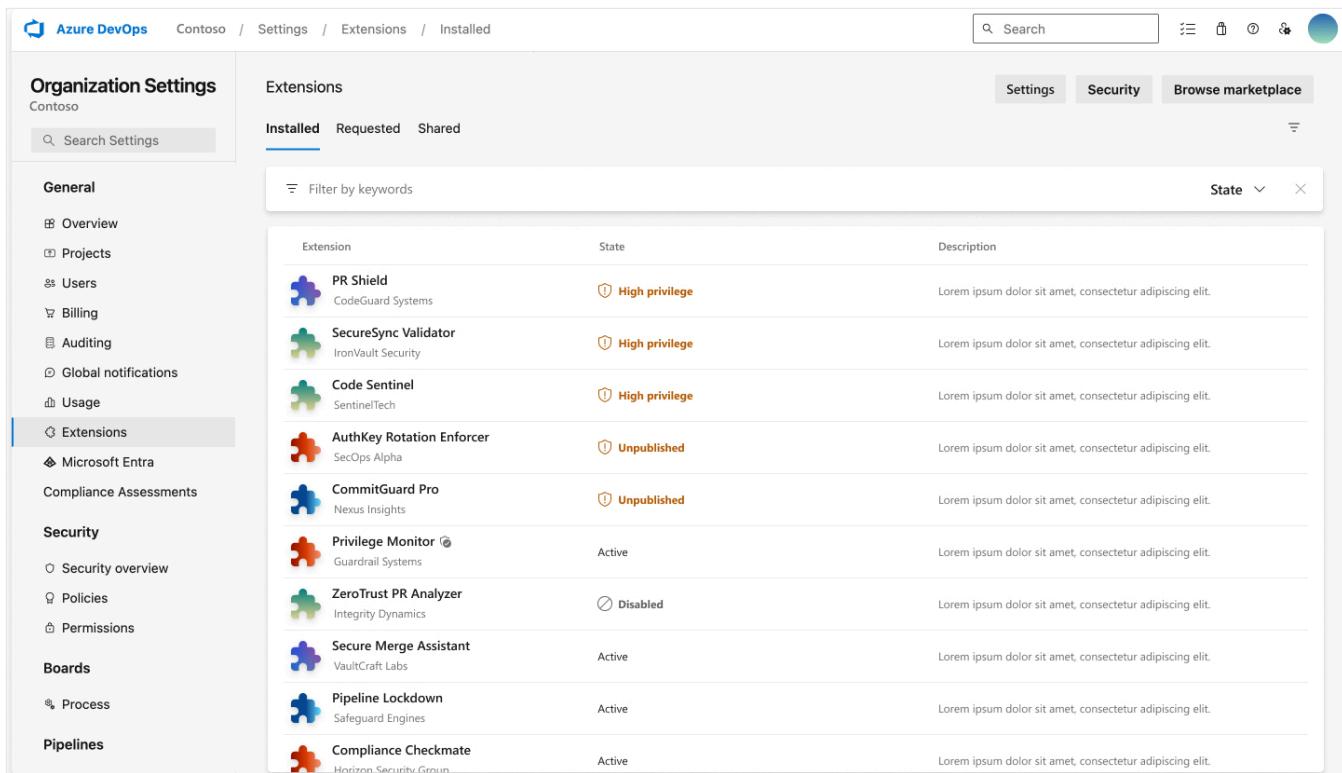
- **Review the requested scopes:** Ensure the requested scopes are necessary for the extension's functionality
- **Limit usage:** Install high privilege extensions only if they're critical to your workflows

# Evaluate the usage of high privilege scopes in Azure DevOps extensions

Few of your already installed extensions might be flagged for high privilege scope usage. You can check their state in the **Extensions** section of **Organization settings**.

We recommend that you only install, update, or use extensions if you trust their code and their publishers.

Microsoft runs virus scans on each new and updated version of an extension; yet this feature only highlights in the user interface whether a specific extension uses high privilege scopes. For more information on the virus scans, see [Publish your extension](#).



The screenshot shows the Azure DevOps Organization Settings page for the 'Contoso' organization. The left sidebar is collapsed, and the main content area is titled 'Extensions'. At the top of the list, there are three tabs: 'Installed' (selected), 'Requested', and 'Shared'. Below the tabs is a search bar labeled 'Filter by keywords' and a dropdown menu labeled 'State'. The table lists ten extensions, each with a small icon, the extension name, its state (e.g., 'High privilege' or 'Unpublished'), and a brief description. The 'PR Shield' extension is highlighted with a yellow warning icon and the text 'High privilege' next to it.

Extension	State	Description
PR Shield CodeGuard Systems	High privilege	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
SecureSync Validator IronVault Security	High privilege	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Code Sentinel SentinelTech	High privilege	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
AuthKey Rotation Enforcer SecOps Alpha	Unpublished	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
CommitGuard Pro Nexus Insights	Unpublished	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Privilege Monitor Guardrail Systems	Active	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
ZeroTrust PR Analyzer Integrity Dynamics	Disabled	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Secure Merge Assistant VaultCraft Labs	Active	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Pipeline Lockdown Safeguard Engines	Active	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Compliance Checkmate Horizon Security Group	Active	Lorem ipsum dolor sit amet, consectetur adipiscing elit.

## Manage extensions with high privilege scopes

If you identify an extension with high privilege scopes, assess whether the called scopes from the extension are essential for your use case. If the extension's functionality doesn't justify the scopes, we recommend not installing or using the extension to safeguard your Azure DevOps organization.

## Extension details

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Publisher

PR Shield

Installed version

1.2.63 (Latest)

last updated

Jan 16, 2018 at 11:11 AM PST

Scopes

Work items (read)

**Work items (read and write) ⓘ**

User profile (read)

Code (read)

**Code (read and write)**

 This extension requires high privilege scopes. Keep this extension in your organization only if you trust the extension's publisher and its code. [Learn more](#)

The Visual Studio Marketplace for Azure DevOps extensions provides similar indications to those extensions shown in the admin page for high privilege scopes. So you can also identify high privilege scopes flagged in the [Azure DevOps Visual Studio Marketplace](#) before the extension is installed in your organization.

When you select any extension and especially an extension with high privilege scopes, think critically whether the extension's functionality justifies the use of these scopes. Only proceed with installation if you trust the publisher and the extension's code.



Project Health

83  
Organization Done

← Review extension details

**Publisher**

Some Thirdparty

**Last update**

Jan 16, 2018 at 6:11 AM PST

**Scopes**

Code (read)  
Work items (read and write)  
Test management (read)  
Release (read)

**Pipeline decorator**

Yes

Extensions that contain a pipeline decorator are considered high privilege, as they can modify all pipelines within your organization. Keep this extension only if you trust the extension's publisher and its code. [Learn more](#)

Select an Azure DevOps organization

test-1



Install

## Use pipeline decorators safely

[Pipeline decorators](#) are private extensions that modify and enhance all pipelines within your organization, they're also classified as high privilege extensions. Use pipeline decorator extensions cautiously and only if you trust their publishers and code.

## Authorize {extension name}

A new version of {extension name} is available. Review the changes in permissions required:

New:

Code (read) 

Removed:

Test management (read)

Unchanged:

Code (read)

Work items (read)

User profile (read)

 This extension requires high privilege scopes, which may introduce security risks to your organization. Install this extension only if you trust the extension's publisher and its code. [Learn more](#)

By clicking Authorize, you authorize this extension on behalf of all users in this organization.

[Authorize](#)

[Cancel](#)

## Discontinue use of unpublished extensions

Beyond high privilege extensions, the extension's administration page visually indicates whether an extension has been unpublished by its publisher.

When an extension is unpublished from the [Visual Studio Marketplace](#) by its publisher, it typically signifies that the extension is no longer maintained.

Discontinue the use of unpublished extensions by uninstalling them from your Azure DevOps organization.

Additionally, with the Azure DevOps Services REST API [version 7.2](#), the string field `unpublished` is now available. This field enables you to programmatically identify extensions that are unpublished from the Visual Studio Marketplace. And for example, you can build your own process of detecting and managing unpublished extensions within your Azure DevOps organization.

## Related content

- [Secure your Azure DevOps environment](#)
- [Publish extensions to the Visual Studio Marketplace](#)
- [Add pipeline decorators](#)

- Reference the extension manifest

# Develop a web extension

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Use extensions to enhance Azure DevOps with new web experiences, dashboard widgets, build tasks, and more. You can develop extensions using standard technologies like HTML, JavaScript, and CSS. This tutorial guides you through creating a web extension for Azure DevOps.

## 💡 Tip

[Explore the extension samples](#) and the newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Prerequisites

expand Expand table

Category	Requirements
Permissions	Owner of the organization.
Tools	<ul style="list-style-type: none"><li>- <a href="#">Node.js</a></li><li>- Extension packaging tool: Run <code>npm install -g tfx-cli</code> from a command prompt, which you use to <a href="#">package your extension</a> later on.</li></ul>

## Create a directory and manifest

An extension is composed of a set of files that includes a required manifest file. You package it into a .vsix file and publish to the Visual Studio Marketplace.

1. Create a directory to hold the files needed for your extension:

```
mkdir my-first-extension
```

2. Initialize a new npm package manifest from this directory:

```
npm init -y
```

This file describes the libraries required by your extension.

3. Install the Azure DevOps Extension SDK package and save it to your npm package manifest:

```
npm install azure-devops-extension-sdk --save
```

This SDK includes a JavaScript library that provides APIs required for communicating with the page your extension is embedded in.

4. Create an extension manifest file named `vss-extension.json` at the root of your extension directory with the following content:

JSON

```
{
  "manifestVersion": 1,
  "id": "my-first-extension",
  "publisher": "",
  "version": "1.0.0",
  "name": "My First Extension",
  "description": "A sample Visual Studio Services extension",
  "public": false,
  "categories": ["Azure Repos"],
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services"
    }
  ],
  "contributions": [
    {
      "id": "my-hub",
      "type": "ms.vss-web.hub",
      "targets": [
        "ms.vss-code-web.code-hub-group"
      ],
      "properties": {
        "name": "My Hub",
        "uri": "my-hub.html"
      }
    }
  ],
  "files": [
    {
      "name": "index.html"
    }
  ]
}
```

```
        "path": "my-hub.html",
        "addressable": true
    },
    {
        "path": "node_modules/azure-devops-extension-sdk",
        "addressable": true,
        "packagePath": "lib"
    }
]
}
```

### ⓘ Important

The `public` property controls whether the extension is visible to everyone on the Visual Studio Marketplace. Keep your extensions private during development.

5. Create a file named `my-hub.html` at the root of your extension directory with the following content, which is for the view (also known as a hub) contributed into the web experience.

#### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/require.js/2.3.6/require.min.js">
</script>
    <script>
        window.requirejs.config({
            enforceDefine: true,
            paths: {
                'SDK': './lib/SDK.min'
            }
        });
        window.requirejs(['SDK'], function (SDK) {
            if (typeof SDK !== 'undefined') {
                console.log("SDK is defined. Trying to initialize...");
                SDK.init();
                SDK.ready().then(() => {
                    console.log("SDK is ready");
                    document.getElementById("name").innerText =
SDK.getUser().displayName;
                });
            } else {
                console.log('SDK is not defined');
            }
        });
    </script>
    <style>
        body {
```

```
        background-color: rgb(0, 67, 117);
        color: white;
        margin: 10px;
        font-family: "Segoe UI VSS (Regular)", "-apple-
system", BlinkMacSystemFont, "Segoe UI", sans-serif;
    }
</style>
</head>
<body>
    <h1>Hello, <span id="name"></span></h1>
</body>
</html>
```

6. Your extension directory should look like the following example.

```
|-- my-hub.html
|-- node_modules
|   |-- @types
|       |-- azure-devops-extension-sdk
|-- package.json
|-- vss-extension.json
```

Need help? Post questions to the [Azure DevOps Services Developer Community](#).

## Next steps

[Package and publish your extension](#)

## Related content

- [Developer Formula Design System](#)
- [Contribution model](#)

# Extensibility points

10/03/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

Extensions add capabilities to the Azure DevOps UI and REST surface. This article lists the most common extensibility points you can target and shows the IDs you use in your extension manifest. For an overview of the extension model and contribution patterns, see the [Contribution model](#).

## 💡 Tip

If you're starting a new Azure DevOps extension, try these maintained sample collections first—they work with current product builds and cover modern scenarios (for example, adding tabs on pull request pages).

- Azure DevOps extension sample (GitHub)—a compact starter sample that demonstrates common extension patterns: <https://github.com/microsoft/azure-devops-extension-sample> ↗
- Azure DevOps extension samples (legacy collection and contributions guide)—install to inspect UI targets, or view the source:  
<https://marketplace.visualstudio.com/items/ms-samples.samples-contributions-guide> ↗ and <https://github.com/Microsoft/vso-extension-samples/tree/master/contributions-guide> ↗
- Microsoft Learn samples (browse Azure DevOps samples)—curated, up-to-date samples across Microsoft docs: /samples/browse/?terms=azure%20devops%20extension

If a sample doesn't work in your organization, install it into a personal or test organization and compare the extension manifest's target IDs and API versions with the current docs. For reference and APIs, see:

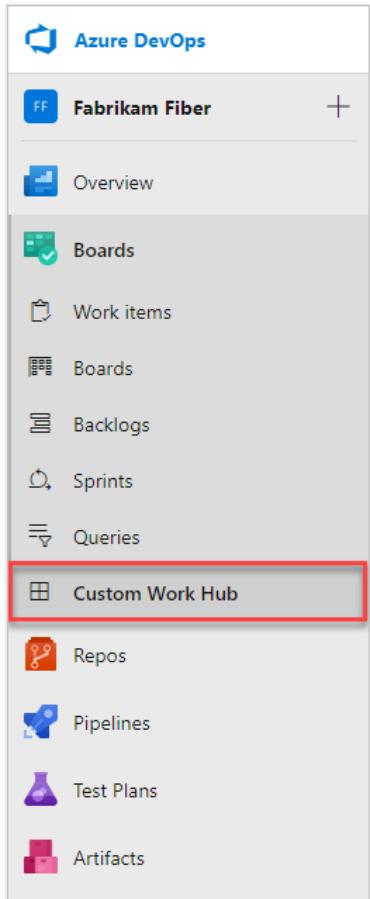
- [azure-devops-extension-api](#)
- [azure-devops-extension-sdk](#)
- [installed extension API](#)

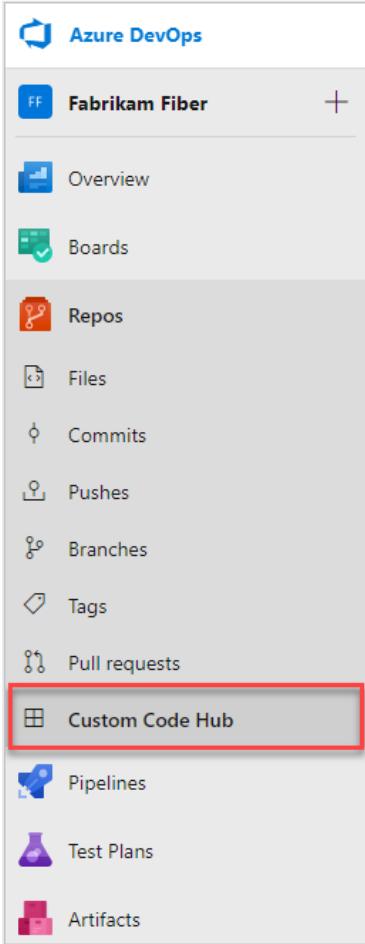
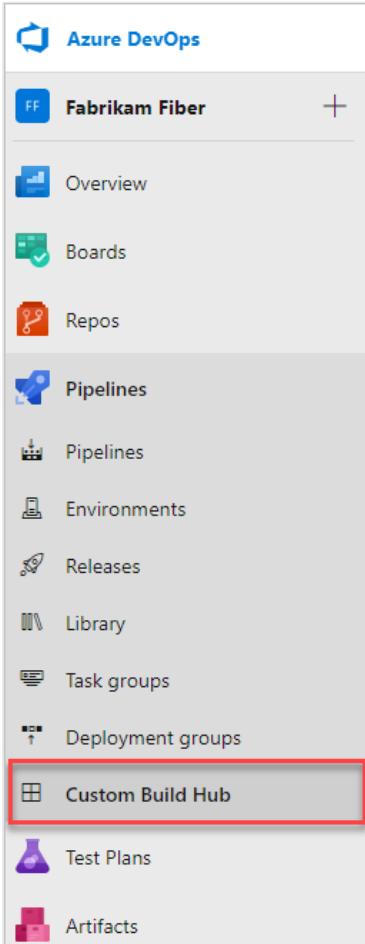
## Hubs and hub groups

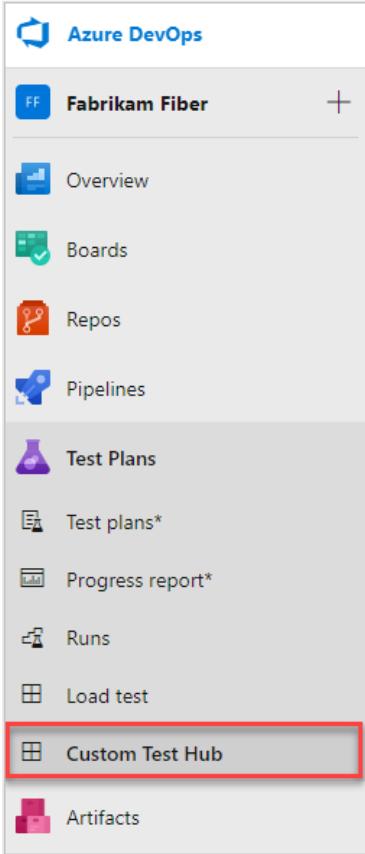
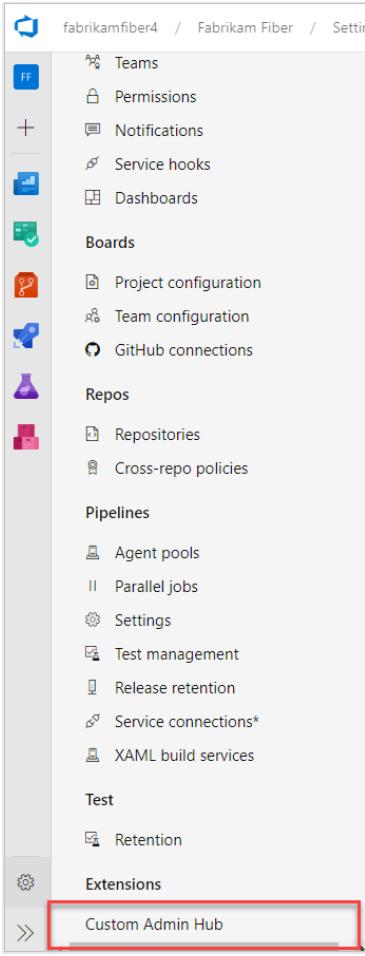
Hubs and hub groups provide primary navigation in Azure DevOps (for example, **Files**, **Releases**, **Backlogs**, **Queries**). A hub belongs to a hub group; for example, the **Files** hub belongs to the project-level **Azure Repos** hub group. Hub groups can exist at the organization/collection level or at the project level. Most extensions contribute at the project level.

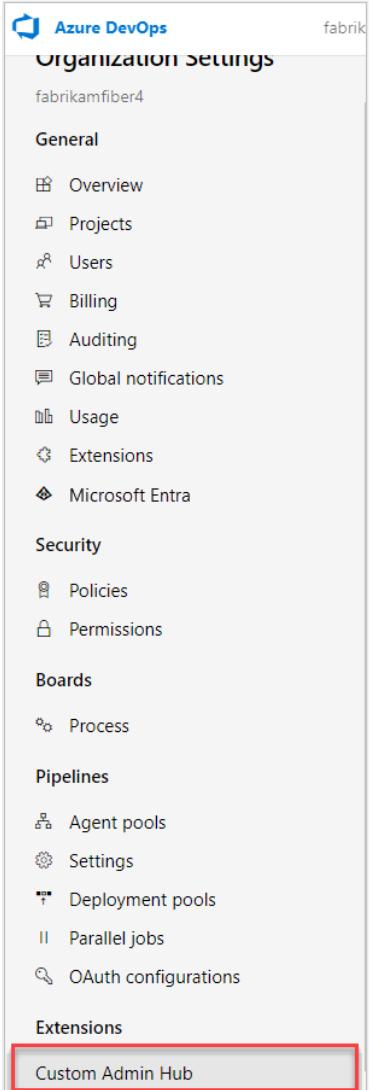
The following table lists common hub groups and their contribution IDs.

 Expand table

Name	ID	Level	Preview image
Azure Boards	ms.vss-work-web.work-hub-group	Project/team	

Name	ID	Level	Preview image
Azure Repos	ms.vss-code-web.code-hub-group	Project/team	 <p>The screenshot shows the Azure DevOps interface for a project named "Fabrikam Fiber". The left sidebar lists various project navigation items: Overview, Boards, Repos, Files, Commits, Pushes, Branches, Tags, Pull requests, Pipelines, Test Plans, and Artifacts. The "Custom Code Hub" item is highlighted with a red box.</p>
Azure Pipelines	ms.vss-build-web.build-release-hub-group	Project/team	 <p>The screenshot shows the Azure DevOps interface for a project named "Fabrikam Fiber". The left sidebar lists various project navigation items: Overview, Boards, Repos, Pipelines, Pipelines (highlighted with a gray background), Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The "Custom Build Hub" item is highlighted with a red box.</p>

Name	ID	Level	Preview image
Azure Test Plans	ms.vss-test-web.test-hub-group	Project/team	 <p>The screenshot shows the 'Test Plans' section of the Azure DevOps interface. It includes links for 'Test plans*', 'Progress report*', 'Runs', 'Load test', and 'Custom Test Hub'. The 'Custom Test Hub' link is highlighted with a red box.</p>
Project settings	ms.vss-web.project-admin-hub-group	Project	 <p>The screenshot shows the 'Project configuration' section of the Azure DevOps interface. It includes links for 'Teams', 'Permissions', 'Notifications', 'Service hooks', 'Dashboards', 'Boards', 'Project configuration', 'Team configuration', 'GitHub connections', 'Repos', 'Repositories', 'Cross-repo policies', 'Pipelines', 'Agent pools', 'Parallel jobs', 'Settings', 'Test management', 'Release retention', 'Service connections*', 'XAML build services', 'Test', 'Retention', and 'Extensions'. At the bottom, there is a 'Custom Admin Hub' link, which is highlighted with a red box.</p>

Name	ID	Level	Preview image
Organization settings	ms.vss-web.collection-admin-hub-group	Organization/collection	 <p>The screenshot shows the 'Organization Settings' page in Azure DevOps. The left sidebar includes sections for General, Overview, Projects, Users, Billing, Auditing, Global notifications, Usage, Extensions, Microsoft Entra, Security, Policies, and Permissions. The 'Extensions' section is expanded, showing Agent pools, Settings, Deployment pools, Parallel jobs, and OAuth configurations. At the bottom of the 'Extensions' list, there is a link labeled 'Custom Admin Hub' which is highlighted with a red rectangular box.</p>

## Example: contribute a hub

This example shows a hub contribution that targets the Code hub group:

JSON

```
{
  "contributions": [
    {
      "id": "my-custom-hub",
      "type": "ms.vss-web.hub",
      "targets": [
        "ms.vss-code-web.code-hub-group"
      ],
      "properties": {
        "name": "Code Hub",
        "order": 30,
        "uri": "/views/code/custom.html"
      }
    }
  ]
}
```

```
]  
}
```

- `ms.vss-web.hub` is the type of contribution. This type is defined in the `vss-web` extension published under the `ms` publisher. This type declares optional and required properties that are required by contributions of this type (for example, name, order, and so on).
- `ms.vss-code-web.code-hub-group` is the full ID of the hub group contribution that this hub targets. This contribution is declared in the `vss-code-web` extension published under the `ms` publisher.
- `my-custom-hub` is the short ID of this contribution; `{publisherId}.{extensionId}.my-custom-hub` is the full ID.

## Add an icon to your menu or toolbar

Add an icon property, so it can be used directly by name.

We recommend providing your own icon. Using your own icon example:

```
"properties":  
  
    "name": "Sample hub",  
    "uri": "dist/Hub/Hub.html",  
    "icon": "asset://static/sample-icon.png",  
    "supportsMobile": true  
}
```

Using the [Office UI Fabric Icons](#) example:

```
"properties":  
  
    "iconName": "Code",  
    "name": "Code Hub",  
    "order": 30,  
    "uri": "/views/code/custom.html"  
}
```

## Settings for menus and toolbars

 Expand table

Name	Target ID
Organization/collection overview toolbar	ms.vss-admin-web.collection-overview-toolbar-menu
Collection overview projects grid	ms.vss-admin-web.projects-grid-menu
Project overview toolbar	ms.vss-admin-web.project-overview-toolbar-menu
Project overview teams grid	ms.vss-admin-web.teams-grid-menu

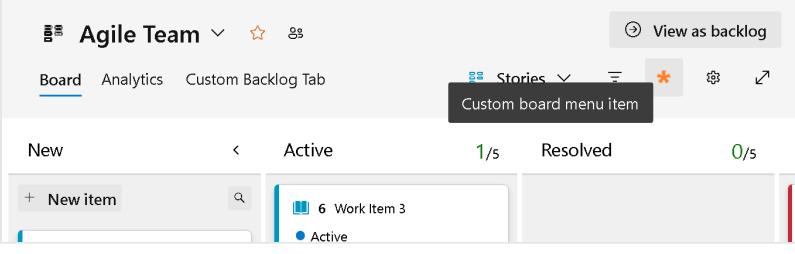
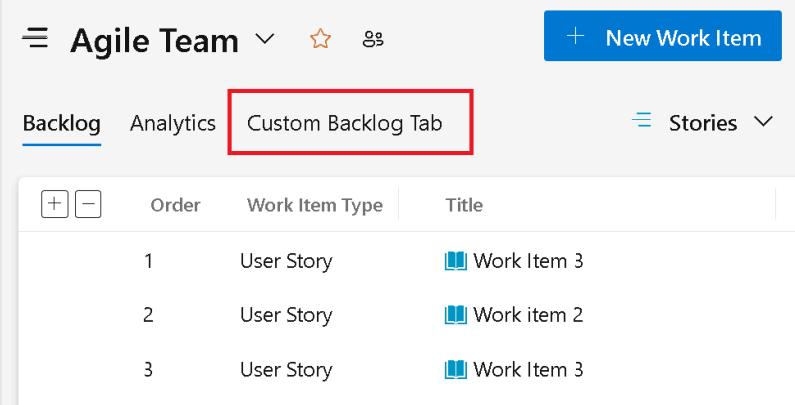
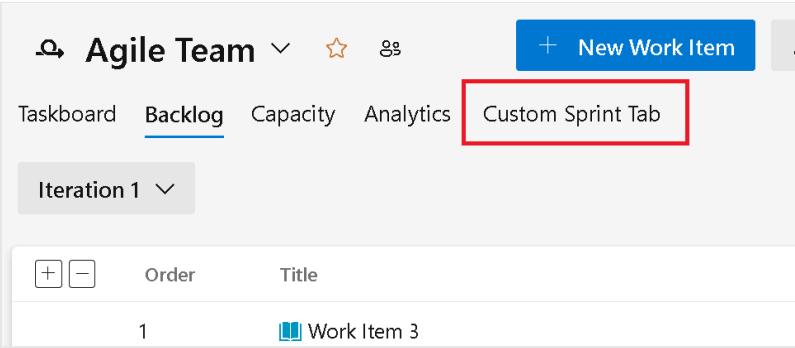
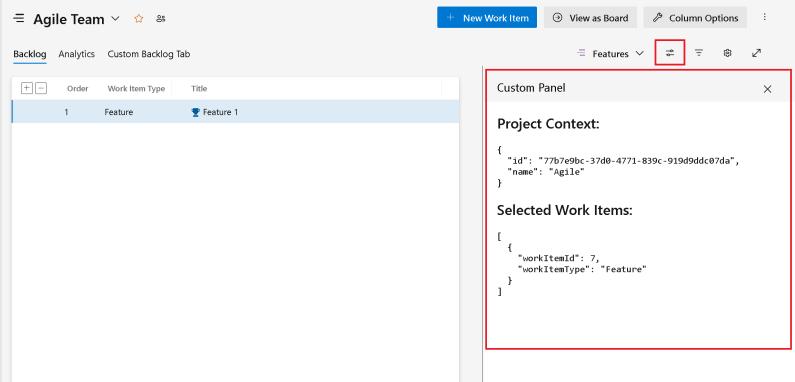
## Azure Boards menu and toolbar

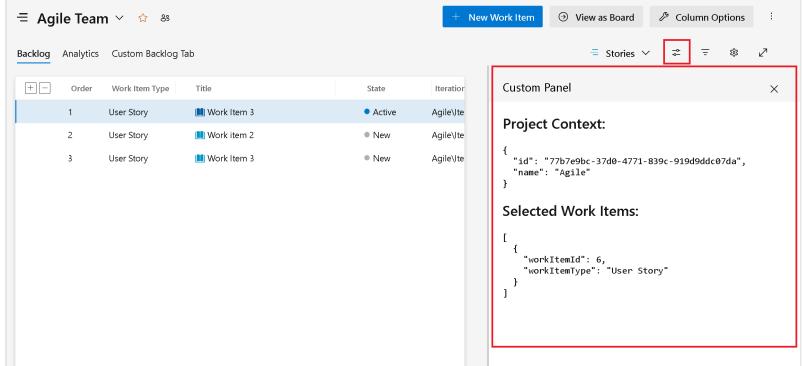
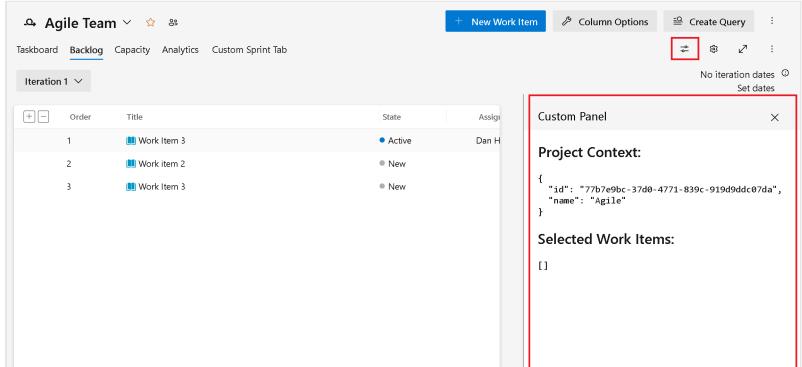
[Expand table](#)

Name	Target ID	Preview image
Work item query menu	ms.vss-work-web.work-item-query-menu	
Work item query results toolbar menu	ms.vss-work-web.work-item-query-results-toolbar-menu	

Name	Target ID	Preview image									
Work item query results menu item	ms.vss-work-web.query-result-work-item-menu	<p>The screenshot shows a 'Assigned to me' query results page with one work item selected: Bug 4. A context menu is open, showing various actions such as Edit, Change type, Unfollow, Assign to, Delete, Templates, Link to a new work item, Link to an existing item, Move to team project, Copy as HTML, Copy link, Email, Clone work item, Create copy of work item, New branch, and Custom query result menu item.</p>									
Work item query results tab	ms.vss-work-web.query-tabs	<p>The screenshot shows a 'Assigned to me' query results page with one work item selected: Bug 4. The 'Custom Tab' tab is highlighted with a red box. The table below shows the work items.</p> <table border="1"> <thead> <tr> <th>ID</th> <th>Work Item Type</th> <th>Title</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Bug</td> <td>Bug 4</td> </tr> <tr> <td>2</td> <td>User Story</td> <td>Work Item 3</td> </tr> </tbody> </table>	ID	Work Item Type	Title	1	Bug	Bug 4	2	User Story	Work Item 3
ID	Work Item Type	Title									
1	Bug	Bug 4									
2	User Story	Work Item 3									

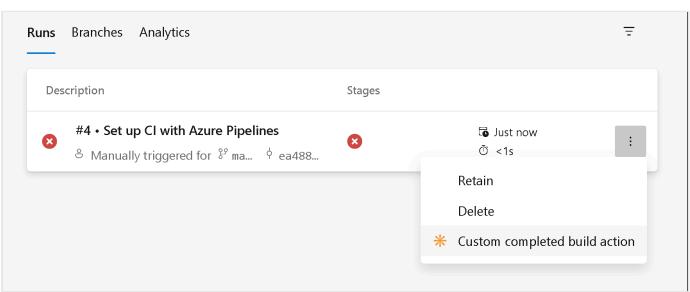
Name	Target ID	Preview image
Work item for context menu	ms.vss-work-web.work-item-toolbar-menu	<p>The screenshot shows a 'USER STORY 2' work item with the identifier '2 Work Item 3'. The context menu on the right includes options like 'New linked work item', 'Change type...', 'Move to team project', 'Create copy of work item...', 'Email work item', 'Delete', 'Copy link', 'Templates', 'New branch...', 'Customize', and a 'Custom work item toolbar action' option.</p>
Backlog item menu	ms.vss-work-web.backlog-item-menu	<p>The screenshot shows a 'Backlog' tab with a backlog item titled 'Work Item 3' at order 1. The context menu on the right includes options like 'Edit', 'Assign to', 'Copy as HTML', 'Copy link', 'Delete', 'Add link', 'Move to iteration', 'Change parent', 'Move to position...', 'Change type...', 'Move to team project...', 'Email...', 'Templates', 'New branch...', and a 'Custom backlog item action' option.</p>
Sprint board pivot filter menu	ms.vss-work-web.sprint-board-pivot-filter-menu	<p>The screenshot shows a 'Sprint Board' tab with a pivot filter menu. The context menu on the right includes options like 'New Work Item', 'Column Options', 'Sprint board menu', and 'Set dates'.</p>

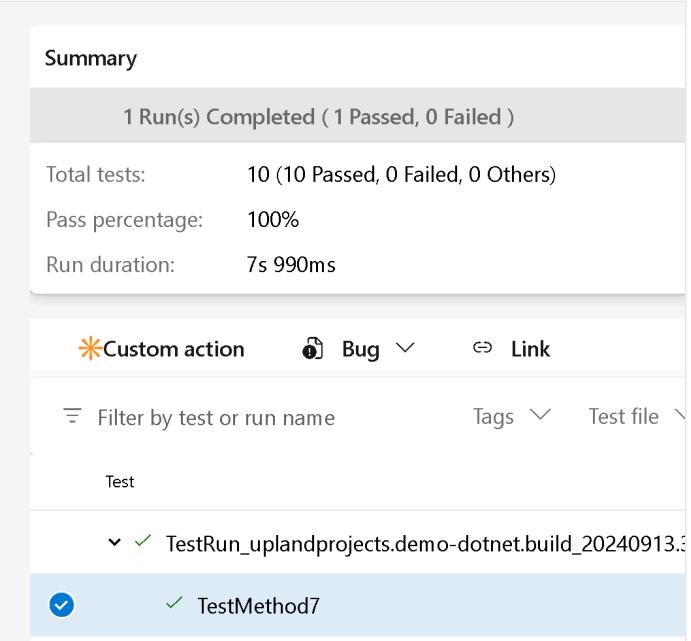
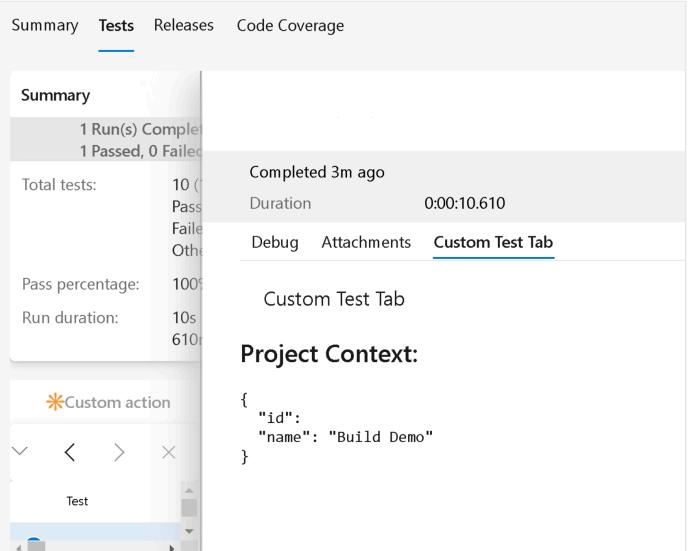
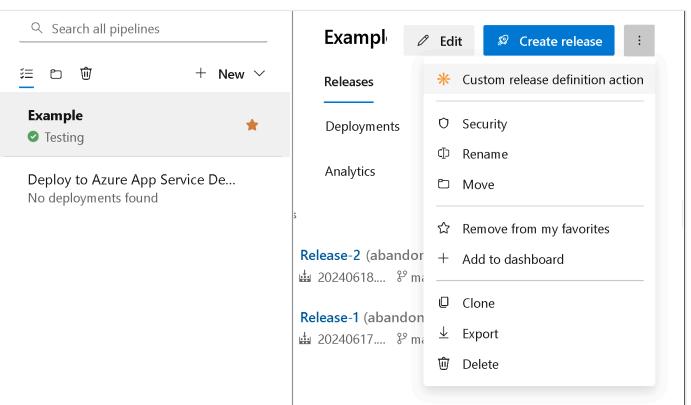
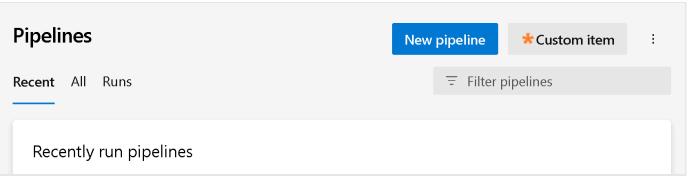
Name	Target ID	Preview image
Board pivot filter menu	ms.vss-work-web.backlog-board-pivot-filter-menu	
Card menu	ms.vss-work-web.backlog-board-card-item-menu	
Product backlog tab	ms.vss-work-web.product-backlog-tabs	
Iteration backlog tab	ms.vss-work-web.iteration-backlog-tabs	
Portfolio backlog pane	ms.vss-work-web.portfolio-backlog-toolpane	

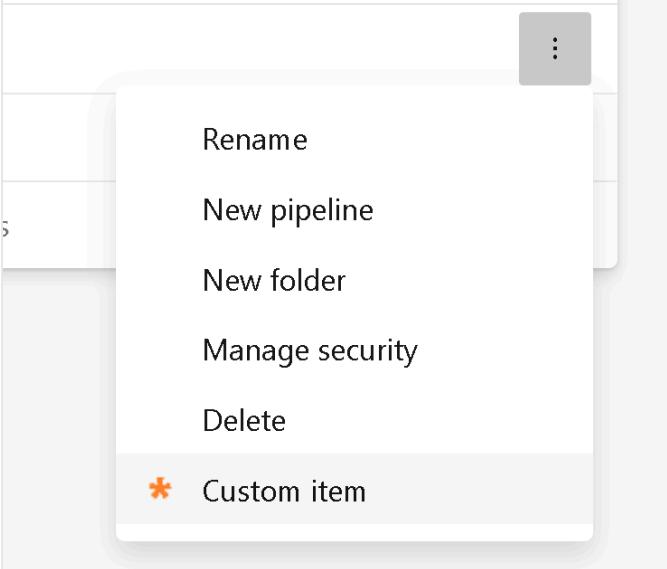
Name	Target ID	Preview image
Product backlog pane	ms.vss-work-web.requirement-backlog-toolpane	
Iteration backlog pane	ms.vss-work-web.iteration-backlog-toolpane	

## Azure Pipelines menu and toolbar

 Expand table

Name	Target ID	Preview
Completed build menu	ms.vss-build-web.completed-build-menu	

Name	Target ID	Preview
Test results toolbar action	ms.vss-test-web.test-results-actions-menu	 <p>Summary</p> <p>1 Run(s) Completed ( 1 Passed, 0 Failed )</p> <p>Total tests: 10 (10 Passed, 0 Failed, 0 Others)</p> <p>Pass percentage: 100%</p> <p>Run duration: 7s 990ms</p> <p>Custom action Bug Link</p> <p>Filter by test or run name Tags Test file</p> <p>Test</p> <p>✓ ✓ TestRun_uploadprojects.demo-dotnet.build_20240913.3</p> <p>✓ ✓ TestMethod7</p>
Test result details tab	ms.vss-test-web.test-result-details-tab-items	 <p>Summary Tests Releases Code Coverage</p> <p>Completed 3m ago Duration 0:00:10.610</p> <p>Debug Attachments Custom Test Tab</p> <p>Custom Test Tab</p> <p>Project Context:</p> <pre>{   "id": "Build Demo",   "name": "Build Demo" }</pre>
Release pipeline explorer context menu	ms.vss-releaseManagement-web.release-definition-explorer-context-menu	 <p>Example Edit Create release</p> <p>Releases</p> <p>Custom release definition action</p> <ul style="list-style-type: none"> <li>Security</li> <li>Rename</li> <li>Move</li> <li>Remove from my favorites</li> <li>Add to dashboard</li> </ul> <p>Deploy to Azure App Service De...</p> <p>No deployments found</p> <p>Release-2 (abandon) 20240618... 8 mi</p> <p>Release-1 (abandon) 20240617.... 8 mi</p> <p>Clone Export Delete</p>
Pipeline details view, header button	ms.vss-build-web.pipelines-header-menu	 <p>Pipelines</p> <p>New pipeline * Custom item</p> <p>Recent All Runs</p> <p>Filter pipelines</p> <p>Recently run pipelines</p>

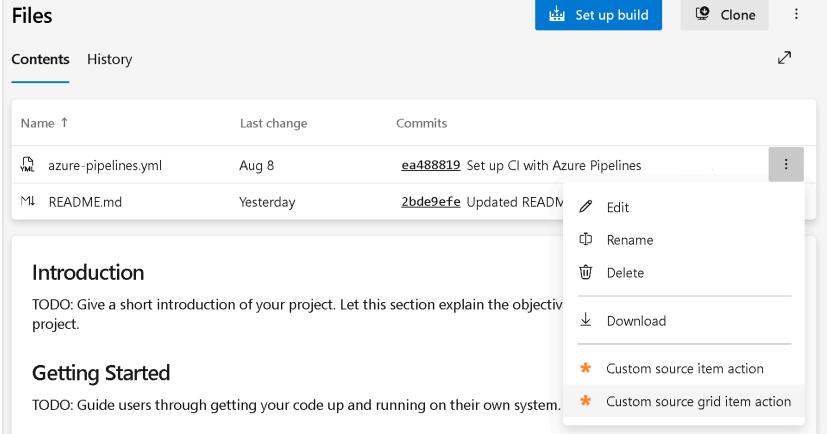
Name	Target ID	Preview
Pipeline details view, folder context menu	ms.vss-build-web.pipelines-folder-menu	

## Azure Pipelines tasks

Tasks perform work in a build or release. For more information, see [Add a custom pipelines task extension](#).

## Azure Repos menu and toolbar

[Expand table](#)

Name	Target ID	Preview image
Source item (grid) menu	ms.vss-code-web.source-grid-item-menu	

Name	Target ID	Preview image				
Source item (tree) menu	ms.vss-code-web.source-tree-item-menu		<b>Files</b> <a href="#">Contents</a> <a href="#">History</a>  <table border="1"> <thead> <tr> <th>Name ↑</th> </tr> </thead> <tbody> <tr> <td> azure-pipelines.yml</td> </tr> <tr> <td> README.md</td> </tr> </tbody> </table>	Name ↑	azure-pipelines.yml	README.md
Name ↑						
azure-pipelines.yml						
README.md						
Source item (grid and tree) menu	ms.vss-code-web.source-item-menu		<b>Introducing</b> <a href="#">TODO: Give</a>  <table border="1"> <thead> <tr> <th>Name ↑</th> </tr> </thead> <tbody> <tr> <td> azure-pipelines.yml</td> </tr> <tr> <td> README.md</td> </tr> </tbody> </table>	Name ↑	azure-pipelines.yml	README.md
Name ↑						
azure-pipelines.yml						
README.md						
Change list item menu	ms.vss-code-web.change-list-item-menu		<b>Introducing</b> <a href="#">TODO: Give</a>  <table border="1"> <thead> <tr> <th>Name ↑</th> </tr> </thead> <tbody> <tr> <td> azure-pipelines.yml</td> </tr> <tr> <td> README.md</td> </tr> </tbody> </table>	Name ↑	azure-pipelines.yml	README.md
Name ↑						
azure-pipelines.yml						
README.md						
Change list summary item menu	ms.vss-code-web.change-list-summary-item-menu		<b>Introducing</b> <a href="#">TODO: Give</a>  <table border="1"> <thead> <tr> <th>Name ↑</th> </tr> </thead> <tbody> <tr> <td> azure-pipelines.yml</td> </tr> <tr> <td> README.md</td> </tr> </tbody> </table>	Name ↑	azure-pipelines.yml	README.md
Name ↑						
azure-pipelines.yml						
README.md						

Name	Target ID	Preview image																					
Git branches tree menu	ms.vss-code-web.git-branches-tree-menu	<p>Branches</p> <p>Mine All Stale</p> <table border="1"> <thead> <tr> <th>Branch</th> <th>C</th> <th>Author</th> <th>A...</th> <th>Behind   Ahead</th> <th>S</th> <th>P</th> </tr> </thead> <tbody> <tr> <td>branch-1</td> <td>2bc</td> <td>Y...</td> <td>0 1</td> <td>view history</td> <td>83</td> <td>star</td> </tr> <tr> <td>main</td> <td>Default</td> <td>Compare</td> <td>ear</td> <td>A...</td> <td>Compare branches</td> <td>Set as compare branch</td> </tr> </tbody> </table> <ul style="list-style-type: none"> <li>branch-1</li> <li>main</li> <li>Default</li> <li>Compare</li> <li>ear</li> <li>A...</li> <li>0 1</li> <li>view history</li> <li>Compare branches</li> <li>Set as compare branch</li> <li>Set as default branch</li> <li>Lock</li> <li>Branch policies</li> <li>Branch security</li> <li>Custom Git branch tree action</li> </ul>	Branch	C	Author	A...	Behind   Ahead	S	P	branch-1	2bc	Y...	0 1	view history	83	star	main	Default	Compare	ear	A...	Compare branches	Set as compare branch
Branch	C	Author	A...	Behind   Ahead	S	P																	
branch-1	2bc	Y...	0 1	view history	83	star																	
main	Default	Compare	ear	A...	Compare branches	Set as compare branch																	
Git pull request actions menu	ms.vss-code-web.pull-request-action-menu	<p>Approve Complete</p> <p>o main</p> <p>ab</p> <p>Show everything (1)</p> <p>Reviewers</p> <p>Required</p> <p>Optional</p> <p>Tags</p> <p>No tags</p> <p>Work items</p>																					

Name	Target ID	Preview image														
Git pull request tabs (pivots)	ms.vss-code-web.pr-tabs	<p>Updated README.md</p> <p>Active !1  proposes to merge <a href="#">branch-1</a> into <a href="#">main</a></p> <p>Overview Files Updates Commits <b>Custom PR Details Tab</b></p> <p>Custom Backlog Tab</p> <p><b>Project Context:</b></p> <pre>{   "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee",   "name": "Agile" }</pre>														
Git commit listing menu	ms.vss-code-web.git-commit-list-menu	<p>Overview Files Updates <b>Commits</b> Custom PR Details Tab</p> <p>Updated README.md 2bde9efe Yesterday at 1:43 PM</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Copy full SHA</li> <li><input type="checkbox"/> Browse files</li> <li><input type="checkbox"/> New branch...</li> <li><input type="checkbox"/> Create tag...</li> <li><input checked="" type="checkbox"/> Custom commit list action</li> </ul>														
Git commit detail menu	ms.vss-code-web.git-commit-details-menu	<p>Updated README.md</p> <p>2bde9efe  committed Yesterday <a href="#">branch-1</a></p> <p>Files Details</p> <p>Parent 1 → This commit Filter 1 changed file</p> <table border="1"> <thead> <tr> <th>Agile</th> <th>README</th> </tr> </thead> <tbody> <tr> <td>MI</td> <td>/README</td> </tr> <tr> <td>18</td> <td>- [ASP.NET Core](https://github.com/aspnet/Ho</td> </tr> <tr> <td>19</td> <td>- [Visual Studio Code](https://github.com/Mic</td> </tr> <tr> <td>20</td> <td>- [Chakra Core](https://github.com/Microsoft/</td> </tr> <tr> <td>21</td> <td>+ - Change made</td> </tr> <tr> <td>22</td> <td></td> </tr> </tbody> </table> <ul style="list-style-type: none"> <li><input type="checkbox"/> Cherry-pick...</li> <li><input type="checkbox"/> Revert...</li> <li><input type="checkbox"/> Create tag...</li> <li><input type="checkbox"/> New branch...</li> <li><input checked="" type="checkbox"/> Custom commit details action</li> </ul>	Agile	README	MI	/README	18	- [ASP.NET Core](https://github.com/aspnet/Ho	19	- [Visual Studio Code](https://github.com/Mic	20	- [Chakra Core](https://github.com/Microsoft/	21	+ - Change made	22	
Agile	README															
MI	/README															
18	- [ASP.NET Core](https://github.com/aspnet/Ho															
19	- [Visual Studio Code](https://github.com/Mic															
20	- [Chakra Core](https://github.com/Microsoft/															
21	+ - Change made															
22																

## Azure Test Plans menu and toolbar

Expand table

Name	Target ID	Preview image
Test run grid menu	ms.vss-test-web.test-run-grid-menu	<p>Test Points (2 items)</p> <p>Test Case 1: Passed</p> <p>Test Case 2: Failed</p> <ul style="list-style-type: none"> <li>View execution history</li> <li>Mark Outcome</li> <li>Run</li> <li>Reset test to active</li> <li>Edit test case</li> <li>Assign tester</li> <li>View test result</li> <li>Custom test run action</li> </ul>
Test plan suites tree menu	ms.vss-test-web.test-plans-suites-context	<p>My Example Test Plan (2)</p> <p>New Suite</p> <p>Assign configurations</p> <p>Export</p> <p>Assign testers to run all tests</p> <p>Import test suites</p> <p>Custom test suite action</p>

Name	Target ID	Preview image
Test plan hub pivot tab	ms.vss-test-web.test-plan-pivot-tabs	<p><b>My Example Test Plan (ID: 14)</b></p> <p>Define Execute Chart <b>Custom Test Plan Tab</b></p> <p>Custom Test Plan Tab</p> <p><b>Project Context:</b></p> <pre>{   "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee" ,   "name": "Agile" }</pre>

## Other extensibility points

- **Dashboard widget:** An extension can contribute a new type of widget that can be added by users to a [dashboard](#). Learn how to [contribute a dashboard widget](#).
- **Work item form:** The work item form is enhanced by extensions with new sections, tabs, actions, and custom field renderers. For more information, learn how to [extend the work item form](#).
- **Service hooks:** A *consumer* is the service that events are sent to in Service Hooks. An extension can contribute consumer services. These services get configured by a user (or programmatically), to send events to that service. For more information, see [Create a custom consumer for service hooks](#).
- **Features: Name:** Preview feature (hosted only)  
Target ID: ms.vss-web.managed-features

# Contribution model

10/03/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

Extensions add capabilities to Azure DevOps by declaring contribution types and contribution instances. A contribution type defines a contract (the properties and behavior) that contributions of that type must implement. A contribution is a concrete instance of a contribution type (for example, a hub or a build task).

## 💡 Tip

If you're starting a new Azure DevOps extension, try these maintained sample collections first—they work with current product builds and cover modern scenarios (for example, adding tabs on pull request pages).

- Azure DevOps extension sample (GitHub)—a compact starter sample that demonstrates common extension patterns: <https://github.com/microsoft/azure-devops-extension-sample> ↗
- Azure DevOps extension samples (legacy collection and contributions guide)—install to inspect UI targets, or view the source:  
<https://marketplace.visualstudio.com/items/ms-samples.samples-contributions-guide> ↗ and <https://github.com/Microsoft/vso-extension-samples/tree/master/contributions-guide> ↗
- Microsoft Learn samples (browse Azure DevOps samples)—curated, up-to-date samples across Microsoft docs: /samples/browse/?terms=azure%20devops%20extension

If a sample doesn't work in your organization, install it into a personal or test organization and compare the extension manifest's target IDs and API versions with the current docs. For reference and APIs, see:

- [azure-devops-extension-api](#)
- [azure-devops-extension-sdk](#)
- [installed extension API](#)

For more information, see:

- [Extensibility points](#)
- [azure-devops-extension-api](#)

- [azure-devops-extension-sdk](#)

# Contribution types

A contribution type defines the properties and rules that contributions of that type must follow. A contribution type can extend another contribution type, inheriting its properties.

Common contribution types include:

- hub
- action
- build-task

Each property definition includes:

- property type (for example, string or boolean)
- whether the property is required
- an optional default value

## Example

A contribution type declaration in a manifest looks like this:

JSON

```
{
  "contributionTypes": [
    {
      "id": "hub",
      "name": "Web Access Hub",
      "description": "A hub that appears in the hubs menu at the top of web pages.",
      "properties": {
        "name": {
          "description": "The text to display for the hub",
          "type": "string",
          "required": true
        },
        "uri": {
          "description": "URI of the contents of the hub page",
          "type": "string",
          "required": true
        },
        "order": {
          "description": "Optional ordering value indicating the hub's position within the hub group",
          "type": "integer"
        }
      }
    }
  ]
}
```

```
        }
    ]
}
```

## Contributions

A **contribution** is an instance of a contribution type. For example, the `Queries` hub under the Work hub group is a contribution of type `hub` and the `Publish Test Results` build task is a contribution of type `build-task`.

All contributions must specify a type and specify values for any properties required by that contribution type.

## Example

Here's an example of a hub contribution declaration in an extension manifest:

JavaScript

```
{
  "contributions": [
    {
      "id": "build-explorer-hub",
      "type": "ms.vss-web.hub",
      "targets": [
        ".build-hub-group"
      ],
      "properties": {
        "name": "Explorer",
        "uri": "/_build",
        "order": 22
      }
    }
  ]
}
```

## Target contributions

A contribution can **target** one or more other contributions, which creates a relationship between the contribution and each of its targets. The system can discover contributions for the target at runtime. For example, a `hub` contribution (`Explorer`) might target a specific `hub-group` contribution (`Build`).

## JSON

```
{  
  "id": "build-explorer-hub",  
  "type": "ms.vss-web.hub",  
  "targets": [  
    ".build-hub-group"  
  ]  
}
```

When the hub group renders, the system can query for all hub contributions that target the hub group to know which hubs to render.

## Identify contributions and types

Every contribution and contribution type must have a unique ID within the extension it's declared in.

A *full* contribution identifier includes the following items, which you separate with a dot `.` :

- Publisher ID
- Extension ID
- Contribution/type ID

For example: `ms.vss-web.hub` is the full identifier for the following contribution:

- Publisher ID: `ms`
- Extension ID: `vss-web`
- Contribution/type ID: `hub`

You can use *relative* contribution references within an extension manifest for a contribution's reference to another contribution or contribution type within that same extension. In this case, the publisher and extension IDs aren't included, and the ID is a dot `.` followed by the contribution ID. For example, `.hub` might be used within the `vss-web` extension mentioned previously as a shortcut for `ms.vss-web.hub`.

ⓘ Note: The author created this article with assistance from AI. [Learn more](#)

# Add a custom pipelines task extension

08/05/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

This guide walks you through creating, testing, and publishing custom build or release tasks as Azure DevOps extensions. Custom pipeline tasks let you extend Azure DevOps with specialized functionality tailored to your team's workflows, from simple utilities to complex integrations with external systems.

Learn how to do the following tasks:

- Set up the development environment and project structure
- Create task logic using TypeScript and the Azure Pipelines Task Library
- Implement comprehensive unit testing with mock frameworks
- Package your extension for distribution
- Publish to the Visual Studio Marketplace
- Set up automated CI/CD pipelines for extension maintenance

For more information about Azure Pipelines, see [What is Azure Pipelines?](#)

 **Note**

This article covers agent tasks in agent-based extensions. For information about server tasks and server-based extensions, see [Server Task Authoring](#).

## Prerequisites

Before you begin, ensure you have the following requirements in place:

 [Expand table](#)

Component	Requirement	Description
Azure DevOps organization	Required	<a href="#">Create an organization</a> if you don't have one
Text editor	Recommended	<a href="#">Visual Studio Code</a> for IntelliSense and debugging support
Node.js	Required	Install the <a href="#">latest version</a> (Node.js 20 or later recommended)
TypeScript compiler	Required	Install the <a href="#">latest version</a> (version 4.6.3 or later)

Component	Requirement	Description
Azure DevOps CLI (tfx-cli)	Required	Install using <code>npm i -g tfx-cli</code> to package extensions
Azure DevOps Extension SDK	Required	Install the <a href="#">azure-devops-extension-sdk</a> package
Testing framework	Required	Mocha for unit testing (installed during setup)

## Project structure

Create a `home` directory for your project. After you complete this tutorial, your extension should have the following structure:

```

|--- README.md
|--- images
|   |--- extension-icon.png
|--- buildandreleasetask           // Task scripts location
|   |--- task.json                 // Task definition
|   |--- index.ts                  // Main task logic
|   |--- package.json              // Node.js dependencies
|   |--- tests/
|       |--- _suite.ts
|       |--- success.ts
|       |--- failure.ts
|--- vss-extension.json           // Extension manifest

```

### ⓘ Important

Your development machine must run the [latest version of Node.js](#) to ensure compatibility with the production environment. Update your `task.json` file to use Node 20:

JSON

```

"execution": {
  "Node20_1": {
    "target": "index.js"
  }
}

```

## 1. Create a custom task

This section guides you through creating the basic structure and implementation of your custom task. All files in this step should be created within the `buildandreleasetask` folder inside your project's `home` directory.

### ! Note

This walkthrough uses Windows with PowerShell. The steps work on all platforms, but environment variable syntax differs. On Mac or Linux, replace `$env:<var>=<val>` with `export <var>=<val>`.

## Set up the task scaffolding

Create the basic project structure and install required dependencies:

1. To initialize the Node.js project, open PowerShell, go to your `buildandreleasetask` folder, and run:

```
PowerShell
```

```
npm init --yes
```

The `package.json` file gets created with default settings. The `--yes` flag accepts all default options automatically.

### 💡 Tip

Azure Pipelines agents expect task folders to include node modules. Copy `node_modules` to your `buildandreleasetask` folder. To manage VSIX file size (50-MB limit), consider running `npm install --production` or `npm prune --production` before packaging.

2. Install the Azure Pipelines Task Library:

```
PowerShell
```

```
npm install azure-pipelines-task-lib --save
```

3. Install TypeScript type definitions:

```
PowerShell
```

```
npm install @types/node --save-dev
npm install @types/q --save-dev
```

#### 4. Set up version control exclusions

PowerShell

```
echo node_modules > .gitignore
```

Your build process should run `npm install` to rebuild `node_modules` each time.

#### 5. Install testing dependencies:

PowerShell

```
npm install mocha --save-dev -g
npm install sync-request --save-dev
npm install @types/mocha --save-dev
```

#### 6. Install TypeScript compiler:

PowerShell

```
npm install typescript@4.6.3 -g --save-dev
```

##### ⓘ Note

Install TypeScript globally to ensure the `tsc` command is available. Without it, TypeScript 2.3.4 is used by default.

#### 7. Configure TypeScript compilation:

PowerShell

```
tsc --init --target es2022
```

The `tsconfig.json` file gets created with ES2022 target settings.

## Implement the task logic

With scaffolding complete, create the core task files that define functionality and metadata:

1. Create the task definition file: Create `task.json` in the `buildandreleasetask` folder. This file describes your task to the Azure Pipelines system, defining inputs, execution settings, and UI presentation.

```
JSON

{
    "$schema": "https://raw.githubusercontent.com/Microsoft/azure-pipelines-task-lib/master/tasks.schema.json",
    "id": "{{taskguid}}",
    "name": "{{taskname}}",
    "friendlyName": "{{taskfriendlyname}}",
    "description": "{{taskdescription}}",
    "helpMarkDown": "",
    "category": "Utility",
    "author": "{{taskauthor}}",
    "version": {
        "Major": 0,
        "Minor": 1,
        "Patch": 0
    },
    "instanceNameFormat": "Echo $(samplestring)",
    "inputs": [
        {
            "name": "samplestring",
            "type": "string",
            "label": "Sample String",
            "defaultValue": "",
            "required": true,
            "helpMarkDown": "A sample string"
        }
    ],
    "execution": {
        "Node20_1": {
            "target": "index.js"
        }
    }
}
```

 **Note**

Replace `{{placeholders}}` with your task's actual information. The `taskguid` must be unique. Generate one using PowerShell: `(New-Guid).Guid`

2. To implement the task logic, create `index.ts` with your task's main functionality:

```
TypeScript
```

```

import tl = require('azure-pipelines-task-lib/task');

async function run() {
    try {
        const inputString: string | undefined = tl.getInput('samplestring', true);
        if (inputString == 'bad') {
            tl.setResult(tl.TaskResult.Failed, 'Bad input was given');
            return;
        }
        console.log('Hello', inputString);
    }
    catch (err: any) {
        tl.setResult(tl.TaskResult.Failed, err.message);
    }
}

run();

```

### 3. Compile TypeScript to JavaScript:

PowerShell

tsc

The `index.js` file gets created from your TypeScript source.

## Understanding task.json components

The `task.json` file is the heart of your task definition. Here are the key properties:

[Expand table](#)

Property	Description	Example
<code>id</code>	Unique GUID identifier for your task	Generated using <code>(New-Guid).Guid</code>
<code>name</code>	Task name without spaces (used internally)	<code>MyCustomTask</code>
<code>friendlyName</code>	Display name shown in the UI	<code>My Custom Task</code>
<code>description</code>	Detailed description of task functionality	<code>Performs custom operations on files</code>
<code>author</code>	Publisher or author name	<code>My Company</code>
<code>instanceNameFormat</code>	How the task appears in pipeline steps	<code>Process \$(inputFile)</code>

Property	Description	Example
inputs	Array of input parameters	See the following input types
execution	Execution environment specification	Node20_1, PowerShell13, etc.
restrictions	Security restrictions for commands and variables	Recommended for new tasks

## Security restrictions

For production tasks, add security restrictions to limit command usage and variable access:

JSON

```
"restrictions": {
  "commands": {
    "mode": "restricted"
  },
  "settableVariables": {
    "allowed": [ "variable1", "test*" ]
  }
}
```

Restricted mode allows only these commands:

- logdetail, logissue, complete, setprogress
- setsecret, setvariable, debug, settaskvariable
- prependpath, publish

Variable allowlist controls which variables can be set via `setvariable` or `prependpath`. Supports basic regex patterns.

! Note

This feature requires [agent version 2.182.1](#) or later.

## Input types and examples

Common input types for task parameters:

JSON

```
"inputs": [
  {
```

```

        "name": "stringInput",
        "type": "string",
        "label": "Text Input",
        "defaultValue": "",
        "required": true,
        "helpMarkDown": "Enter a text value"
    },
    {
        "name": "boolInput",
        "type": "boolean",
        "label": "Enable Feature",
        "defaultValue": "false",
        "required": false
    },
    {
        "name": "picklistInput",
        "type": "pickList",
        "label": "Select Option",
        "options": {
            "option1": "First Option",
            "option2": "Second Option"
        },
        "defaultValue": "option1"
    },
    {
        "name": "fileInput",
        "type": "filePath",
        "label": "Input File",
        "required": true,
        "helpMarkDown": "Path to the input file"
    }
]

```

## Test your task locally

Before packaging, test your task to ensure it works correctly:

1. Test with missing input (should fail):

PowerShell

```
node index.js
```

Expected output:

```
##vso[task.debug]agent.workFolder=undefined
##vso[task.debug]loading inputs and endpoints
##vso[task.debug]loaded 0
```

```
##vso[task.debug]task result: Failed  
##vso[task.issue type=error;]Input required: samplestring  
##vso[task.complete result=Failed;]Input required: samplestring
```

## 2. Test with valid input (should succeed):

PowerShell

```
$env:INPUT_SAMPLESTRING="World"  
node index.js
```

Expected output:

```
##vso[task.debug]agent.workFolder=undefined  
##vso[task.debug]loading inputs and endpoints  
##vso[task.debug]loading INPUT_SAMPLESTRING  
##vso[task.debug]loaded 1  
##vso[task.debug]samplestring=World  
Hello World
```

## 3. Test error handling:

PowerShell

```
$env:INPUT_SAMPLESTRING="bad"  
node index.js
```

This action should trigger the error handling path in your code.

 **Tip**

For information about task runners and Node.js versions, see [Node runner update guidance](#).

For more information, see the [Build/release task reference](#).

## 2. Implement comprehensive unit testing

Testing your task thoroughly ensures reliability and helps catch issues before deployment to production pipelines.

# Install testing dependencies

Install the required testing tools:

PowerShell

```
npm install mocha --save-dev -g
npm install sync-request --save-dev
npm install @types/mocha --save-dev
```

## Create test

1. Create a `tests` folder in your task directory containing a `_suite.ts` file:

TypeScript

```
import * as path from 'path';
import * as assert from 'assert';
import * as ttm from 'azure-pipelines-task-lib/mock-test';

describe('Sample task tests', function () {

    before( function() {
        // Setup before tests
    });

    after(() => {
        // Cleanup after tests
    });

    it('should succeed with simple inputs', function(done: Mocha.Done) {
        // Success test implementation
    });

    it('should fail if tool returns 1', function(done: Mocha.Done) {
        // Failure test implementation
    });
});
```

### 💡 Tip

Your test folder should be located in the task folder (for example, `buildandreleasetask`). If you encounter a sync-request error, install it in the task folder: `npm i --save-dev sync-request`.

2. Create `success.ts` in your test directory to simulate successful task execution:

TypeScript

```
import ma = require('azure-pipelines-task-lib/mock-answer');
import tmrm = require('azure-pipelines-task-lib/mock-run');
import path = require('path');

let taskPath = path.join(__dirname, '.', 'index.js');
let tmr: tmrm.TaskMockRunner = new tmrm.TaskMockRunner(taskPath);

// Set valid input for success scenario
tmr.setInput('samplestring', 'human');

tmr.run();
```

3. Add the success test to your `_suite.ts` file:

TypeScript

```
it('should succeed with simple inputs', function(done: Mocha.Done) {
    this.timeout(1000);

    let tp: string = path.join(__dirname, 'success.js');
    let tr: ttm.MockTestRunner = new ttm.MockTestRunner(tp);

    tr.runAsync().then(() => {
        console.log(tr.succeeded);
        assert.equal(tr.succeeded, true, 'should have succeeded');
        assert.equal(tr.warningIssues.length, 0, "should have no warnings");
        assert.equal(tr.errorIssues.length, 0, "should have no errors");
        console.log(tr.stdout);
        assert.equal(tr.stdout.indexOf('Hello human') >= 0, true, "should
display Hello human");
        done();
    }).catch((error) => {
        done(error); // Ensure the test case fails if there's an error
    });
});
```

4. Create `failure.ts` in your test directory to test error handling:

TypeScript

```
import ma = require('azure-pipelines-task-lib/mock-answer');
import tmrm = require('azure-pipelines-task-lib/mock-run');
import path = require('path');

let taskPath = path.join(__dirname, '.', 'index.js');
let tmr: tmrm.TaskMockRunner = new tmrm.TaskMockRunner(taskPath);

// Set invalid input to trigger failure
tmr.setInput('samplestring', 'bad');
```

```
tmr.run();
```

5. Add the failure test to your `_suite.ts` file:

TypeScript

```
it('should fail if tool returns 1', function(done: Mocha.Done) {
  this.timeout(1000);

  const tp = path.join(__dirname, 'failure.js');
  const tr: ttm.MockTestRunner = new ttm.MockTestRunner(tp);

  tr.runAsync().then(() => {
    console.log(tr.succeeded);
    assert.equal(tr.succeeded, false, 'should have failed');
    assert.equal(tr.warningIssues.length, 0, 'should have no warnings');
    assert.equal(tr.errorIssues.length, 1, 'should have 1 error issue');
    assert.equal(tr.errorIssues[0], 'Bad input was given', 'error issue
output');
    assert.equal(tr.stdout.indexOf('Hello bad'), -1, 'Should not display
Hello bad');
    done();
  });
});
```

## Run your tests

Execute the test suite:

PowerShell

```
# Compile TypeScript
tsc

# Run tests
mocha tests/_suite.js
```

Both tests should pass. For verbose output (similar to build console output), set the trace environment variable:

PowerShell

```
$env:TASK_TEST_TRACE=1
mocha tests/_suite.js
```

## Test coverage best practices

- **Test all input combinations:** Valid inputs, invalid inputs, missing required inputs
- **Test error scenarios:** Network failures, file system errors, permission issues
- **Mock external dependencies:** Don't rely on external services in unit tests
- **Validate outputs:** Check console output, task results, and generated artifacts
- **Performance testing:** Consider adding tests for tasks that process large files

## Security best practices

- **Input validation:** Always validate and sanitize inputs
- **Secrets handling:** Use `setSecret` for sensitive data
- **Command restrictions:** Implement command restrictions for production tasks
- **Minimal permissions:** Request only necessary permissions
- **Regular updates:** Keep dependencies and Node.js versions current

After testing your task locally and implementing comprehensive unit tests, package it into an extension for Azure DevOps.

## Install packaging tools

Install the Cross Platform Command Line Interface (tfx-cli):

```
PowerShell
```

```
npm install -g tfx-cli
```

## Create the extension manifest

The extension manifest (`vss-extension.json`) contains all information about your extension, including references to your task folders and images.

1. Create an images folder with an `extension-icon.png` file
2. Create `vss-extension.json` in your extension's root directory (not in the task folder):

```
JSON
```

```
{  
  "manifestVersion": 1,  
  "id": "my-custom-tasks",  
  "name": "My Custom Tasks",  
  "version": "1.0.0",  
  "publisher": "Your Company Name",  
  "description": "A brief description of your extension.",  
  "categories": ["Build", "Deployment"],  
  "inputs": [{"name": "Input A", "type": "String"}, {"name": "Input B", "type": "Number"}],  
  "outputs": [{"name": "Output C", "type": "String"}, {"name": "Output D", "type": "Number"}],  
  "tasks": [{"name": "Task A", "path": "task-a"}, {"name": "Task B", "path": "task-b"}],  
  "resources": [{"name": "Resource A", "path": "resource-a"}, {"name": "Resource B", "path": "resource-b"}]  
}
```

```

"publisher": "your-publisher-id",
"targets": [
    {
        "id": "Microsoft.VisualStudio.Services"
    }
],
"description": "Custom build and release tasks for Azure DevOps",
"categories": [
    "Azure Pipelines"
],
"icons": {
    "default": "images/extension-icon.png"
},
"files": [
    {
        "path": "MyCustomTask"
    }
],
"contributions": [
    {
        "id": "my-custom-task",
        "type": "ms.vss-distributed-task.task",
        "targets": [
            "ms.vss-distributed-task.tasks"
        ],
        "properties": {
            "name": "MyCustomTask"
        }
    }
]
}

```

## Key manifest properties

[ ] Expand table

Property	Description
<code>publisher</code>	Your marketplace publisher identifier
<code>contributions.id</code>	Unique identifier within the extension
<code>contributions.properties.name</code>	Must match your task folder name
<code>files.path</code>	Path to your task folder relative to the manifest

### (!) Note

Change the **publisher** value to your publisher name. For information about creating a publisher, see [Create your publisher](#).

## Package your extension

Package your extension into a .vsix file:

PowerShell

```
tfx extension create --manifest-globs vss-extension.json
```

## Version management

- **Extension version:** Increment the version in `vss-extension.json` for each update
- **Task version:** Increment the version in `task.json` for each task update
- **Auto-increment:** Use `--rev-version` to automatically increment the patch version

PowerShell

```
tfx extension create --manifest-globs vss-extension.json --rev-version
```

### Important

Both the task version and extension version must be updated for changes to take effect in Azure DevOps.

## Versioning strategy

Follow semantic versioning principles for your task updates:

- **Major version:** Breaking changes to inputs/outputs
- **Minor version:** New features, backward compatible
- **Patch version:** Bug fixes only

Update process:

1. Update `task.json` version
2. Update `vss-extension.json` version
3. Test thoroughly in a test organization
4. Publish and monitor for issues

# Publish to Visual Studio Marketplace

## 1. Create your publisher

1. Sign in to the [Visual Studio Marketplace Publishing Portal](#)
2. Create a new publisher if prompted:
  - **Publisher identifier:** Used in your extension manifest (for example, `mycompany-myteam`)
  - **Display name:** Public name shown in the marketplace (for example, `My Team`)
3. Review and accept the [Marketplace Publisher Agreement](#)

## 2. Upload your extension

Web interface method:

1. Select **Upload new extension**
2. Choose your packaged `.vsix` file
3. Select **Upload**

Command-line method:

PowerShell

```
tfx extension publish --manifest-globs vss-extension.json --share-with  
yourOrganization
```

## 3. Share your extension

1. Right-click your extension in the marketplace
2. Select **Share**
3. Enter your organization name
4. Add more organizations as needed

 **Important**

Publishers must be verified to share extensions publicly. For more information, see [Package/Publish/Install](#).

## 4. Install to your organization

After sharing, install the extension to your Azure DevOps organization:

1. Navigate to **Organization Settings > Extensions**
2. Browse for your extension
3. Select **Get it free** and install

## 3. Package and publish your extension

### Verify your extension

After installation, verify your task works correctly:

1. Create or edit a pipeline.
2. Add your custom task:
  - Select **Add task** in the pipeline editor
  - Search for your custom task by name
  - Add it to your pipeline
3. Configure task parameters:
  - Set required inputs
  - Configure optional settings
4. Run the pipeline to test functionality
5. Monitor execution:
  - Check task logs for proper execution
  - Verify expected outputs
  - Ensure no errors or warnings

## 4. Automate extension publishing with CI/CD

To maintain your custom task effectively, create automated build and release pipelines that handle testing, packaging, and publishing.

### Prerequisites for automation

- **Azure DevOps Extension Tasks:** [Install the extension](#) for free
- **Variable group:** Create a [pipeline library variable group](#) with these variables:
  - `publisherId`: Your marketplace publisher ID
  - `extensionId`: Extension ID from `vss-extension.json`

- `extensionName`: Extension name from `vss-extension.json`
- `artifactName`: Name for the VSIX artifact
- **Service connection:** Create a Marketplace service connection with pipeline access permissions

## Complete CI/CD pipeline

Create a YAML pipeline with comprehensive stages for testing, packaging, and publishing:

YAML

```
trigger:
- main

pool:
  vmImage: "ubuntu-latest"

variables:
- group: extension-variables # Your variable group name

stages:
- stage: Test_and_validate
  displayName: 'Run Tests and Validate Code'
  jobs:
    - job: RunTests
      displayName: 'Execute unit tests'
      steps:
        - task: TfxInstaller@4
          displayName: 'Install TFX CLI'
          inputs:
            version: "v0.x"

        - task: Npm@1
          displayName: 'Install task dependencies'
          inputs:
            command: 'install'
            workingDir: '/MyCustomTask' # Update to your task directory

        - task: Bash@3
          displayName: 'Compile TypeScript'
          inputs:
            targetType: "inline"
            script: |
              cd MyCustomTask # Update to your task directory
              tsc

        - task: Npm@1
          displayName: 'Run unit tests'
          inputs:
            command: 'custom'
            workingDir: '/MyCustomTask' # Update to your task directory
```

```
    customCommand: 'test' # Ensure this script exists in package.json

  - task: PublishTestResults@2
    displayName: 'Publish test results'
    inputs:
      testResultsFormat: 'JUnit'
      testResultsFiles: '**/test-results.xml'
      searchFolder: '$(System.DefaultWorkingDirectory)'

  - stage: Package_extension
    displayName: 'Package Extension'
    dependsOn: Test_and_validate
    condition: succeeded()
    jobs:
      - job: PackageExtension
        displayName: 'Create VSIX package'
        steps:
          - task: TfxInstaller@4
            displayName: 'Install TFX CLI'
            inputs:
              version: "v0.x"

          - task: Npm@1
            displayName: 'Install dependencies'
            inputs:
              command: 'install'
              workingDir: '/MyCustomTask'

          - task: Bash@3
            displayName: 'Compile TypeScript'
            inputs:
              targetType: "inline"
              script: |
                cd MyCustomTask
                tsc

          - task: QueryAzureDevOpsExtensionVersion@4
            name: QueryVersion
            displayName: 'Query current extension version'
            inputs:
              connectTo: 'VsTeam'
              connectedServiceName: 'marketplace-connection'
              publisherId: '$(publisherId)'
              extensionId: '$(extensionId)'
              versionAction: 'Patch'

          - task: PackageAzureDevOpsExtension@4
            displayName: 'Package extension'
            inputs:
              rootFolder: '$(System.DefaultWorkingDirectory)'
              publisherId: '$(publisherId)'
              extensionId: '$(extensionId)'
              extensionName: '$(extensionName)'
              extensionVersion: '$(QueryVersion.Extension.Version)'
              updateTasksVersion: true
```

```

updateTasksVersionType: 'patch'
extensionVisibility: 'private'
extensionPricing: 'free'

- task: PublishBuildArtifacts@1
  displayName: 'Publish VSIX artifact'
  inputs:
    PathToPublish: '$(System.DefaultWorkingDirectory)/*.vsix'
    ArtifactName: '$(artifactName)'
    publishLocation: 'Container'

- stage: Publish_to_marketplace
  displayName: 'Publish to Marketplace'
  dependsOn: Package_extension
  condition: and(succeeded(), eq(variables['Build.SourceBranch'],
'refs/heads/main'))
  jobs:
    - deployment: PublishExtension
      displayName: 'Deploy to marketplace'
      environment: 'marketplace-production'
      strategy:
        runOnce:
          deploy:
            steps:
              - task: TfxInstaller@4
                displayName: 'Install TFX CLI'
                inputs:
                  version: "v0.x"

              - task: PublishAzureDevOpsExtension@4
                displayName: 'Publish to marketplace'
                inputs:
                  connectTo: 'VsTeam'
                  connectedServiceName: 'marketplace-connection'
                  fileType: 'vsix'
                  vsixFile: '$(Pipeline.Workspace)/$(artifactName)/*.vsix'
                  publisherId: '$(publisherId)'
                  extensionId: '$(extensionId)'
                  extensionName: '$(extensionName)'
                  updateTasksVersion: false
                  extensionVisibility: 'private'
                  extensionPricing: 'free'

```

## Configure package.json for testing

Add test scripts to your `package.json`:

JSON

```
{
  "scripts": {
    "test": "mocha tests/_suite.js --reporter xunit --reporter-option output=test-
```

```
results.xml",
  "test-verbose": "cross-env TASK_TEST_TRACE=1 npm test"
}
}
```

## Pipeline stage breakdown

### Stage 1: Test and validate

- **Purpose:** Ensure code quality and functionality
- **Actions:** Install dependencies, compile TypeScript, run unit tests, publish results
- **Validation:** All tests must pass to proceed

### Stage 2: Package extension

- **Purpose:** Create deployable VSIX package
- **Actions:** Query current version, increment version, package extension, publish artifacts
- **Versioning:** Automatically handles version increments

### Stage 3: Publish to marketplace

- **Purpose:** Deploy to Visual Studio Marketplace
- **Conditions:** Only runs on main branch after successful packaging
- **Environment:** Uses deployment environment for approval gates

## Best practices for CI/CD

- **Branch protection:** Only publish from main/release branches
- **Environment gates:** Use deployment environments for production releases
- **Version management:** Automate version increments to avoid conflicts
- **Test coverage:** Ensure comprehensive test coverage before packaging
- **Security:** Use service connections instead of hardcoded credentials
- **Monitoring:** Set up alerts for failed deployments

For classic build pipelines, follow these steps to set up extension packaging and publishing:

1. Add the `Bash` task to compile the TypeScript into JavaScript.
2. To query the existing version, add the `Query Extension Version` task using the following inputs:
  - Connect to: Visual Studio Marketplace

- Visual Studio Marketplace (Service connection): Service Connection
- Publisher ID: ID of your Visual Studio Marketplace publisher
- Extension ID: ID of your extension in the `vss-extension.json` file
- Increase version: Patch
- Output Variable: `Task.Extension.Version`

3. To package the extensions based on manifest Json, add the **Package Extension** task using the following inputs:

- Root manifests folder: Points to root directory that contains manifest file. For example, `$(System.DefaultWorkingDirectory)` is the root directory
- Manifest file: `vss-extension.json`
- Publisher ID: ID of your Visual Studio Marketplace publisher
- Extension ID: ID of your extension in the `vss-extension.json` file
- Extension Name: Name of your extension in the `vss-extension.json` file
- Extension Version: `$(Task.Extension.Version)`
- Override tasks version: checked (true)
- Override Type: Replace Only Patch (1.0.r)
- Extension Visibility: If the extension is still in development, set the value to *private*. To release the extension to the public, set the value to *public*.

4. To copy to published files, add the **Copy files** task using the following inputs:

- Contents: All of the files to be copied for publishing them as an artifact
- Target folder: The folder that the files get copied to
  - For example: `$(Build.ArtifactStagingDirectory)`

5. Add **Publish build artifacts** to publish the artifacts for use in other jobs or pipelines. Use the following inputs:

- Path to publish: The path to the folder that contains the files that are being published
  - For example: `$(Build.ArtifactStagingDirectory)`
- Artifact name: The name given to the artifact
- Artifacts publish location: Choose **Azure Pipelines** to use the artifact in future jobs

## Stage 3: Download build artifacts and publish the extension

1. To install the tfx-cli onto your build agent, add **Use Node CLI for Azure DevOps (tfx-cli)**.
2. To download the artifacts onto a new job, add the **Download build artifacts** task using the following inputs:

- Download artifacts produced by: If you're downloading the artifact on a new job from the same pipeline, select **Current build**. If you're downloading on a new pipeline, select **Specific build**
- Download type: Choose **Specific artifact** to download all files that were published.
- Artifact name: The published artifact's name
- Destination directory: The folder where the files should be downloaded

3. To get the **Publish Extension** task, use the following inputs:

- Connect to: Visual Studio Marketplace
- Visual Studio Marketplace connection: ServiceConnection
- Input file type: VSIX file
- VSIX file: `/Publisher.*.vsix`
- Publisher ID: ID of your Visual Studio Marketplace publisher
- Extension ID: ID of your extension in the `vss-extension.json` file
- Extension Name: Name of your extension in the `vss-extension.json` file
- Extension visibility: Either private or public

## Optional: Install and test your extension

After you publish your extension, it needs to be installed in Azure DevOps organizations.

### Install extension to organization

Install your shared extension in a few steps:

1. Go to **Organization settings** and select **Extensions**.
2. Locate your extension in the **Extensions Shared With Me** section:
  - Select the extension link
  - Select **Get it free or Install**
3. Check that the extension appears in your **Installed** extensions list:
  - Confirm it's available in your pipeline task library

#### Note

If you don't see the **Extensions** tab, ensure you're at the organization administration level ([https://dev.azure.com/{organization}/\\_admin](https://dev.azure.com/{organization}/_admin)) and not at the project level.

# End-to-end testing

After installation, perform comprehensive testing:

## 1. Create a test pipeline:

- Add your custom task to a new pipeline
- Configure all input parameters
- Test with various input combinations

## 2. Validate functionality:

- Run the pipeline and monitor execution
- Check task outputs and logs
- Verify error handling with invalid inputs

## 3. Test performance:

- Test with large input files (if applicable)
- Monitor resource usage
- Validate timeout behavior

# Frequently asked questions

## Q: How is task cancellation handled?

A: The pipeline agent sends `SIGINT` and `SIGTERM` signals to task processes. While the [task library](#) doesn't provide explicit cancellation handling, your task can implement signal handlers. For details, see [Agent jobs cancellation](#).

## Q: How can I remove a task from my organization?

A: Automatic deletion isn't supported as it would break existing pipelines. Instead:

1. Deprecate the task: [Mark the task as deprecated](#)
2. Version management: [Bump the task version](#)
3. Communication: Notify users about the deprecation timeline

## Q: How can I upgrade my task to the latest Node.js version?

A: Upgrade to [the latest Node version](#) for better performance and security. For migration guidance, see [Upgrading tasks to Node 20](#).

Support multiple Node versions by including multiple execution sections in `task.json`:

JSON

```
"execution": {  
  "Node20_1": {  
    "target": "index.js"  
  },  
  "Node10": {  
    "target": "index.js"  
  }  
}
```

Agents with Node 20 use the preferred version, while older agents fall back to Node 10.

To upgrade your tasks:

- To ensure your code behaves as expected, test your tasks on the various Node runner versions.
- In your task's execution section, update from `Node` or `Node10` to `Node16` or `Node20`.
- To support older server versions, you should leave the `Node`/`Node10` target. Older Azure DevOps Server versions might not have the latest Node runner version included.
- You can choose to share the entry point defined in the target or have targets optimized to the Node version used.

JSON

```
"execution": {  
  "Node10": {  
    "target": "bash10.js",  
    "argumentFormat": ""  
  },  
  "Node16": {  
    "target": "bash16.js",  
    "argumentFormat": ""  
  },  
  "Node20_1": {  
    "target": "bash20.js",  
    "argumentFormat": ""  
  }  
}
```

 **Important**

If you don't add support for the Node 20 runner to your custom tasks, they fail on agents installed from the `pipelines-agent-*` [release feed](#).

## Related content

- [Extension manifest reference](#)
- [Integrate custom build pipeline tasks with extensions](#)
- [Build/Release task examples ↗](#)

ⓘ **Note:** The author created this article with assistance from AI. [Learn more](#)

# Author a pipeline decorator

08/29/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022

Pipeline decorators let you add steps to the beginning and end of every job. The process of authoring a pipeline decorator is different than adding steps to a single definition because it applies to all pipelines in an organization.

Suppose your organization requires running a virus scanner on all build outputs that could be released. Pipeline authors don't need to remember to add that step. We create a decorator that automatically injects the step. Our pipeline decorator injects a custom task that does virus scanning at the end of every pipeline job.

## 💡 Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## 1. Add contributions to an extension

The following example assumes you're familiar with the [contribution models](#).

1. [Create an extension](#). Once your extension gets created, you have a `vss-extension.json` file.
2. Add contributions to the `vss-extension.json` file for our new pipeline decorator.

### `vss-extension.json`

JSON

```
{  
  "manifestVersion": 1,  
  "contributions": [  
    {  
      "id": "my-required-task",  
      "type": "ms.azure-pipelines.pipeline-decorator",  
      "targets": [  
        "ms.azure-pipelines-agent-job.post-job-tasks"  
      ],  
      "properties": {  
        "template": "my-decorator.yml"  
      }  
    }]
```

```

        }
    ],
    "files": [
        {
            "path": "my-decorator.yml",
            "addressable": true,
            "contentType": "text/plain"
        }
    ]
}

```

## Contribution options

Let's take a look at the properties and what they're used for:

[\[ \] Expand table](#)

Property	Description
<code>id</code>	Contribution identifier. Must be unique among contributions in this extension.
<code>type</code>	Specifies that this contribution is a pipeline decorator. Must be the string <code>ms.azure-pipelines.pipeline-decorator</code> .
<code>targets</code>	Decorators can run before your job/specifed task, after, or both. See the following table for available options.
<code>properties.template</code>	(Required) The template is a YAML file included in your extension, which defines the steps for your pipeline decorator. It's a relative path from the root of your extension folder.
<code>properties.targettask</code>	The target task ID used for <code>ms.azure-pipelines-agent-job.pre-task-tasks</code> or <code>ms.azure-pipelines-agent-job.post-task-tasks</code> targets. Must be GUID string like <code>89b8ac58-8cb7-4479-a362-1baaacc6c7ad</code>

## Targets

[\[ \] Expand table](#)

Target	Description
<code>ms.azure-pipelines-agent-job.pre-job-tasks</code>	Run before other tasks in a classic build or YAML pipeline. Due to differences in how source code checkout happens, this target runs after checkout in a YAML pipeline but before checkout in a classic build pipeline.
<code>ms.azure-pipelines-agent-job.post-checkout-</code>	Run after the last <code>checkout</code> task in a classic build or YAML pipeline.

Target	Description
<code>tasks</code>	
<code>ms.azure-pipelines-agent.job.post-job-tasks</code>	Run after other tasks in a classic build or YAML pipeline.
<code>ms.azure-pipelines-agent.job.pre-task-tasks</code>	Run before specified task in a classic build or YAML pipeline.
<code>ms.azure-pipelines-agent.job.post-task-tasks</code>	Run after specified task in a classic build or YAML pipeline.
<code>ms.azure-release-pipeline-agent.job.pre-task-tasks</code>	Run before specified task in a classic RM pipeline.
<code>ms.azure-release-pipeline-agent.job.post-task-tasks</code>	Run after specified task in a classic RM pipeline.
<code>ms.azure-release-pipeline-agent.job.pre-job-tasks</code>	Run before other tasks in a classic RM pipeline.
<code>ms.azure-release-pipeline-agent.job.post-job-tasks</code>	Run after other tasks in a classic RM pipeline.

### (!) Note

Deployment jobs in a YAML pipeline only support `ms.azure-pipelines-agent.job.pre-job-tasks` and `ms.azure-pipelines-agent.job.post-job-tasks` targets. Jobs support all YAML pipeline targets. Deployment jobs are not supported in classic release pipelines.

In this example, we use `ms.azure-pipelines-agent.job.post-job-tasks` because we want to run at the end of all build jobs.

This extension contributes a pipeline decorator. Next, we create a template YAML file to define the decorator's behavior.

## 2. Create a decorator YAML file

In the extension's properties, we chose the name "my-decorator.yml". Create that file in the root of your contribution. It holds the set of steps to run after each job. We start with a basic example and work up to the full task.

## my-decorator.yml (initial version)

YAML

```
steps:
- task: CmdLine@2
  displayName: 'Run my script (injected from decorator)'
  inputs:
    script: dir
```

### ! Note

Pipeline decorator tasks with service connection usage are not supported for classic release pipelines.

## 3. Install the decorator

To add a pipeline decorator to your organization, you must [install an extension](#). **Only private extensions can contribute pipeline decorators**. The extension must be authored and shared with your organization before it can be used.

Once the extension has been shared with your organization, [search for the extension](#) and install it.

Save the file, then [build and install the extension](#). Create and run a basic pipeline. The decorator automatically injects our `dir` script at the end of every job. A pipeline run looks similar to the following example.

- ✓ Prepare job · succeeded
- ✓ Initialize job · succeeded
- ✓ Checkout · succeeded
- ✓ Command Line Script · succeeded
- ✓ Bash Script · succeeded
- ✓ Run my script (injected from decorator) · succeeded
- ✓ Post-job: Checkout · succeeded
- ✓ Report build status · succeeded

⚠ Note

The decorator runs on every job in every pipeline in the organization. In later steps, we add logic to control when and how the decorator runs.

## 4. Inject conditions

In our example, we only need to run the virus scanner if the build outputs might be released to the public. Let's say that only builds from the default branch (typically `main`) are ever released. We should limit the decorator to jobs running against the default branch.

The updated file looks like this:

### my-decorator.yml (revised version)

YAML

```
steps:
- ${{ if eq(resources.repositories['self'].ref,
resources.repositories['self'].defaultBranch) }}:
  - task: CmdLine@2
    displayName: 'Run my script (injected from decorator)'
    inputs:
      script: dir
```

You can start to see the power of this extensibility point. Use the context of the current job to conditionally inject steps at runtime. Use [YAML expressions](#) to make decisions about what steps to inject and when. See [pipeline decorator expression context](#) for a full list of available data.

There's another condition we need to consider: what if the user already included the virus scanning step? We shouldn't waste time running it again. In this simple example, we'll pretend that any `script` task found in the job is running the virus scanner. (In a real implementation, you'd have a custom task to check for that instead.)

The script task's ID is `d9bafed4-0b18-4f58-968d-86655b4d2ce9`. If we see another script task, we shouldn't inject ours.

## my-decorator.yml (final version)

YAML

```
steps:
- ${{ if and(eq(resources.repositories['self'].ref,
resources.repositories['self'].defaultBranch),
not(containsValue(job.steps.*.task.id, 'd9bafed4-0b18-4f58-968d-86655b4d2ce9')))}
}:
  - task: CmdLine@2
    displayName: 'Run my script (injected from decorator)'
    inputs:
      script: dir
```

## 5. Specify a target task

You can specify target [task ID](#), and inject tasks before or after this target task. To specify target task, you can modify `vss-extension.json` manifest file like the following example.

### vss-extension.json

## JSON

```
{  
  "contributions": [  
    {  
      "id": "my-required-task",  
      "type": "ms.azure-pipelines.pipeline-decorator",  
      "targets": [  
        "ms.azure-pipelines-agent-job.pre-task-tasks",  
        "ms.azure-pipelines-agent-job.post-task-tasks"  
      ],  
      "properties": {  
        "template": "my-decorator.yml",  
        "targettask": "target-task-id"  
      }  
    },  
    ...  
  ]  
}
```

When you set up the 'targettask' property, you can specify ID of a target task. Tasks will be injected before/after all instances of specified target task.

## Specify target task's inputs injection

You can specify a list of inputs of the target task that you want to inject as inputs to the injected task.

This feature is designed to work with [custom pipeline tasks](#). It isn't intended to provide access to target pipeline task inputs via pipeline variables.

To get access to the target pipeline task inputs (inputs with the `target_` prefix), the injected pipeline task should use methods from the [azure-pipelines-tasks-task-lib](#), and not the pipeline variables, for example `const inputString = tl.getInput('target_targetInput');`.

To do so, you can create your own custom pipeline [task](#) and use the target inputs there. If you need the functionality of one of the out-of-box tasks, like [CmdLine@2](#), you can create a copy of the [CmdLine@2 task](#) and publish it with your decorator extension.

### ⓘ Note

This functionality is only available for tasks that are injected before or after the target task.

To specify this list of inputs, you can modify `vss-extension.json` manifest file like the following example.

## vss-extension.json (injected task inputs version)

JSON

```
{  
  "contributions": [  
    {  
      "id": "my-required-task",  
      "type": "ms.azure-pipelines.pipeline-decorator",  
      "targets": [  
        "ms.azure-pipelines-agent-job.pre-task-tasks",  
        "ms.azure-pipelines-agent-job.post-task-tasks"  
      ],  
      "properties": {  
        "template": "my-decorator.yml",  
        "targettask": "target-task-id",  
        "targettaskinputs": ["target-task-input", "target-task-second-  
input"]  
      }  
    },  
    ...  
  ]  
}
```

By setting up of 'targettaskinputs' property, you can specify the list of inputs that are expected to inject. These inputs will be injected into the task with the prefix "`target_`" and will be available in the injected task like `target_target-task-input`.

### ➊ Note

Target task inputs that get secret values with variables or get them from other tasks won't be injected.

## Debug

You might need to debug when you create your decorator. You also might want to see what data you have available in the context.

You can set the `system.debugContext` variable to `true` when you queue a pipeline. Then, look at the pipeline summary page.

You see something similar to the following image.

✓	Prepare job	· succeeded
✓	Initialize job	· succeeded
⌚	Pipeline decorator context (macOS/Linux)	· skipped ⓘ
✓	Pipeline decorator context (Windows)	· succeeded
✓	Checkout	· succeeded
✓	Command Line Script	· succeeded
✓	Virus Scan (injected from decorator)	· succeeded
✓	Post-job: Checkout	· succeeded
✓	Report build status	· succeeded

Select the task to see the logs, which show runtime values and that the context is available.

## Related content

- [About YAML expression syntax](#)
- [Pipeline decorator expression context](#)
- [Develop a web extension](#)
- [Authentication guide](#)

# Extension samples

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Start developing your extension by working from a sample.

## Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

The following Microsoft samples show the capabilities of the extension framework and how to contribute to various areas. Each sample illustrates one or more contributions. We limit the number of contributions for each sample to increase understanding of the extension framework. For source information, see the [Azure DevOps extension samples repo ↗](#).

[ ] Expand table

Sample	Contributions	Description
BreadcrumbService	Breadcrumb Service, Hub	Adds a breadcrumb service, which adds a "Sample Breadcrumb Item" to the sample hub. To see this item, go to the <b>Sample Hub</b> in the <b>Pipelines</b> hub group.
CodeEditorContribution	Code Editor	Adds a language definition and a JSON schema for the code editor.
Feature	Feature, Hub, Property Provider	Shows how to hook into the <b>Preview Features</b> panel under the user profile menu. Adds a simple hub that only shows when you turn on an "ABC" feature. You can toggle the feature on and off, per user or per organization.
Hub	Hub	Adds a hub named <b>Sample Hub</b> into the <b>Pipelines</b> hub group. The Sample Hub is on a project-level page, under the <b>Pipelines</b> navigation element.
Menu	Build Definition Menu Item	Adds a <b>Sample build definition menu item</b> to the <b>Builds</b> hub in the dropdown actions menu. The menu handler gets the current build definition from the context passed, makes a REST call, and then shows the result in a message box.
Panel	Panel Content, Hub	Applied within the <b>Hub</b> sample. Contains a toggle button along with <b>OK</b> and <b>Cancel</b> buttons. Can be used as a custom

Sample	Contributions	Description
		panel or dialog content.
Pivot	Web Tab	Adds a <b>Sample Pivot</b> tab to the organization or project collection home page, next to <b>Projects</b> , <b>My work items</b> , and <b>My pull requests</b> .
Pills	Pill Provider	Adds pills to the title of the Pipeline definition (Runs) page.
QueryParamsHandler	Event Subscription	Adds a service that loads on any page whenever a <code>showMyPanel</code> query parameter presents in the URL when any page gets loaded. The startup service shows the custom panel from the Panel sample, using an optional <code>myPanelTitle</code> query parameter as the panel title.
RepositoryActions	Menu Item	Adds a <b>Sample repository action</b> menu item to the repository picker in the header of code hub pages. If the <code>href</code> property shows, select the action to go to the given URL. If the <code>uri</code> property is provided, that code executes when you select the action.
RepositoryServiceHub	Hub	Adds a <b>Repository Information</b> hub to the <b>Code</b> hub group. Demonstrates how to interact with the <code>IVersionControlRepositoryService</code> to obtain basic information about a user's Git repository.
WorkItemFormGroup	Work Item Form Group	Adds a <b>Sample WorkItem Form Group</b> extension to the work item form to show how to interact with the <code>IWorkItemFormService</code> service and <code>IWorkItemNotificationListener</code> . Provides a UI to show case how to change field values using the form service and displaying work item form notification events.
WorkItemOpen	Hub	Adds a <b>Sample WorkItem Open</b> hub to the <b>Boards</b> hub group to show how to interact with the <code>IWorkItemFormNavigationService</code> service. Provides a UI for you to open an existing work item by ID, or open the work item form for a new work item by work item type. Either of these options open a dialog in the host frame.

## DevLabs examples

Other open source examples that you might be interested in.

 Expand table

Sample	Source	Contributions	Description
<a href="#">Team Calendar</a>	<a href="#">GitHub</a>	Hub, Event sources	Track events important to your team, view and manage days off, quickly see when sprints start and end, and more.
<a href="#">WSJF (Weighted Shortest Job First)</a>	<a href="#">GitHub</a>	Notification (work item), Context menu action (work item)	Auto calculates WSJF (weighted shortest job first) per work item and stores it in a work item field.
<a href="#">Cascading Lists</a>	<a href="#">GitHub</a>	Work Item Form	Define cascading behavior for picklists in work item form.
<a href="#">Retrospectives</a>	<a href="#">GitHub</a>	Hub	First-class experience for retrospectives and general feedback board scenarios. Collect feedback on your project milestones, organize and prioritize, and create and track actionable tasks, which can help your team improve over time.
<a href="#">Estimate</a>	<a href="#">GitHub</a>	Hub, Work item action menu	Play Planning Poker in Azure DevOps. Select work from an iteration, query, or your backlog, estimate the effort of those items with your team, and immediately update the work items.
<a href="#">Multivalue control</a>	<a href="#">GitHub</a>	Work item form	A work item form control, which allows selection of multiple values.
<a href="#">Azure DevOps Extension Tasks</a>	<a href="#">GitHub</a>	Build and release tasks	Azure Pipelines tasks for packaging and publishing Azure Devops and Visual Studio extensions to the Visual Studio Marketplace.

## Get started

To get started as quickly as possible, use the [seed project](#) that contains the files required to build an extension using TypeScript. There's a grunt script to automate building, packaging, and publishing the extension.

## Related content

- [Develop a web extension](#)
- [Use the Developer Formula Design System](#)
- [Use the Azure DevOps sample extensions repo](#)
- [View the contribution model](#)
- [Package and publish extensions](#)

# Vertical navigation guidance

Article • 10/04/2022

## Azure DevOps Services

Vertical navigation brings with it changes that affect some extensions. These outcomes include support for extension icons along with changes to team context.

### Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Team context

In traditional horizontal navigation, users could go to a project or team by selecting from a picker that's in the top left of the page header. This picker presented a list of recent teams and a way to browse for all teams. In the new vertical navigation, a user can only navigate into a project (and not into a team). This change was made to simplify the overall experience. But, it introduced a challenge for web extensions that rely on users' ability to switch teams using the traditional team picker in the page header.

`SDK.getWebContext()` is a client-side API provided by the SDK that provides information about the current organization, project, and team the user is operating in:

JSON

```
{  
  "account": {  
    "name": "Fabrikam",  
    "id": "50e49b94-4bb6-40e7-8c85-a6d756d8a4d8"  
  },  
  "project": {  
    "name": "Fabrikam Dev",  
    "id": "049b1af2-fc87-4e50-95e4-151357f04b7f"  
  },  
  "team": {  
    "name": "Ops Team",  
    "id": "9b3a6b80-fe95-4c8c-8aa1-1e5d535d7af1"  
  }  
}
```

We don't recommend relying on `SDK.getWebContext().team`. Instead, follow the guidance below, based on the category your extension falls under.

## Hub extensions that are team aware

If your extension needs to provide users a way to select a team, you can use the Teams REST API to get a list of teams for the current project. The following example shows how to call this API from your extension.

JavaScript

```
import { getClient } from "azure-devops-extension-api";
import { CoreRestClient } from "azure-devops-extension-api/Core";
import * as SDK from "azure-devops-extension-sdk";

private async getTeams() {
    const client = getClient(CoreRestClient);

    client.getTeams(SDK.getWebContext().project.id).then(
        function(teams) {
            console.log(teams);
        }
    );
}
```

For an example of an extension that provides a team picker control, see [Team Calendar](#).

## Pivots/Panels extensions that are in team aware hubs like Backlogs and Dashboard

Your extension can check the *configuration* object passed to your contribution. This object has different properties depending on the contribution type and where the contribution is hosted. Example shows reading team from the *configuration* and falling back to reading team from *webContext* to support both new vertical navigation and older horizontal navigation in on-premise releases.

JavaScript

```
function getCurrentTeam() {
    let webContext = SDK.getWebContext();
    let configuration = SDK.getConfiguration();

    if ("team" in configuration) {
        return configuration.team;
    } else if ("team" in webContext) {
```

```
        return webContext.team;
    } else {
        return null; // should only happen if not in a project context
    }
}
```

## Actions extensions that are in team aware hubs like Backlogs and Dashboard

Your extension can check the *actionContext* object passed to the callback invoked when a user selects the contributed menu item. Example shows reading team from the *actionContext*.

JavaScript

```
var menuContributionHandler = (function () {
    "use strict";
    return {
        // This is a callback that gets invoked when a user selects the
        // newly contributed menu item
        execute: function (actionContext) {
            if("team" in actionContext) {
                alert(`Team. Id is ${actionContext.team.id}, Name is
${actionContext.team.name}`);
            }
        }
    };
})();
```

## Hub icon

You can optionally set an asset (like a .png or .jpg) as the icon for your hub. This icon appears next to the hub in the vertical navigation bar. It must be packaged with your extension.

### ⓘ Note

These icons don't appear in horizontal navigation.

Complete the following steps to set an icon for your hub.

1. Set the `iconAsset` property of the hub contribution to the fully qualified asset identifier, which follows the pattern: `{publisher-id}.{extension-id}/{asset-path}`.

2. Add an entry for this asset in the `includesData` contribution property.
3. Package the asset with your extension by listing it in the `files` property at the root of your manifest.

### Example #1:

JSON

```
{
  "id": "my-extension",
  "publisherId": "my-publisher",
  ...
  "contributions": [
    {
      "id": "example-hub",
      "type": "ms.vss-web.hub",
      "targets": [
        "ms.vss-code-web.code-hub-group"
      ],
      "properties": {
        "name": "My Hub",
        "iconAsset": "my-publisher.my-extension/images/fabrikam-
logo.png",
        "includesData": {
          "assets": [
            "my-publisher.my-extension/images/fabrikam-logo.png"
          ]
        }
      }
    },
    {
      "files": [
        {
          "path": "images/fabrikam-logo.png",
          "addressable": true
        }
      ]
    }
  ]
}
```

### Example #2:

When themes like light versus dark get applied, you can specify the icons in your extension manifest as follows.

JSON

```
{
  "id": "hub",
  "type": "ms.vss-web.hub",
```

```
"targets": [
    "ms.vss-work-web.work-hub-group"
],
"properties": {
    "name": "Hub",
    "description": "Something",
    "uri": "pages/Hub.html",
    "icon": {
        "light": "img/hub-light.png",
        "dark": "img/hub-dark.png"
    }
}
```

# Work with URLs in extensions and integrations

Article • 04/17/2025

## Azure DevOps Services

With Azure DevOps, organizational resources and APIs are accessible through either of the following URLs:

- `https://dev.azure.com/{organization}` (new)
- `https://{{organization}}.visualstudio.com` (legacy)

Users, tools, and integrations can interact with organization-level REST APIs using either URL, regardless of when the organization was created.

This article explains how you can best work with URLs in extensions, integrations, or tools. For more information, see the [Azure DevOps Services REST API Reference](#).

## Primary URL

Each organization has a designated *primary* URL that's in either the new form or the legacy form. The primary URL is used by Azure DevOps to construct URLs in certain scenarios. The default primary URL for an organization is determined by when the organization was created, but can be changed by an administrator:

[ ] [Expand table](#)

When the organization was created	Default primary URL
On or after 9/10/2018	New
Before 9/10/2018	Legacy

## How the primary URL is used

The primary URL is the base URL for all URLs constructed by Azure DevOps in background jobs and other automated scenarios. See the following examples.

- URLs provided to Azure Pipelines tasks via environment variables (like `SYSTEM_TEAMFOUNDATIONCOLLECTIONURI`)
- URLs included in service hooks event payloads (like URLs in `resourceContainers`)
- URLs in email, Slack, Microsoft Teams, and similar notifications

For example, the following task snippet displays the organization URL provided to the task:

```
PowerShell
```

```
$orgUrl = $env:SYSTEM_TEAMFOUNDATIONCOLLECTIONURI  
Write-Host $orgUrl
```

If this task is executed on an organization where the primary URL is in the new URL form, the output is `https://dev.azure.com/{organization}`. The same task executed on an organization where the primary URL is the legacy URL form outputs

```
https://{{organization}}.visualstudio.com.
```

Therefore, it's important that Azure Pipelines tasks and services that receive events from service hooks handle both URL forms.

## URLs returned in REST APIs

Regardless of an organization's primary URL, URLs returned in the response to a REST API call use the same base URL as the requesting URL. This function ensures that clients calling a REST API using a legacy URL continue to get back URLs in the same (legacy) form. For example, when the Projects REST API is called using a legacy URL, URLs in the response use the legacy form:

## Request

```
HTTP
```

```
GET https://Fabrikam.visualstudio.com/_apis/projects/MyProject
```

## Response

```
JavaScript
```

```
{  
  "id": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5c5",  
  "name": "MyProject",  
  "url": "https://Fabrikam.visualstudio.com/_apis/projects/MyProject"  
}
```

Calling the same API using the new URL

(`https://dev.azure.com/Fabrikam/_apis/projects/MyProject`) results in URLs returned in the

new URL form.

## Best practices

To ensure your extension, tool, or integration is resilient to changing organization URL forms and to possible future changes to the location (domain) of a REST API:

- Assume the form of the organization URL can change over time.
- Avoid parsing a URL to construct another URL.
- Don't assume a particular REST API always resides on the same domain.
- Avoid storing URLs in your service.
- When possible, use Microsoft-provided [.NET](#), [TypeScript \(web\)](#), [Node.js](#) , and [Python](#) client libraries with Azure DevOps.

## How to get an organization's URL

With just the organization's name or ID, you can get its base URL by using the global Resource Areas REST API ([https://dev.azure.com/\\_apis/resourceAreas](https://dev.azure.com/_apis/resourceAreas)). This API doesn't require authentication. It also provides information about the location (URL) of the organization and the base URL for REST APIs, which can live on different domains.

A resource area is a group of related REST API resources and endpoints. Each resource area has a well-known identifier. Each resource area has an organization-specific base URL that can be used to form URLs for APIs in that resource area. For example, the base URL for *build* REST APIs for the Fabrikam might be <https://dev.azure.com/Fabrikam>, but the base URL for *release management* REST APIs might be <https://vsrm.dev.azure.com/Fabrikam>.

### ! Note

The Resource Areas REST API returns URLs for the organization based on that organization's designated primary URL.

## Using the organization's name

There are a few ways to get the base URL for an organization by using its name.

HTTP

## Request

Replace `{organizationName}` with organization's name, for example *Fabrikam*. `79134C72-4A58-4B42-976C-04E7115F32BF` is the ID for the *core* resource area, which is where important resources like *projects* are.

HTTP

```
GET https://dev.azure.com/_apis/resourceAreas/79134C72-4A58-4B42-976C-04E7115F32BF  
?accountName={organizationName}&api-version=5.0-preview.1
```

## Response

JavaScript

```
{  
  "id": "79134C72-4A58-4B42-976C-04E7115F32BF",  
  "name": "Core",  
  "locationUrl": "https://dev.azure.com/Fabrikam"  
}
```

The `locationUrl` reflects the organization's base URL.

## Using the organization's ID

To get the URL for an organization by using its GUID identifier, use the `hostId` query parameter in the previous examples (instead of `accountName`). For example:

HTTP

```
GET https://dev.azure.com/_apis/resourceAreas/79134C72-4A58-4B42-976C-04E7115F32BF?hostId={organizationId}&api-version=5.0-preview.1
```

## How to get the base URL for a REST API

Starting from an organization's URL, you can use the Resource Areas REST API to look up the correct base URL for any REST API you need to call. This process ensures that your code is

resilient to the location (domain) of a REST API changing in the future and avoids potentially brittle logic.

#### (!) Note

If you're using the Microsoft-provided .NET, TypeScript (web), Node.js, or Python client library, URL lookup is handled for you. For example, when you construct a `VssConnection` and call `GetClient<T>` in .NET, the returned client is properly bound to the correct base URL for the REST APIs called by this client.

If you aren't using a Microsoft-provided client library:

1. Use the [following table](#) to find the resource area ID for the REST API you need to call. The resource area name usually appears after `/_apis/` in the REST API route. For example, the `/_apis/release/definitions` REST API belongs to the `release` resource area, which has an ID of `efc2f575-36ef-48e9-b672-0c6fb4a48ac5`.
2. Call the organization-level Resource Areas REST API

```
({organizationUrl}/_apis/resourceAreas/{resourceAreaId}?api-version=5.0-preview.1)
```

and pass the resource area ID. For example:

HTTP

```
GET https://dev.azure.com/Fabrikam/_apis/resourceAreas/efc2f575-36ef-48e9-b672-0c6fb4a48ac5?api-version=5.0-preview.1
```

3. Use the `locationUrl` field from the JSON response as the base URL for calling other REST APIs for this area. In this example, the base URL for Release Management REST APIs is `https://vsrm.dev.azure.com/Fabrikam`.

Like the global Resource Areas REST API described earlier, no credentials are required to call the organization-level Resource Areas REST API.

## Example: Pipelines task calling an Azure Pipelines releases REST API

In this example, a build task needs to call the Azure Pipelines releases REST API. The task forms the correct base URL for this REST API call by using the organization URL (provided in an environment variable) and the Resource Areas REST API.

#### (!) Note

Resource area IDs are fixed and can be safely embedded in tasks and other logic.

PowerShell

```
$orgUrl = $env:SYSTEM_TEAMFOUNDATIONCOLLECTIONURI
$releaseManagementAreaId = "efc2f575-36ef-48e9-b672-0c6fb4a48ac5"

# Build the URL for calling the org-level Resource Areas REST API for the RM APIs
$orgResourceAreasUrl = [string]::Format("{0}/_apis/resourceAreas/{1}?api-
preview=5.0-preview.1", $orgUrl, $releaseManagementAreaId)

# Do a GET on this URL (this returns an object with a "locationUrl" field)
$results = Invoke-RestMethod -Uri $orgResourceAreasUrl

# The "locationUrl" field reflects the correct base URL for RM REST API calls
$rmUrl = $results.locationUrl

# Construct the URL to the release definitions REST API
$releaseDefinitionsUrl = [string]::Format("{0}/_apis/release/definitions?api-
preview=5.0-preview.1", $rmUrl)
```

## Reference: Resource area IDs

This table shows the IDs for common resource areas. See the previous section for details on how to use this table.

 Note

Resource area IDs are fixed and are consistent across all organizations in Azure DevOps Services.

 Expand table

Resource area ID	Name
0d55247a-1c47-4462-9b1f-5e2125590ee6	account
5d6898bb-45ec-463f-95f9-54d49c71752e	build
79bea8f8-c898-4965-8c51-8bbc3966faa8	collection
79134c72-4a58-4b42-976c-04e7115f32bf	core
31c84e0a-3ece-48fd-a29d-100849af99ba	dashboard
a0848fa1-3593-4aec-949c-694c73f4c4ce	delegatedAuth

Resource area ID	Name
6823169a-2419-4015-b2fd-6fd6f026ca00	discussion
a85b8835-c1a1-4aac-ae97-1c3d0ba72dbd	distributedtask
7bf94c77-0ce1-44e5-a0f3-263e4ebbf327	drop
6c2b0933-3600-42ae-bf8b-93d4f7e83594	extensionManagement
67349c8b-6425-42f2-97b6-0843cb037473	favorite
4e080c62-fa21-4fbc-8fef-2a10a2b38049	git
4e40f190-2e3f-4d9f-8331-c7788e833080	graph
68ddce18-2501-45f1-a17b-7931a9922690	memberEntitlementManagement
b3be7473-68ea-4a81-bfc7-9530baaa19ad	NuGet
4c83cf1-f33a-477e-a789-29d38ffca52e	npm
45fb9450-a28d-476d-9b0f-fb4aedddff73	package
7ab4e64e-c4d8-4f50-ae73-5ef2e21642a5	packaging
2e0bf237-8973-4ec9-a581-9c3d679d1776	pipelines
fb13a388-40dd-4a04-b530-013a739c72ef	policy
8ccfef3d-2b87-4e99-8ccb-66e343d2daa8	profile
efc2f575-36ef-48e9-b672-0c6fb4a48ac5	release
57731fdf-7d72-4678-83de-f8b31266e429	reporting
ea48a0a1-269c-42d8-b8ad-ddc8fcdfc578	search
3b95fb80-fdda-4218-b60e-1052d070ae6b	test
c83eaf52-edf3-4034-ae11-17d38f25404c	testresults
8aa40520-446d-40e6-89f6-9c9f9ce44c48	tfvc
970aa69f-e316-4d78-b7b0-b7137e47a22c	user
5264459e-e5e0-4bd8-b118-0985e68a4ec5	wit
1d4f49f9-02b9-4e26-b826-2cdb6195f2a9	work
85f8c7b6-92fe-4ba6-8b6d-fbb67c809341	worktracking

# Next step

Make your extension or integration public

# Data storage

Article • 01/29/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Azure DevOps extensions can store user preferences and complex data structures directly on Microsoft-provided infrastructure, which ensures your user's data is secure and backed up just like other organization and project data. It also means for simple data storage needs, you, as the extension provider, aren't required to set up, manage, or pay for third-party data storage services.

There are two methods to engage with the data storage service: through REST APIs or via a client service provided by Microsoft, which is part of the VSS SDK. We advise extension developers to utilize the provided client service APIs, as they offer a user-friendly encapsulation of the REST APIs.

## ⓘ Note

Looking for Azure DevOps REST APIs? See the latest [Azure DevOps REST API reference](#).

For information about .NET client libraries, see [.NET client libraries for Azure DevOps](#).

## What you can store

The service is designed to let you store and manage two different types of data:

- **Settings:** simple key-value settings (like user preferences)
- **Documents:** collections of similar complex objects (documents)

A collection is as an indexed container for documents. A document is a JSON blob that belongs to a collection. Other than a few reserved property names, you control and manage the schema of these documents.

## How you can scope data

Settings and document collections can be scoped to either the:

- **Project Collection:** shared by all users of the project collection to which the extension is installed

- **User**: a single user of a project collection to which the extension is installed

## Settings storage

The two main methods for managing settings are `getValue()` and `setValue()`:

- `getValue()` accepts a string key (along with other options like scope) and returns an IPromise. The resolved value of this promise is the value associated with the provided key.
- `setValue()` accepts a string key, a value, and other options such as scope, and returns an IPromise. The resolved value of this promise is the updated value of the setting.

Here's an example of how to set a value:

JavaScript

```
private async initializeState(): Promise<void> {
    await SDK.ready();
    const accessToken = await SDK.getAccessToken();
    const extDataService = await SDK.getService<IExtensionDataService>
(CommonServiceIds.ExtensionDataService);
    this._dataManager = await
extDataService.getExtensionDataManager(SDK.getExtensionContext().id,
accessToken);

    this._dataManager.getValue<string>("test-id").then((data) => {
        this.setState({
            dataText: data,
            persistedText: data,
            ready: true
        });
    }, () => {
        this.setState({
            dataText: "",
            ready: true
        });
    });
}
```

Here's an example of how to retrieve a setting value:

JavaScript

```
// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    // Get value in user scope
    dataService.getValue("userScopedKey", {scopeType:
```

```
"User"}).then(function(value) {
    console.log("User scoped key value is " + value);
});
```

If `scopeType` isn't specified, the settings are stored at the project collection level and they're accessible to all users in that project collection using the extension. Here's an example of how to set a setting value at the project collection level:

JavaScript

```
// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    // Set value (default is project collection scope)
    dataService.setValue("someKey", "abcd-efgh").then(function(value) {
        console.log("Key value is " + value);
    });
});
```

## Data (collections of documents) storage

To handle more complex data beyond key-value pairs, you can utilize the concept of documents to execute CRUD operations on your extension's data. A document is a JSON blob, enhanced with two special properties: ID and `_etag`. If they're crucial to an extension's data model, IDs can be user-defined, or if left unspecified, the system generates them. These IDs must be unique within a specific collection. Since a collection refers to a particular scope and instance of an extension, it implies that the same document ID can be reused across different collections.

The following document operations are available:

- Get a document
- Create a document
- Set a document (create or update)
- Update a document
- Delete a document

There's also a single operation that can be performed on a collection: Get all documents

### Get a document by ID

Obtaining a document from a collection using its identifier is straightforward, like the following example:

JavaScript

```
// Acquire data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    // Retrieve document by id
    dataService.getDocument("MyCollection",
"MyDocumentId").then(function(doc) {
        // Assuming document has a property named foo
        console.log("Doc foo: " + doc.foo);
    });
});
```

This operation tries to fetch a document with the ID "MyDocumentId" from the "MyCollection" collection. In the absence of a provided scope, the service defaults to using the collection scoped to the entire instance of this extension. If either this collection or a document with the specified ID doesn't exist, a 404 error is returned, which the extension should handle. The returned document is a JSON object that includes all its properties, along with the special ID and `_etag` properties utilized by the data storage service.

JavaScript

```
// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    // Get document by id
    dataService.getDocument("MyCollection",
"MyDocumentId").then(function(doc) {
        // Assuming document has a property named foo
        console.log("Doc foo: " + doc.foo);
    });
});
```

This call attempts to retrieve a document with the ID "MyDocumentId," from the collection "MyCollection." Since no scope is provided, the collection that the service uses gets scoped to the default of the entire instance of this extension. If this collection doesn't exist or a document with that ID doesn't exist, then a 404 gets returned, which the extension should handle. The document that is returned is a JSON object containing all of its own properties, in addition to the special ID and `_etag` properties used by the data storage service.

## Create a document

To create a new document, perform a call like the following example:

JavaScript

```

// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    // Prepare document first
    var newDoc = {
        fullScreen: false,
        screenWidth: 500
    };

    dataService.createDocument("MyCollection",
newDoc).then(function(doc) {
        // Even if no ID was passed to createDocument, one gets
generated
        console.log("Doc id: " + doc.id);
    });
});

```

If the collection with the name and scope provided, doesn't yet exist, it gets created dynamically before the document itself is created.

If the document provided contains an `id` property, that value gets used as the unique ID for the document. Please note that the provided `id` should be limited to 50 characters. If that field doesn't exist, a GUID gets generated by the service and included on the document that is returned when the promise is resolved.

If another document in the collection already exists with the same ID as the one provided on the document, the operation fails. If the desired behavior is create a new document if the ID doesn't exist, but modify the existing document if it does, then the `setDocument()` method should be used.

## Set a document (update or create)

The `setDocument()` function carries out an "upsert" operation - it modifies an existing document if its ID is present and matches a document in the collection. If the ID is absent or doesn't correspond to any document in the collection, then a new document is added to the collection.

JavaScript

```

// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    // Prepare document first
    var myDoc = {
        id: 1,
        fullScreen: false,
        screenWidth: 500
    };

```

```
    dataService.setDocument("MyCollection", myDoc).then(function(doc) {
        console.log("Doc id: " + doc.id);
    });
});
```

## Update a document

The `updateDocument` function requires that the document being altered already resides in the collection. An exception is thrown if no ID is supplied or if the provided ID doesn't correspond to any document in the collection.

Here's an example of how update is used:

JavaScript

```
// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    var collection = "MyCollection";
    var docId = "1234-4567-8910";
    // Get document first
    dataService.getDocument(collection, docId, { scopeType: "User" })
}).then(function(doc) {
    // Update the document
    doc.name = "John Doe";
    dataService.updateDocument(collection, doc, { scopeType: "User" })
}).then(function(d) {
    // Check the new version
    console.log("Doc version: " + d.__etag);
});
```

## Delete a document

This function deletes the document with the provided ID from the provided collection. If the collection doesn't exist or the document doesn't exist, a 404 gets returned.

Here's an example usage:

JavaScript

```
// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    var docId = "1234-4567-8910";
    // Delete document
```

```
    dataService.deleteDocument("MyCollection", docId).then(function() {
        console.log("Doc deleted");
    });
});
```

## Get all documents in a collection

The following example retrieves all documents from the "MyCollection" collection using the data service, and then logs the number of documents to the console:

JavaScript

```
// Get data service
SDK.getService(SDK.getContributionId()).then(function(dataService) {
    // Get all document under the collection
    dataService.getDocuments("MyCollection").then(function(docs) {
        console.log("There are " + docs.length + " in the collection.");
    });
});
```

This call retrieves all documents in a scoped collection, with a limit of 100,000 documents. If the collection is nonexistent, it returns a 404 error.

## Advanced

### How settings get stored

This call encapsulates the `setDocument` client method, supplying it with multiple pieces of data. As stated before, settings are internally saved as documents. Therefore, a basic document is dynamically generated, where the document's ID is the key given in the `setValue()` method. The document has two more properties. The `value` property holds the value passed to the method, and the `revision` property is set to `-1`. While the `revision` property is elaborated more in the "Working with Documents" section, in the context of settings, setting `revision` to `-1` in the document signifies that we're not concerned with the versioning of this settings document.

Because settings are stored as documents, we need to provide a collection name, indicating where to store the document. To keep things simple, when working with the `setValue()/getValue()` methods, the collection name is always the special name `$settings`. The previous call issues a PUT Request at the following endpoint:

httprequest

```
GET  
_apis/ExtensionManagement/InstalledExtensions/{publisherName}/{extensionName}  
}/Data/Scopes/User/Me/Collections/%24settings/Documents
```

The request payload is like the following example:

JSON

```
{  
    "id": "myKey",  
    "__etag": -1,  
    "value": "myValue"  
}
```

## REST APIs

Assuming this snippet is executed after the value is set, you should see an alert message containing the text "The value is myValue." The getValue method is again a wrapper around the REST APIs, issuing a GET request to the following endpoint:

httprequest

```
GET  
_apis/ExtensionManagement/InstalledExtensions/{publisherName}/{extensionName}  
}/Data/Scopes/User/Me/Collections/%24settings/Documents/myKey
```

## etags

The `__etag` field is used by the Data Storage Service for document concurrency management. Before an update is saved, the service verifies that the `__etag` of the currently stored document matches the `__etag` of the updated document. If they match, the `__etag` is incremented and the updated document is returned to the caller. If they don't match, it indicates that the document to be updated is outdated, and an exception is thrown. The extension writer is responsible for handling this exception gracefully, either by retrieving the latest `__etag` of the document, merging the changes, and retrying the update, or by notifying the user.

For some types of documents, the level of concurrency provided might not be necessary, and a last-in-wins model might be more suitable. In such cases, while editing the document, input -1 as the `__etag` value to signify this functionality. The previously mentioned settings service employs this model for storing settings and preferences.

---

# Feedback

Was this page helpful?

 Yes

 No

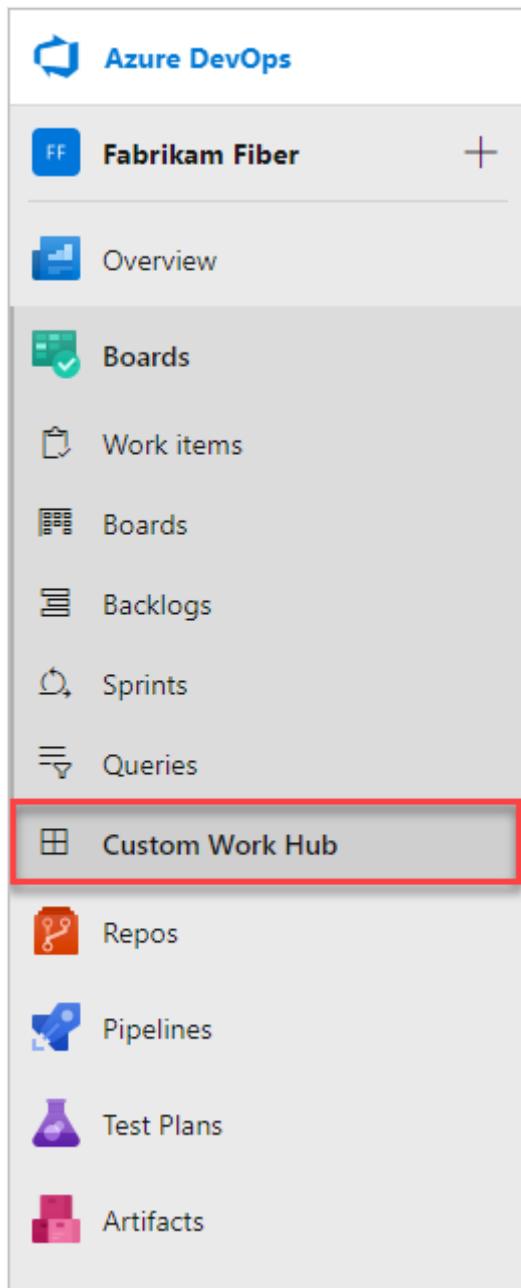
Provide product feedback 

# Add a hub

07/17/2025

## Azure DevOps Services

In this article, we'll create a new hub that displays in Azure Boards after the *Sprints* and *Queries* hubs.



## Structure of an extension

no-highlight

```
|--- README.md  
|--- sdk  
|--- node_modules
```

```
|--- scripts
|   |--- SDK.js
|--- images
|   |--- icon.png
|--- scripts                         // not used in this tutorial
|--- hello-world.html                 // html page to be used for your hub
|--- vss-extension.json               // extension's manifest
```

## Get the client SDK: `SDK.js`

The core SDK script, `SDK.js`, enables web extensions to communicate to the host, Azure DevOps Services, frame. This script also initializes, notifies that the extension loaded, or gets context about the current page. Get the Client SDK `SDK.js` file and add it to your web app. Place it in the `home/sdk/scripts` folder.

Use the '`npm install`' command via the command line (requires [Node](#)) to retrieve the SDK:

```
no-highlight
```

```
npm install azure-devops-extension-sdk
```

### ! Note

For more information, see [Azure DevOps Web Extension SDK](#).

## Your hub page: `hello-world.html`

- Every hub displays a web page
- Check out the targetable hub groups in the [extension points reference](#)

Create a `hello-world.html` file in the `home` directory of your extension. Reference the SDK and call `init()` and `notifyLoadSucceeded()`.

### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Hello World</title>
    <script src="sdk/scripts/SDK.js"></script>
</head>
<body>
    <script type="text/javascript">SDK.init();</script>
    <h1>Hello World</h1>
```

```
<script type="text/javascript">SDK.notifyLoadSucceeded();</script>
</body>
</html>
```

## Your extension's manifest file: `vss-extension.json`

- *Every* extension must have an extension manifest file
- Read the [Extension manifest reference](#)
- Find out more about the contribution points in [Extensibility points](#)

Create a json file (`vss-extension.json`, for example) in the `home` directory with the following contents:

JSON

```
{
  "manifestVersion": 1,
  "id": "sample-extension",
  "version": "0.1.0",
  "name": "My first sample extension",
  "description": "A sample Visual Studio Services extension.",
  "publisher": "fabrikamdev",
  "categories": ["Azure Boards"],
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services"
    }
  ],
  "icons": {
    "default": "images/logo.png"
  },
  "contributions": [
    {
      "id": "Fabrikam.HelloWorld",
      "type": "ms.vss-web.hub",
      "description": "Adds a 'Hello' hub to the Work hub group.",
      "targets": [
        "ms.vss-work-web.work-hub-group"
      ],
      "properties": {
        "name": "Hello",
        "order": 99,
        "uri": "hello-world.html"
      }
    }
  ],
  "scopes": [
    "vso.work"
  ],
  "files": [
```

```

        {
          "path": "hello-world.html", "addressable": true
        },
        {
          "path": "sdk/scripts", "addressable": true
        },
        {
          "path": "images/logo.png", "addressable": true
        }
      ]
    }
  }
}

```

### ⓘ Note

Change the **publisher** to your publisher name. To create a publisher, see [Package, publish, and install](#).

## Icons

The **icons** stanza specifies the path to your extension's icon in your manifest.

Add a square image titled `logo.png`, as shown in the extension manifest.

## Contributions

The **contributions** stanza adds your contribution - the Hello hub - to your extension manifest.

For each contribution in your extension, the manifest defines the following:

- type of contribution, hub
- contribution target, the work hub group (check out all of the [targetable hub groups](#),
- the properties specific to each type of contribution. A hub has the following properties.

[] [Expand table](#)

Property	Description
name	Name of the hub.
order	Placement of the hub in the hub group.
uri	Path (relative to the extension base URI) of the page to surface as the hub.

## Scopes

Include the [scopes](#) that your extension requires.

In this case, we need `vso.work` to access work items.

## Files

The **files** stanza states the files that you want to include in your package - your HTML page, your scripts, the SDK script, and your logo.

Set `addressable` to `true` unless you include other files that don't need to be URL-addressable.

 Note

For more information about the [extension manifest file](#), such as properties and function, check out the [extension manifest reference](#).

## Next steps

[Package, publish, and install extensions](#)

## Related content

- [Testing and debugging extensions](#)
- [Extension manifest reference](#)
- [Azure DevOps web extension SDK ↗](#)
- [Azure DevOps Formula Design System](#)

# Add a hub group

Article • 10/04/2022

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

We'll create a hub group and add a hub to it. If you haven't already, [create the Hello hub](#) first, and then follow these steps to create the hub group.

1. Add a hub group to your app's manifest file in contributions, like this.

JSON

```
"contributions": [
  {
    "id": "sample-hub-group",
    "type": "ms.vss-web.hub-group",
    "description": "Adds a 'Samples' hub group at the project/team-level",
    "targets": [
      "ms.vss-web.project-hub-groups-collection"
    ],
    "properties": {
      "name": "Samples",
      "order": 100
    }
  },
]
```

Look at the contribution targets reference to see the [available hub groups](#) that can be contributed to.

2. Change the hub contribution so that it's in the samples hub group that you just created. Just update the targets to the relative contribution ID of the hub group you just added.

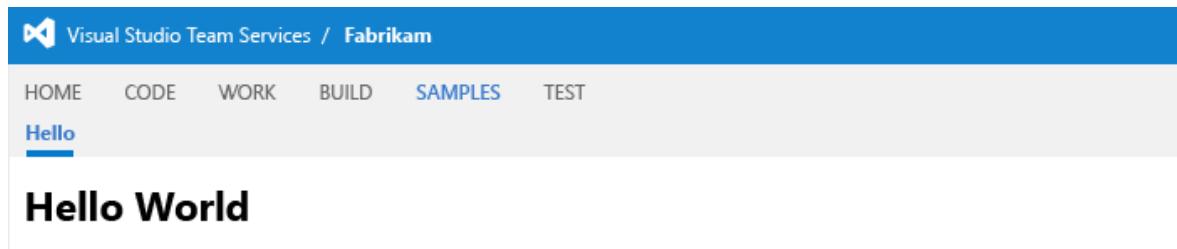
JSON

```
{
  "id": "Fabrikam.HelloWorld",
  "type": "ms.vss-web.hub",
  "description": "Adds a 'Hello' hub to the Work hub group.",
  "targets": [
    ".sample-hub-group"
  ],
  "properties": {
    "name": "Hello",
    "order": 99,
```

```
        "uri": "hello-world.html"
    }
}
```

### 3. Install your extension.

Now your hub appears under your Samples hub group.



The screenshot shows a web browser window for Visual Studio Team Services. The URL is 'Visual Studio Team Services / Fabrikam'. The navigation bar includes links for HOME, CODE, WORK, BUILD, SAMPLES, and TEST. The 'SAMPLES' link is highlighted. Below the navigation bar, there is a sub-navigation menu with 'Hello' selected. The main content area displays the title 'Hello World'.

Here's the complete extension manifest with Hello in the samples hub group.

JSON

```
{
    "namespace": "Fabrikam.myextension",
    "name": "My Extension",
    "description": "This is my first extension",
    "version": "1.0",
    "provider": {
        "name": "Fabrikam Fiber Inc"
    },
    "baseUri": "https://localhost:port",
    "icon": "images/logo.png",
    "links": {
        "info": "info.html",
        "support": "support.html",
        "termsOfService": "terms-of-service.html"
    },
    "contributions": {
        "vss.web#hubGroups.project": [
            {
                "id": "samples",
                "name": "Samples",
                "order": 30
            }
        ],
        "vss.web#hubs": [
            {
                "id": "myhub",
                "name": "Hello",
                "groupId": "samples",
                "uri": "hello-world.html"
            }
        ]
    }
}
```

 **Tip**

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

# Add a menu action

Article • 01/28/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

In this example, we add an action to the query context menu in the work item queries hub.

## Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Prerequisites

- Develop a web extension.
- Create a web app for your action.

## Update extension manifest file

Below is the code snippet that adds your action to the contributions section of your extension manifest.

JSON

```
...
    "contributions": [
        {
            "id": "myAction",
            "type": "ms.vss-web.action",
            "description": "Run in Hello hub action",
            "targets": [
                "ms.vss-work-web.work-item-query-menu"
            ],
            "properties": {
                "text": "Run in Hello hub",
                "title": "Run in Hello hub",
                "icon": "images/icon.png",
                "groupId": "actions",
                "uri": "action.html"
            }
        }
    ]
```

]

...

## Properties

[+] Expand table

Property	Description
text	Text that appears on the menu item.
title	Tooltip text that appears on the menu item.
icon	URL to an icon that appears on the menu item. Relative URLs are resolved using baseUri.
groupId	Determines where this menu item appears in relation to the others.
uri	URI to a page that registers the menu action handler (see below).
registeredObjectId	(Optional) Name of the registered menu action handler. Defaults to the contributor ID.

Learn about all of the places where you can add actions in [Extensibility points](#).

## Your HTML page

Your menu action is represented by a JavaScript script embedded in an HTML file. Save the following contents in a file and location that matches the reference to it in your extension's manifest file.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Action Sample</title>
</head>
<body>
    <div>
        The end user doesn't see the content on this page.
        It is only in the background to handle the contributed menu item
        being selected.
    </div>
</body>
</html>
```

# Your JavaScript

The script below registers the handler object to handle the action, place it in the `head` section of the previous HTML page.

We aliased `lib` to be `node_modules/azure-devops-extension-sdk/lib` in our `sdk-extension.json` manifest file.

TypeScript

```
<script src="lib/SDK.min.js"></script>
<script>
    SDK.init();

    // Use an IIFE to create an object that satisfies the
    IContributedMenuSource contract
    var menuContributionHandler = (function () {
        "use strict";
        return {
            // This is a callback that gets invoked when a user selects the
            newly contributed menu item
            // The actionContext parameter contains context data surrounding
            the circumstances of this
            // action getting invoked.
            execute: function (actionContext) {
                alert("Hello, world");
            }
        };
    }());
}

// Associate the menuContributionHandler object with the "myAction" menu
contribution from the manifest.
SDK.register(SDK.getContributionId(), menuContributionHandler);
</script>
```

## Tip

For more information, see [Extensibility points, menus and toolbars](#), the [Contribution model](#) the [Formula Design System](#), [REST API reference](#), [Extension samples](#) , and resources in the [Developer Community](#).

## Next steps

Now that you've written your extension, the next steps are to Package, Publish, and Install your extension. You can also check out the documentation for Testing and

Debugging your extension.

- Package, publish, and install extensions
  - Testing and debugging extensions
- 

## Feedback

Was this page helpful?

 Yes

 No

Provide product feedback ↗

# Create a service endpoint

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Service endpoints are a way for Azure DevOps to connect to external systems or services. They're a bundle of properties securely stored by Azure DevOps, which includes but isn't limited to the following properties:

- Service name
- Description
- Server URL
- Certificates or tokens
- User names and passwords

Extensions are then able to use the service endpoint to acquire the stored details to do the necessary operations on that service. Follow this guide to create a new service endpoint contribution and use it in your extension.

## 💡 Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Task overview

You can develop a service endpoint by creating an example extension for Azure DevOps that includes the following items:

- A custom service endpoint with data sources, which enables a build task or dashboard widget to call a REST endpoint on the service/server defined by the endpoint.
- A build task, which defines two properties: The service endpoint & a picklist, which has values populated from the REST endpoint data source.

## ⓘ Note

When you create a service endpoints, it's at the project level, not the organization level.

The steps involved in completing this task are:

- [1. Create the extension manifest file](#)
- [2. Create the build task pipeline, in the task.json file](#)

! Note

This tutorial refers to the home directory for your project as "home".

## Create the manifest file: `vss-extension.json`

The [manifest file](#) defines the custom endpoint and links to the task.json manifest for the build task.

In this article, the manifest file creation is separated into the following three parts:

- [Create a service endpoint](#)
  - [Task overview](#)
  - [Create the manifest file: vss-extension.json](#)
    - [Create basic manifest file](#)
    - [Add the custom endpoint contribution](#)
    - [Add the build task contribution](#)
  - [Create the build task](#)
    - [task.json components](#)
  - [Authentication](#)
  - [Next steps](#)
  - [Related content](#)

## Create basic manifest file

Create a json file (`vss-extension.json`, for example) in the `home` directory of your extension.

JSON

```
{  
  "manifestVersion": 1,  
  "id": "service-endpoint-tutorial",  
  "version": "0.1.1",  
  "name": "Sample extension that leverages a service endpoint",  
  "description": "A sample Azure DevOps extension which shows how to create a  
  custom endpoint and dynamic build task parameters taking value from a REST API.",  
  "publisher": "francistotten",  
  "targets": [  
    {  
      "id": "Microsoft.VisualStudio.Services"  
    }]
```

```
],
  "files": [
    {
      "path": "BuildTaskFolder"
    }
  ]
}
```

### ⓘ Note

Update the `publisher` property. The `BuildTaskFolder` is the path where we'll eventually place our build task pipeline.

## Add the custom endpoint contribution

Add the following `contributions` array underneath the `targets` array of the basic manifest content.

### ⓘ Important

Service connection parameters must be fetched by service connection ID.

#### JSON

```
"contributions": [
  {
    "id": "service-endpoint",
    "description": "Service endpoint type for Fabrikam connections",
    "type": "ms.vss-endpoint.service-endpoint-type",
    "targets": [ "ms.vss-endpoint.endpoint-types" ],
    "properties": {
      "name": "fabrikam",
      "displayName": "Fabrikam server connection",
      "url": {
        "displayName": "Server Url",
        "helpText": "Url for the Fabrikam server to connect to."
      },
      "dataSources": [
        {
          "name": "Fabrikam Projects",
          "endpointUrl": "{{endpoint.url}}api/projects/index",
          "resultSelector": "jsonpath:$[*].nm"
        }
      ],
      "authenticationSchemes": [
        {
          "name": "Basic Authentication"
        }
      ]
    }
  }
]
```

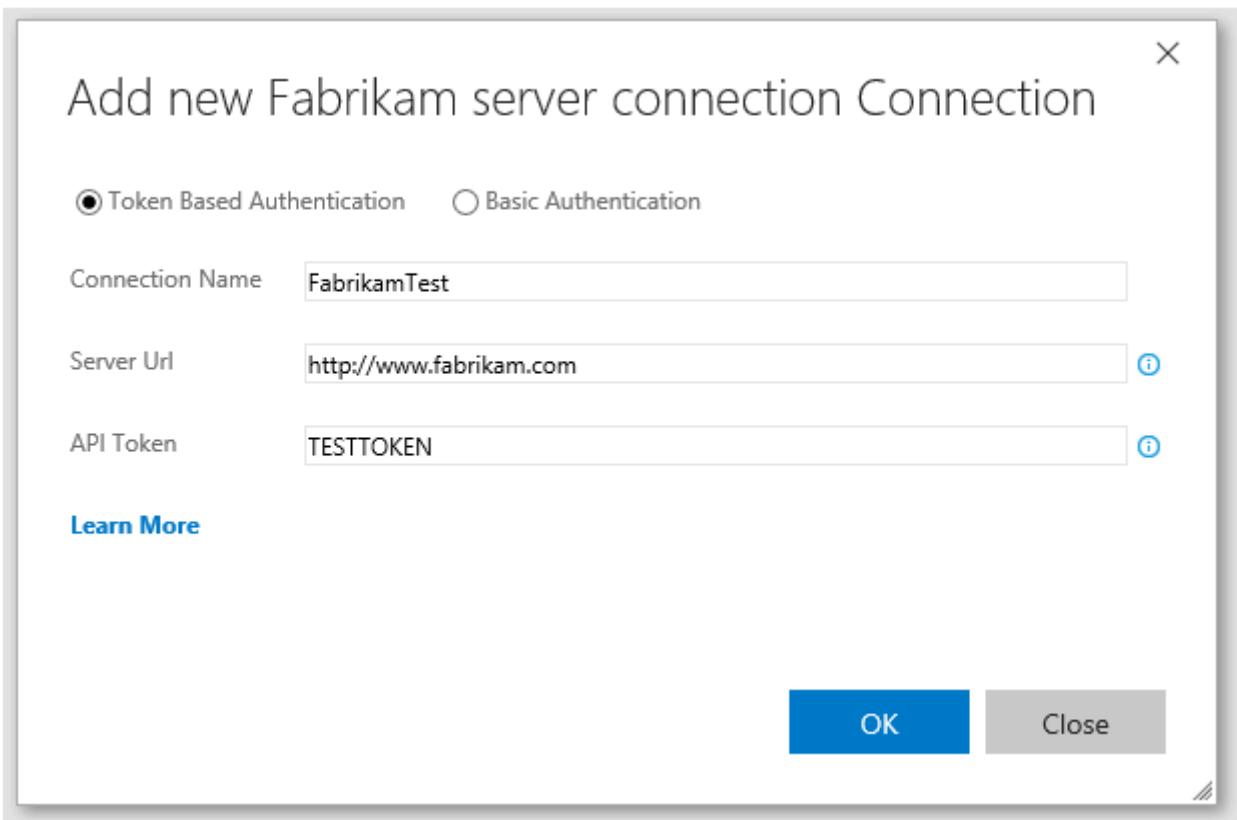
```

    "type": "ms.vss-endpoint.endpoint-auth-scheme-token"
  },
  {
    "type": "ms.vss-endpoint.endpoint-auth-scheme-basic",
    "inputDescriptors": [
      {
        "id": "username",
        "name": "Username",
        "description": "Username",
        "inputMode": "textbox",
        "validation": {
          "isRequired": false,
          "dataType": "string"
        }
      },
      {
        "id": "password",
        "name": "Password",
        "description": "Password",
        "inputMode": "passwordbox",
        "isConfidential": true,
        "validation": {
          "isRequired": false,
          "dataType": "string"
        }
      }
    ]
  }
],
"helpMarkDown": "<a href=\"url-to-documentation\" target=\"_blank\">
<b>Learn More</b></a>"
}
},
],

```

If you've successfully added the service contribution, you see the Fabrikam endpoint when you try to add a new service endpoint to your organization.

Create a service endpoint using the Fabrikam endpoint.



### 💡 Tip

You can add inputDescriptors without authenticationSchemes. For more information, see [InputDescriptor interface](#).

## Add the build task contribution

Inside the `contributions` array from the previous step, add the following object to the end.

```
JSON

{
  "id": "build-task",
  "description": "Task with a dynamic property getting data from an endpoint REST data source",
  "type": "ms.vss-distributed-task.task",
  "targets": [ "ms.vss-distributed-task.tasks" ],
  "properties": {
    "name": "BuildTaskFolder"
  }
}
```

The `dataSource` endpoint URL gets computed from the URL of the endpoint or a fixed URL, and some other values. For this tutorial, this REST call returns nothing and is meant to be replaced by any REST calls you wish to make to your service.

It's possible to use other parameters than the endpoint URL for the REST URL, for instance some endpoint properties. For instance, assuming that we had a property in the endpoint named subscriptionId, the REST URL could use it with the following syntax:  
\$(endpoint.subscription).

## Create the build task

The `task.json` file describes your build task.

### ! Note

For more information, check out the following articles:

- [Build task reference](#) to find the schema for the build task json file
- [Add a custom pipelines task extension](#)
- [Integrate build task](#)

Create a `task.json` file in your `BuildTaskFolder` directory, if you haven't created this folder yet, do so now.

### JSON

```
{  
  "id": "6557a6d2-4caf-4247-99ea-5131286a8753",  
  "name": "build-task",  
  "friendlyName": "Build Task that uses the service endpoint",  
  "description": "Task with a dynamic property getting data from an endpoint REST  
data source",  
  "author": "francistotten",  
  "helpMarkDown": "Replace with Markdown to show in help",  
  "category": "Build",  
  "visibility": [  
    "Build",  
    "Release"  
>],  
  "demands": [],  
  "version": {  
    "Major": "0",  
    "Minor": "1",  
    "Patch": "1"  
>},  
  "minimumAgentVersion": "1.95.0",  
  "instanceNameFormat": "Service Endpoint Build Task $(project)",  
  "inputs": [  
    {  
      "name": "FabrikamService",  
      "type": "connectedService:Fabrikam",  
    }  
  ]  
}
```

```

    "label": "Fabrikam service/server end point",
    "defaultValue": "",
    "required": true,
    "helpMarkDown": "Select the Fabrikam end point to use. If needed, select 'manage', and add a new service endpoint of type 'Fabrikam server connection'"
  },
  {
    "name": "project",
    "type": "pickList",
    "label": "Fabrikam Project",
    "required": true,
    "helpMarkDown": "Select the name of the Fabrikam Project to analyze.",
    "properties": {
      "EditableOptions": "True"
    }
  }
],
"dataSourceBindings": [
  {
    "target": "project",
    "endpointId": "$(FabrikamService)",
    "dataSourceName": "Fabrikam Projects"
  }
],
"execution": {
  "Node": {
    "target": "sample.js",
    "argumentFormat": ""
  },
  "PowerShell3": {
    "target": "sample.ps1"
  }
}
}

```

## task.json components

### The `FabrikamService` input object

This field is the first of type `connectedService:Fabrikam.connectedService` expresses that it's an endpoint type, and that Fabrikam is the name of the object.

### The `project` input object

This field is second. It's a picklist.

- This field is populated by a REST call.
- The values from the field "project" are taken from the "Projects" REST data source of the custom endpoint.
- Expressed in the `dataSourceBindings` array.

- The target is the name of the build task field to be populated ("project").
- The endpointId is the name of the build task field containing the custom endpoint type.
- The REST call is chosen by the dataSourceName.

If you've added the Build Task successfully, you should now see the Build Task when you're adding tasks to a build pipeline.

## Add tasks

**All**    **Build**    **Utility**    **Test**    **Package**    **Deploy**

 Android Build (deprecated; use Gradle)	Build an Android app using Gradle and optionally start the emulator for unit tests	<b>Add</b>
 Android Signing	Sign and align Android APK files	<b>Add</b>
 Ant	Build with Apache Ant	<b>Add</b>
 Build Task that uses the service endpoint	Task with a dynamic property getting data from an endpoint REST data source	<b>Add</b>
 CMake	Build with the CMake cross-platform build system	<b>Add</b>
 Gradle	Build using a Gradle wrapper script	<b>Add</b>
 Grunt	The JavaScript Task Runner	<b>Add</b>
 Gulp	Node.js streaming task based build system	<b>Add</b>
 Index Sources & Publish Symbols		<b>Add</b>
<b>(i) Don't see what you need? Check out our Marketplace.</b> 		

**Close**

Once you've added the Build Task to your pipeline, confirm that it can see the Fabrikam endpoint you created. The projects dropdown in this tutorial is blank since we aren't using a real service. Once you replace Fabrikam with your service, replace the Projects call with your own REST API call to use dynamic data inside your build task.

**Service Endpoint Build Task**

Fabrikam service/server endpoint	FabrikamTest	x  Manage
Fabrikam Project *		
<b>Control Options</b>		
Enabled	<input checked="" type="checkbox"/>	
Continue on error	<input type="checkbox"/>	
Always run	<input type="checkbox"/>	
Timeout	0	
Replace with markdown to show in help		

## Authentication

The authentication scheme in a service endpoint determines the credentials that would be used to connect to the external service. For more information and to see the following authentication schemes, see the [authentication schemes documentation](#).

- Basic authentication
- Token-based authentication
- Certificate-based authentication
- No authentication

## Next steps

[Package, publish, and install extensions](#)

## Related content

- [Test and debug extensions](#)
- [Develop a web extension](#)
- [Add a pipeline decorator](#)

# Create a custom consumer for service hooks

10/03/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

Use service hooks to notify non-Microsoft systems about events that occur in your project. A custom consumer sends an HTTP message to the endpoint defined in your extension's manifest.

## 💡 Tip

If you're starting a new Azure DevOps extension, try these maintained sample collections first—they work with current product builds and cover modern scenarios (for example, adding tabs on pull request pages).

- Azure DevOps extension sample (GitHub)—a compact starter sample that demonstrates common extension patterns: <https://github.com/microsoft/azure-devops-extension-sample> ↗
- Azure DevOps extension samples (legacy collection and contributions guide)—install to inspect UI targets, or view the source:  
<https://marketplace.visualstudio.com/items/ms-samples.samples-contributions-guide> ↗ and <https://github.com/Microsoft/vso-extension-samples/tree/master/contributions-guide> ↗
- Microsoft Learn samples (browse Azure DevOps samples)—curated, up-to-date samples across Microsoft docs: /samples/browse/?terms=azure%20devops%20extension

If a sample doesn't work in your organization, install it into a personal or test organization and compare the extension manifest's target IDs and API versions with the current docs.

For reference and APIs, see:

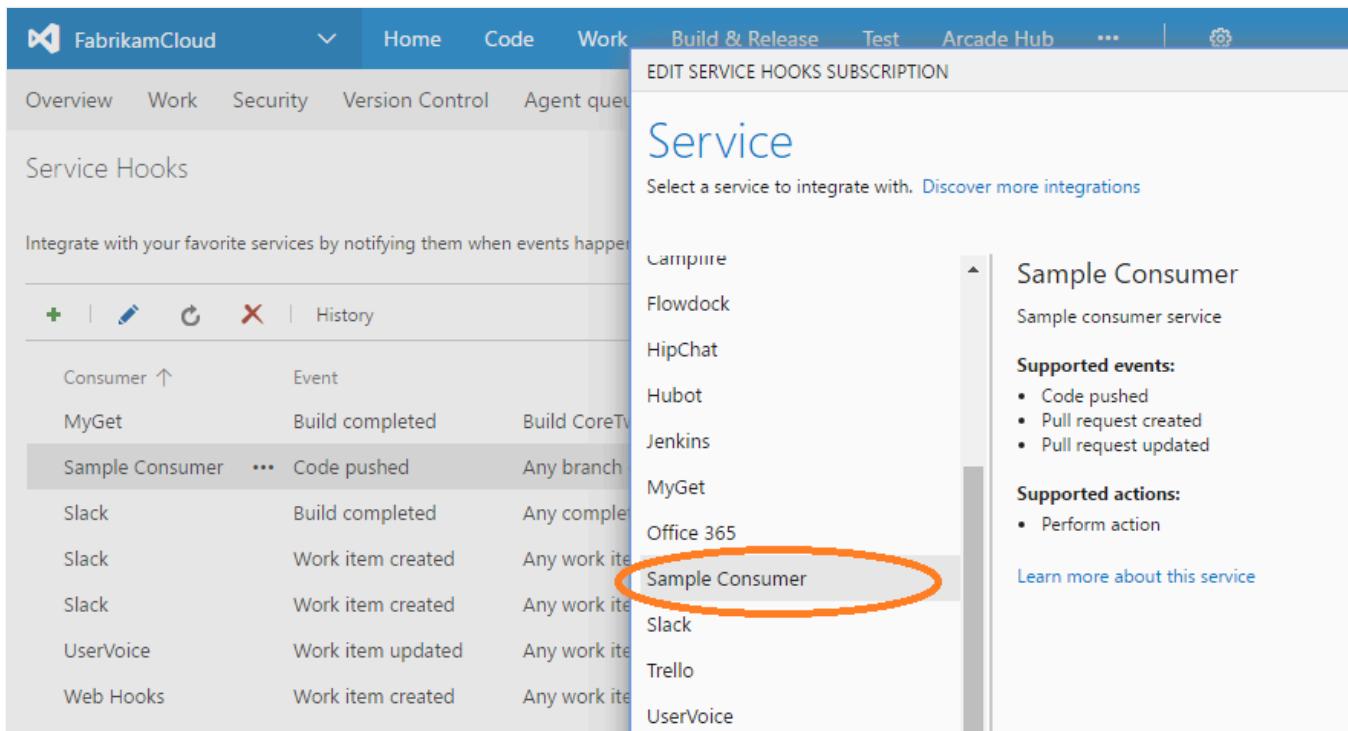
- [azure-devops-extension-api](#)
- [azure-devops-extension-sdk](#)
- [installed extension API](#)

This article walks through developing an extension that implements a **sample consumer service**, which includes the following events and actions.

- Supported events that trigger the following actions:
  - Code pushed
  - Pull request created
  - Pull request updated
- Supported actions to take when events occur:
  - Send an HTTP message

 **Note**

This article refers to the home directory for your project as *home*.



The screenshot shows the 'Service Hooks' page in the Azure DevOps interface. On the left, there's a list of existing subscriptions. A new subscription is being configured on the right, titled 'Sample Consumer'. The 'Service' dropdown is set to 'Sample Consumer'. Under 'Supported events', it lists 'Code pushed', 'Pull request created', and 'Pull request updated'. Under 'Supported actions', it lists 'Perform action'. The 'Sample Consumer' service card is highlighted with an orange circle.

Consumer ↑	Event	Action
MyGet	Build completed	Build CoreTV
Sample Consumer	Code pushed	Any branch
Slack	Build completed	Any completed
Slack	Work item created	Any work item
Slack	Work item created	Any work item
UserVoice	Work item updated	Any work item
Web Hooks	Work item created	Any work item

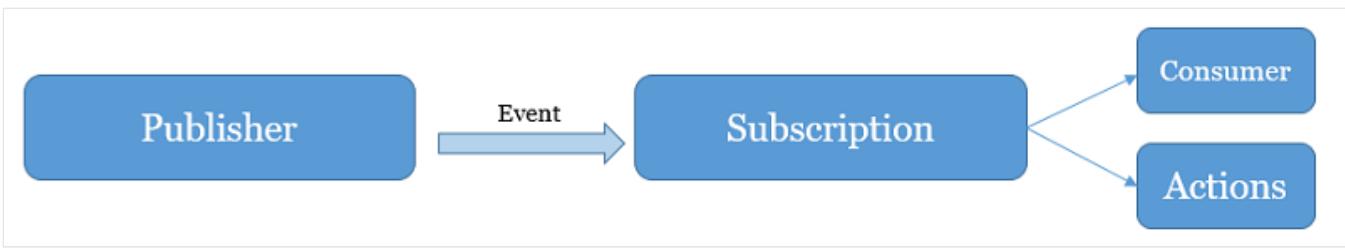
For more information, see the [Extension example GitHub repo](#). For a list of all supported events you can use as triggers for your custom consumer extension, see [List of event types](#).

 **Tip**

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## How service hooks work

Service hook publishers define a set of *events*. Subscriptions listen for the *events* and define actions that run when an event triggers.



This diagram shows the general service hook flow: publishers emit events, subscriptions match events, and actions run when a match occurs. In this article's example, an extension implements the consumer. When a supported event occurs, the consumer's configured action sends an HTTP message to the endpoint you specify in the extension manifest.

## Create the extension

1. [See how to create your extension from scratch](#).
2. Add the specific contribution for custom consumer implementation to your basic [manifest file](#). See the following example of how your manifest should look after you add the contribution.

JSON

```
{
  "manifestVersion": 1,
  "id": "samples-service-hooks-consumer",
  "version": "0.1.2",
  "name": "Service Hooks Sample",
  "description": "A simple extension that demonstrates how to contribute a consumer service into service hooks.",
  "publisher": "fabrikam",
  "public": false,
  "icons": {
    "default": "images/logo.png"
  },
  "scopes": [],
  "files": [
    {
      "path": "images",
      "addressable": true
    }
  ],
  "content": {
    "details": {
      "path": "readme.md"
    }
  },
  "categories": [
    "Developer samples"
  ],
  "targets": [
    ...
  ]
}
```

```

    {
      "id": "Microsoft.VisualStudio.Services"
    }
  ],
  "contributions": [
    {
      "id": "consumer",
      "type": "ms.vss-servicehooks.consumer",
      "targets": [
        "ms.vss-servicehooks.consumers"
      ],
      "properties": {
        "id": "consumer",
        "name": "Sample Consumer",
        "description": "Sample consumer service",
        "informationUrl": "https://aka.ms/vsoextensions",
        "inputDescriptors": [
          {
            "id": "url",
            "isRequired": true,
            "name": "URL",
            "description": "URL to post event payload to",
            "inputMode": "textbox"
          }
        ],
        "actions": [
          {
            "id": "performAction",
            "name": "Perform action",
            "description": "Posts a standard event payload",
            "supportedEventTypes": [
              "git.push",
              "git.pullrequest.created",
              "git.pullrequest.updated"
            ],
            "publishEvent": {
              "url": "{{{url}}}",
              "resourceDetailsToSend": "all",
              "messagesToSend": "all",
              "detailedMessagesToSend": "all"
            }
          }
        ]
      }
    }
  ]
}

```

## ⚠ Note

Remember to update the `publisher` property.

For each contribution in your extension, the manifest defines the following items.

- Type of contribution - consumer service (`ms.vss-servicehooks.consumer`) in this case.
- Contribution target - consumer services (`ms.vss-servicehooks.consumers`) in this case.
- Properties that are specific to each type of contribution.

Consumers have the following properties.

 [Expand table](#)

Property	Description
<code>id</code>	Unique ID for your consumer service.
<code>name</code>	Name of the custom consumer, which is visible during service hook subscription creation.
<code>description</code>	Describes your consumer service.
<code>informationUrl</code>	Find more info about your extension.
<code>inputDescriptors</code>	Inputs to be used by users that are creating subscriptions with the consumer service.
<code>actions</code>	Describes the actions to take and which events trigger those actions.

Actions for your consumer have the following properties:

 [Expand table](#)

Property	Description
<code>id</code>	ID for your action service.
<code>name</code>	Name of the action.
<code>description</code>	Detailed description of the action.
<code>supportedEventTypes</code>	Array of trigger types for which this action can be used. For more information, see <a href="#">List of event types</a> .
<code>publishEvent.url</code>	The endpoint URL that receives the HTTP message. You can template this value with tokens from the <code>inputDescriptors</code> ; users provide the actual values when they create the subscription.

3. Deploy your extension to your Azure DevOps organization and test it.

## Next steps

## Related content

- [Service hook consumers](#)
- [Available services for service hooks](#)
- [Create a service hook subscription programmatically](#)
- [Test and debug extensions in the browser](#)
- [Extension example GitHub repository ↗](#)
- [List of event types](#)

# Create a pull request status server with Node.js

07/03/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

The pull request (PR) workflow provides developers with an opportunity to get feedback on their code from peers and from automated tools. Non-Microsoft tools and services can participate in the PR workflow by using the PR [Status API](#). This article guides you through the process of creating a status server to validate PRs in an Azure DevOps Services Git repository. For more information about PR status, see [Customize and extend pull request workflows with pull request status](#).

## Prerequisites

[ ] [Expand table](#)

Category	Requirements
Organization	An <a href="#">organization in Azure DevOps</a> with a Git repository.
Tools	<ul style="list-style-type: none"><li>- <a href="#">Visual Studio Code</a> or other code editor of your choice.</li><li>- <a href="#">Node.js</a>. The download contains an installer, which you can run to install the Node.js runtime on your local machine. When installing Node.js, be sure to keep the <a href="#">npm package manager</a> portion of the install, which is selected by default.</li></ul>
Authentication	<a href="#">Microsoft Entra ID token</a> with the <a href="#">Code (status)</a> scope to have permission to change PR status. For more information, see <a href="#">Microsoft Entra authentication</a> .

## Create a basic web server using Express

The steps in this section use [Express](#), which is a lightweight web framework for Node.js that provides many HTTP utility methods that simplify creating a web server. This framework provides you with the basic functions needed to listen to PR events.

1. From the command line, create a new project folder for your web server.

```
mkdir pr-server  
cd pr-server
```

2. Use the `npm init` command to create a new `package.json` file for the project.

```
npm init
```

Select **Enter** to accept the defaults for all of the options except the entry point. Change it to `app.js`

```
entry point: (index.js) app.js
```

3. Install Express in the `pr-server` directory using the following command. This installs Express and saves it to the dependencies list.

```
npm install express
```

4. Create an Express app to build upon for the PR status server. The following steps are based on the Express [Hello world example ↗](#).

a. Open the project folder in Visual Studio Code by running the following command from the `pr-server` folder.

```
code .
```

b. Create a new file (`Ctrl + N`) and paste in the following sample code to create a basic Express server.

JavaScript

```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.listen(3000, function () {
```

```
    console.log('Example app listening on port 3000!')
  })
```

c. Save the file as `app.js`.

5. Run the basic web server using the following command:

```
node app.js
```

Verify the server is running by browsing to <http://localhost:3000/>.

## Listen for HTTP POST requests

The web server is going to receive `POST` requests from Azure DevOps Services, so you need to handle those requests in your server.

1. At the end of the `app.js` file, add the following code, and save the file.

JavaScript

```
app.post('/', function (req, res) {
  res.send('Received the POST')
})
```

2. Rerun your web server using the following command:

```
node app.js
```

## Configure a service hook for PR events

Service hooks are an Azure DevOps Services feature that can alert external services when certain events occur. For this sample, set up two service hooks for PR events, so the status server can be notified. The first is for the **Pull request created** event and the second is for the **Pull request updated** event.

To receive the service hook notifications, expose a port to the public internet. The `ngrok` utility is useful for doing so in a development environment.

1. Download and unzip the appropriate `ngrok` release for your platform.

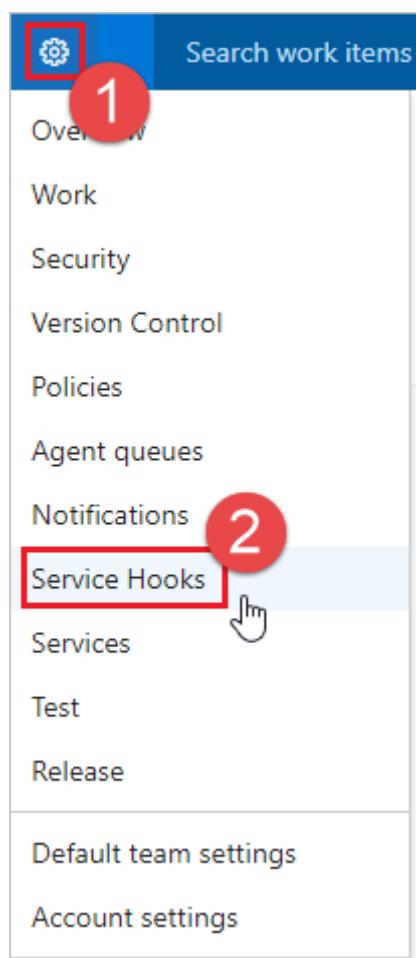
2. Use ngrok to start listening on the same port as your sample server - port 3000. Run the following command in a new command window.

```
ngrok http 3000
```

Ngrok creates a public URL that forwards to `localhost:3000`. Make a note of the URL, as you need it in the next step. It looks like the following example:

```
http://c3c1bffa.ngrok.io
```

3. Browse to your project in Azure DevOps, for example, `https://dev.azure.com/<your account>/<your project name>`
4. From the navigation menu, hover over the gear and select **Service Hooks**.



5. If it's your first service hook, select **+ Create subscription**.

The screenshot shows the 'Service Hooks' page in the VSTS interface. At the top, there's a navigation bar with 'MyFirstProject' selected. Below it is a secondary navigation bar with links for 'Overview', 'Work', 'Security', 'Version Control', 'Policies', 'Agent queues', and 'Notifications'. The main content area is titled 'Service Hooks' and contains the text 'Integrate with your favorite services by notifying them when events happen in your project.' A prominent blue button labeled '+ Create subscription' is centered, with a red box drawn around it to indicate it as the target for step 6.

If you already have other service hooks configured, select the plus (+) to create a new service hook subscription.

This screenshot shows a list of service hook subscriptions. At the top left, there's a heading 'Service Hooks' followed by a descriptive text: 'Integrate with your favorite services by notifying them when events happen in your project.' Below this is a toolbar with four icons: a green plus sign, a blue pencil, a blue circular arrow, and a red X. To the right of the toolbar are the labels 'Event' and 'Action'. Underneath the toolbar, a button labeled 'Create a new subscription...' is visible, also highlighted with a red box.

6. On the New Service Hooks Subscription dialog, select **Web Hooks** from the list of services, then select **Next**.

# Service

Select a service to integrate with. [Discover more integrations](#)

Campfire

Flowdock

HipChat

Hubot

Jenkins

Microsoft Teams

MyGet

Office 365

Slack

Trello

UserVoice

Web Hooks

Zapier

Zendesk

## Web Hooks

Provides event communication via HTTP

### Supported events:

- All events

### Supported actions:

- Post via HTTP

[Learn more about this service](#)

1

2

Previous

Next

Test

Finish

Cancel

7. Select **Pull request created** from the list of event triggers, then select **Next**.

## NEW SERVICE HOOKS SUBSCRIPTION

x

## Trigger

Select an event to trigger on and configure any filters.

Trigger on this type of event  
Pull request created

i Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.

## FILTERS

Repository i optional  
[Any]

Target branch i optional  
[Any]

Requested by a member of group: i optional  
[Any]

Reviewer includes group: i optional  
[Any]

2

Previous Next Test Finish Cancel

8. In the Action page, enter the URL from ngrok in the URL box. Select Test to send a test event to your server.

# Action

Select and configure the action to perform.

## Perform this action

Post via HTTP

This action posts a JSON object representation of the event to the specified URL. Secure, HTTPS endpoints are recommended due to the potential for private data in the event payload. [Learn More](#)

### SETTINGS

URL i  required ✓

HTTP headers i  optional

Basic authentication username i  optional

Basic authentication password i  optional

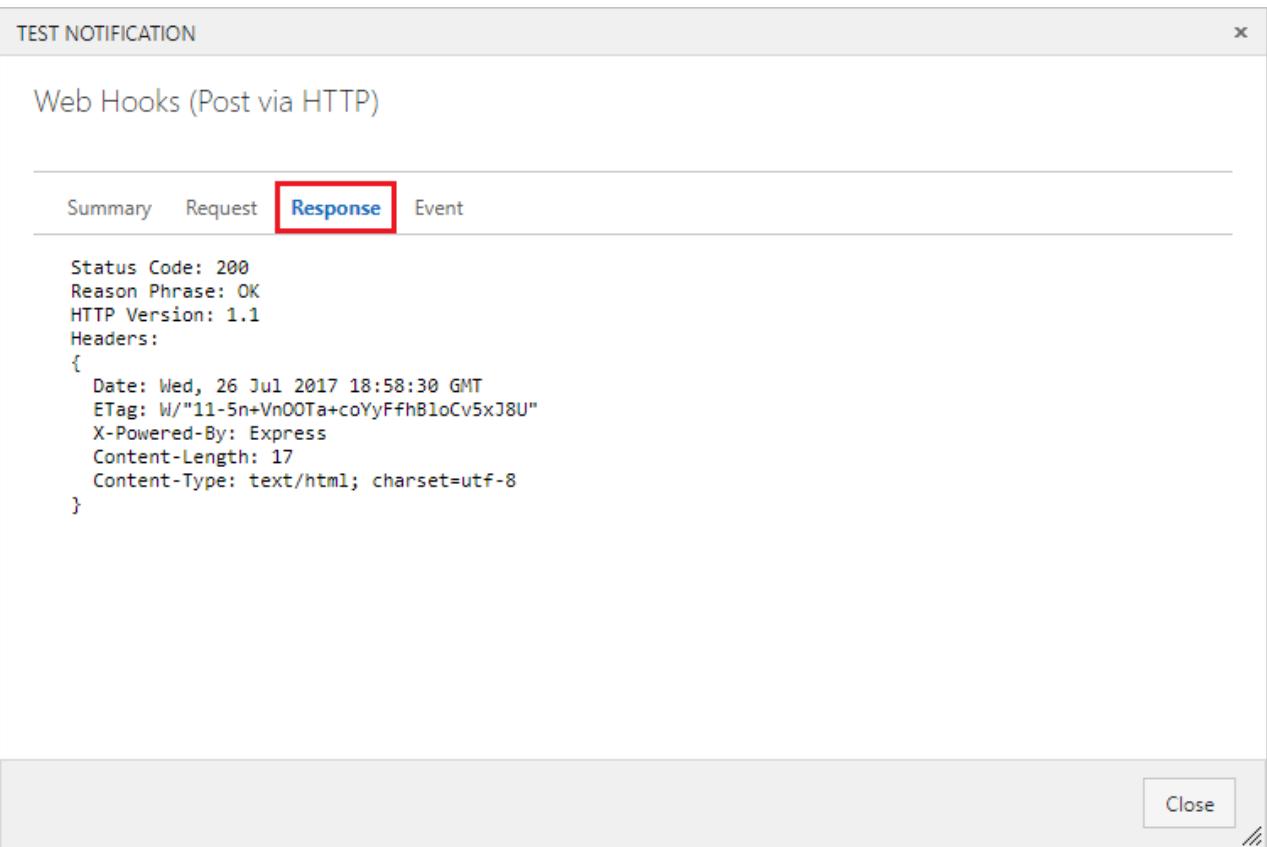
Resource details to send i  optional

[Previous](#) [Next](#) [Test](#) [Finish](#) [Cancel](#)

In the ngrok console window, an incoming `POST` returns a `200 OK`, indicating your server received the service hook event.

```
HTTP Requests
-----
POST /          200 OK
```

In the Test Notification window, select the Response tab to see the details of the response from your server. You should see a content length of 17 that matches the length of the string from your POST handler (for example, "Received the POST").



9. Close the Test Notification window, and select **Finish** to create the service hook.

Go through steps 3-9 again but this time configure the **Pull request updated** event.

**ⓘ Important**

Be sure to go through the preceding steps twice and create service hooks for both the **Pull request created** and **Pull request updated** events.

## Post status to PRs

Now that your server can receive service hook events when new PRs are created, update it to post back status to the PR.

1. Service hook requests include a JSON payload describing the event. To help parse the JSON returned by the service hook, install the [body-parser](#) package.

```
npm install body-parser
```

2. Update `app.js` to use body-parser for parsing `application/json`.

```
JavaScript
```

```
var bodyParser = require('body-parser')

app.use(bodyParser.json())
```

3. To simplify making REST API calls to Azure Repos, install the [azure-devops-node-api](#) package.

```
npm install azure-devops-node-api
```

4. Update `app.js` to use the `azure-devops-node-api` package, set up the details for a connection to your account, and get an instance of the Git API.

JavaScript

```
const vsts = require("azure-devops-node-api")

const collectionURL = process.env.COLLECTIONURL
const token = process.env.TOKEN

var authHandler = vsts.getBearerHandler(token)
var connection = new vsts.WebApi(collectionURL, authHandler)
var vstsGit = connection.getGitApi()
```

5. Create an environment variable for your collection URL, replacing `<your account>` with the name of your Azure DevOps organization.

```
setx COLLECTIONURL "https://dev.azure.com/<your account>"
```

6. Get a Microsoft Entra ID token for your app to use. Microsoft Entra ID tokens are the recommended authentication method for Azure DevOps REST APIs. You can get these tokens through the following ways:

- Option 1: Azure CLI (for development/testing)

Bash

```
az account get-access-token --resource 499b84ac-1321-427f-aa17-
267ca6975798 --query "accessToken" --output tsv
```

- Option 2: Service Principal (for production)

- a. Register an application in Microsoft Entra ID
- b. Create a client secret for the application
- c. Grant the application appropriate permissions in Azure DevOps
- d. Use the service principal credentials to obtain tokens programmatically

For more information, see [Microsoft Entra authentication](#).

7. Create an environment variable for your Microsoft Entra ID token.

```
setx TOKEN "your-entra-id-token-here"
```

## Obtaining Microsoft Entra ID tokens programmatically (Recommended for production)

For production applications, you should obtain Microsoft Entra ID tokens programmatically rather than using static tokens. Here's how to implement this using the Microsoft Authentication Library (MSAL) for Node.js:

1. Install the MSAL Node package:

Bash

```
npm install @azure/msal-node
```

2. Create a token provider module (`tokenProvider.js`):

JavaScript

```
const { ConfidentialClientApplication } = require('@azure/msal-node');

const clientConfig = {
    auth: {
        clientId: process.env.CLIENT_ID,
        clientSecret: process.env.CLIENT_SECRET,
        authority:
            `https://login.microsoftonline.com/${process.env.TENANT_ID}`
    }
};

const cca = new ConfidentialClientApplication(clientConfig);

async function getAccessToken() {
    const clientCredentialRequest = {
        scopes: ['499b84ac-1321-427f-aa17-267ca6975798/.default']
    };
}
```

```

    try {
        const response = await
cca.acquireTokenByClientCredential(clientCredentialRequest);
        return response.accessToken;
    } catch (error) {
        console.error('Error acquiring token:', error);
        throw error;
    }
}

module.exports = { getAccessToken };

```

3. Update your `app.js` to use the token provider:

JavaScript

```

const { getAccessToken } = require('./tokenProvider');

// Instead of using a static token, get a fresh token
app.post("/", async function (req, res) {
    try {
        const token = await getAccessToken();
        var authHandler = vsts.getBearerHandler(token);
        var connection = new vsts.WebApi(collectionURL, authHandler);

        // ... rest of your POST handler code
    } catch (error) {
        console.error('Authentication error:', error);
        res.status(500).send('Authentication failed');
    }
});

```

4. Update the `post()` function to read the PR details from the service hook payload. You need these values to post back status.

JavaScript

```

var repoId = req.body.resource.repository.id
var pullRequestId = req.body.resource.pullRequestId
var title = req.body.resource.title

```

5. Build the status object to post on the PR.

`State` is an enum of type [GitStatusState](#). Use `succeeded` to indicate that the PR passed the status check and is ready to merge.

The `description` is a string value that displays to the user in the Status section and activity feed in the PR details view.

The `targetUrl` is a URL that's used to create a link for the description text in the Status section and activity feed, which is where users can go to get more information about the status, for example, a build report or test run. If no URL is specified, the description appears as text with no link.

The context `name` and `genre` are used to categorize the status and distinguish it from other services posting status.

JavaScript

```
var prStatus = {
    "state": "succeeded",
    "description": "Ready for review",
    "targetUrl": "https://visualstudio.microsoft.com",
    "context": {
        "name": "wip-checker",
        "genre": "continuous-integration"
    }
}
```

6. Instead of posting the `succeeded` status right away, inspect the PR title to see if the user indicated if the PR is a work in progress by adding `WIP` to the title. If so, change the status posted back to the PR.

JavaScript

```
if (title.includes("WIP")) {
    prStatus.state = "pending"
    prStatus.description = "Work in progress"
}
```

7. Finally, post the status using the `createPullRequestStatus()` method. It requires the status object, the repo ID, and the pull request ID. Output the response to the node console so you can see the result of the post.

JavaScript

```
vstsGit.createPullRequestStatus(prStatus, repoId, pullRequestId).then( result
=> {
    console.log(result)
})
```

8. The resulting method should look something like this:

JavaScript

```

app.post("/", async function (req, res) {
  try {
    // Get the details about the PR from the service hook payload
    var repoId = req.body.resource.repository.id
    var pullRequestId = req.body.resource.pullRequestId
    var title = req.body.resource.title

    // Build the status object that we want to post.
    // Assume that the PR is ready for review...
    var prStatus = {
      "state": "succeeded",
      "description": "Ready for review",
      "targetUrl": "https://visualstudio.microsoft.com",
      "context": {
        "name": "wip-checker",
        "genre": "continuous-integration"
      }
    }

    // Check the title to see if there is "WIP" in the title.
    if (title.includes("WIP")) {
      // If so, change the status to pending and change the
      description.
      prStatus.state = "pending"
      prStatus.description = "Work in progress"
    }

    // Get the Git API instance and post the status to the PR
    const gitApi = await vstsGit
    const result = await gitApi.createPullRequestStatus(prStatus, repoId,
pullRequestId)
    console.log(result)

    res.send("Received the POST")
  } catch (error) {
    console.error('Error processing PR status:', error)
    res.status(500).send('Error processing request')
  }
})

```

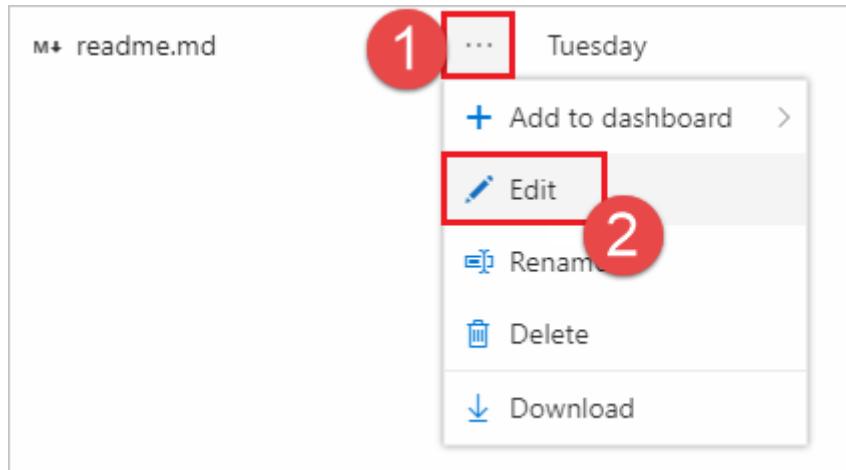
9. Save `app.js` and restart your node app.

```
node app.js
```

## Create a new PR for testing the status server

Now that your server is running and listening for service hook notifications, create a pull request to test it out.

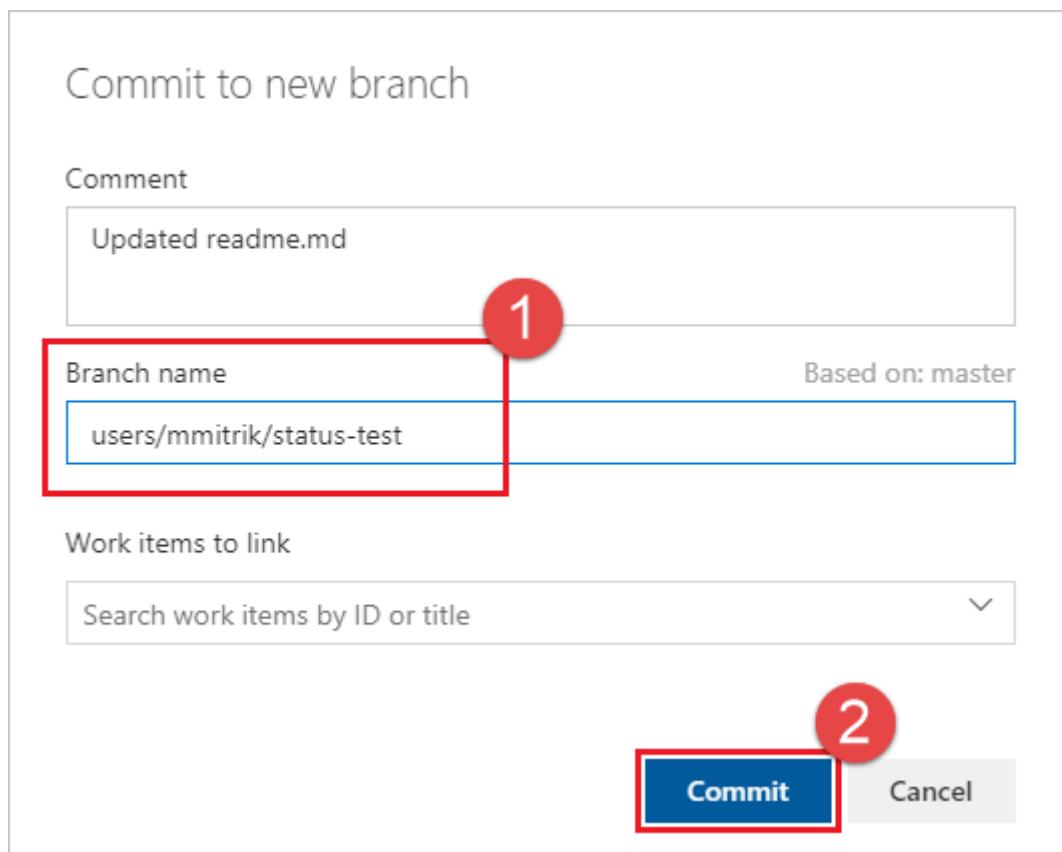
1. Start in the files view. Edit the readme.md file in your repo (or any other file if you don't have a readme.md).



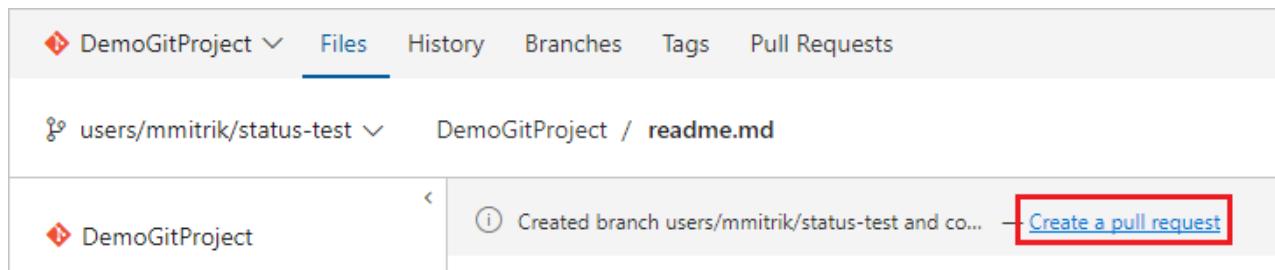
2. Make an edit and commit the changes to the repo.



3. Be sure to commit the changes to a new branch so you can create a PR in the next step.



4. Select the Create a pull request link.



5. Add WIP in the title to test the functionality of the app. Select **Create** to create the PR.

The screenshot shows the 'New Pull Request' dialog. It has fields for 'From' (users/mmitrik/status-test) and 'into' (master). The 'Title' field (marked with a red box and red number '1') contains '[WIP] Updated readme.md'. The 'Description' section contains 'Updated readme.md'. Below the description is a note 'Markdown supported.' followed by a rich text editor toolbar. The 'Reviewers' section shows '[DemoGitProject]\DemoGitProject Team' selected. The 'Work Items' section has a search bar 'Search work items by ID or title'. The bottom right corner features a large red circle with the number '2' over the 'Create' button, which is also highlighted with a red box.

6. Once the PR is created, the status section shows with the **Work in progress** entry that links to the URL specified in the payload.

The screenshot shows a GitHub pull request page for a repository named 'users/mmitrik/status-test'. The pull request has been merged into the 'master' branch. The title of the PR is '[WIP] Updated readme.md'. The status bar at the top indicates '190 ACTIVE' pull requests. On the left, there's a sidebar with 'Policies' (Required: 0 of 1 reviewers approved, No work items linked; Optional: Build succeeded), 'Status' (Work in progress, continuous-integration/wip-checker), and 'No related work items'. The main right panel shows the PR details: Description: 'Updated readme.md', a 'Show everything' dropdown, and a comment section. A comment from Raisa Pokrovskaya was added just now, and the PR was created by Matthew Mitrik just now.

7. Update the PR title and remove the **WIP** text and note that the status changes from **Work in progress** to **Ready for review**.

## Related articles

- REST API documentation
- Configure a branch policy for an external service

# Use Azure Functions to create custom branch policies

07/03/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

The pull request (PR) workflow allows developers to receive feedback on their code from peers and automated tools. Non-Microsoft tools and services can also participate in the PR workflow by using the PR [Status API](#). This article guides you through creating a custom branch policy using [Azure Functions](#) to validate PRs in an Azure DevOps Git repository. Azure Functions eliminates the need to provision and maintain servers, even as your workload grows. They provide a fully managed compute platform with high reliability and security.

For more information about PR status, see [Customize and extend pull request workflows with pull request status](#).

## Prerequisites

  Expand table

Category	Requirements
Organization	An <a href="#">organization in Azure DevOps</a> with a Git repository.
Azure Function	An <a href="#">Azure Function</a> , which implements a serverless, event-driven solution that integrates with Azure DevOps to create custom branch policies and automate PR validation.
Service Hooks	<a href="#">Configure service hooks</a> for PR events to notify your Azure function when a pull request changes.
Authentication	<a href="#">Microsoft Entra ID token</a> with the <b>Code (status)</b> scope to have permission to change PR status. For more information, see <a href="#">Microsoft Entra authentication</a> .

## Create a basic Azure Function to listen to Azure Repos events

Create your first [Azure function](#). Then, modify the code in the sample to look like the following code:

```
cs  
  
using System;  
using System.Net;  
using System.Net.Http;
```

```

using System.Net.Http.Headers;
using System.Text;
using Newtonsoft.Json;

public static async Task<HttpResponseMessage> Run(HttpRequestMessage req,
TraceWriter log)
{
    try
    {
        log.Info("Service Hook Received.");

        // Get request body
        dynamic data = await req.Content.ReadAsAsync<object>();

        log.Info("Data Received: " + data.ToString());

        // Get the pull request object from the service hooks payload
        dynamic jobject = JsonConvert.DeserializeObject(data.ToString());

        // Get the pull request id
        int pullRequestId;
        if (!Int32.TryParse(jObject.resource.pullRequestId.ToString(), out
pullRequestId))
        {
            log.Info("Failed to parse the pull request id from the service hooks
payload.");
        };

        // Get the pull request title
        string pullRequestTitle = jobject.resource.title;

        log.Info("Service Hook Received for PR: " + pullRequestId + " " +
pullRequestTitle);

        return req.CreateResponse(HttpStatusCode.OK);
    }
    catch (Exception ex)
    {
        log.Info(ex.ToString());
        return req.CreateResponse(HttpStatusCode.InternalServerError);
    }
}

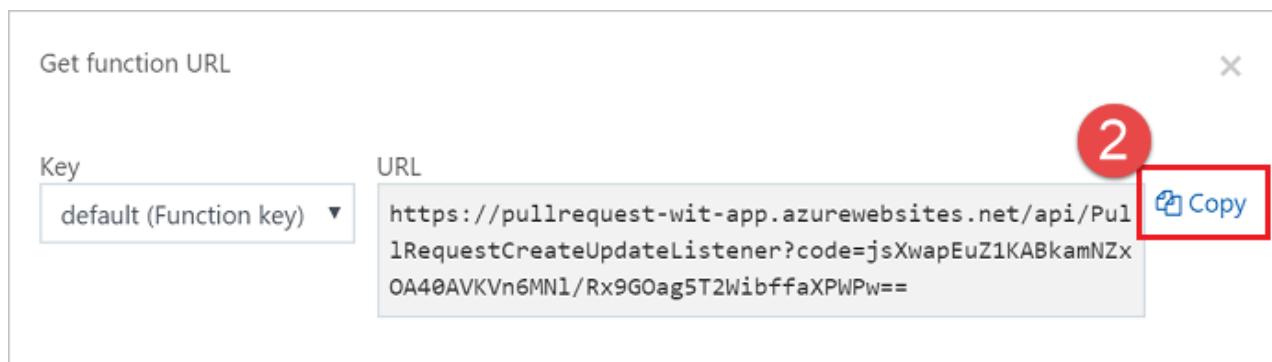
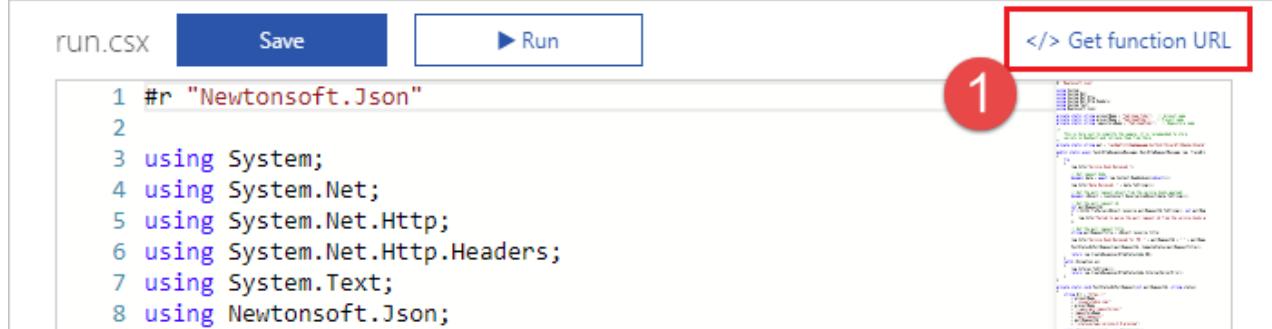
```

## Configure a service hook for PR events

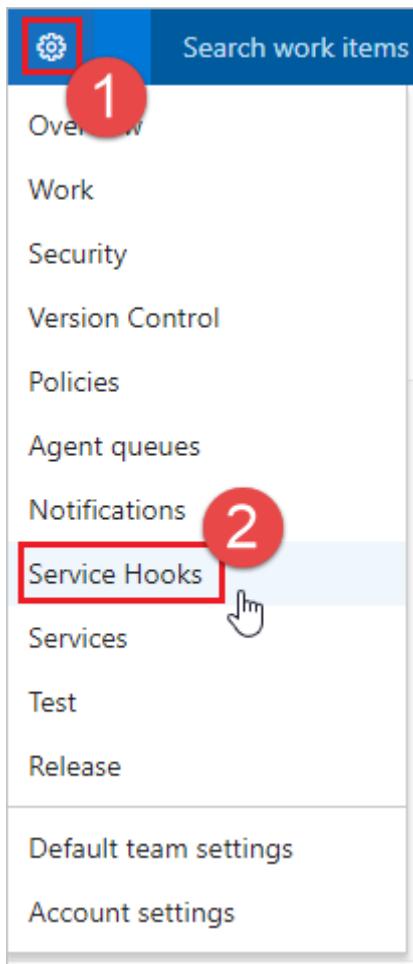
Service hooks are an Azure DevOps feature that can alert external services when certain events occur. For this sample, set up a service hook for PR events, your Azure function is notified when a pull request changes. In order to receive `POST` requests when pull requests change, provide the service hook with the Azure function URL.

For this sample, configure two service hooks. The first is for the **Pull request created** event and the second is for the **Pull request updated** event.

1. Get the function URL from the Azure portal by clicking the **Get function URL** in your Azure function view and copy the URL.



2. Browse to your project in Azure DevOps, for example, <https://dev.azure.com/<your organization>/<your project name>>
3. From the navigation menu, hover over the **gear** and select **Service Hooks**.



4. If it's your first service hook, select **+ Create subscription**.

A screenshot of the 'Service Hooks' page. At the top, there is a header bar with a speaker icon, the project name 'MyFirstProject', and navigation links for Dashboards, Code, Work, and Build & Release. Below the header, there is a horizontal navigation bar with links for Overview, Work, Security, Version Control, Policies, Agent queues, and Notifications. The main content area is titled 'Service Hooks' and contains the text 'Integrate with your favorite services by notifying them when events happen in your project.' At the bottom of this section is a blue button with a white plus sign and the text '+ Create subscription', which is highlighted with a red box.

If you already have other service hooks configured, select the green plus (+) to create a new service hook subscription.

## Service Hooks

Integrate with your favorite services by notifying them when events happen in your project.

The screenshot shows the 'Service Hooks' interface. At the top, there are three icons: a green plus sign, a blue pencil, and a red X. Below these is a button labeled 'Create a new subscription...'. To the right of the button are tabs for 'Event' and 'Action'. A horizontal bar at the bottom has the text 'Event' and 'Action'.

5. On the New Service Hooks Subscription dialog, select **Web Hooks** from the list of services, then select **Next**.

The screenshot shows the 'NEW SERVICE HOOKS SUBSCRIPTION' dialog. On the left, a list of services includes: Campfire, Flowdock, HipChat, Hubot, Jenkins, Microsoft Teams, MyGet, Office 365, Slack, Trello, UserVoice, Web Hooks (highlighted with a red box and a red number '1'), Zapier, and Zendesk. On the right, details for 'Web Hooks' are shown: 'Provides event communication via HTTP', 'Supported events: All events', 'Supported actions: Post via HTTP', and a link to 'Learn more about this service'. At the bottom, there are buttons for 'Previous', 'Next' (highlighted with a red box and a red number '2'), 'Test', 'Finish', and 'Cancel'.

6. Select **Pull request created** from the list of event triggers, then select **Next**.

NEW SERVICE HOOKS SUBSCRIPTION

## Trigger

Select an event to trigger on and configure any filters.

1

Trigger on this type of event

Pull request created

*Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.*

FILTERS

Repository optional  
[Any]

Target branch optional  
[Any]

Requested by a member of group: optional  
[Any]

Reviewer includes group: optional  
[Any]

2

Previous Next Test Finish Cancel

7. In the Action page, enter the URL that you copied in step 1 in the URL box. Select **Test** to send a test event to your server.

# Action

Select and configure the action to perform.

## Perform this action

Post via HTTP

This action posts a JSON object representation of the event to the specified URL. Secure, HTTPS endpoints are recommended due to the potential for private data in the event payload. [Learn More](#)

### SETTINGS

URL i required  
xxxx ✓

HTTP headers i optional

Basic authentication username i optional

Basic authentication password i optional

Resource details to send i optional  
All

Previous Next Test Finish Cancel

In the Azure function log window, you see an incoming `POST` that returned a `200 OK`, indicating your function received the service hook event.

HTTP Requests

-----

POST / 200 OK

In the Test Notification window, select the Response tab to see the details of the response from your server. You should see the response from your server.

TEST NOTIFICATION X

Web Hooks (Post via HTTP)

Summary Request **Response** Event

```
Status Code: 200
Reason Phrase: OK
HTTP Version: 1.1
Headers:
{
    Pragma: no-cache
    Cache-Control: no-cache
    Date: Thu, 07 Dec 2017 22:27:39 GMT
    Server: Microsoft-IIS/8.0
    X-AspNet-Version: 4.0.30319
    X-Powered-By: ASP.NET
    Content-Length: 0
    Expires: -1
}
```

Close ≡

8. Close the Test Notification window, and select **Finish** to create the service hook.

Go through steps 2-8 again but this time configure the **Pull request updated** event.

i **Important**

Be sure to go through the preceding steps twice and create service hooks for both the **Pull request created** and **Pull request updated** events.

Create a pull request to verify your Azure function is receiving notifications.

## Post status to PRs

Now that your server can receive service hook events when new PRs are created, update it to post back status to the PR. You can use the JSON payload posted by the service hook in order to determine what status to set on your PR.

Update the code of your Azure function, similar to the following example.

Make sure to update the code with your organization name, project name, repository name, and Microsoft Entra ID token. In order to have permission to change PR status, the token requires [vso.code\\_status](#) scope, which you can obtain through Microsoft Entra authentication.

## Important

This sample code stores the token in code, simplifying the sample. It is recommended to store secrets in Azure Key Vault and retrieve them from there using managed identity for enhanced security.

This sample inspects the PR title to see if the user indicated if the PR is a work in progress by adding **WIP** to the title. If so, the sample code changes the status posted back to the PR. Replace the code in your Azure function with the following code, which updates the status posted back to the PR.

```
cs

using System;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using Newtonsoft.Json;

private static string organizationName = "[Organization Name]"; // Organization name
private static string projectName = "[Project Name]"; // Project name
private static string repositoryName = "[Repo Name]"; // Repository name

/*
    This is here just to simplify the sample, it is recommended to store secrets in Azure Key Vault and retrieve them using managed identity.
*/
private static string accessToken = "[MICROSOFT_ENTRA_TOKEN]";

public static async Task<HttpResponseMessage> Run(HttpRequestMessage req,
TraceWriter log)
{
    try
    {
        log.Info("Service Hook Received.");

        // Get request body
        dynamic data = await req.Content.ReadAsAsync<object>();

        log.Info("Data Received: " + data.ToString());

        // Get the pull request object from the service hooks payload
        dynamic jObject = JsonConvert.DeserializeObject(data.ToString());

        // Get the pull request id
        int pullRequestId;
        if (!Int32.TryParse(jObject.resource.pullRequestId.ToString(), out
```

```

        pullRequestId))
    {
        log.Info("Failed to parse the pull request id from the service hooks
payload.");
    };

    // Get the pull request title
    string pullRequestTitle = jobject.resource.title;

    log.Info("Service Hook Received for PR: " + pullRequestId + " " +
pullRequestTitle);

    PostStatusOnPullRequest(pullRequestId, ComputeStatus(pullRequestTitle));

    return req.CreateResponse(HttpStatusCode.OK);
}
catch (Exception ex)
{
    log.Info(ex.ToString());
    return req.CreateResponse(HttpStatusCode.InternalServerError);
}
}

private static void PostStatusOnPullRequest(int pullRequestId, string status)
{
    string Url = string.Format(
        @"https://dev.azure.com/{0}/{1}/_apis/git/repositories/{2}/pullrequests/{3}/status
es?api-version=4.1",
        organizationName,
        projectName,
        repositoryName,
        pullRequestId);

    using (HttpClient client = new HttpClient())
    {
        client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
        client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", accessToken);

        var method = new HttpMethod("POST");
        var request = new HttpRequestMessage(method, Url)
        {
            Content = new StringContent(status, Encoding.UTF8, "application/json")
        };

        using (HttpResponseMessage response = client.SendAsync(request).Result)
        {
            response.EnsureSuccessStatusCode();
        }
    }
}

private static string ComputeStatus(string pullRequestTitle)

```

```

{
    string state = "succeeded";
    string description = "Ready for review";

    if (pullRequestTitle.ToLower().Contains("wip"))
    {
        state = "pending";
        description = "Work in progress";
    }

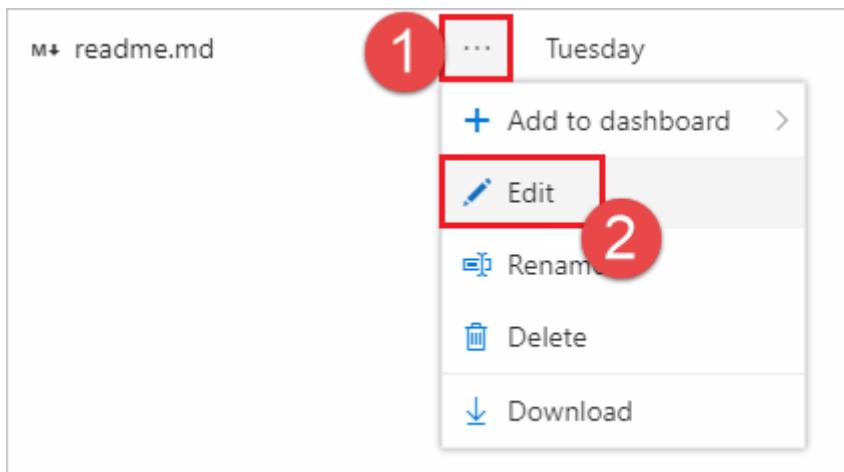
    return JsonConvert.SerializeObject(
        new
        {
            State = state,
            Description = description,
            TargetUrl = "https://visualstudio.microsoft.com",
            Context = new
            {
                Name = "PullRequest-WIT-App",
                Genre = "pr-azure-function-ci"
            }
        });
}

```

## Create a new PR and test the status server

Now that your server is running and listening for service hook notifications, create a pull request to test it out.

1. Start in the files view. Edit the readme.md file in your repo (or any other file if you don't have a readme.md).



2. Make an edit and commit the changes to the repo.

Contents Preview Highlight changes

1 #ReadMe  
2 Testing the status API

Commit... Discard

3. Be sure to commit the changes to a new branch so you can create a PR in the next step.

### Commit to new branch

Comment

Updated readme.md

1

Branch name

users/mmitrik/status-test

Based on: master

Work items to link

Search work items by ID or title

2

Commit Cancel

4. Select the Create a pull request link.

DemoGitProject Files History Branches Tags Pull Requests

users/mmitrik/status-test DemoGitProject / readme.md

DemoGitProject < Created branch users/mmitrik/status-test and co... [Create a pull request](#)

5. Add WIP in the title to test the functionality of the app. Select **Create** to create the PR.

New Pull Request

into

Title \*

[WIP] Updated readme.md

Description

Updated readme.md

Markdown supported.

Aa **I** *I* *I* *I* *I* *I* @ #

Updated readme.md

Reviewers

[DemoGitProject]\DemoGitProject Team X Search users and groups to add as reviewers

Work Items

Search work items by ID or title

Create

- Once the PR gets created, the status section displays, with the **Work in progress** entry that links to the URL specified in the payload.

190 ACTIVE [WIP] Updated readme.md

Matthew Mitrik users/mmitrik/status-test into master

Approve Set auto-complete ...

Overview Files Updates Commits

Policies

Required  
X 0 of 1 reviewers approved  
X No work items linked

Optional  
✓ Build succeeded

Status

Work in progress

continuous-integration/wip-checker

Description

Updated readme.md

Show everything *X*

Add a comment...

Raisa Pokrovskaya was added as a reviewer for readme.md just now

Created by Matthew Mitrik just now

No related work items

- Update the PR title and remove the **WIP** text and note that the status changes from **Work in progress** to **Ready for review**.

## Next steps

Configure a branch policy for an external service

# Add tabs on backlog pages

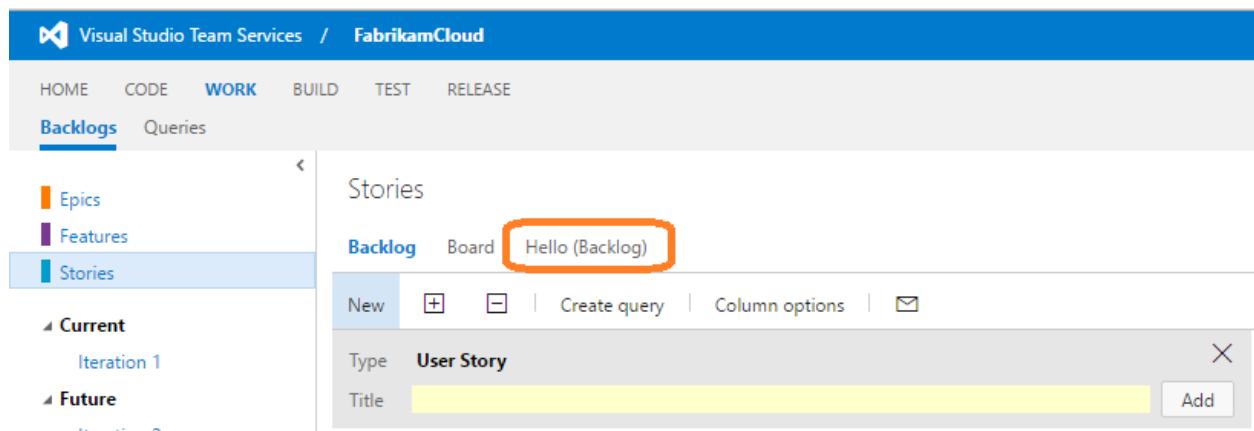
Article • 10/04/2022

## Azure DevOps Services

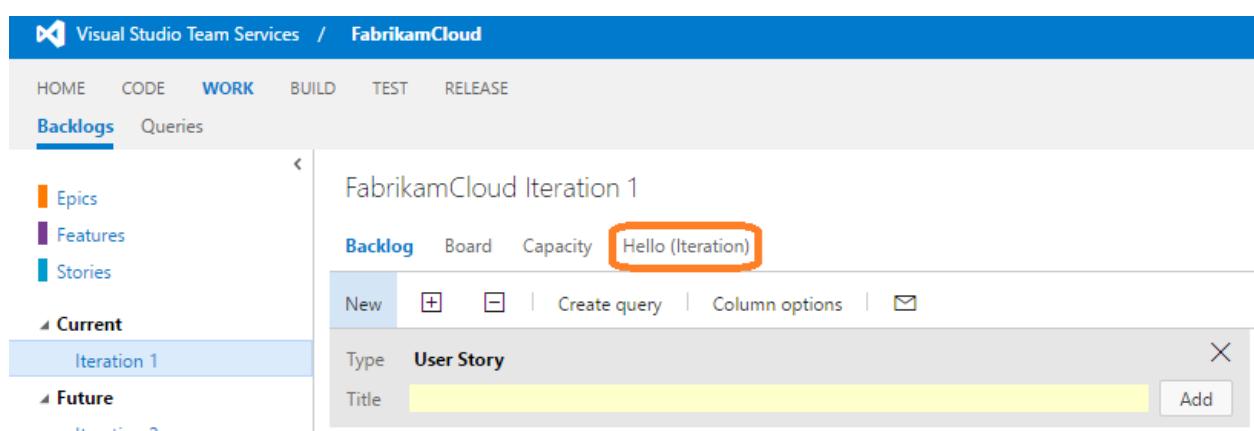
### Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

If you have a web page that can be hosted in an iframe, it can also be hosted in Azure DevOps Services. This webpage would be a tab on the backlog pages. In this example, we'll add a Hello World tab on the Product Backlog and the Iteration Backlog.



The screenshot shows the 'Backlogs' section of the Azure DevOps interface. On the left, there's a sidebar with 'Epics', 'Features', and 'Stories' (which is selected). Below that, under 'Current', is 'Iteration 1'. Under 'Future' is 'Iteration 2'. The main area shows 'Stories' with tabs for 'Backlog', 'Board', and a new tab 'Hello (Backlog)' which is highlighted with a red box. A modal window is open for creating a new User Story, with 'Type' set to 'User Story' and a yellow 'Add' button.



The screenshot shows the 'Backlogs' section of the Azure DevOps interface, specifically for 'FabrikamCloud Iteration 1'. The sidebar shows 'Iteration 1' is selected under 'Current'. The main area shows 'Stories' with tabs for 'Backlog', 'Board', 'Capacity', and a new tab 'Hello (Iteration)' which is highlighted with a red box. A modal window is open for creating a new User Story, with 'Type' set to 'User Story' and a yellow 'Add' button.

## Create your web page

1. Get the Client SDK `SDK.js` file and add it to your web app. Place it in the `home/sdk/scripts` folder.
  - a. Use the 'npm install' command to retrieve the SDK: `npm install azure-devops-extension-sdk`.

b. To learn more about the SDK, visit the [Client SDK GitHub Page](#).

2. Add the web page that you want to display as a hub. We're doing a simple `hello-world.html` page here, added to the `home` directory.

3. In your HTML page, add a reference to the SDK, and call `init()` and `notifyLoadSucceeded()`.

HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Hello World</title>
    <script src="sdk/scripts/SDK.js"></script>
</head>
<body>
    <script type="text/javascript">SDK.init();</script>
    <h1>Hello World</h1>
    <script type="text/javascript">SDK.notifyLoadSucceeded();</script>
</body>
</html>
```

## Update your extension manifest

Update your [extension manifest](#) file with the following code:

JSON

...

```
"contributions": [
    {
        "id": "Fabrikam.HelloWorld.Backlog.Tabs",
        "type": "ms.vss-web.tab",
        "description": "Adds a 'Hello' tab to the Product and Iteration backlog tabs.",
        "targets": [
            "ms.vss-work-web.product-backlog-tabs",
            "ms.vss-work-web.iteration-backlog-tabs"
        ],
        "properties": {
            "name": "Hello",
            "uri": "hello-world.html",
            "registeredObjectId": "backlogTabObject"
        }
    }
],
"scopes": [
```

```

    "vso.work"
],
"files": [
{
    "path": "hello-world.html", "addressable": true
},
{
    "path": "scripts", "addressable": true
},
{
    "path": "sdk/scripts", "addressable": true
},
{
    "path": "images/logo.png", "addressable": true
}
]
...

```

## Contributions

The **contributions** stanza contains information about your tasks.

For each contribution in your extension, the manifest defines

- the type of contribution (tab in this case),
- the contribution target (the product or iteration backlog tabs in this case),
- and the properties that are specific to each type of contribution. For a tab, we have

Property	Description
name	Name of the hub
uri	Path (relative to the extension's base URI) of the page to surface as the tab
registeredObjectId	ID of the object registered for the tab. Include code like the example below in the html file indicated in the "uri" property of the contribution shown previously.

## Scopes

It includes the **scopes** that your extension requires. In this case, we need `vso.work` to access work items.

## Files

Include all of the files your extension accesses.

For your files, set `addressable` to `true` unless you include other files that don't need to be URL-addressable.

## Example registeredObjectId

JavaScript

```
SDK.register("backlogTabObject", {
  pageTitle: function(state) {
    return "Hello Tab";
  },
  updateContext: function(tabContext) {
  },
  isInvisible: function(state) {
    return false;
  },
  isDisabled: function(state) {
    return false;
  }
});
```

Learn about all of the places where you can add a hub in [Extensibility points](#).

## Next steps

[Package, Publish, and Install](#)

or

[Test and Debug](#)

# Add panels on backlog pages

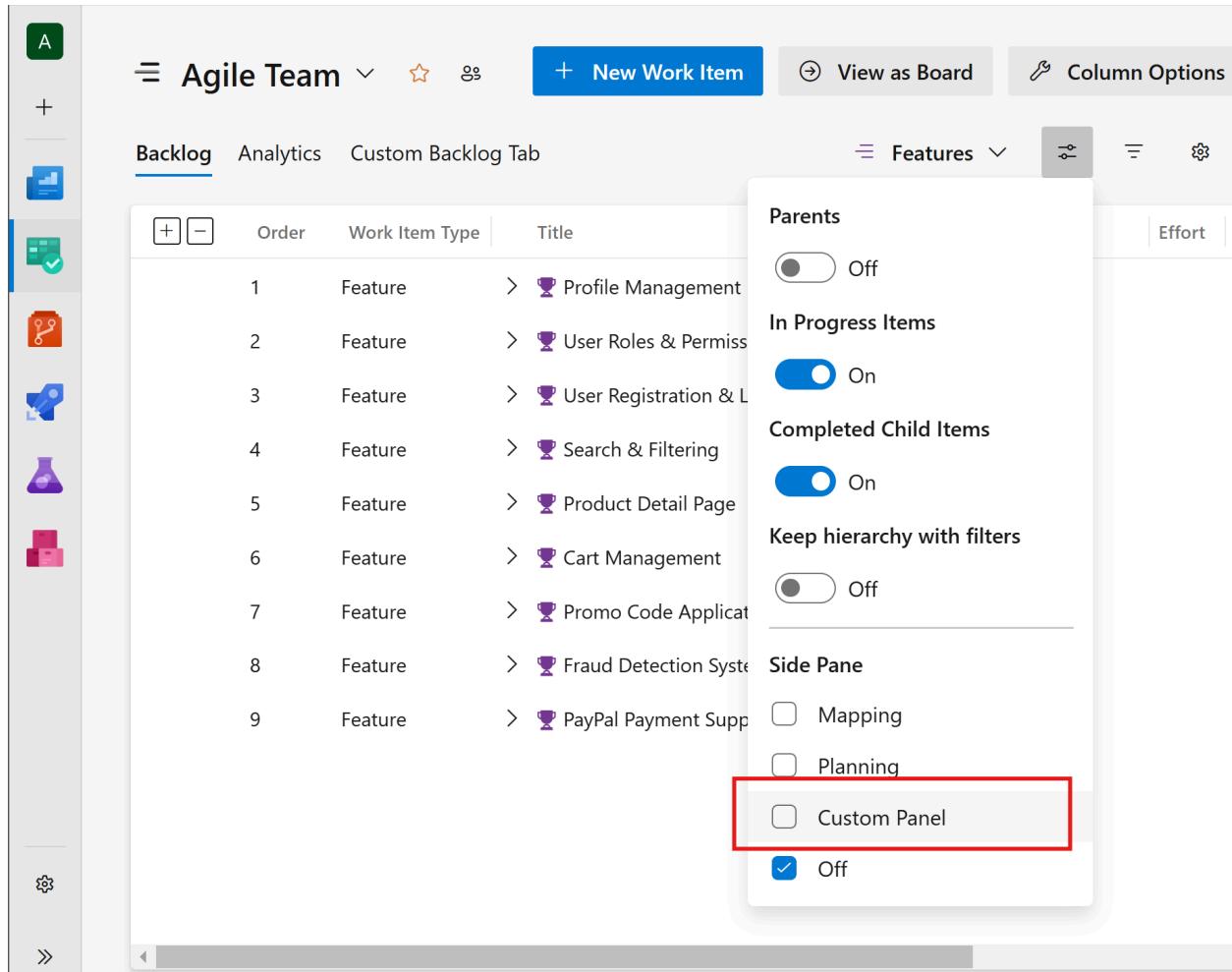
Article • 02/22/2025

## Azure DevOps Services

Here, we add a simple Hello World extension as a panel on the Portfolio backlog, Product backlog, and Iteration backlog.

### 💡 Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).



The screenshot shows the Azure DevOps Backlog page for the 'Agile Team' project. The backlog table lists nine items, all of which are 'Feature' type work items. To the right of the backlog, there is a 'Features' pane with several settings:

- Parents:** Off (toggle switch)
- In Progress Items:** On (toggle switch)
- Completed Child Items:** On (toggle switch)
- Keep hierarchy with filters:** Off (toggle switch)
- Side Pane:**
  - Mapping
  - Planning
  - Custom Panel (this option is highlighted with a red box)
- Off:** On (checkbox)

The custom panel opens in the same space that the mapping panel opens if it were selected.

A screenshot of the Azure DevOps Agile Team backlog page. On the left, there's a sidebar with various icons. The main area shows a backlog of features. A 'Custom Panel' extension is open on the right, titled 'Features'. The backlog table has columns for Order, Work Item Type, and Title. The backlog items are numbered 1 to 9, all listed as 'Feature' type with a trophy icon and a detailed title.

There are three types of backlogs that can be targets for panel extensions: Portfolio backlogs, Product backlogs, and Iteration backlogs. For the Agile template, this breakdown is as below. This is representative of Scrum and CMMI as well. For custom templates, please consult your process to see which backlogs are requirement or portfolio category.

[Expand table](#)

Backlog Category	Contribution point
Portfolio (Epic, Feature)	ms.vss-work-web.portfolio-backlog-toolpane
Requirements (User Story, Product Backlog Item)	ms.vss-work-web.requirement-backlog-toolpane
Sprint Backlog	ms.vss-work-web.iteration-backlog-toolpane

For more information, see the [Azure DevOps Services Extension Sample](#).

## Update your extension manifest

Update your [extension manifest](#) file with the following code:

JSON

```
...
  "contributions": [
    {
```

```

    "id": "Fabrikam.HelloWorld.Backlogs.Panel",
    "type": "ms.vss-work-web.backlog-panel",
    "description": "Adds a 'Hello' panel to Product and Iteration backlog pages.",
    "targets": [
        "ms.vss-work-web.requirement-backlog-toolpane",
        "ms.vss-work-web.portfolio-backlog-toolpane",
        "ms.vss-work-web.iteration-backlog-toolpane"
    ],
    "properties": {
        "title": "Hello Panel Pane",
        "name": "Hello Panel",
        "uri": "index.html",
        "registeredObjectId": "backlogPanelObject"
    }
},
],
"scopes": [
    "vso.work"
]
...

```

## Contribution

For each contribution in your extension, the manifest defines

- the type of contribution (backlog panel in this case),
- the contribution target (the requirements, portfolio, and iteration backlogs in this case),
- and the properties that are specific to each type of contribution. For panels, we have

[\[ \] Expand table](#)

Property	Description
title	Tooltip text that appears on the menu item
name	What appears in the dropdown for panel selection
uri	Path (relative to the extension's base URI) of the page to surface as the panel
registeredObjectId	ID of the object registered for the panel

Learn about all of the places where you can add an extension in [Extensibility points](#).

## Scopes

Include the [scopes](#) that your extension requires. In this case, we need `vso.work` to access work items.

## Get selection events

To get selection events (information about what work items are selected) implement this interface on your registered object.

JavaScript

```
...
    IContributedPanel {
        workItemSelectionChanged: (selectedWorkItems) => void;
    }
...
```

## Next steps

[Package, Publish, and Install](#) or

[Test and Debug](#)

---

## Feedback

Was this page helpful?

[!\[\]\(4b121ac00d27afaa77bea843bd8b0b45\_img.jpg\) Yes](#)

[!\[\]\(9e93f449b7f67d34a5a611dbb1aabcc8\_img.jpg\) No](#)

[Provide product feedback ↗](#)

# Extend the work item form

10/03/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

This article explains how to use extensions to customize the work item form. You can add an action, an observer, a group, or a page to the work item form in Azure DevOps.

## 💡 Tip

If you're starting a new Azure DevOps extension, try these maintained sample collections first—they work with current product builds and cover modern scenarios (for example, adding tabs on pull request pages).

- Azure DevOps extension sample (GitHub)—a compact starter sample that demonstrates common extension patterns: <https://github.com/microsoft/azure-devops-extension-sample> ↗
- Azure DevOps extension samples (legacy collection and contributions guide)—install to inspect UI targets, or view the source:  
<https://marketplace.visualstudio.com/items/ms-samples.samples-contributions-guide> ↗ and <https://github.com/Microsoft/vso-extension-samples/tree/master/contributions-guide> ↗
- Microsoft Learn samples (browse Azure DevOps samples)—curated, up-to-date samples across Microsoft docs: /samples/browse/?terms=azure%20devops%20extension

If a sample doesn't work in your organization, install it into a personal or test organization and compare the extension manifest's target IDs and API versions with the current docs.

For reference and APIs, see:

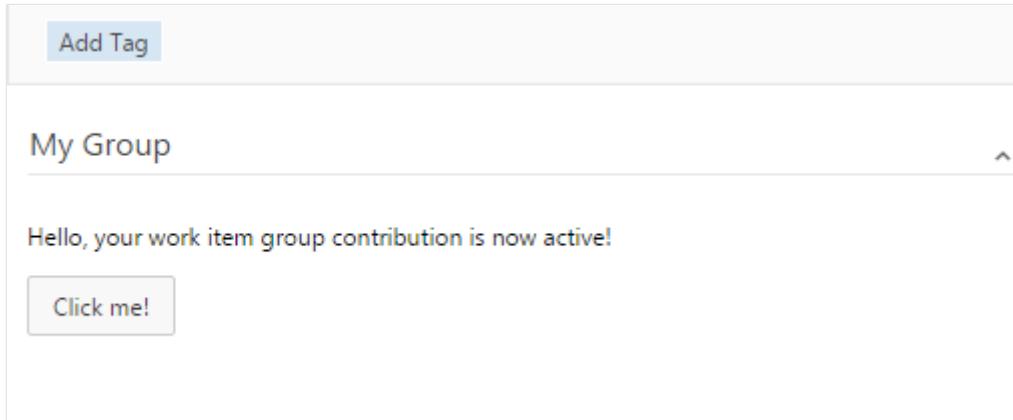
- [azure-devops-extension-api](#)
- [azure-devops-extension-sdk](#)
- [installed extension API](#)

- [Add a group to the main page](#)
- [Add a page \(tab\)](#)
- [Add an action to the context menu](#)
- [Add a custom control](#)
- [Listen for events on the form](#)

- Configure contributions in work item form

For the full source code, see the [UI example](#) in the [Azure DevOps extension samples](#) on GitHub.

## Add a group



To add a group to the main page, add a contribution to your extension manifest. The type of this contribution should be `ms.vss-work-web.work-item-form-group` and it should target the `ms.vss-work-web.work-item-form` contribution.

```
JSON

"contributions": [
  {
    "id": "sample-work-item-form-group",
    "type": "ms.vss-work-web.work-item-form-group",
    "description": "Custom work item form group",
    "targets": [
      "ms.vss-work-web.work-item-form"
    ],
    "properties": {
      "name": "My Group",
      "uri": "workItemGroup.html",
      "height": 600
    }
  }
]
```

## Properties

Expand table

Property	Description
name	Text that appears on the group.
uri	URI to a page that hosts the html that shows on the work item form and its scripts.
height	(Optional) Defines the height of the group. When omitted, it's 100%.

## JavaScript sample

This example shows how to register an object that's called when events occur on the work item form that might affect your contributed group.

JavaScript

```
import { IWorkItemFormService, WorkItemTrackingServiceIds } from "azure-devops-extension-api/WorkItemTracking";
import * as SDK from "azure-devops-extension-sdk";

// Get the WorkItemFormService. This service allows you to get/set fields/links
// on the 'active' work item (the work item
// that currently is displayed in the UI).
async function getWorkItemFormService()
{
    const workItemFormService = await SDK.getService<IWorkItemFormService>
(WorkItemTrackingServiceIds.WorkItemFormService);
    return workItemFormService;
}

// Register a listener for the work item group contribution after initializing the
// SDK.
SDK.register(SDK.getContributionId(), function () {
    return {
        // Called when the active work item is modified
        onFieldChanged: function(args) {
            $(".events").append($("<div/>").text("onFieldChanged - " +
JSON.stringify(args)));
        },
        // Called when a new work item is being loaded in the UI
        onLoad: function (args) {

            getWorkItemFormService().then(function(service) {
                // Get the current values for a few of the common fields
                service.getFieldValues(["System.Id", "System.Title",
"System.State", "System.CreatedDate"]).then(
                    function (value) {
                        $(".events").append($("<div/>").text("onLoaded - " +
JSON.stringify(value)));
                    });
            });
        };
    };
}
```

```

    },
    // Called when the active work item is being unloaded in the UI
    onUnloaded: function (args) {
        $(".events").empty();
        $(".events").append($(".<div/>").text("onUnloaded - " +
JSON.stringify(args)));
    },
    // Called after the work item has been saved
    onSaved: function (args) {
        $(".events").append($(".<div/>").text("onSaved - " +
JSON.stringify(args)));
    },
    // Called when the work item is reset to its unmodified state (undo)
    onReset: function (args) {
        $(".events").append($(".<div/>").text("onReset - " +
JSON.stringify(args)));
    },
    // Called when the work item has been refreshed from the server
    onRefreshed: function (args) {
        $(".events").append($(".<div/>").text("onRefreshed - " +
JSON.stringify(args)));
    }
});

```

## Events

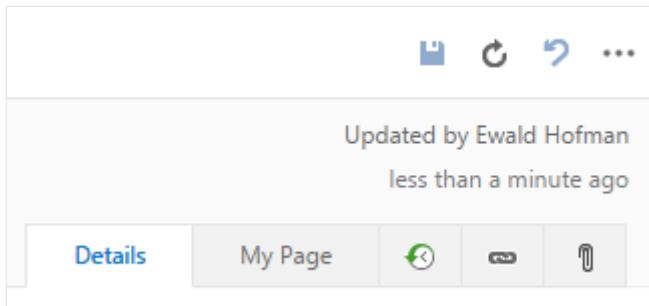
[] Expand table

Event	Event description	Argument	Argument description
onFieldChanged	Fired after a field has changed. If the field change ran rules that updated other fields, all these changes are part of a single event.	ID	The ID of the work item
changedFields	Array with the reference name of all changed fields.	ID	The ID of the work item
onLoaded	Fired after the data gets loaded in the work item form, when the user opens a work item, or when the user navigates to another work item in the triage view.	ID	The ID of the work item
onReset	Fired after the user undoes the changes to the work item.	ID	The ID of the work item

Event	Event description	Argument	Argument description
onRefreshed	Fired after the user refreshes the work item manually.	ID	The ID of the work item
onSaved	Fired after a work item is saved. For work items in a dialog, you should target the <code>ms.vss-work-web.work-item-notifications</code> type to ensure the event fires since once the dialog closes, this contribution type gets unloaded. For more information, see <a href="#">Listen for events</a> .	ID	The ID of the work item
onUnloaded	Fired before the user closes the dialog, or before the user moves to another work item in the triage view.	ID	The ID of the work item

## Add a page

A new page is rendered as a tab on the work item form. New pages appear next to the **Details** tab.



To add a page to the work item form, add a contribution to your extension manifest. The type of this contribution should be `ms.vss-work-web.work-item-form-page` and it should target the `ms.vss-work-web.work-item-form` contribution.

### JSON

```
"contributions": [
  {
    "id": "sample-work-item-form-page",
    "type": "ms.vss-work-web.work-item-form-page",
    "description": "Custom work item form page",
    "targets": [
      "ms.vss-work-web.work-item-form"
    ],
    "properties": {
      "name": "My Page",
      "uri": "workItemPage.html"
    }
  }
]
```

```
        }  
    ]
```

## Properties

[Expand table](#)

Property	Description
name	Text that appears on the tab page.
uri	URI to a page that hosts the html that shows on the work item form and its scripts.

## JavaScript sample

See the JavaScript sample in the form group section. The name of the registered object should match the `id` of the contribution.

## Events

[Expand table](#)

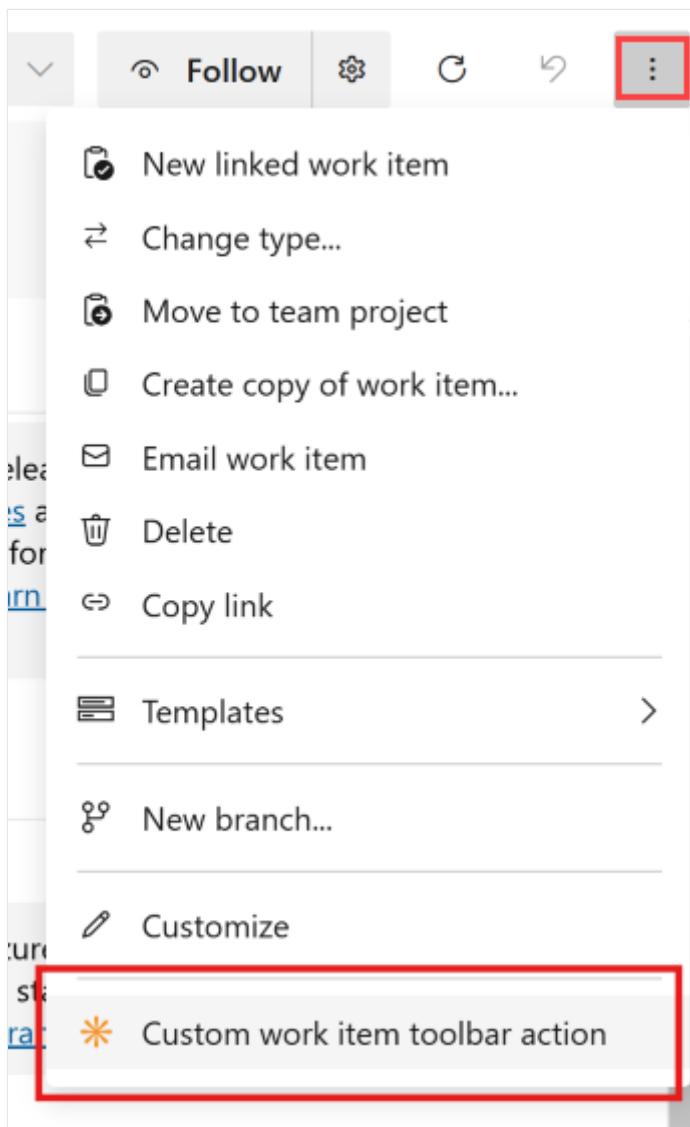
Event	Event description	Argument	Argument description
<code>onFieldChanged</code>	Fired after a field has changed. If the field change ran rules that updated other fields, all these changes are part of a single event.	ID	The ID of the work item
<code>changedFields</code>	Array with the reference name of all changed fields.	ID	The ID of the work item
<code>onLoaded</code>	Fired after the data gets loaded in the work item form, when the user opens a work item, or when the user navigates to another work item in the triage view.	ID	The ID of the work item
<code>onReset</code>	Fired after the user undoes the changes to the work item.	ID	The ID of the work item
<code>onRefreshed</code>	Fired after the user refreshes the work item manually.	ID	The ID of the work item
<code>onSaved</code>	Fired after a work item is saved. For work items in a dialog, you should target the <code>ms.vss-work-web.work-item-notifications</code> type to ensure the event fires since once	ID	The ID of the work item

Event	Event description	Argument	Argument description
	the dialog closes, this contribution type gets unloaded. For more information, see <a href="#">Listen for events</a> .		
onUnloaded	Fired before the user closes the dialog, or before the user moves to another work item in the triage view.	ID	The ID of the work item

## Configure contributions in work item form

In Azure DevOps Services, by default, the group extensions appear in the end of the second column of the form, and page contributions appear after all the work item form pages as a tab. Control contributions aren't shown in the form by default, so users have to manually add them to the form. In Azure DevOps Server, to show/hide or move the control, group and page contributions in work item form, see [Configure work item form extensions](#).

### Add menu action



To add an item to the work item toolbar, add this contribution to your extension manifest. Select the vertical ellipsis in the work item form to see the dropdown menu.

JSON

```
"contributions": [
  {
    "id": "sample-work-item-menu",
    "type": "ms.vss-web.action",
    "description": "Sample toolbar item which updates the title of a work item",
    "targets": [
      "ms.vss-work-web.work-item-context-menu"
    ],
    "properties": {
      "text": "Try me!",
      "title": "Updates the title of the work item from the extension",
      "toolbarText": "Try me!",
      "icon": "images/show-properties.png",
      "uri": "menu-workItemToolbarButton.html"
    }
  }
]
```

## Properties

[ ] Expand table

Property	Description
<code>text</code>	Text that appears on the toolbar item.
<code>title</code>	Tooltip text that appears on the menu item.
<code>toolbarText</code>	Text that appears when the item is being hovered over.
<code>uri</code>	URI to a page that registers the toolbar action handler.
<code>icon</code>	URL to an icon that appears on the menu item. Relative URLs are resolved using <code>baseUri</code> .
<code>group</code>	Determines where the menu item appears, related to others. Toolbar items with the same group name are grouped and divided by a separator from the rest of the items.
<code>registeredObjectId</code>	(Optional) Name of the registered menu action handler. Defaults to the contribution ID.

## Listen for events

To add an observer to the work item, which listens to the work item events, add this contribution to your extension manifest. There's no visualization for observers on the work item form. This is the best way to listen to work item form *onSaved* event since the observer lives outside of the form and doesn't get destroyed when form closes, which might happen right after save.

JSON

```
"contributions": [
  {
    "id": "sample-work-item-form-observer",
    "type": "ms.vss-work-web.work-item-notifications",
    "description": "Gets events about the current work item form for the 'Try Me!' toolbar button",
    "targets": [
      "ms.vss-work-web.work-item-form"
    ],
    "properties": {
      "uri": "myformobserver.html"
    }
  }
]
```

## Properties

[ ] Expand table

Property	Description
uri	URI to a page that hosts the scripts listening to events.

## Events

[ ] Expand table

Event	Event description	Argument	Argument description
onFieldChanged	Fired after a field has changed. If the field change ran rules that updated other fields, all these changes are part of a single event.	ID	The ID of the work item
changedFields	Array with the reference name of all changed fields.	ID	The ID of the work item

Event	Event description	Argument	Argument description
onLoaded	Fired after the data gets loaded in the work item form, when the user opens a work item, or when the user navigates to another work item in the triage view.	ID	The ID of the work item
onReset	Fired after the user undoes the changes to the work item.	ID	The ID of the work item
onRefreshed	Fired after the user refreshes the work item manually.	ID	The ID of the work item
onSaved	Fired after a work item is saved. For work items in a dialog, you should target the <code>ms.vss-work-web.work-item-notifications</code> type to ensure the event fires since once the dialog closes, this contribution type gets unloaded. For more information, see <a href="#">Listen for events</a> .	ID	The ID of the work item
onUnloaded	Fired before the user closes the dialog, or before the user moves to another work item in the triage view.	ID	The ID of the work item

## HTML/JavaScript sample

### HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Work item extension page sample</title>
</head>

<body>
    <script src="sdk/scripts/SDK.js"></script>

    <script>
        SDK.init({
            usePlatformScripts: true
        });

        SDK.ready(function () {
            // Register a listener for the work item page contribution.
            SDK.register(SDK.getContributionId(), function () {
                return {
                    // Called when the active work item is modified
                    onFieldChanged: function(args) {

                },
            })
        });
    </script>
</body>
</html>
```

```

        // Called when a new work item is being loaded in the UI
onLoaded: function (args) {

    },

        // Called when the active work item is being unloaded in the
UI
onUnloaded: function (args) {

    },

        // Called after the work item has been saved
onSaved: function (args) {

    },

        // Called when the work item is reset to its unmodified state
(undo)
onReset: function (args) {

    },

        // Called when the work item has been refreshed from the
server
onRefreshed: function (args) {

    }
}
});
});
});
</script>
</body>
</html>

```

## Changes with New Boards Hub

! Note

The New Boards Hub feature is enabled by default. We strongly suggest that you test your internal extensions with the New Boards Hub to ensure compatibility.

## Use the latest SDKs

Make sure your extension is using the latest version of [azure-devops-extension-sdk](#)

When using the new SDK, you should also be using the [azure-devops-extension-api](#) package for REST APIs. We update the methods and interfaces every sprint to ensure it includes all of

the latest features.

## When to use action or action-provider

Use `ms.vss-web.action-provider` when dynamically loading menu items using `getMenuItems` on the menu handler. Avoid using `ms.vss-web.action-provider` when your menu items are static and defined in your manifest. Instead `ms.vss-web.action` should be used.

## Package `require("VSS/Events/Document")` is no longer supported

`require("VSS/Events/Document")` import is no longer supported with the New Boards Hub.

ⓘ Note: The author created this article with assistance from AI. [Learn more](#)

# Add a custom control to the work item form

Article • 09/09/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Custom controls allow you to change how users view and interact with a field on the work item form. The following article walks you through how this sample custom control was built. This article shows you how to build your own custom control.

## Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Prerequisites

Include the `azure-devops-extension-sdk` in your project:

1. Install the SDK via npm: `npm install azure-devops-extension-sdk`.
2. Initialize it in your JavaScript code:

JavaScript

```
import * as SDK from "azure-devops-extension-sdk";
SDK.init();
```

## Add the custom control

To add a control to the main page, add a contribution to your [extension manifest](#). The type of contribution should be `ms.vss-work-web.work-item-form-control` and it should target the `ms.vss-work-web.work-item-form` contribution.

JSON

```
"contributions": [
{
  "id": "sample-work-item-form-control",
  "type": "ms.vss-work-web.work-item-form-control",
```

```

    "description": "Custom work item form control",
    "targets": [
        "ms.vss-work-web.work-item-form"
    ],
    "properties": {
        "name": "My Control",
        "uri": "workItemControl.html",
        "height": 600
    }
}
],
"manifestVersion": 1.0,
"scopes": [
    "vso.work"
]

```

The work item form adds an IFrame to host the custom control.

## Properties

[\[ \] Expand table](#)

Property	Description
<code>name</code>	Text that appears on the group.
<code>uri</code>	URI to a page that hosts the html that is loaded by the IFrame.
<code>height</code>	(Optional) Defines the height of the IFrame. When omitted, it's 50 pixels.
<code>inputs</code>	The values a user provides within the form.

If you want to dynamically resize the IFrame, you can use the `resize method` available in the client SDK.

A custom control on the work item form is another type of [contribution](#), like [group & page contribution](#). The main difference is that a control contribution can take a set of user inputs while group and page contributions can't.

## Control contribution inputs

To define the inputs for your control contribution, use the `inputs` property in the contribution object in the manifest.

In the following sample you see two inputs: `FieldName` and `Colors`. `FieldName` specifies with which field the control associates. `Colors` configures which colors map to which

values in the control.

The values for the inputs get provided by the users when they add to the work item form. These values get passed to the control contribution when it's loaded on the form.

#### JSON

```
"inputs": [
  {
    "id": "FieldName",
    "description": "The field associated with the control.",
    "type": "WorkItemField",
    "properties": {
      "workItemFieldTypes": ["String"]
    },
    "validation": {
      "dataType": "String",
      "isRequired": true
    }
  },
  {
    "id": "Colors",
    "description": "The colors that match the values in the control.",
    "type": "string",
    "validation": {
      "dataType": "String",
      "isRequired": false
    }
  }
]
```

The following properties define a user input that the contribution can use:

- **id** - A unique ID for the input.
- **description** - A few sentences describing the input.
- **type (optional)** - The type of input.
  - Valid values:
    - `WorkItemField` - Indicates that the input is a Work Item field. The value provided by the user for this input should be a reference name for the valid work item field.
- **properties (optional)** - Custom properties for the input.
  - Valid keys:
    - `workItemFieldTypes` - Defines an array of field types that this input supports.  
Valid values:
      - `String`
      - `Integer`
      - `DateTime`

- PlainText
- HTML
- TreePath
- History
- Double
- Guid
- Boolean
- Identity
- PicklistString
- PicklistInteger
- PicklistDouble

- validation - A set of properties that defines which values are considered valid for the input.
- Valid keys:
  - dataType - Defines the data type of the input value. Valid values for this property:
    - String
    - Number
    - Boolean
    - Field
  - isRequired - A boolean value, which indicates whether the input is required to have a value.

## JavaScript sample

A control extension works like a group or page extension with one difference that it can take certain user inputs. To read the user input values, use

`vss.getConfiguration().witInputs`. The following sample shows how to register an object that's called when events occur on the work item form that may affect your contributed control. It also shows how to read input values provided by user for this control.

TypeScript

```
import { Control } from "control";
import * as SDK from "azure-devops-extension-sdk";
import * as ExtensionContracts from "azure-devops-extension-
api/WorkItemTracking/WorkItemTracking";

let control;
```

```

const provider = () => {
  return {
    onLoaded: (workItemLoadedArgs) => {
      // create the control
      const config = SDK.getConfiguration();
      const fieldName = config.witInputs["FieldName"];
      const colors = config.witInputs["Colors"];
      control = new Control(fieldName, colors);
    },
    onFieldChanged: (fieldChangedEventArgs) => {
      const changedValue =
        fieldChangedEventArgs.changedFields[control.getFieldName()];
      if (changedValue !== undefined) {
        control.updateExternal(changedValue);
      }
    }
  };
};

SDK.init();
SDK.ready().then(() => {
  SDK.register(SDK.getContributionId(), provider);
});

```

The following screenshot shows a sample custom work item control for the *Priority* field.

Planning

Priority

- 1 - Critical
- 2 - High
- 3 - Medium
- 4 - Low

## Feedback

Was this page helpful?

 Yes

 No

Provide product feedback ↗

# Add extensions in work item form via work item type definition xml

Article • 02/06/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

You can export a work item type as xml using the `witadmin` tool, which includes the layout of the work item form. As part of this example, we add the page, group, and control contributions to the layout. We also add the control to the Agile "User Story" work item type. For more information, see [WebLayout xml reference](#).

## ⓘ Note

Work item form customization via xml is supported only on Azure DevOps Server, not Azure DevOps Services.

## Add extension in work item form

1. Install work item form extensions in Azure DevOps Server.
2. Open the `Developer Command Prompt`. Export the xml file to your desktop with the following command.

```
witadmin exportwitd /collection:CollectionURL /p:Project /n:TypeName  
/f:FileName
```

A file gets created in the directory that you specified.

3. Inside this file, go to the **WebLayout** section. Inside the WebLayout section, a comment blob specifies which installed extensions target work item forms for your collection. For each extension, all its form contributions are listed with their IDs and inputs (if it's a Control contribution). In the following example, the comment shows the *color-control-dev* extension installed on the collection. The extension has one control contribution that takes two inputs.

XML

```

<!--*****Work Item
Extensions*****


Extension:
  Name: color-control-dev
  Id: example.color-control-dev

  Control contribution:
    Id: example.color-control-dev.color-control-contribution
    Description:
    Inputs:
      Id: FieldName
      Description: The field associated with the control.
      Type: Field
      IsRequired: true

      Id: Colors
      Descriptions: The colors that match the values in the
control.
      Type: String
      IsRequired: false

```

4. Find your extension ID in the **Work Item Extensions** section:

XML

```

<!--*****Work Item
Extensions*****


Extension:
  Name: color-control-dev
  Id: example.color-control-dev
  ...

```

5. Add an extension tag below the **Work Item Extensions** section, shown as follows, to make your extension available to the work item form. To place a contribution inside the form, its extension must be specified in the **Extensions** section.

XML

```

<!--*****Work Item
Extensions*****


  ...

  Note: For more information on work item extensions use the
following topic:
  https://go.microsoft.com/fwlink/?LinkId=816513
  -->

  <Extensions>

```

```
<Extension Id="example.color-control-dev" />
</Extensions>
```

6. Specifying the extensions in the xml automatically places both the **page** and **group** contributions defined in the extensions inside the form. You can move the contributions in the following examples.

## Add page contribution

XML

```
<Page Id="Details">
<PageContribution Id="<page contribution id>" />
...
...
```

## Add group contribution

XML

```
<Page Id="Details">
...
<Section>
...
<GroupContribution Id="<group contribution id>" />
...
...
```

A page contribution and a group contribution can't take any other layout elements.

## Add control contribution

Unlike *page* and *group* contributions, specifying the extensions in the xml doesn't automatically place *control* contributions. To add these contributions in the form, add them with a contribution tag inside the form. The following example adds the *ControlContribution* to the *Planning* group.

If a control contribution has any required input defined, users must give a value for that input. For any non-required input, users can decide whether to set a value to the input. In the following example, the `FieldName` and `Colors` inputs get set.

XML

```
<Page Id="Details">
...
...
```

```
<Section>
...
    <Group Id="Planning">
        ...
            <ControlContribution Label="Priority" Id="example.color-
control-dev.color-control-contribution">
                <Inputs>
                    <Input Id="FieldName" Value="Microsoft.Azure DevOps
Services.Common.Priority" />
                    <Input Id="Colors" Value="red;green" />
                </Inputs>
            </ControlContribution>

            <Control Label="Risk" Type="FieldControl"
FieldName="Microsoft.Azure DevOps Services.Common.Risk" />

```

7. Import this xml file, using `witadmin`.

```
witadmin importwitd /collection:CollectionURL /p:Project /f:FileName
```

Your extension is configured via the work item form!

# Add tabs on query result pages

Article • 10/04/2022

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

If you have a web page that can be hosted in an iframe, it can be hosted in Azure DevOps as a tab on the query result pages. In this example, we'll add a simple Hello World tab on query results.

The screenshot shows the Azure DevOps interface for query results. At the top, there are tabs: 'Results' (selected), 'Editor', 'Charts', and a custom tab labeled 'Hello' which is highlighted with a green box. Below the tabs is a toolbar with icons for 'Save query', 'Refresh', 'Copy', and 'Checklist'. The main area displays a table with columns: 'ID', 'Work Item...', and 'Title'. The first row of the table is partially visible.

## Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Create your web page

1. Get the Client SDK `SDK.js` file and add it to your web app. Place it in the `home/sdk/scripts` folder.
  - a. Use the 'npm install' command to retrieve the SDK: `npm install azure-devops-extension-sdk`.
  - b. To learn more about the SDK, visit the [Client SDK GitHub Page](#).
2. Add the web page that you want to display as a hub. We're doing a simple `hello-world.html` page here, added to the `home` directory.
3. In your HTML page, add a reference to the SDK, and call `init()` and `notifyLoadSucceeded()`.

HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Hello World</title>
    <script src="sdk/scripts/SDK.js"></script>
</head>
<body>
    <script type="text/javascript">SDK.init();</script>
    <h1>Hello World</h1>
    <script type="text/javascript">SDK.notifyLoadSucceeded();</script>
</body>
</html>
```

## Update your extension manifest

Update your [extension manifest](#) file with the following code:

JSON

...

```
"contributions": [
    {
        "id": "Fabrikam.HelloTab.Query.Tabs",
        "type": "ms.vss-web.tab",
        "description": "Adds a 'Hello' tab to the query results",
        "targets": [
            "ms.vss-work-web.query-tabs"
        ],
        "properties": {
            "uri": "hello-query-tab.html",
            "title": "Hello",
            "name": "Hello"
        }
    }
],
"scopes": [
    "vso.work"
],
"files": [
    {
        "path": "hello-query-tab.html", "addressable": true
    },
    {
        "path": "scripts", "addressable": true
    },
    {
        "path": "sdk/scripts", "addressable": true
    },
    {
        "path": "images/logo.png", "addressable": true
    }
]
```

```
        }
    ]
...
}
```

## Contributions

The **contributions** stanza contains information about your tasks.

For each contribution in your extension, the manifest defines

- the type of contribution (tab in this case),
- the contribution target (the query tabs in this case),
- and the properties that are specific to each type of contribution. For a tab, we have

Property	Description
name	Name of the tab
uri	Path (relative to the extension's base URI) of the page to surface as the tab
title	Title of the tab to display

## Scopes

It includes the **scopes** that your extension requires. In this case, we need `vso.work` to access work items.

## Files

Include all of the files your extension accesses.

For your files, set `addressable` to `true` unless you include other files that don't need to be URL-addressable.

## Example

JavaScript

```
SDK.register(SDK.getContributionId(), {
    pageTitle: function(state) {
        return "Hello";
    },
    updateContext: function(tabContext) {
    },
    isVisible: function(state) {
```

```
        return false;
    },
    isDisabled: function(state) {
        return false;
    }
});
```

Learn about all of the places where you can add a hub in [Extensibility points](#).

## Next steps

[Package, Publish, and Install](#) or

[Test and Debug](#)

# Add a dashboard widget

10/03/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

Widgets are implemented as [contributions](#) in the [extension framework](#). A single extension can include multiple widget contributions. This article shows how to create an extension that provides one or more widgets.

## 💡 Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## 💡 Tip

If you're starting a new Azure DevOps extension, try these maintained sample collections first—they work with current product builds and cover modern scenarios (for example, adding tabs on pull request pages).

- Azure DevOps extension sample (GitHub)—a compact starter sample that demonstrates common extension patterns: <https://github.com/microsoft/azure-devops-extension-sample> ↗
- Azure DevOps extension samples (legacy collection and contributions guide)—install to inspect UI targets, or view the source:  
<https://marketplace.visualstudio.com/items/ms-samples.samples-contributions-guide> ↗ and <https://github.com/Microsoft/vso-extension-samples/tree/master/contributions-guide> ↗
- Microsoft Learn samples (browse Azure DevOps samples)—curated, up-to-date samples across Microsoft docs: /samples/browse/?terms=azure%20devops%20extension

If a sample doesn't work in your organization, install it into a personal or test organization and compare the extension manifest's target IDs and API versions with the current docs. For reference and APIs, see:

- [azure-devops-extension-api](#)
- [azure-devops-extension-sdk](#)

- [installed extension API](#)

[+] [Expand table](#)

Requirement	Description
Programming knowledge	JavaScript, HTML, and CSS knowledge for widget development
Azure DevOps organization	<a href="#">Create an organization</a>
Text editor	We use <a href="#">Visual Studio Code</a> for tutorials
Node.js	<a href="#">Latest version of Node.js</a>
Cross-platform CLI	<a href="#">tfx-cli</a> to package extensions Install using: <code>npm i -g tfx-cli</code>
Project directory	Home directory with this structure after completing the tutorial:
	<pre>  --- README.md  --- sdk  --- node_modules  --- scripts  --- VSS.SDK.min.js  --- img  --- logo.png  --- scripts  --- hello-world.html // html page for your widget  --- vss-extension.json // extension manifest </pre>

## Tutorial overview

This tutorial teaches widget development through three progressive examples:

[+] [Expand table](#)

Part	Focus	What you learn
<a href="#">Part 1: Hello World</a>	Basic widget creation	Create a widget that displays text
<a href="#">Part 2: REST API integration</a>	Azure DevOps API calls	Add REST API functionality to fetch and display data

Part	Focus	What you learn
Part 3: Widget configuration	User customization	Implement configuration options for your widget

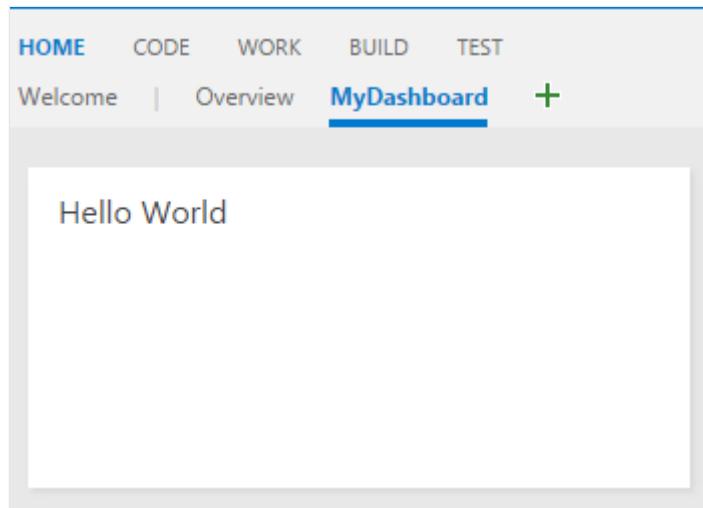
### 💡 Tip

If you prefer to jump straight to working examples, the included samples (see the previous note) show a set of widgets you can package and publish.

Before you begin, review the [basic widget styles](#) and structural guidance we provide.

## Part 1: Hello World

Create a basic widget that displays "Hello World" using JavaScript. This foundation demonstrates the core widget development concepts.



## Step 1: Install the client SDK

The VSS SDK enables your widget to communicate with Azure DevOps. Install it using npm:

```
Windows Command Prompt
```

```
npm install vss-web-extension-sdk
```

Copy the `vss.SDK.min.js` file from `vss-web-extension-sdk/lib` to your `home/sdk/scripts` folder.

For more SDK documentation, see the [Client SDK GitHub page](#).

## Step 2: Create the HTML structure

Create `hello-world.html` in your project directory. This file provides the widget's layout and references to required scripts.

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <script src="sdk/scripts/VSS.SDK.min.js"></script>
  </head>
  <body>
    <div class="widget">
      <h2 class="title"></h2>
    </div>
  </body>
</html>
```

Widgets run in iframes, so most HTML head elements except `<script>` and `<link>` get ignored by the framework.

## Step 3: Add widget JavaScript

To implement the widget functionality, add this script to the `<head>` section of your HTML file:

HTML

```
<script type="text/javascript">
  VSS.init({
    explicitNotifyLoaded: true,
    usePlatformStyles: true
  });

  VSS.require(["AzureDevOps/Dashboards/WidgetHelpers"], function (WidgetHelpers)
{
  WidgetHelpers.IncludeWidgetStyles();
  VSS.register("HelloWorldWidget", function () {
    return {
      load: function (widgetSettings) {
        var $title = $('h2.title');
        $title.text('Hello World');

        return WidgetHelpers.WidgetStatusHelper.Success();
      }
    };
  });
  VSS.notifyLoadSucceeded();
}</script>
```

```
});  
</script>
```

## Key JavaScript components

[ ] Expand table

Function	Purpose
VSS.init()	Initializes communication between widget and Azure DevOps
VSS.require()	Loads required SDK libraries and widget helpers
VSS.register()	Registers your widget with a unique identifier
WidgetHelpers.IncludeWidgetStyles()	Applies default Azure DevOps styling
VSS.notifyLoadSucceeded()	Notifies the framework that loading completed successfully

### (i) Important

The widget name in `VSS.register()` must match the `id` in your extension manifest (Step 5).

## Step 4: Add extension images

Create the required images for your extension:

- **Extension logo:** 98x98 pixel image named `logo.png` in the `img` folder
- **Widget catalog icon:** 98x98 pixel image named `CatalogIcon.png` in the `img` folder
- **Widget preview:** 330x160 pixel image named `preview.png` in the `img` folder

These images display in the Marketplace and widget catalog when users browse available extensions.

## Step 5: Create the extension manifest

Create `vss-extension.json` in your project's root directory. This file defines your extension's metadata and contributions:

JSON

```
{  
  "manifestVersion": 1,  
  "id": "azure-devops-extensions-myExtensions",  
  "version": "1.0.0",  
  "name": "My First Set of Widgets",  
  "description": "Samples containing different widgets extending dashboards",  
  "publisher": "fabrikam",  
  "categories": ["Azure Boards"],  
  "targets": [  
    {  
      "id": "Microsoft.VisualStudio.Services"  
    }  
  ],  
  "icons": {  
    "default": "img/logo.png"  
  },  
  "contributions": [  
    {  
      "id": "HelloWorldWidget",  
      "type": "ms.vss-dashboards-web.widget",  
      "targets": [  
        "ms.vss-dashboards-web.widget-catalog"  
      ],  
      "properties": {  
        "name": "Hello World Widget",  
        "description": "My first widget",  
        "catalogIconUrl": "img/CatalogIcon.png",  
        "previewImageUrl": "img/preview.png",  
        "uri": "hello-world.html",  
        "supportedSizes": [  
          {  
            "rowSpan": 1,  
            "columnSpan": 2  
          }  
        ],  
        "supportedScopes": ["project_team"]  
      }  
    }  
  ],  
  "files": [  
    {  
      "path": "hello-world.html",  
      "addressable": true  
    },  
    {  
      "path": "sdk/scripts",  
      "addressable": true  
    },  
    {  
      "path": "img",  
      "addressable": true  
    }  
  ]  
}
```

```
    ]  
}
```

### Important

Replace "publisher": "fabrikam" with your actual publisher name. Learn how to [create a publisher](#).

## Essential manifest properties

 [Expand table](#)

Section	Purpose
Basic info	Extension name, version, description, and publisher
Icons	Paths to your extension's visual assets
Contributions	Widget definitions including ID, type, and properties
Files	All files to include in the extension package

For complete manifest documentation, see [Extension manifest reference](#).

## Step 6: Package and publish your extension

Package your extension and publish it to the Visual Studio Marketplace.

### Install the packaging tool

```
Windows Command Prompt
```

```
npm i -g tfx-cli
```

### Create your extension package

From your project directory, run:

```
Windows Command Prompt
```

```
tfx extension create --manifest-globs vss-extension.json
```

This action creates a `.vsix` file that contains your packaged extension.

## Set up a publisher

1. Go to the [Visual Studio Marketplace Publishing Portal](#).
2. Sign in and create a publisher if you don't have one.
3. Choose a unique publisher identifier (used in your manifest file).
4. Update your `vss-extension.json` to use your publisher name instead of "fabrikam."

## Upload your extension

1. In the Publishing Portal, select **Upload new extension**.
2. Choose your `.vsix` file and upload it.
3. Share the extension with your Azure DevOps organization.

Alternatively, use the command line:

```
Windows Command Prompt
```

```
tfx extension publish --manifest-globs vss-extension.json --share-with  
yourOrganization
```

### 💡 Tip

Use `--rev-version` to automatically increment the version number when updating an existing extension.

## Step 7: Install and test your widget

To test, add your widget to a dashboard:

1. Go to your Azure DevOps project:  
`https://dev.azure.com/{Your_Organization}/{Your_Project}`.
2. Go to **Overview > Dashboards**.
3. Select **Add a widget**.
4. Find your widget in the catalog and select **Add**.

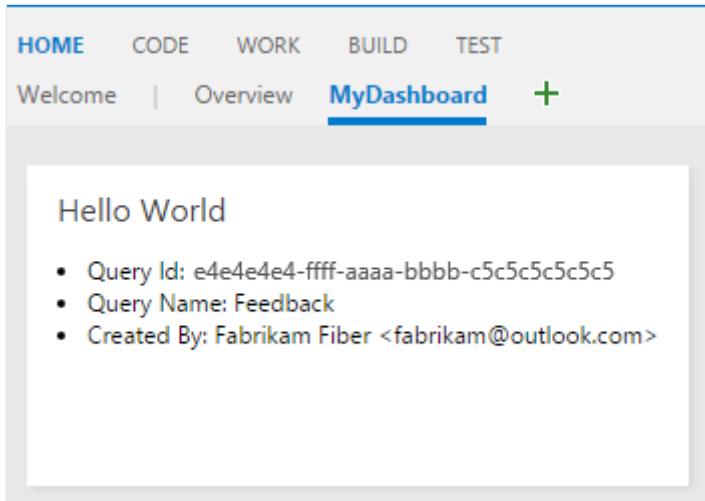
Your "Hello World" widget appears on the dashboard, displaying the text you configured.

**Next step:** Continue to [Part 2](#) to learn how to integrate Azure DevOps REST APIs into your widget.

## Part 2: Hello World with Azure DevOps REST API

Extend your widget to interact with Azure DevOps data using REST APIs. This example demonstrates how to fetch query information and display it dynamically in your widget.

In this part, use the [Work Item Tracking REST API](#) to retrieve information about an existing query and display the query details below the "Hello World" text.



### Step 1: Create the enhanced HTML file

Create a new widget file that builds on the previous example. Copy `hello-world.html` and rename it to `hello-world2.html`. Your project structure now includes:

```
|--- README.md
|--- node_modules
|--- sdk/
|   |--- scripts/
|   |   |--- VSS.SDK.min.js
|--- img/
|   |--- logo.png
|--- scripts/
|--- hello-world.html          // Part 1 widget
|--- hello-world2.html         // Part 2 widget (new)
|--- vss-extension.json        // Extension manifest
```

### Update the widget HTML structure

Make these changes to `hello-world2.html`:

1. Add a container for query data: Include a new `<div>` element to display query information.

2. Update the widget identifier: Change the widget name from `HelloWorldWidget` to `HelloWorldWidget2` for unique identification.

#### HTML

```
<!DOCTYPE html>
<html>
  <head>
    <script src="sdk/scripts/VSS.SDK.min.js"></script>
    <script type="text/javascript">
      VSS.init({
        explicitNotifyLoaded: true,
        usePlatformStyles: true
      });

      VSS.require(["AzureDevOps/Dashboards/WidgetHelpers"], function
(WidgetHelpers) {
        WidgetHelpers.IncludeWidgetStyles();
        VSS.register("HelloWorldWidget2", function () {
          return {
            load: function (widgetSettings) {
              var $title = $('h2.title');
              $title.text('Hello World');

              return WidgetHelpers.WidgetStatusHelper.Success();
            }
          }
        });
        VSS.notifyLoadSucceeded();
      });
    </script>
  </head>
  <body>
    <div class="widget">
      <h2 class="title"></h2>
      <div id="query-info-container"></div>
    </div>
  </body>
</html>
```

## Step 2: Configure API access permissions

Before making REST API calls, configure the required permissions in your extension manifest.

### Add the work scope

The `vso.work` scope grants read-only access to work items and queries. Add this scope to your `vss-extension.json`:

JSON

```
{  
  "scopes": [  
    "vso.work"  
  ]  
}
```

## Complete manifest example

For a complete manifest with other properties, structure it like this:

JSON

```
{  
  "name": "example-widget",  
  "publisher": "example-publisher",  
  "version": "1.0.0",  
  "scopes": [  
    "vso.work"  
  ]  
}
```

### Important

**Scope limitations:** Adding or changing scopes after publishing isn't supported. If you already published your extension, you must remove it from the Marketplace first. Go to the [Visual Studio Marketplace Publishing Portal](#), find your extension, and select Remove.

## Step 3: Implement REST API integration

Azure DevOps provides JavaScript REST client libraries through the SDK. These libraries wrap AJAX calls and map API responses to usable objects.

### Update the widget JavaScript

Replace the `vss.require` call in your `hello-world2.html` to include the Work Item Tracking REST client:

JavaScript

```

VSS.require(["AzureDevOps/Dashboards/WidgetHelpers",
"AzureDevOps/WorkItemTracking/RestClient"],
  function (WidgetHelpers, WorkItemTrackingRestClient) {
    WidgetHelpers.IncludeWidgetStyles();
    VSS.register("HelloWorldWidget2", function () {
      var projectId = VSS.getWebContext().project.id;

      var getQueryInfo = function (widgetSettings) {
        // Get a WIT client to make REST calls to Azure DevOps Services
        return WorkItemTrackingRestClient.getClient().getQuery(projectId,
"Shared Queries/Feedback")
          .then(function (query) {
            // Process query data (implemented in Step 4)

            return WidgetHelpers.WidgetStatusHelper.Success();
          }, function (error) {
            return
WidgetHelpers.WidgetStatusHelper.Failure(error.message);
          });
    }

    return {
      load: function (widgetSettings) {
        // Set your title
        var $title = $('h2.title');
        $title.text('Hello World');

        return getQueryInfo(widgetSettings);
      }
    });
  });
  VSS.notifyLoadSucceeded();
});

```

## Key implementation details

[ ] Expand table

Component	Purpose
<code>WorkItemTrackingRestClient.getClient()</code>	Gets an instance of the Work Item Tracking REST client
<code>getQuery()</code>	Retrieves query information wrapped in a promise
<code>WidgetStatusHelper.Failure()</code>	Provides consistent error handling for widget failures
<code>projectId</code>	Current project context required for API calls

💡 Tip

**Custom query paths:** If you don't have a "Feedback" query in "Shared Queries", replace "Shared Queries/Feedback" with the path to any query that exists in your project.

## Step 4: Display API response data

Render the query information in your widget by processing the REST API response.

### Add query data rendering

Replace the `// Process query data` comment with this implementation:

JavaScript

```
// Create a list with query details
var $list = $('<ul>');
$list.append($('<li>').text("Query ID: " + query.id));
$list.append($('<li>').text("Query Name: " + query.name));
$list.append($('<li>').text("Created By: " + (query.createdBy ?
query.createdBy.displayName : "<unknown>")));
// Append the list to the query-info-container
var $container = $('#query-info-container');
$container.empty();
$container.append($list);
```

The `getQuery()` method returns a `Contracts.QueryHierarchyItem` object with properties for query metadata. This example displays three key pieces of information below the "Hello World" text.

### Complete working example

Your final `hello-world2.html` file should look like this:

HTML

```
<!DOCTYPE html>
<html>
<head>
    <script src="sdk/scripts/VSS.SDK.min.js"></script>
    <script type="text/javascript">
        VSS.init({
            explicitNotifyLoaded: true,
            usePlatformStyles: true
        });

        VSS.require(["AzureDevOps/Dashboards/WidgetHelpers",
```

```

"AzureDevOps/WorkItemTracking/RestClient"],
    function (WidgetHelpers, WorkItemTrackingRestClient) {
        WidgetHelpers.IncludeWidgetStyles();
        VSS.register("HelloWorldWidget2", function () {
            var projectId = VSS.getWebContext().project.id;

            var getQueryInfo = function (widgetSettings) {
                // Get a WIT client to make REST calls to Azure DevOps
Services
                return
WorkItemTrackingRestClient.getClient().getQuery(projectId, "Shared
Queries/Feedback")
                    .then(function (query) {
                        // Create a list with query details
                        var $list = $('<ul>');
                        $list.append($('<li>').text("Query ID: " +
query.id));
                        $list.append($('<li>').text("Query Name: " +
query.name));
                        $list.append($('<li>').text("Created By: " +
(query.createdBy ? query.createdBy.displayName : "<unknown>")));

                        // Append the list to the query-info-container
                        var $container = $('#query-info-container');
                        $container.empty();
                        $container.append($list);

                        // Use the widget helper and return success as
Widget Status
                        return WidgetHelpers.WidgetStatusHelper.Success();
                    }, function (error) {
                        // Use the widget helper and return failure as
Widget Status
                        return
WidgetHelpers.WidgetStatusHelper.Failure(error.message);
                    });
            }

            return {
                load: function (widgetSettings) {
                    // Set your title
                    var $title = $('h2.title');
                    $title.text('Hello World');

                    return getQueryInfo(widgetSettings);
                }
            });
        });
        VSS.notifyLoadSucceeded();
    });
</script>

</head>
<body>
<div class="widget">

```

```
<h2 class="title"></h2>
<div id="query-info-container"></div>
</div>
</body>
</html>
```

## Step 5: Update the extension manifest

To make it available in the widget catalog, add your new widget to the extension manifest.

### Add the second widget contribution

Update `vss-extension.json` to include your REST API-enabled widget. Add this contribution to the `contributions` array:

JSON

```
{
  "contributions": [
    // ...existing HelloWorldWidget contribution...
    {
      "id": "HelloWorldWidget2",
      "type": "ms.vss-dashboards-web.widget",
      "targets": [
        "ms.vss-dashboards-web.widget-catalog"
      ],
      "properties": {
        "name": "Hello World Widget 2 (with API)",
        "description": "My second widget",
        "previewImageUrl": "img/preview2.png",
        "uri": "hello-world2.html",
        "supportedSizes": [
          {
            "rowSpan": 1,
            "columnSpan": 2
          }
        ],
        "supportedScopes": ["project_team"]
      }
    }
  ],
  "files": [
    {
      "path": "hello-world.html",
      "addressable": true
    },
    {
      "path": "hello-world2.html",
      "addressable": true
    }
  ]
}
```

```
{  
    "path": "sdk/scripts",  
    "addressable": true  
,  
{  
    "path": "img",  
    "addressable": true  
}  
,  
"scopes": [  
    "vso.work"  
]  
}
```

### 💡 Tip

**Preview image:** Create a `preview2.png` image (330x160 pixels) and place it in the `img` folder to show users what your widget looks like in the catalog.

## Step 6: Package, publish, and share

[Package, publish, and share your extension](#). If you already published the extension, you can repackage and update it directly in the Marketplace.

## Step 7: Test your REST API widget

To view the REST API integration in action, add the new widget to your dashboard:

1. Go to your Azure DevOps project:

`https://dev.azure.com/{Your_Organization}/{Your_Project}`.

2. Select **Overview > Dashboards**.
3. Select **Add a widget**.
4. Find "Hello World Widget 2 (with API)" and select **Add**.

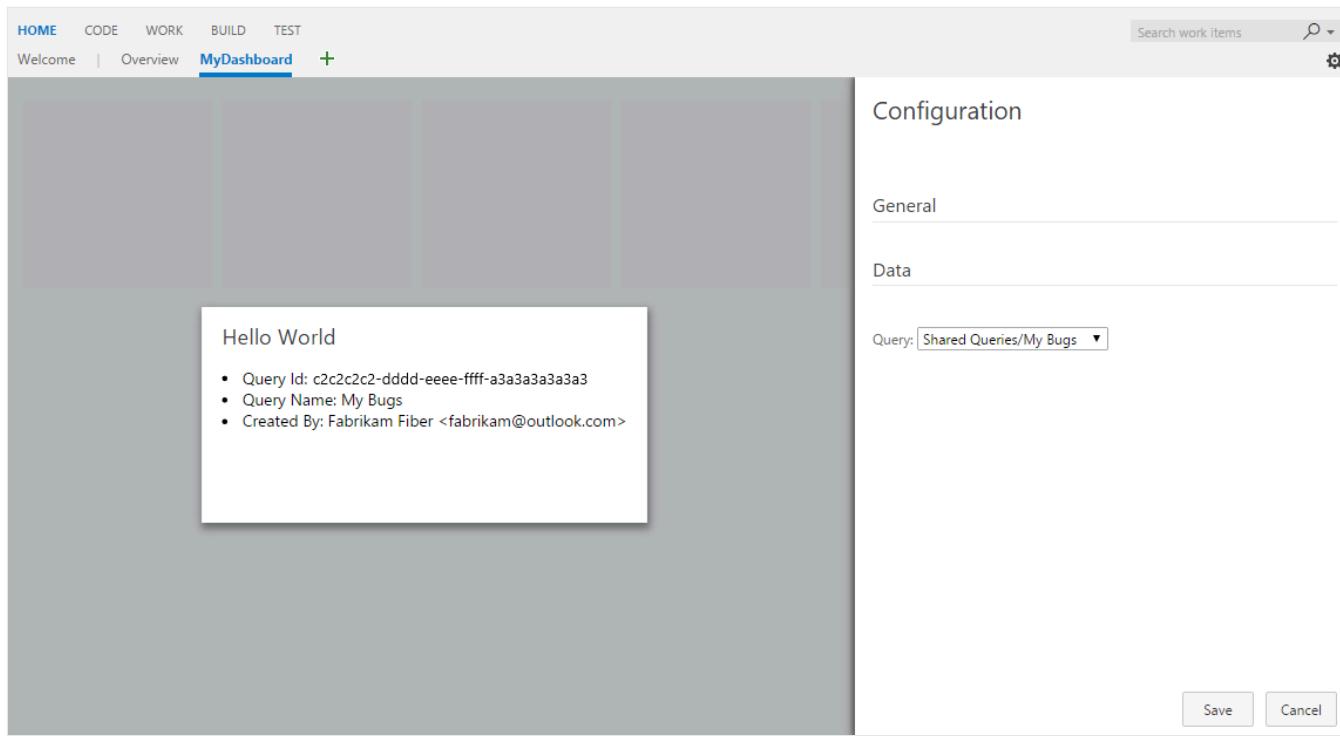
Your enhanced widget displays both the "Hello World" text and live query information from your Azure DevOps project.

**Next steps:** Continue to [Part 3](#) to add configuration options that let users customize which query to display.

## Part 3: Configure Hello World

Build on [Part 2](#) by adding user configuration capabilities to your widget. Instead of hard-coding the query path, create a configuration interface that lets users select which query to display, with live preview functionality.

This part demonstrates how to create configurable widgets that users can customize to their specific needs while providing real-time feedback during configuration.



## Step 1: Create configuration files

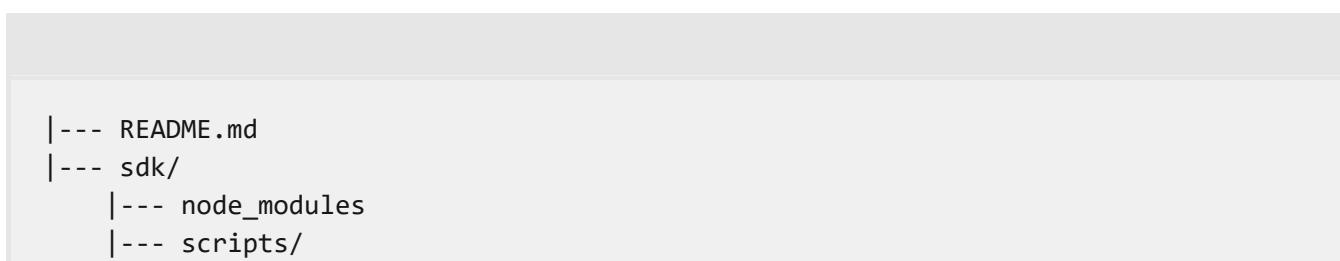
Widget configurations share many similarities with widgets themselves—both use the same SDK, HTML structure, and JavaScript patterns, but serve different purposes within the extension framework.

### Set up the project structure

To support widget configuration, create two new files:

1. Copy `hello-world2.html` and rename it to `hello-world3.html`, your configurable widget.
2. Create a new file called `configuration.html`, which handles the configuration interface.

Your project structure now includes:



```
|--- VSS.SDK.min.js
|--- img/
|   |--- logo.png
|--- scripts/
|--- configuration.html          // New: Configuration interface
|--- hello-world.html           // Part 1: Basic widget
|--- hello-world2.html          // Part 2: REST API widget
|--- hello-world3.html          // Part 3: Configurable widget (new)
|--- vss-extension.json         // Extension manifest
```

## Create the configuration interface

Add this HTML structure to `configuration.html`, which creates a dropdown selector for choosing queries:

HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script src="sdk/scripts/VSS.SDK.min.js"></script>
  </head>
  <body>
    <div class="container">
      <fieldset>
        <label class="label">Query: </label>
        <select id="query-path-dropdown" style="margin-top:10px">
          <option value="" selected disabled hidden>Please select a
          query</option>
          <option value="Shared Queries/Feedback">Shared
          Queries/Feedback</option>
          <option value="Shared Queries/My Bugs">Shared Queries/My
          Bugs</option>
          <option value="Shared Queries/My Tasks">Shared Queries/My
          Tasks</option>
        </select>
      </fieldset>
    </div>
  </body>
</html>
```

## Step 2: Implement configuration JavaScript

Configuration JavaScript follows the same initialization pattern as widgets, but implements the `IWidgetConfiguration` contract instead of the basic `IWidget` contract.

### Add configuration logic

Insert this script into the `<head>` section of `configuration.html`:

### HTML

```
<script type="text/javascript">
    VSS.init({
        explicitNotifyLoaded: true,
        usePlatformStyles: true
    });

    VSS.require(["AzureDevOps/Dashboards/WidgetHelpers"], function (WidgetHelpers) {
        VSS.register("HelloWorldWidget.Configuration", function () {
            var $queryDropdown = $("#query-path-dropdown");

            return {
                load: function (widgetSettings, widgetConfigurationContext) {
                    var settings = JSON.parse(widgetSettings.customSettings.data);
                    if (settings && settings.queryPath) {
                        $queryDropdown.val(settings.queryPath);
                    }
                }

                return WidgetHelpers.WidgetStatusHelper.Success();
            },
            onSave: function() {
                var customSettings = {
                    data: JSON.stringify({
                        queryPath: $queryDropdown.val()
                    })
                };
                return
            }
        });
        WidgetHelpers.WidgetConfigurationSave.Valid(customSettings);
    }
});
VSS.notifyLoadSucceeded();
});
</script>
```

## Configuration contract details

The `IWidgetConfiguration` contract requires these key functions:

[+] Expand table

Function	Purpose	When called
<code>load()</code>	Initialize configuration UI with existing settings	When configuration dialog opens
<code>onSave()</code>	Serialize user input and validate settings	When user selects <b>Save</b>

## 💡 Tip

**Data serialization:** This example uses JSON to serialize settings. The widget accesses these settings via `widgetSettings.customSettings.data` and must deserialize them accordingly.

## Step 3: Enable live preview functionality

Live preview allows users to see widget changes immediately as they modify configuration settings, providing instant feedback before saving.

### Implement change notifications

To enable live preview, add this event handler within the `load` function:

#### JavaScript

```
$queryDropdown.on("change", function () {
    var customSettings = {
        data: JSON.stringify({
            queryPath: $queryDropdown.val()
        })
    };
    var eventName = WidgetHelpers.WidgetEvent.ConfigurationChange;
    var eventArgs = WidgetHelpers.WidgetEvent.Args(customSettings);
    widgetConfigurationContext.notify(eventName, eventArgs);
});
```

## Complete configuration file

Your final `configuration.html` should look like this:

#### HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <script src="sdk/scripts/VSS.SDK.min.js"></script>
        <script type="text/javascript">
            VSS.init({
                explicitNotifyLoaded: true,
                usePlatformStyles: true
            });

            VSS.require(["AzureDevOps/Dashboards/WidgetHelpers"], function
(WidgetHelpers) {
```

```

        VSS.register("HelloWorldWidget.Configuration", function () {
            var $queryDropdown = $("#query-path-dropdown");

            return {
                load: function (widgetSettings,
widgetConfigurationContext) {
                    var settings =
JSON.parse(widgetSettings.customSettings.data);
                    if (settings && settings.queryPath) {
                        $queryDropdown.val(settings.queryPath);
                    }

                    $queryDropdown.on("change", function () {
                        var customSettings = {data:
JSON.stringify({queryPath: $queryDropdown.val()})};
                        var eventName =
WidgetHelpers.WidgetEvent.ConfigurationChange;
                        var eventArgs =
WidgetHelpers.WidgetEvent.Args(customSettings);
                        widgetConfigurationContext.notify(eventName,
eventArgs);
                    });
                }

                return WidgetHelpers.WidgetStatusHelper.Success();
            },
            onSave: function() {
                var customSettings = {data: JSON.stringify({queryPath:
$queryDropdown.val()})};
                return
WidgetHelpers.WidgetConfigurationSave.Valid(customSettings);
            }
        });
    VSS.notifyLoadSucceeded();
});
</script>
</head>
<body>
    <div class="container">
        <fieldset>
            <label class="label">Query: </label>
            <select id="query-path-dropdown" style="margin-top:10px">
                <option value="" selected disabled hidden>Please select a
query</option>
                <option value="Shared Queries/Feedback">Shared
Queries/Feedback</option>
                <option value="Shared Queries/My Bugs">Shared Queries/My
Bugs</option>
                <option value="Shared Queries/My Tasks">Shared Queries/My
Tasks</option>
            </select>
        </fieldset>
    </div>
</body>
</html>

```

### Important

**Enable Save button:** The framework requires at least one configuration change notification to enable the **Save** button. The change event handler ensures this action occurs when users select an option.

## Step 4: Make the widget configurable

Transform your widget from Part 2 to use configuration data instead of hard-coded values. This step requires implementing the `IConfigurableWidget` contract.

### Update widget registration

In `hello-world3.html`, make these changes:

1. **Update widget ID:** Change from `HelloWorldWidget2` to `HelloWorldWidget3`.
2. **Add reload function:** Implement the `IConfigurableWidget` contract.

#### JavaScript

```
return {
  load: function (widgetSettings) {
    // Set your title
    var $title = $('#h2.title');
    $title.text('Hello World');

    return getQueryInfo(widgetSettings);
  },
  reload: function (widgetSettings) {
    return getQueryInfo(widgetSettings);
  }
}
```

### Handle configuration data

Update the `getQueryInfo` function to use configuration settings instead of hard-coded query paths:

#### JavaScript

```
var settings = JSON.parse(widgetSettings.customSettings.data);
if (!settings || !settings.queryPath) {
```

```

var $container = $('#query-info-container');
$container.empty();
$container.text("Please configure a query path to display data.");

return WidgetHelpers.WidgetStatusHelper.Success();
}

```

## Widget lifecycle differences

[+] Expand table

Function	Purpose	Usage guidelines
load()	Initial widget rendering and one-time setup	Heavy operations, resource initialization
reload()	Update widget with new configuration	Lightweight updates, data refresh

### 💡 Tip

**Performance optimization:** Use `load()` for expensive operations that only need to run once, and `reload()` for quick updates when configuration changes.

## (Optional) Add a lightbox for detailed information

Dashboard widgets have limited space, making it challenging to display comprehensive information. A lightbox provides an elegant solution by showing detailed data in a modal overlay without navigating away from the dashboard.

## Why use a lightbox in widgets?

[+] Expand table

Benefit	Description
Space efficiency	Keep widget compact while offering detailed views
User experience	Maintain dashboard context while showing more information
Progressive disclosure	Show summary data in widget, details on demand
Responsive design	Adapt to different screen sizes and widget configurations

## Implement clickable elements

Update your query data rendering to include clickable elements that trigger the lightbox:

JavaScript

```
// Create a list with clickable query details
var $list = $('ul class="query-summary">');
$list.append($('li').text("Query ID: " + query.id));
$list.append($('li').text("Query Name: " + query.name));
$list.append($('li').text("Created By: " + (query.createdBy ?
query.createdBy.displayName : "<unknown>")));

// Add a clickable element to open detailed view
var $detailsLink = $('button class="details-link">View Details</button>');
$detailsLink.on('click', function() {
    showQueryDetails(query);
});

// Append to the container
var $container = $('#query-info-container');
$container.empty();
$container.append($list);
$container.append($detailsLink);
```

## Create the lightbox functionality

Add this lightbox implementation to your widget JavaScript:

JavaScript

```
function showQueryDetails(query) {
    // Create lightbox overlay
    var $overlay = $('div class="lightbox-overlay">');
    var $lightbox = $('div class="lightbox-content">');

    // Add close button
    var $closeBtn = $('button class="lightbox-close">&times;</button>');
    $closeBtn.on('click', function() {
        $overlay.remove();
    });

    // Create detailed content
    var $content = $('div class="query-details">');
    $content.append($('h3').text(query.name || 'Query Details'));
    $content.append($('p').html('<strong>ID:</strong> ' + query.id));
    $content.append($('p').html('<strong>Path:</strong> ' + query.path));
    $content.append($('p').html('<strong>Created:</strong> ' +
(query.createdDate ? new Date(query.createdDate).toLocaleDateString() :
'Unknown')));
    $content.append($('p').html('<strong>Modified:</strong> ' +
```

```

(query.lastModifiedDate ? new Date(query.lastModifiedDate).toLocaleDateString() : 
'Unknown')));
$content.append($('p').html('<strong>Created By:</strong> ' +
(query.createdBy ? query.createdBy.displayName : 'Unknown')));
$content.append($('p').html('<strong>Modified By:</strong> ' +
(query.lastModifiedBy ? query.lastModifiedBy.displayName : 'Unknown')));

if (query.queryType) {
    $content.append($('p').html('<strong>Type:</strong> ' +
query.queryType));
}

// Assemble lightbox
$lightbox.append($closeBtn);
$lightbox.append($content);
$overlay.append($lightbox);

// Add to document and show
$('body').append($overlay);

// Close on overlay click
$overlay.on('click', function(e) {
    if (e.target === $overlay[0]) {
        $overlay.remove();
    }
});

// Close on Escape key
$(document).on('keydown.lightbox', function(e) {
    if (e.keyCode === 27) { // Escape key
        $overlay.remove();
        $(document).off('keydown.lightbox');
    }
});
}

```

## Add lightbox styling

Include CSS styles for the lightbox in your widget HTML `<head>` section:

HTML

```

<style>
.query-summary {
    list-style: none;
    padding: 0;
    margin: 10px 0;
}

.query-summary li {
    padding: 2px 0;
    font-size: 12px;
}

```

```
}

.details-link {
    background: #0078d4;
    color: white;
    border: none;
    padding: 4px 8px;
    font-size: 11px;
    cursor: pointer;
    border-radius: 2px;
    margin-top: 8px;
}

.details-link:hover {
    background: #106ebe;
}

.lightbox-overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.7);
    z-index: 10000;
    display: flex;
    align-items: center;
    justify-content: center;
}

.lightbox-content {
    background: white;
    border-radius: 4px;
    padding: 20px;
    max-width: 500px;
    max-height: 80vh;
    overflow-y: auto;
    position: relative;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
}

.lightbox-close {
    position: absolute;
    top: 10px;
    right: 15px;
    background: none;
    border: none;
    font-size: 24px;
    cursor: pointer;
    color: #666;
    line-height: 1;
}

.lightbox-close:hover {
    color: #000;
}
```

```
}

.query-details h3 {
    margin-top: 0;
    color: #323130;
}

.query-details p {
    margin: 8px 0;
    font-size: 14px;
    line-height: 1.4;
}
</style>
```

## Enhanced widget implementation

Your complete enhanced widget with lightbox functionality:

### HTML

```
<!DOCTYPE html>
<html>
<head>
    <script src="sdk/scripts/VSS.SDK.min.js"></script>
    <style>
        /* Lightbox styles from above */
        .query-summary {
            list-style: none;
            padding: 0;
            margin: 10px 0;
        }

        .query-summary li {
            padding: 2px 0;
            font-size: 12px;
        }

        .details-link {
            background: #0078d4;
            color: white;
            border: none;
            padding: 4px 8px;
            font-size: 11px;
            cursor: pointer;
            border-radius: 2px;
            margin-top: 8px;
        }

        .details-link:hover {
            background: #106ebe;
        }
    </style>
</head>
<body>
    <ul class="query-summary">
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
    <a href="#" class="details-link">View Details</a>
</body>
</html>
```

```
.lightbox-overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.7);
    z-index: 10000;
    display: flex;
    align-items: center;
    justify-content: center;
}

.lightbox-content {
    background: white;
    border-radius: 4px;
    padding: 20px;
    max-width: 500px;
    max-height: 80vh;
    overflow-y: auto;
    position: relative;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
}

.lightbox-close {
    position: absolute;
    top: 10px;
    right: 15px;
    background: none;
    border: none;
    font-size: 24px;
    cursor: pointer;
    color: #666;
    line-height: 1;
}

.lightbox-close:hover {
    color: #000;
}

.query-details h3 {
    margin-top: 0;
    color: #323130;
}

.query-details p {
    margin: 8px 0;
    font-size: 14px;
    line-height: 1.4;
}
</style>
<script type="text/javascript">
VSS.init({
    explicitNotifyLoaded: true,
    usePlatformStyles: true
```

```

});

VSS.require(["AzureDevOps/Dashboards/WidgetHelpers",
"AzureDevOps/WorkItemTracking/RestClient"],
    function (WidgetHelpers, WorkItemTrackingRestClient) {
        WidgetHelpers.IncludeWidgetStyles();

        function showQueryDetails(query) {
            // Lightbox implementation from above
        }

VSS.register("HelloWorldWidget2", function () {
    var projectId = VSS.getWebContext().project.id;

    var getQueryInfo = function (widgetSettings) {
        return
WorkItemTrackingRestClient.getClient().getQuery(projectId, "Shared
Queries/Feedback")
        .then(function (query) {
            // Enhanced display with lightbox trigger
            var $list = $('

');
            $list.append($('- ').text("Query ID: " +
query.id));
            $list.append($('- ').text("Query Name: " +
query.name));
            $list.append($('- ').text("Created By: " +
(query.createdBy ? query.createdBy.displayName : "<unknown>")));
            var $detailsLink = $('View Details</button>');
            $detailsLink.on('click', function() {
                showQueryDetails(query);
            });

            var $container = $('#query-info-container');
            $container.empty();
            $container.append($list);
            $container.append($detailsLink);

            return WidgetHelpers.WidgetStatusHelper.Success();
        }, function (error) {
            return
WidgetHelpers.WidgetStatusHelper.Failure(error.message);
        });
    }

    return {
        load: function (widgetSettings) {
            // Set your title
            var $title = $('h2.title');
            $title.text('Hello World');

            return getQueryInfo(widgetSettings);
        }
    }
}

```

```

        });
        VSS.notifyLoadSucceeded();
    });
</script>
</head>
<body>
    <div class="widget">
        <h2 class="title"></h2>
        <div id="query-info-container"></div>
    </div>
</body>
</html>

```

**Accessibility considerations:** Ensure your lightbox is keyboard accessible and includes proper labels for screen readers. Test with Azure DevOps' built-in accessibility features.

### ⓘ Important

**Performance:** Lightboxes should load quickly. Consider lazy-loading detailed data only when the lightbox opens, rather than fetching everything upfront.

## Step 5: Configure the extension manifest

Register both the configurable widget and its configuration interface in your extension manifest.

### Add widget and configuration contributions

Update `vss-extension.json` to include two new contributions:

JSON

```
{
    "contributions": [
        {
            "id": "HelloWorldWidget3",
            "type": "ms.vss-dashboards-web.widget",
            "targets": [
                "ms.vss-dashboards-web.widget-catalog",
                "fabrikam.azuredevops-extensions-
myExtensions.HelloWorldWidget.Configuration"
            ],
            "properties": {
                "name": "Hello World Widget 3 (with config)",
                "description": "My third widget",
                "previewImageUrl": "img/preview3.png",
                "uri": "hello-world3.html",
            }
        }
    ]
}
```

```

        "supportedSizes": [
            {
                "rowSpan": 1,
                "columnSpan": 2
            },
            {
                "rowSpan": 2,
                "columnSpan": 2
            }
        ],
        "supportedScopes": ["project_team"]
    }
},
{
    "id": "HelloWorldWidget.Configuration",
    "type": "ms.vss-dashboards-web.widget-configuration",
    "targets": [ "ms.vss-dashboards-web.widget-configuration" ],
    "properties": {
        "name": "HelloWorldWidget Configuration",
        "description": "Configures HelloWorldWidget",
        "uri": "configuration.html"
    }
},
],
"files": [
    {
        "path": "hello-world.html", "addressable": true
    },
    {
        "path": "hello-world2.html", "addressable": true
    },
    {
        "path": "hello-world3.html", "addressable": true
    },
    {
        "path": "configuration.html", "addressable": true
    },
    {
        "path": "sdk/scripts", "addressable": true
    },
    {
        "path": "img", "addressable": true
    }
]
}

```

## Configuration contribution requirements

 Expand table

Property	Purpose	Required value
<code>type</code>	Identifies contribution as widget configuration	<code>ms.vss-dashboards-web.widget-configuration</code>
<code>targets</code>	Where configuration appears	<code>ms.vss-dashboards-web.widget-configuration</code>
<code>uri</code>	Path to configuration HTML file	Your configuration file path

## Widget targeting pattern

For configurable widgets, the `targets` array must include a reference to the configuration:

```
<publisher>.<extension-id>.<configuration-id>
```

### ⚠ Warning

**Configuration button visibility:** If the widget doesn't properly target its configuration contribution, the **Configure** button doesn't appear. Verify the publisher and extension names match your manifest exactly.

## Step 6: Package, publish, and share

Deploy your enhanced extension with configuration capabilities.

If it's your first publication, follow [Step 6: Package, publish, and share](#). For existing extensions, repackage and update directly in the Marketplace.

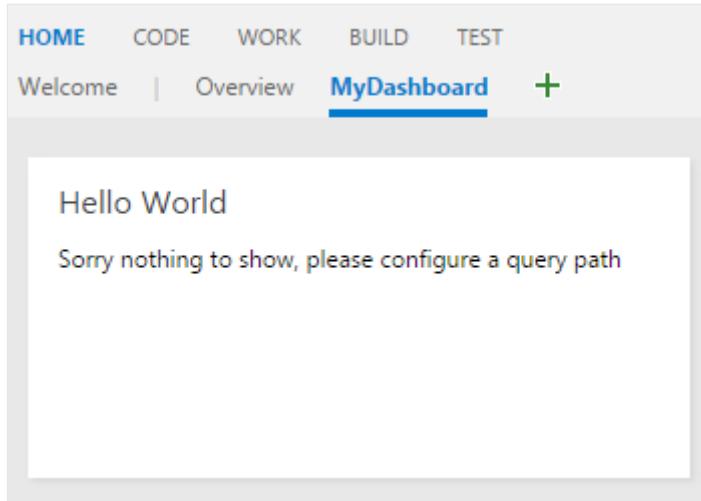
## Step 7: Test the configurable widget

Experience the full configuration workflow by adding and configuring your widget.

### Add the widget to your dashboard

1. Go to [https://dev.azure.com/{Your\\_Organization}/{Your\\_Project}](https://dev.azure.com/{Your_Organization}/{Your_Project}).
2. Go to **Overview > Dashboards**.
3. Select **Add a widget**.
4. Find "Hello World Widget 3 (with config)" and select **Add**.

A configuration prompt displays since the widget requires setup:



## Configure the widget

Access configuration through either method:

- **Widget menu:** Hover over the widget, select the ellipsis (...), then **Configure**
- **Dashboard edit mode:** Select **Edit** on the dashboard, then the configure button on the widget

The configuration panel opens with a live preview in the center. Select a query from the dropdown to see immediate updates, then select **Save** to apply your changes.

## Step 8: Add advanced configuration options

Extend your widget with more built-in configuration features like custom names and sizes.

### Enable name and size configuration

Azure DevOps provides two configurable features out-of-the-box:

  [Expand table](#)

Feature	Manifest property	Purpose
Custom names	<code>isNameConfigurable: true</code>	Users can override the default widget name
Multiple sizes	Multiple <code>supportedSizes</code> entries	Users can resize widgets

### Enhanced manifest example

JSON

```
{  
  "contributions": [  
    {  
      "id": "HelloWorldWidget3",  
      "type": "ms.vss-dashboards-web.widget",  
      "targets": [  
        "ms.vss-dashboards-web.widget-catalog",  
        "fabrikam.azuredevops-extensions-  
myExtensions.HelloWorldWidget.Configuration"  
      ],  
      "properties": {  
        "name": "Hello World Widget 3 (with config)",  
        "description": "My third widget",  
        "previewImageUrl": "img/preview3.png",  
        "uri": "hello-world3.html",  
        "isNameConfigurable": true,  
        "supportedSizes": [  
          {  
            "rowSpan": 1,  
            "columnSpan": 2  
          },  
          {  
            "rowSpan": 2,  
            "columnSpan": 2  
          }  
        ],  
        "supportedScopes": ["project_team"]  
      }  
    }  
  ]  
}
```

## Display configured names

To show custom widget names, update your widget to use `widgetSettings.name`:

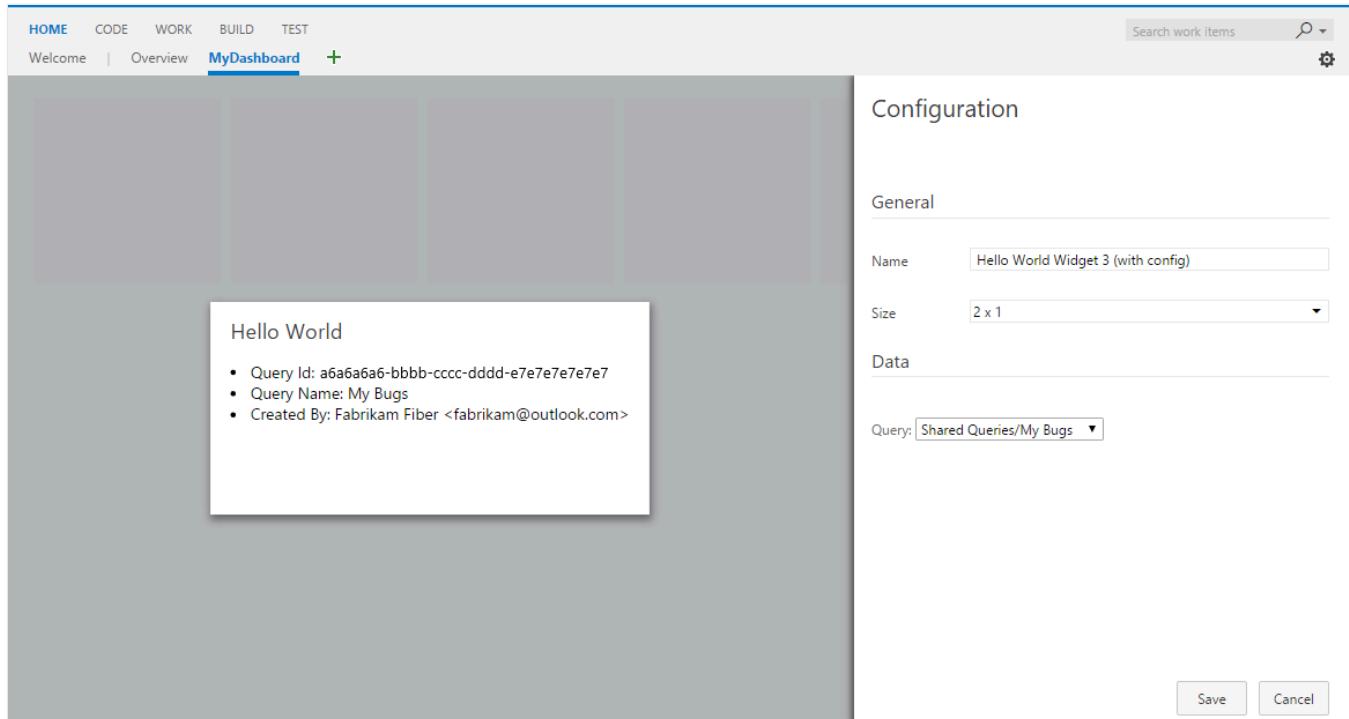
JavaScript

```
return {  
  load: function (widgetSettings) {  
    // Display configured name instead of hard-coded text  
    var $title = $('#h2.title');  
    $title.text(widgetSettings.name);  
  
    return getQueryInfo(widgetSettings);  
  },  
  reload: function (widgetSettings) {  
    // Update name during configuration changes  
    var $title = $('#h2.title');
```

```
$title.text(widgetSettings.name);

    return getQueryInfo(widgetSettings);
}
}
```

After you update your extension, you can configure both the widget name and size:



[Repackage](#) and update your extension to enable these advanced configuration options.

**Congratulations!** You created a complete, configurable Azure DevOps dashboard widget with live preview capabilities and user customization options.

ⓘ **Note:** The author created this article with assistance from AI. [Learn more](#)

# Call a REST API

Article • 05/30/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

To call a REST API from your extension, get the client service first, and then use that to call the API.

## ⓘ Note

Looking for Azure DevOps REST APIs? See the latest [Azure DevOps REST API reference](#).

For information about .NET client libraries, see [.NET client libraries for Azure DevOps](#).

## Client services

You can find the [full list of available REST clients and their references here](#).

## Call an API

You'll have to add the scope `"scopes": ["vso.work"]`, to your `vss-extension.json` to use the work item tracking client.

1. Get the REST client. In this case, we're getting the [work item tracking client](#).

JavaScript

```
VSS.require(["VSS/Service", "TFS/WorkItemTracking/RestClient"],  
  function (VSS_Service, TFS_Wit_WebApi) {  
    // Get the REST client  
    var witClient =  
      VSS_Service.getCollectionClient(TFS_Wit_WebApi.WorkItemTrackingHttpCli-  
        nt);  
    // ...  
  });
```

2. Call the API, `getWorkItems`, using the client service, `witClient`, with a callback that handles results.

## JavaScript

```
witClient.getWorkItems(/* some work item IDs */ [1,2,3,4],  
  ["System.Title"]).then(  
    function(workItems) {  
      console.log(JSON.stringify(workItems));  
    });
```

## Next steps

Display the results of your REST API call [in a Grid](#).

# Public project support by Azure DevOps Services extensions

Article • 10/04/2022

## Azure DevOps Services

Before public project support, all Azure DevOps Services projects were private. Private projects only authenticated users with access to the project, so the public couldn't see or interact with it. A public project allows non-member users to view the contents of that project in a read-only state.

A non-member user is either **anonymous** (not authenticated to Azure DevOps Services) or **public** (are authenticated to Azure DevOps Services, but do not belong to the organization).

Non-member users generally see the same views as authenticated users, however non-public functionality is hidden or disabled. An example of non-public functionality, such as settings and actions, includes a queue build.

### ⓘ Note

Azure DevOps Services public project support is currently in **Limited Preview**. Contact [vsts-public@microsoft.com](mailto:vsts-public@microsoft.com) if you are interested in developing extensions that support public projects. To learn more about public projects, see [Azure DevOps Services Public Projects Limited Preview](#).

## Decide whether to make an extension visible to non-member users

As the extension developer, you can make all or part of your extension available to non-member users. These users can only use your extension from within a public project. If you choose to not make your extension available to non-member users, no change is required and there's no impact on members, whether they use your extension from within a public project.

Use this checklist to help decide if you should make your extension available to non-member users:

- ✓ Data presented by your extension is relevant to non-member users

- ✓ Your extension contributes capabilities at the project level
- ✓ Your extension contributes to areas of the product that are accessible by non-member users
- ✓ Your extension doesn't extend or rely on features that are unavailable to non-member users, for example the Data Service or certain Azure DevOps Services REST APIs. For more information, see the [Limitations](#) section of this article.

## Contribution visibility

By default, contributions are only visible to organization members. To give non-member users visibility to a contribution, set the `restrictedTo` attribute on that contribution. The value is a string array that lists which types of users should have visibility to the contribution. The possible values are:

- `member` an authenticated user that is a member of the organization
- `public` an authenticated user that is **not** a member of the organization
- `anonymous` an unauthenticated user

### Example: make a hub visible to anonymous, public, and member users

JSON

```
{
  "contributions": [
    {
      "id": "my-hub",
      "type": "ms.vss-web.hub",
      "targets": [
        "ms.vss-code-web.code-hub-group"
      ],
      "restrictedTo": [
        "member",
        "public",
        "anonymous"
      ],
      "properties": {
        ...
      }
    }
  ]
}
```

You can also set the default visibility for all contributions in your extension by setting the `restrictedTo` attribute at the root of your extension manifest. You can then override this

default on individual contributions.

## Example: make every contribution, except for one, visible to all users

JSON

```
{  
  "id": "my-extension",  
  "name": "My Extension",  
  "version": "1.0.0",  
  ...  
  "restrictedTo": [  
    "anonymous",  
    "public",  
    "member"  
,  
  "contributions": [  
    {  
      "id": "my-member-only-widget",  
      "type": "ms.vss-dashboards-web.widget",  
      "restrictedTo": [  
        "member"  
,  
      "properties": {  
        ...  
      },  
      {  
        "id": "my-hub",  
        "type": "ms.vss-web.hub",  
        "targets": [  
          "ms.vss-code-web.code-hub-group"  
,  
        "properties": {  
          ...  
        },  
        {  
          "id": "my-second-hub",  
          "type": "ms.vss-web.hub",  
          "targets": [  
            "ms.vss-work-web.work-hub-group"  
,  
          "properties": {  
            ...  
          },  
        }  
      ]  
    }  
  ]  
}
```

# Limitations

If you want to make some or all aspects of your contribution available to public users, be aware of the following limitations.

## VSS SDK methods

The core SDK script, VSS.SDK.js, allows web extensions to communicate with the parent frame to do operations like initializing communication and getting context information about the current user. The following VSS SDK methods aren't supported for non-member users:

- `VSS.getAccessToken()`
- `VSS.getAppToken()`

## Extension data service

Because data managed by the [extension data service](#) isn't scoped or secured to a project, non-member users can't read or write any type of extension data.

### Example: handling data access errors

If the data service can't access the data service because of permission limitations by the calling user, the promise returned from the call to `getValue` is rejected. The error passed to the reject function has a name, which can help you understand why the call failed to read or write data.

JavaScript

```
VSS.getService(VSS.ServiceIds.ExtensionData).then(function(dataService) {
    dataService.getValue("someKey").then(function(value) {
        // do something with the value
    }, function(error) {
        if (error.name === "AccessCheckException") {
            alert(error.message);
        }
    });
});
```

## REST APIs

A limited set of Azure DevOps Services REST APIs is available to non-member users. These APIs include most organization-level and project-level APIs for features unavailable to non-member users in general. Consider this information when you're deciding whether to make your extension available to non-member users.

We recommend that you use version 5.0 and later APIs, as certain APIs are only available to non-member users starting with version 5.0.

## Identity references

A majority of Azure DevOps Services REST APIs use a common "contract" to represent a user or group. To protect member information, like email addresses, when a REST API is invoked by an anonymous or public user, certain fields, like `uniqueName` are omitted.

## Checking user permissions

Use permissions to decide whether to surface or enable a capability in your extension. The Security REST API gets used from web extension code to check whether the current user has permission in Azure DevOps Services to complete the task. This way, the user won't think they have permission to do something, only to find that they don't.

### Example: check whether the user has permission to queue builds

This example shows how to use the Security REST client to check whether the user has permissions to queue builds in the current project. By default, non-member users don't have this permission.

JavaScript

```
VSS.require(["VSS/Service", "VSS/security/RestClient"],
  function(VSS_Service, Security_RestClient) {
    var client =
      VSS_Service.getCollectionClient(Security_RestClient.SecurityHttpClient3);

    var securityToken = VSS.getWebContext().project.id;

    client.hasPermissionsBatch({
      evaluations: [
        {
          "securityNamespaceId": "33344D9C-FC72-4d6f-ABA5-FA317101A7E9",
          "token": securityToken,
          "permissions": 128 /* queue builds */
        }
      ],
      alwaysAllowAdministrators: true
    });
  }
);
```

```
}

).then(function(response) {
    console.log("Can user queue builds in this project? " +
response.evaluations[0].value);
})
});
```

## Dashboard widget considerations

Just like other types of contributions, the visibility of dashboard widget contributions is controlled with the `restrictedTo` contribution property. For example to make a widget visible to both non-member and member users:

JSON

```
{
  "contributions": [
    {
      "id": "HelloWorldWidget",
      "type": "ms.vss-dashboards-web.widget",
      "targets": [
        "ms.vss-dashboards-web.widget-catalog"
      ],
      "restrictedTo": [
        "member",
        "public",
        "anonymous"
      ],
      "properties": {
        ...
      }
    }
  ]
}
```

## Widget settings

While controlling visibility of the widget to non-member users, the dashboard framework also provides an optional, open-form storage mechanism for widget settings. Two mechanisms are available for indicate whether widget settings are available for use by non-member users in public projects.

A widget with configurable settings that is visible to non-member users **must** follow one of the following patterns. Not doing so blocks the widget from surfacing to these users.

## Pattern 1: widget declares its instances only store project-specific settings

Set the widget contribution's `canStoreCrossProjectSettings` property to `false`, indicating the widget settings are project-specific.

JSON

```
{  
  "id": "HelloWorldWidget",  
  "type": "ms.vss-dashboards-web.widget",  
  ...  
  "properties": {  
    "canStoreCrossProjectSettings": false  
  }  
}
```

## Pattern 2: a widget instance declares its settings are project-specific

Individual widget instances can also indicate that its settings are project-specific and available to non-member users. When saving the settings, the widget should set the `hasCrossProjectSettings` to `false` in the stringified JSON string:

JSON

```
{  
  "hasCrossProjectSettings": false,  
  "hypotheticalWidgetOption": true,  
  "backlogLevel": "Stories"  
}
```

## Build and release considerations

If your extension contributes a build or release task, no change is required to use that task from a pipeline in a public project.

## Work item tracking considerations

Extensions don't work for non-member users in the context of a public project without changes. This includes the work item form, other work item experiences, or interaction with work item tracking REST APIs.

## Work item form

All work item updates or deletes fail for non-member users.

## Identities

In Azure DevOps Services REST API version 5.0 and later, identities are returned as `IdentityRef` objects instead of strings. As described previously, certain fields, like `uniqueName` aren't returned in these objects if the API call was made by a non-member user.

## APIs

Only project-scoped REST APIs can be invoked by an extension when the current user isn't an organization member. Any REST API calls not scoped to a project are rejected.

## Queries

There are the following limitations for non-member users related to work item queries:

- non-member users can run known queries by ID or path
- Queries must be scoped to the current project. Any work items not belonging to the current project are excluded
- non-member user can't create new queries or execute WIQL queries

# Package and publish extensions

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Once you [develop your extension](#), you can package and publish it to the [Visual Studio Marketplace](#). The Marketplace is a global repository for private and public extensions, integrations, and other offers from Microsoft.

## (!) Note

For information on the discovery properties available in your extension's manifest file that helps users discover and learn about your extension, see the [Extension Manifest Reference](#).

## Prerequisites

The following list of requirements must be met before you publish to the Marketplace.

[+] [Expand table](#)

Category	Requirements
Packaging tool	Install the extension packaging tool (TFX). Run <code>npm install -g tfx-cli</code> from a command prompt.
Image permissions	Ensure you have proper permissions to use any images, like icons, logos, screenshots, and so on.
Marketplace overview	Include a thorough <code>overview.md</code> file to describe your listing in the Marketplace.
Extension icon	Include an icon for your extension that represents your integration, company, or organization, at least 128x128 pixels in size (PNG or JPEG).
Microsoft product names	Use full names for Microsoft products (for example, Azure DevOps instead of AzDO or other abbreviations).
Brand names	Don't use brand names in the name of your extension.

## Create a publisher

Every extension or integration, including those from Microsoft, must have a publisher. Anyone can create a publisher and publish extensions under it. You can also share publisher access with other users, such as your development team.

1. Sign in to the [Visual Studio Marketplace Publishing Portal](#).
2. If you aren't part of an existing publisher, select **+ Create a publisher**.  
Enter a publisher name; the ID field autofills based on your entry.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

 **Note**

- Ensure your publisher name is within 16 characters for multibyte characters.
- Save the publisher ID—you need it in your extension's manifest file.

If you aren't prompted to create a publisher, scroll to **Publish extensions** under *Related sites*.

- Set a unique publisher identifier, such as `mycompany-myteam`. Use this value for the `publisher` attribute in your manifest.
- Set a display name, such as `My Team`.

3. Review the [Marketplace Publisher Agreement](#), then select **Create**.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

After you create the publisher, you can manage items, although no items appear until you publish.

## Package your extension

To upload your extension, package it as a VSIX 2.0-compatible .vsix file. Microsoft provides a cross-platform command-line interface (CLI) to package and publish your extension.

1. Open your extension manifest file (`vss-extension.json`) and set the value of the `publisher` field to the ID of your publisher. For example:

JSON

```
{  
  ...  
  "id": "my-first-extension",  
  "publisher": "AnnetteNielsen",  
  ...  
}
```

2. From a command prompt, run the TFX tool's packaging command from your extension directory.

```
npx tfx-cli extension create
```

A message displays indicating your extension is successfully packaged:

```
==== Completed operation: create extension ===  
- VSIX: C:\my-first-extension\AnnetteNielsen.my-first-extension-1.0.0.vsix  
- Extension ID: my-first-extension  
- Extension Version: 1.0.0  
- Publisher: AnnetteNielsen
```

### ! Note

Increment the version of your extension or integration in the manifest with every update. Use the `--rev-version` command line switch. This switch increments the *patch* version number of your extension and saves the new version to your manifest.

## Check package size

Check the size of the vsix after it gets packaged. If it's greater than 50 MB, you need to optimize it. To do so, see the following considerations:

- Deduplicate common dependencies by stating them once in the extension package.
- Fetch dependencies at runtime or during install time rather than including them in the package. Consider using the tool installer library to pull tool dependencies at runtime. This approach caches the tool by version for private agents, preventing downloads for every build. The tool installer library doesn't work in disconnected scenarios (no internet), which should be mentioned in the task description or documentation.
- Use WebPack to tree shake dependencies in tasks.

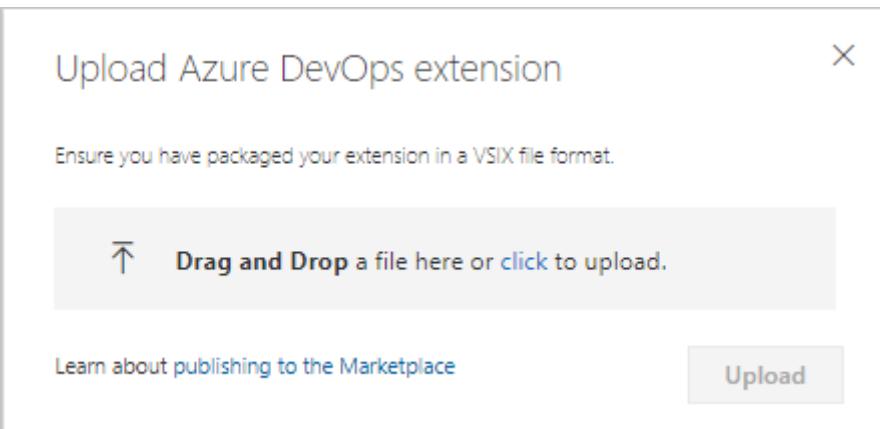
## Publish your extension

Once your extension is packaged, you can upload it to the Marketplace under a publisher. The **publisher** identifier specified in your [extension's manifest file](#) must match the identifier of the publisher the extension is uploaded under.

1. From the [management portal](#), select your publisher from the drop-down menu at the top of the page.
2. Select **New extension > Azure DevOps**.

Microsoft		Annette	+ New extension		
	Owner	Extensions	Details	Members	
	Owner	Name ↑	Visual Studio	Updated	
	Owner	Fabrikam Build Task	Visual Studio Code	2 years ago	
Microsoft Account	Owner		Azure DevOps	2 years ago	
	Owner	My first sample extension	0.1.0	2 years ago	
+ Create publisher					

3. Drag and drop your file or select it to find your VSIX file, which you created in the previous packaging step, and then choose **Upload**.



After quick validation, your extension appears in the list of published extensions. Don't worry, the extension is only visible to you.

Name ↑	Version	Updated	Availability
My First Extension	1.0.0	just now	Private (not shared)

At this point, your extension isn't visible to any accounts. To make it visible to others, you need to share the extension.

#### !**Note**

Microsoft runs a virus scan on each new and updated extension package published. Until the scan is all clear, we don't publish the extension in the Marketplace for public usage. This way we also avoid surfacing inappropriate or offensive content on the Marketplace pages.

## Share your extension

Share your extension with an organization before you can install it in Azure DevOps. To share an extension, do the following tasks:

1. From the [Marketplace management portal](#), select your extension from the list, right-click, and then choose **Share/Unshare** or **Publish/Unpublish**, depending on the extension.

The screenshot shows the 'Extensions' tab selected in the Visual Studio Marketplace. A list of extensions is displayed, with 'My First Extension' at the top. The extension details are shown: Name (My First Extension), Version (1.0.0), and Updated (just now). A context menu is open over the extension, listing options: Reports, View Extension, Update, Remove, Share/Unshare, and Certificate.

2. Select **Organization**, and then enter the name of your organization. Select **Enter**.

The screenshot shows the 'Shared with' panel for 'My First Extension'. It displays a list of accounts shared with the extension. One account, 'annette' from '.visualstudio.com', is listed. The 'Account' button next to the account name has a blue border, indicating it is selected or active.

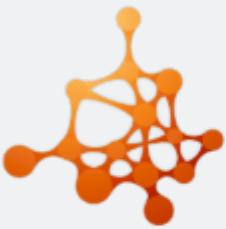
3. Close the panel.

Your extension can now be installed into this organization.

## Install your extension

To install your shared extension, do the following steps.

1. In the Marketplace, select your extension to open its overview page.



## My First Extension

| ★★★★★ (0) | Free

[Get it free](#)[Overview](#)[Q & A](#)[Rating & Review](#)

Sample Azure DevOps extension

**! Note**

Since your extension is private, only you and members of the organization it's shared with can see this page.

2. Select **Get it free** to start the installation process. Select the organization you shared the extension with from the dropdown menu.

The screenshot shows the 'My First Extension' page in the Visual Studio Marketplace. At the top, there's a note: 'Since your extension is private, only you and members of the organization it's shared with can see this page.' Below this, the main content area has a blue sidebar on the left with the extension name 'My First Extension'. The main panel shows the 'Get it free' button, followed by a 'Done' step with an account icon and a checkmark. Below that, it says 'Select a Visual Studio Team Services account' with a dropdown menu showing 'annette'. There's a large 'Install' button. At the bottom, there are links for 'For Team Foundation Server' and 'Download'.

3. Select **Install**.

Congratulations! You installed your extension into an organization and you're ready to try it.

# Try your extension

1. Select **Proceed to organization** at the end of the installation wizard to go to the home page of the organization the extension was installed to (<https://dev.azure.com/{organization}>).
2. Refresh your browser.
3. Open **Organization settings**, and then select **Extensions**.

The screenshot shows the Azure DevOps Organization Settings page for the organization 'fabrikamprime'. The left sidebar includes sections for General (Overview, Projects, Users, Billing, Auditing, Global notifications, Usage, Extensions, Microsoft Entra), and a search bar. The 'Extensions' link in the sidebar is highlighted with a red box. The main content area is titled 'Extensions' and shows three tabs: 'Installed' (selected), 'Requested', and 'Shared'. Under the 'Installed' tab, there are six listed extensions:

Extension	Description
SARIF SAST Scans Tab by Microsoft	Adds a 'Scans' tab to each Build Pipeline
1ES GPT by Microsoft	1ES PT helper extension for running
Accessibility Insights for Azure DevOps	Scan accessibility issues in an Azure
Advanced Security Build Tasks by Microsoft	A set of build tasks for Advanced
Azure DevTest Labs Tasks by Microsoft	Collection of Azure Pipelines task
Build Security Monitoring by 1ES	Enables Build Security Monitoring

You should see the new extension on the **Installed** tab.

## Debug your extension

To debug the extension using Visual Studio or Browser Developer Tools, change the manifest by adding the `baseUri` property. This action speeds up the development without the need to redeploy the extension each time you change source code.

JSON

```
{  
  ...  
  "baseUri": "https://localhost:44300",  
  ...  
}
```

When you change the manifest, it loads the extension from your local web server instance. For example, IISExpress in Visual Studio. After you change the manifest, deploy and install this debugging extension only once.

 **Note**

Run your local web server in SSL mode because Azure DevOps demands that the web page is served from a secure source. Otherwise, you get an error in the browser console during the extension IFRAAME loading.

## Update your extension

To update an extension you already published, do the following steps:

 **Tip**

Update your extension instead of removing and re-uploading it. We recommend maintaining two extensions: `publisher.extension`, public in the Marketplace for customers, and `publisher.extension-dev`, private, shared only with your organization for development and testing. You don't need two copies of your source code—just maintain separate manifest files for each extension. When packaging, provide the appropriate manifest file to the tfx-cli tool. For more information, see [TFX extension commands](#).

1. Select your extension from the list of displayed items.
2. Right-click and select **Update** for the development version, such as `publisher.extension-dev`.
3. Validate your extension.
4. Apply the same updates to the production version, such as `publisher.extension`.

5. Browse to the .vsix file for your extension and upload it.

Azure DevOps automatically installs the updated version for all accounts that already have the extension. New installations also receive the latest version.

## Make your extension public

While you develop your extension or integration for the Marketplace, keep it private. This limits the visibility of the extension to specific accounts that you have shared it with.

To make your extension available publicly, set the [public flag](#) to `true` in your manifest.

## Qualifications

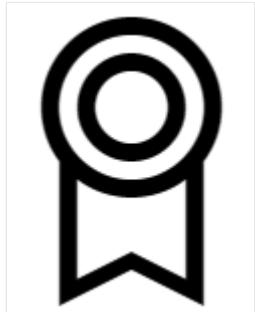
To have a public listing on the Marketplace, your integration or extension must meet the following qualifications:

- Works with or extends Azure DevOps.
- You, or your company, own, develop, and are licensed to distribute and advertise the integration or extension.
- The extension or integration is actively maintained.

Microsoft might also request a demo and to review the content planned for your Marketplace entry.

## Top Publisher

The Top Publisher program is only available for publishers with Azure DevOps extensions or integrations. It's not applicable for Visual Studio IDE and Visual Studio Code extension publishers.



The Top Publisher program recognizes publishers with commitment to their customers and the Marketplace through exemplary policies, quality, reliability, and support. Once you become a Top Publisher, all of your public offerings display the Top Publisher badge.

# Top Publisher requirements

The Top Publisher program in the Marketplace is designed to help you evaluate or acquire Azure DevOps extensions and integrations with confidence. The Top Publisher badge implies that the publisher shows commitment to their customers and the Marketplace through exemplary policies, quality, reliability, and support. It's for publishers with one or more global Azure DevOps extensions or integrations and isn't applicable for Visual Studio IDE and Visual Studio Code extension publishers.

Marketplace assigns the badge to a publisher after carefully reviewing the publisher across the following parameters:

- Privacy policy
- Licensing policy
- Support policy
- Documentation
- Q & A responsiveness
- Ratings and reviews for their offerings
- Active uptake and install count for their offerings
- Management of at least one public extension for Azure DevOps
- Public extension with more than 5,000 installs and an active install count exceeding 1000

You can expect timely support and a good overall experience when you get an extension from a Top Publisher. Check out the offerings from the Top Publishers.

For more information on adding policies to your offering, see the [extension manifest](#).

## 1. Update your publisher profile.

Through the publisher profile, you can showcase all of your offerings in one place along with key publisher-related information. To provide the information, which shows up in the profile, do the following steps:

- a. Sign in to <https://marketplace.visualstudio.com/manage/publishers> using the account with which you publish and manage your offerings in the Visual Studio Marketplace.

b. Select the publisher and complete the **About you** section in the **Details** tab.

**About you** ^

We will use this information to populate your publisher profile page

Description

Microsoft is an multinational technology company with vision to empower every person and every organization on the planet to achieve more.

Logo ⓘ

The Microsoft logo consists of four colored squares arranged in a 2x2 grid: top-left is red, top-right is green, bottom-left is blue, and bottom-right is yellow.

Company website

<https://www.microsoft.com>

Support

<https://support.microsoft.com>

LinkedIn

<https://www.linkedin.com/company/microsoft>

Source code repository

<https://github.com/microsoft>

Twitter

<https://twitter.com/microsoft>

c. Save your changes and select **View profile** to see how it appears to consumers. You can use this profile page to evangelize your offerings.

**!** Note

This program certifies the publisher, not the software, or security of their extensions and integrations. We recommend reviewing the [safety information](#) when evaluating offerings from a publisher. If you got an extension from a Top Publisher and aren't satisfied with your experience, consider engaging with the publisher first.

## Apply to be a Top Publisher

1. Sign in to [Marketplace management portal](#) using the account with which you publish and manage your offerings in Marketplace.
2. Select the publisher and go to its **Top Publisher** tab. Have one or more global Azure DevOps (Server/Service) extensions or integrations for the tab to appear.
3. If you meet the previously listed requirements and are the publisher's owner, you can apply for the program. Upon application, an email gets sent to the Marketplace team to

review your case. They respond within 10 business days with next steps, clarifying questions, or the grant of the badge.

The team likely considers other parameters, such as active uptake of your offerings, install/get started counts, and ratings & reviews across your offerings before granting the badge. Microsoft reserves the right to grant, reject, or revoke the Top Publisher badge at any time.

Once a publisher is a Top Publisher, all its future updates and offerings must meet the previously listed requirements.

## Respond to Marketplace extension reviews

You can respond to reviews that customers leave for your extensions in the Visual Studio Marketplace. Find and select **Reply** next to a review if you have one of the following permissions: owner, creator, or contributor.

You can leave only one response. Avoid using reviews as a support forum. If you need more details, please provide a support alias for the reviewer to contact. You can then resolve their problems externally and update your reply with a resolution.

## Guidelines for publisher responses

Keep the Visual Studio Marketplace an open, inviting, respectful, and helpful place for customers to find, try, install, and review extensions. Communication plays an important role in keeping a healthy community. To help create this environment, here are guidelines for publishers responding to customer reviews. Think deeply about your customer interactions and reflect on the spirit of the customer experience that the Marketplace is trying to create.

- Reserve reviews for customer comments. Use *Reply* only to respond to a review.
- Respect all customer opinions. Treat comments as feedback without debate, criticism, or argument.
- Ensure your responses add value and are relevant to your customers' comments.
- Focus on precisely addressing questions or problems. If you need more details, ask the customer to contact you over email rather than discussing in reviews. When you resolve the problem, update your reply with the resolution. You can edit your reply like customers can edit their reviews.
- Flag any inappropriate reviews, such as spam, abusive, or offensive content, for our review.

## Request to void a review

As a publisher, you can appeal to void a review if the issue reported is because of the Marketplace or underlying platform. If the issue is valid, Marketplace admins void the rating. You can **Appeal** from ratings and review section on your extension hub page.

## Unpublish an extension

You can unpublish free extensions if you no longer want to offer them in the Marketplace.

Consider removing your extension from the Marketplace in the following scenarios:

- You developed a new extension and no longer want to offer the current one.
- Your extension has a problem, and you want to remove it from the Marketplace until you resolve the issue.
- You published your extension as public by mistake.

Certain criteria must be met for an extension to be unpublished or removed:

[+] Expand table

Action	Requirements
Unpublish	Only <b>free extensions</b> might be unpublished.
Remove	Your extension must have <b>zero (0) installs</b> to be removed.

### (i) Important

If you must remove your extension because of legal or security problems, contact [Customer Support at the Developer Community](#). We review the request and manually delete the extension.

1. Select the extension on your [publisher page](#) and choose **Unpublish** on the menu.

Your extension is unpublished immediately from the Marketplace, and new users can't install it. Ratings and reviews for your extension stay intact.

To offer your extension again in the Marketplace, select **Publish** from the menu.

If your extension has zero installs, you can choose to remove it completely from the Marketplace. To do so, select **Remove** from the menu. You can't reverse this action.

## Extension reporting hub

Once your extension is available in the Visual Studio Marketplace, you can use the **Reports** feature. With this feature, you can track and analyze how the extension is performing and take required actions. To visit the extension hub, browse to your [publisher page](#) and select the extension or select the **Reports** link on the extension details page.

## Acquisition

You can view acquisition-related data in this tab for the selected period.

- Aggregated acquisition in the selected period for overall acquisition
- Aggregated acquisition split by extension downloads Azure DevOps connected install for free extension
- Aggregated acquisition split by trials Azure DevOps connected buy for paid extension
- Daily trend of extension page views with acquisition for Azure DevOps and connected server
- Conversion percentage from page views to acquisition

For paid extensions, all transactional details for buy and trials are available with date, organization name, trial end date, and quantity. You can use the **Contact** action to communicate with your users. For more information, see the [Contact](#) section provided later in this article.

## Uninstall

You can view the following statistics:

- How many organizations uninstalled your extension
- Daily trend of uninstall extensions
- Detailed feedback shared during uninstalls
- Top uninstall reasons

You can use search for text and dates to analyze and draw more insights from the detailed feedback.

For paid extensions, you can use the **Contact** action to communicate with your users. [Contact](#) section provided later in this article for more details.

## Ratings and review

This tab gives you the following information:

- Average rating for the selected period versus overall rating
- Average rating by number of reviewers

- Daily trend of average rating

The details section provides all the reviews and your responses in transactional view.

You can **Reply** to a review or **Edit** a previous response and better manage engagement with your extension users. You can also **Appeal** to void a rating if the issue reported is because of the Marketplace or underlying platform. If the issue is valid, we void the rating.

## Manage engagement

The Q & A tab provides a snapshot of all questions from your extension users, with nonresponded queries at the top. You can reply to or edit previous responses to better manage engagement with your extension users.

## Export to Excel

All data elements available in the reports page are also available for download in XLS format to aid creating your own custom reports.

## Contact

For paid extensions, you can use the **Contact** action to communicate with your users. This feature is available only for publishers with Contributor+ access on the extension.

Marketplace brokers the first communication with the user as our privacy policy doesn't allow direct sharing of customer email addresses. Only users who opted in for communication receive the email. The last contacted date for an organization is updated after sending a communication.

### Important

Follow the guidance on transactional and promotional communication. Publishers found to be sending promotional communication or spamming users get added to a blocklist and lose access to the **Contact** feature for all their extensions.

**Transactional communication:** Emails conveying critical information necessary for the continued use of the extension or service, such as:

- Critical security notices
- Transaction confirmations
- Product recall notices
- Specific feedback requests

- Service discontinuation notices

**Promotional emails:** Emails used to market your extension, product, service, website, or event, such as:

- Invitations to events or webcasts
- Information about new marketing or partner programs
- Offers to obtain value-added content
- Newsletters containing promotional content

For more information, see the [Marketplace Publisher Agreement](#).

 [Expand table](#)

Terminology	Description
Page views	Total number of extension detail page views. Repeated views are counted.
Azure DevOps Services installs	Total number of organizations the extension is installed in. Repeated installs on the same organization get counted.
Azure DevOps Server installs	Total number of collections the extension is installed in. Repeated installs on the same collection get counted. Disconnected server data isn't available.

## Related content

- [Develop a web extension](#)
- [Explore extensibility points](#)

 **Note:** The author created this article with assistance from AI. [Learn more](#)

# Package and publish an integration to the Marketplace

06/06/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

This article explains how to publish your tool, service, or product that integrates with Azure DevOps on the Visual Studio Marketplace. Publishing on the Marketplace helps users discover solutions that extend and enhance their Azure DevOps experience. The Marketplace serves as the central hub for individuals and teams to find integrations and extensions.

Browse the Marketplace [↗](#) to see examples of other integrations and extensions.

## ! Note

For packaging and publishing information for extensions, see [Package & Publish Extensions](#).

## Prerequisites

The following list of requirements must be met before you publish to the Marketplace.

[\[+\] Expand table](#)

Category	Requirements
Packaging tool	Install the extension packaging tool (TFX). Run <code>npm install -g tfx-cli</code> from a command prompt.
Image permissions	Ensure you have proper permissions to use any images, like icons, logos, screenshots, and so on.
Marketplace overview	Include a thorough <code>overview.md</code> file to describe your listing in the Marketplace.
Extension icon	Include an icon for your extension that represents your integration, company, or organization, at least 128x128 pixels in size (PNG or JPEG).
Microsoft product names	Use full names for Microsoft products (for example, Azure DevOps instead of AzDO or other abbreviations).
Brand names	Don't use brand names in the name of your extension.

## Gather required assets

- At least one screenshot of your integration.
- Call-to-action or get started URL for users.

### ⓘ Note

- The term **extension** is used in referenced documentation. Extensions are another type of Marketplace item and share many similarities with integrations.
- Need help getting your integration on the Marketplace? [Contact us ↗](#).

## Create a publisher account

Every extension or integration, including those from Microsoft, must have a publisher. Anyone can create a publisher and publish extensions under it. You can also share publisher access with other users, such as your development team.

1. Sign in to the [Visual Studio Marketplace Publishing Portal ↗](#).
2. If you aren't part of an existing publisher, select **+ Create a publisher**.  
Enter a publisher name; the ID field autofills based on your entry.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

 **Note**

- Ensure your publisher name is within 16 characters for multibyte characters.
- Save the publisher ID—you need it in your extension's manifest file.

If you aren't prompted to create a publisher, scroll to **Publish extensions** under *Related sites*.

- Set a unique publisher identifier, such as `mycompany-myteam`. Use this value for the `publisher` attribute in your manifest.
- Set a display name, such as `My Team`.

3. Review the [Marketplace Publisher Agreement](#), then select **Create**.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

After you create the publisher, you can manage items, although no items appear until you publish.

## Organize your manifest and assets

To organize your manifest and assets, do the following steps:

1. Create a `home` folder to store required assets.
2. Create an `images` folder for:
  - Your integration logo (128x128 pixels)
  - Screenshots (1366x768 pixels)
3. Create an `overview.md` file to describe your integration. For more information, see [GitHub Flavored Markdown ↗](#).
4. Create a `vss-integration.json` file, which is your Marketplace listing's manifest file. For more information, see the [extension manifest reference](#).

## Complete the extension manifest

Publishing to the Marketplace starts with creating a manifest file that defines your integration and its key discovery details (screenshots, logos, overview content). This information is used to present your integration to users on the Marketplace.

1. Fill your `vss-integration.json` file with the following JSON:

```
JavaScript

{
  "manifestVersion": 1,
  "id": "myservice",
  "version": "1.0.0",
  "name": "My Service",
  "publisher": "mycompany",
  "description": "Awesome tools to help you and your team do great things everyday.",
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services.Integration"
    }
  ],
  "icons": {
    "default": "images/service-logo.png"
  },
  "categories": [
    "Plan and track"
}
```

```

],
"tags": [
    "working",
    "people person",
    "search"
],
"screenshots": [
    {
        "path": "images/screen1.png"
    },
    {
        "path": "images/screen2.png"
    }
],
"content": {
    "details": {
        "path": "overview.md"
    },
    "license": {
        "path": "fabrikam-license-terms.md"
    }
},
"links": {
    "getstarted": {
        "uri": "https://www.mycompany.com/help/getstarted"
    },
    "learn": {
        "uri": "https://www.mycompany.com/features"
    },
    "support": {
        "uri": "https://www.mycompany.com/support"
    }
},
"branding": {
    "color": "rgb(34, 34, 34)",
    "theme": "dark"
}
}

```

## 2. Update the JSON using the following references:

The following properties are required:

[Expand table](#)

Property	Description	Notes
manifestVersion	A <i>number</i> corresponding to the version of the manifest format.	Should be 1.

Property	Description	Notes
ID	<i>The extension's identifier.</i>	The ID is a string that must be unique among extensions from the same publisher. It must start with an alphabetic or numeric character and contain 'A' through 'Z', 'a' through 'z', '0' through '9', and '-' (hyphen). Example: <code>sample-extension</code> .
version	<i>A string specifying the version of an extension.</i>	Should be in the format <code>major.minor.patch</code> , for example <code>0.1.2</code> or <code>1.0.0</code> . You can also add a fourth number for the following format: <code>0.1.2.3</code>
name	<i>A short, human-readable name of the extension. Limited to 200 characters.</i>	Example: <code>"Fabrikam Agile Board Extension"</code> .
publisher	<i>The identifier of the publisher.</i>	This identifier must match the identifier the extension is published under. See <a href="#">Create and manage a publisher</a> .
categories	<i>Array of strings representing the categories your extension belongs to. At least one category must be provided and there's no limit to how many categories you may include.</i>	<p>Valid values: <code>Azure Repos</code>, <code>Azure Boards</code>, <code>Azure Pipelines</code>, <code>Azure Test Plans</code>, and <code>Azure Artifacts</code>.</p> <p>Notes:</p> <ul style="list-style-type: none"> <li>- Use version <math>\geq 0.6.3</math> of the tfx-cli if you're publishing the extension programmatically.</li> <li>- If you're using <a href="#">Azure DevOps Extension Tasks extension</a> to publish, ensure that its version is <math>\geq 1.2.8</math>. You might have to approve the extension update because of recent scope changes.</li> <li>- The categories previously mentioned are natively present in Visual Studio Marketplace and Azure DevOps Server 2019 &amp; above.</li> </ul>
targets	<i>The products and services supported by your integration or extension. For more information, see <a href="#">installation targets</a>.</i>	<p>An array of objects, where each object has an <code>id</code> field indicating one of the following:</p> <ul style="list-style-type: none"> <li>- <code>Microsoft.VisualStudio.Services</code> (extensions that work with Azure DevOps),</li> <li>- <code>Microsoft.TeamFoundation.Server</code> (extension that works with Azure DevOps Server),</li> <li>- <code>Microsoft.VisualStudio.Services.Integration</code>,</li> <li>- <code>Microsoft.TeamFoundation.Server.Integration</code> (integrations that work with Azure DevOps Server)</li> </ul>

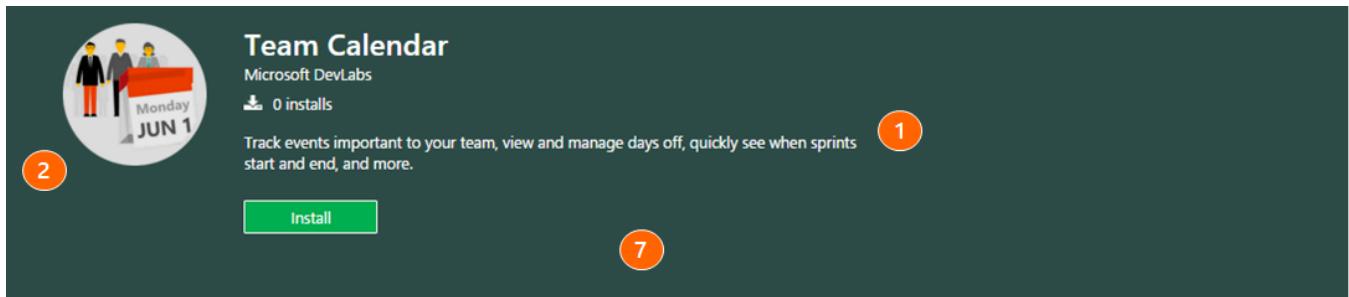
The following optional properties help users discover and learn about your extension:

Property	Description	Notes
<b>description</b>	<i>A few sentences describing the extension. Limited to 200 characters.</i>	The description should be your extension's "elevator pitch" - a couple of lines to describe your extension in the Marketplace and make people want to install it. See the example below
<b>icons</b>	<i>Dictionary of icons representing the extension.</i>	Valid keys: <code>default</code> (128x128 pixels) of type BMP, GIF, EXIF, JPG, PNG and TIFF). Other keys such as <code>large</code> (512x512 pixels) may be supported in the future. The value of each key is the path to the icon file in the extension
<b>tags</b>	<i>Array of string tags to help users find your extension.</i>	Examples: <code>agile</code> , <code>project management</code> , <code>task timer</code> , and so on.
<b>screenshots</b>	<i>Array of images that couldn't be included in your content.</i>	Screenshots are more valuable when featured in your <b>content</b> , and should be used there to help make a quality market details page for your extension. Use <b>screenshots</b> for less important images not featured in your <b>content</b> . Each image should be 1366x768 pixels. The <code>path</code> of each item is the path to the file in the extension.
<b>content</b>	<i>Dictionary of content files that describe your extension to users.</i>	<i>Every extension should include solid content. This is how you'll show users what your extension can do. Make it rich, consumable, and include screenshots where necessary. Include an <code>overview.md</code> file as your base content piece. Each file is assumed to be in GitHub Flavored Markdown ↗ format. The <code>path</code> of each item is the path to the Markdown file in the extension.</i> Valid keys: <code>details</code> . Other keys may be supported in the future.
<b>links</b>	<i>Dictionary of links that help users learn more about your extension, get support, and move.</i>	Valid keys: <code>getstarted</code> - first steps, how to setup or use. <code>learn</code> - deeper content to help users better understand your extension or service. <code>license</code> - end-user license agreement. <code>privacypolicy</code> - privacy policy for an extension. <code>support</code> - get help and support for an extension. The value of each key is an object with a <code>uri</code> field, which is the absolute URL of the link
<b>repository</b>	<i>Dictionary of properties describing the source code repository for the extension</i>	Valid Keys: <code>type</code> - Type of repository. Example: <code>git</code> . <code>uri</code> - Absolute URL of the repository.
<b>badges</b>	<i>Array of links to external metadata badges like TravisCI, Appveyor, and so on,</i>	Valid keys: <code>href</code> - Link the user navigates to when selecting the badge. <code>uri</code> - The absolute URL of the badge image to be displayed. <code>description</code> - Description of the badge, to be displayed on hover.

Property	Description	Notes
	<i>from the approved badges sites</i>	
branding	<i>Dictionary of brand-related properties.</i>	Valid keys: <code>color</code> - primary color of the extension or publisher; can be a hex (#ff00ff), RGB (rgb(100,200,50)), or supported HTML color names (blue). <code>theme</code> - complements the color; use <code>dark</code> for dark branding colors, or <code>light</code> for lighter branding colors.

## Understand the details page

- 1 - description
- 2 - icon
- 3 - categories
- 4 - screenshots
- 5 - content (details)
- 6 - links
- 7 - branding



### Stay on track

5

Team Calendar is an extension for Visual Studio Online that helps busy teams stay on track and informed about important deadlines, sprint schedules, and upcoming milestones. Team Calendar is the one place to see and manage the date important to your teams, including:

- Iterations
- Days off (for individuals or the team)
- Custom events (single or multi-day)

### ⚠ Warning

Set the `public` attribute to `false` or omit it to prevent your integration from becoming visible to all Marketplace users before you're ready.

# Package your manifest and assets

## Install the package tool (tfx-cli)

Install or update the cross-platform CLI for Azure DevOps (tfx-cli) using `npm`:

```
Bash
```

```
npm i -g tfx-cli
```

## Package your integration in a .vsix file

```
no-highlight
```

```
tfx extension create --manifest-globs vss-extension.json
```

 **Note**

Increment the version of your extension or integration with every update.

If you haven't updated the version in your manifest, use the `--rev-version` command line switch. This switch automatically increments the *patch* version number and saves the new version to your manifest.

## Publish your integration to the Marketplace

Once your extension is packaged, you can upload it to the Marketplace under a publisher. The `publisher` identifier specified in your [extension's manifest file](#) must match the identifier of the publisher the extension is uploaded under.

1. From the [management portal](#), select your publisher from the drop-down menu at the top of the page.
2. Select **New extension > Azure DevOps**.

Visual Studio | Marketplace

## Manage Publishers & Extensions

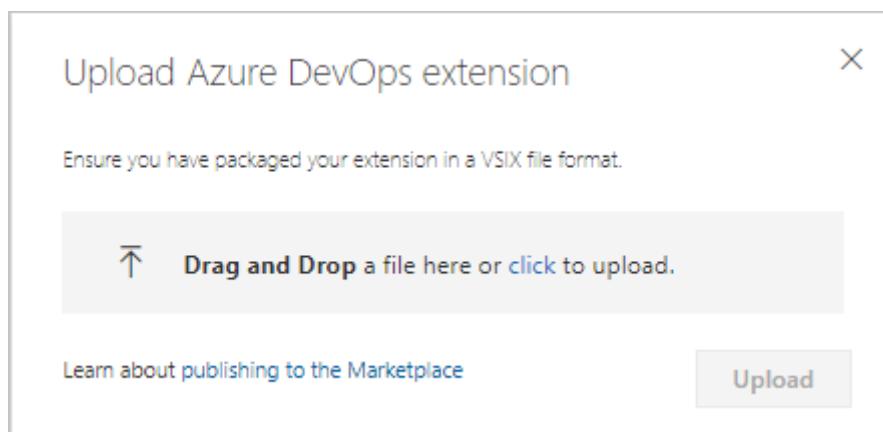
Microsoft	Owner
Microsoft Account	Owner
+ Create publisher	

Annette

Extensions Details Members + New extension ▾

Name ↑	Version	Updated
Visual Studio		Updated
Visual Studio Code		2 years ago
Azure DevOps		2 years ago
Fabrikam Build Task	0.1.0	2 years ago
My first sample extension		

3. Drag and drop your file or select it to find your VSIX file, which you created in the previous packaging step, and then choose **Upload**.



After quick validation, your extension appears in the list of published extensions. Don't worry, the extension is only visible to you.

Manage Publishers & Extensions

Annette	Owner
+ Create publisher	

Annette (Annette) ▲

Extensions Details Members + New extension ▾

Name ↑	Version	Updated	Availability
My First Extension	1.0.0	just now	Private (not shared)

At this point, your extension isn't visible to any accounts. To make it visible to others, you need to share the extension.

**! Note**

Microsoft runs a virus scan on each new and updated extension package published. Until the scan is all clear, we don't publish the extension in the Marketplace for public usage. This way we also avoid surfacing inappropriate or offensive content on the Marketplace pages.

## Share your integration

Before installing an integration in an Azure DevOps organization, share it with that organization. Sharing is required for development and testing, as it's the only way to run an integration during these stages.

To share an integration do the following steps:

1. Select an integration from the list of displayed items
2. Select the **Share** button
3. Specify the name of the organization to make this integration visible to. For example, to make an integration visible to the `dev.azure.com/fabrikam-fiber-inc` organization, specify `fabrikam-fiber-inc`.

## Update an item

To update an extension you already published, do the following steps:

### Tip

Update your extension instead of removing and re-uploading it. We recommend maintaining two extensions: `publisher.extension`, public in the Marketplace for customers, and `publisher.extension-dev`, private, shared only with your organization for development and testing. You don't need two copies of your source code—just maintain separate manifest files for each extension. When packaging, provide the appropriate manifest file to the tfx-cli tool. For more information, see [TFX extension commands](#).

1. Select your extension from the list of displayed items.
2. Right-click and select **Update** for the development version, such as `publisher.extension-dev`.
3. Validate your extension.
4. Apply the same updates to the production version, such as `publisher.extension`.
5. Browse to the .vsix file for your extension and upload it.

Azure DevOps automatically installs the updated version for all accounts that already have the extension. New installations also receive the latest version.

## Make your integration public

For information on making your integration visible to everyone, see [Make your listing public](#).

# Package and publish extensions

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Once you [develop your extension](#), you can package and publish it to the [Visual Studio Marketplace](#). The Marketplace is a global repository for private and public extensions, integrations, and other offers from Microsoft.

## (!) Note

For information on the discovery properties available in your extension's manifest file that helps users discover and learn about your extension, see the [Extension Manifest Reference](#).

## Prerequisites

The following list of requirements must be met before you publish to the Marketplace.

[+] [Expand table](#)

Category	Requirements
Packaging tool	Install the extension packaging tool (TFX). Run <code>npm install -g tfx-cli</code> from a command prompt.
Image permissions	Ensure you have proper permissions to use any images, like icons, logos, screenshots, and so on.
Marketplace overview	Include a thorough <code>overview.md</code> file to describe your listing in the Marketplace.
Extension icon	Include an icon for your extension that represents your integration, company, or organization, at least 128x128 pixels in size (PNG or JPEG).
Microsoft product names	Use full names for Microsoft products (for example, Azure DevOps instead of AzDO or other abbreviations).
Brand names	Don't use brand names in the name of your extension.

## Create a publisher

Every extension or integration, including those from Microsoft, must have a publisher. Anyone can create a publisher and publish extensions under it. You can also share publisher access with other users, such as your development team.

1. Sign in to the [Visual Studio Marketplace Publishing Portal](#).
2. If you aren't part of an existing publisher, select **+ Create a publisher**.  
Enter a publisher name; the ID field autofills based on your entry.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

 **Note**

- Ensure your publisher name is within 16 characters for multibyte characters.
- Save the publisher ID—you need it in your extension's manifest file.

If you aren't prompted to create a publisher, scroll to **Publish extensions** under *Related sites*.

- Set a unique publisher identifier, such as `mycompany-myteam`. Use this value for the `publisher` attribute in your manifest.
- Set a display name, such as `My Team`.

3. Review the [Marketplace Publisher Agreement](#), then select **Create**.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

After you create the publisher, you can manage items, although no items appear until you publish.

## Package your extension

To upload your extension, package it as a VSIX 2.0-compatible .vsix file. Microsoft provides a cross-platform command-line interface (CLI) to package and publish your extension.

1. Open your extension manifest file (`vss-extension.json`) and set the value of the `publisher` field to the ID of your publisher. For example:

JSON

```
{  
  ...  
  "id": "my-first-extension",  
  "publisher": "AnnetteNielsen",  
  ...  
}
```

2. From a command prompt, run the TFX tool's packaging command from your extension directory.

```
npx tfx-cli extension create
```

A message displays indicating your extension is successfully packaged:

```
==== Completed operation: create extension ===  
- VSIX: C:\my-first-extension\AnnetteNielsen.my-first-extension-1.0.0.vsix  
- Extension ID: my-first-extension  
- Extension Version: 1.0.0  
- Publisher: AnnetteNielsen
```

### ! Note

Increment the version of your extension or integration in the manifest with every update. Use the `--rev-version` command line switch. This switch increments the *patch* version number of your extension and saves the new version to your manifest.

## Check package size

Check the size of the vsix after it gets packaged. If it's greater than 50 MB, you need to optimize it. To do so, see the following considerations:

- Deduplicate common dependencies by stating them once in the extension package.
- Fetch dependencies at runtime or during install time rather than including them in the package. Consider using the tool installer library to pull tool dependencies at runtime. This approach caches the tool by version for private agents, preventing downloads for every build. The tool installer library doesn't work in disconnected scenarios (no internet), which should be mentioned in the task description or documentation.
- Use WebPack to tree shake dependencies in tasks.

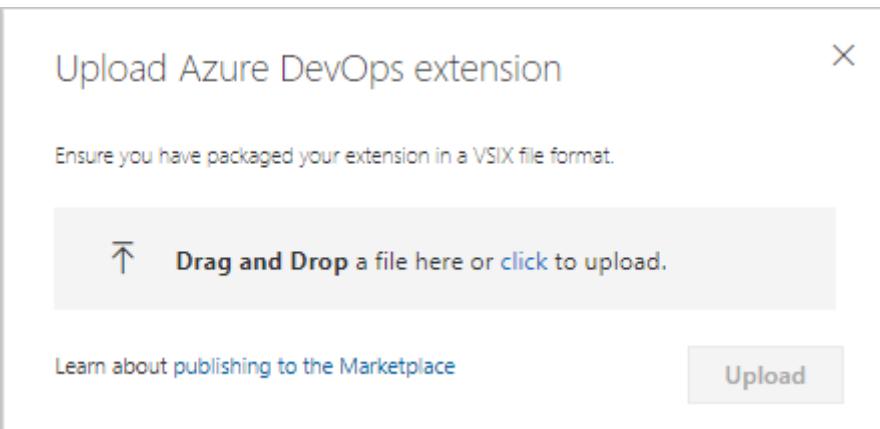
## Publish your extension

Once your extension is packaged, you can upload it to the Marketplace under a publisher. The **publisher** identifier specified in your [extension's manifest file](#) must match the identifier of the publisher the extension is uploaded under.

1. From the [management portal](#), select your publisher from the drop-down menu at the top of the page.
2. Select **New extension > Azure DevOps**.

Microsoft		Annette	+ New extension		
	Owner	Extensions	Details	Members	
	Owner	Name ↑	Visual Studio	Updated	
	Owner	Fabrikam Build Task	Visual Studio Code	2 years ago	
Microsoft Account	Owner		Azure DevOps	2 years ago	
	Owner	My first sample extension	0.1.0	2 years ago	
+ Create publisher					

3. Drag and drop your file or select it to find your VSIX file, which you created in the previous packaging step, and then choose **Upload**.



After quick validation, your extension appears in the list of published extensions. Don't worry, the extension is only visible to you.

Name ↑	Version	Updated	Availability
My First Extension	1.0.0	just now	Private (not shared)

At this point, your extension isn't visible to any accounts. To make it visible to others, you need to share the extension.

#### !**Note**

Microsoft runs a virus scan on each new and updated extension package published. Until the scan is all clear, we don't publish the extension in the Marketplace for public usage. This way we also avoid surfacing inappropriate or offensive content on the Marketplace pages.

## Share your extension

Share your extension with an organization before you can install it in Azure DevOps. To share an extension, do the following tasks:

1. From the [Marketplace management portal](#), select your extension from the list, right-click, and then choose **Share/Unshare** or **Publish/Unpublish**, depending on the extension.

The screenshot shows the 'Extensions' tab selected in the top navigation bar. Below it is a table with columns: Name, Version, and Updated. A single row is selected, showing 'My First Extension' with version 1.0.0 updated 'just now'. A context menu is open over this row, listing the following options: Reports, View Extension, Update, Remove, Share/Unshare, and Certificate.

2. Select **Organization**, and then enter the name of your organization. Select **Enter**.

The screenshot shows the 'Shared with' panel for the extension 'My First Extension'. It displays a list of accounts shared with, with one account listed: 'annette' from '.visualstudio.com'. There is a 'Remove' button next to the account entry.

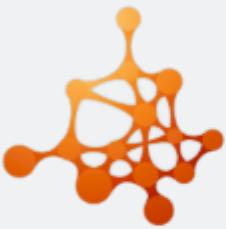
3. Close the panel.

Your extension can now be installed into this organization.

## Install your extension

To install your shared extension, do the following steps.

1. In the Marketplace, select your extension to open its overview page.



## My First Extension

| ★★★★★ (0) | Free

[Get it free](#)[Overview](#)[Q & A](#)[Rating & Review](#)

Sample Azure DevOps extension

**! Note**

Since your extension is private, only you and members of the organization it's shared with can see this page.

2. Select **Get it free** to start the installation process. Select the organization you shared the extension with from the dropdown menu.

The screenshot shows the 'My First Extension' page in the Visual Studio Marketplace. At the top, there's a note: 'Since your extension is private, only you and members of the organization it's shared with can see this page.' Below this, the main content area has a blue sidebar on the left with the extension name 'My First Extension'. The main panel starts with a 'Select a Visual Studio Team Services account' section, which includes a dropdown menu set to 'annette' and an 'Install' button. Below that is a 'For Team Foundation Server' section with a 'Download' button.

3. Select **Install**.

Congratulations! You installed your extension into an organization and you're ready to try it.

# Try your extension

1. Select **Proceed to organization** at the end of the installation wizard to go to the home page of the organization the extension was installed to (<https://dev.azure.com/{organization}>).
2. Refresh your browser.
3. Open **Organization settings**, and then select **Extensions**.

The screenshot shows the Azure DevOps Organization Settings page for the organization 'fabrikamprime'. The left sidebar includes sections for General (Overview, Projects, Users, Billing, Auditing, Global notifications, Usage, Extensions, Microsoft Entra), and a search bar. The 'Extensions' link in the sidebar is highlighted with a red box. The main content area is titled 'Extensions' and shows three tabs: 'Installed' (selected), 'Requested', and 'Shared'. Under the 'Installed' tab, there are six listed extensions:

Extension	Description
SARIF SAST Scans Tab by Microsoft	Adds a 'Scans' tab to each Build Pipeline
1ES GPT by Microsoft	1ES PT helper extension for running
Accessibility Insights for Azure DevOps	Scan accessibility issues in an Azure
Advanced Security Build Tasks by Microsoft	A set of build tasks for Advanced
Azure DevTest Labs Tasks by Microsoft	Collection of Azure Pipelines task
Build Security Monitoring by 1ES	Enables Build Security Monitoring

You should see the new extension on the **Installed** tab.

## Debug your extension

To debug the extension using Visual Studio or Browser Developer Tools, change the manifest by adding the `baseUri` property. This action speeds up the development without the need to redeploy the extension each time you change source code.

JSON

```
{  
  ...  
  "baseUri": "https://localhost:44300",  
  ...  
}
```

When you change the manifest, it loads the extension from your local web server instance. For example, IISExpress in Visual Studio. After you change the manifest, deploy and install this debugging extension only once.

 **Note**

Run your local web server in SSL mode because Azure DevOps demands that the web page is served from a secure source. Otherwise, you get an error in the browser console during the extension IFRAAME loading.

## Update your extension

To update an extension you already published, do the following steps:

 **Tip**

Update your extension instead of removing and re-uploading it. We recommend maintaining two extensions: `publisher.extension`, public in the Marketplace for customers, and `publisher.extension-dev`, private, shared only with your organization for development and testing. You don't need two copies of your source code—just maintain separate manifest files for each extension. When packaging, provide the appropriate manifest file to the tfx-cli tool. For more information, see [TFX extension commands](#).

1. Select your extension from the list of displayed items.
2. Right-click and select **Update** for the development version, such as `publisher.extension-dev`.
3. Validate your extension.
4. Apply the same updates to the production version, such as `publisher.extension`.

5. Browse to the .vsix file for your extension and upload it.

Azure DevOps automatically installs the updated version for all accounts that already have the extension. New installations also receive the latest version.

## Make your extension public

While you develop your extension or integration for the Marketplace, keep it private. This limits the visibility of the extension to specific accounts that you have shared it with.

To make your extension available publicly, set the [public flag](#) to `true` in your manifest.

## Qualifications

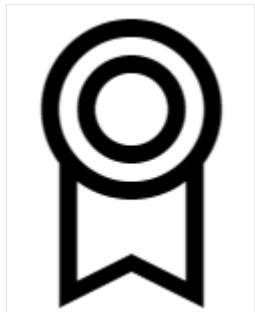
To have a public listing on the Marketplace, your integration or extension must meet the following qualifications:

- Works with or extends Azure DevOps.
- You, or your company, own, develop, and are licensed to distribute and advertise the integration or extension.
- The extension or integration is actively maintained.

Microsoft might also request a demo and to review the content planned for your Marketplace entry.

## Top Publisher

The Top Publisher program is only available for publishers with Azure DevOps extensions or integrations. It's not applicable for Visual Studio IDE and Visual Studio Code extension publishers.



The Top Publisher program recognizes publishers with commitment to their customers and the Marketplace through exemplary policies, quality, reliability, and support. Once you become a Top Publisher, all of your public offerings display the Top Publisher badge.

# Top Publisher requirements

The Top Publisher program in the Marketplace is designed to help you evaluate or acquire Azure DevOps extensions and integrations with confidence. The Top Publisher badge implies that the publisher shows commitment to their customers and the Marketplace through exemplary policies, quality, reliability, and support. It's for publishers with one or more global Azure DevOps extensions or integrations and isn't applicable for Visual Studio IDE and Visual Studio Code extension publishers.

Marketplace assigns the badge to a publisher after carefully reviewing the publisher across the following parameters:

- Privacy policy
- Licensing policy
- Support policy
- Documentation
- Q & A responsiveness
- Ratings and reviews for their offerings
- Active uptake and install count for their offerings
- Management of at least one public extension for Azure DevOps
- Public extension with more than 5,000 installs and an active install count exceeding 1000

You can expect timely support and a good overall experience when you get an extension from a Top Publisher. Check out the offerings from the Top Publishers.

For more information on adding policies to your offering, see the [extension manifest](#).

## 1. Update your publisher profile.

Through the publisher profile, you can showcase all of your offerings in one place along with key publisher-related information. To provide the information, which shows up in the profile, do the following steps:

- a. Sign in to <https://marketplace.visualstudio.com/manage/publishers> using the account with which you publish and manage your offerings in the Visual Studio Marketplace.

b. Select the publisher and complete the **About you** section in the **Details** tab.

**About you** ^

We will use this information to populate your publisher profile page

Description

Microsoft is an multinational technology company with vision to empower every person and every organization on the planet to achieve more.

Logo ⓘ

The Microsoft logo consists of four colored squares arranged in a 2x2 grid: top-left is red, top-right is green, bottom-left is blue, and bottom-right is yellow.

Company website

<https://www.microsoft.com>

Support

<https://support.microsoft.com>

LinkedIn

<https://www.linkedin.com/company/microsoft>

Source code repository

<https://github.com/microsoft>

Twitter

<https://twitter.com/microsoft>

c. Save your changes and select **View profile** to see how it appears to consumers. You can use this profile page to evangelize your offerings.

**!** Note

This program certifies the publisher, not the software, or security of their extensions and integrations. We recommend reviewing the [safety information](#) when evaluating offerings from a publisher. If you got an extension from a Top Publisher and aren't satisfied with your experience, consider engaging with the publisher first.

## Apply to be a Top Publisher

1. Sign in to [Marketplace management portal](#) using the account with which you publish and manage your offerings in Marketplace.
2. Select the publisher and go to its **Top Publisher** tab. Have one or more global Azure DevOps (Server/Service) extensions or integrations for the tab to appear.
3. If you meet the previously listed requirements and are the publisher's owner, you can apply for the program. Upon application, an email gets sent to the Marketplace team to

review your case. They respond within 10 business days with next steps, clarifying questions, or the grant of the badge.

The team likely considers other parameters, such as active uptake of your offerings, install/get started counts, and ratings & reviews across your offerings before granting the badge. Microsoft reserves the right to grant, reject, or revoke the Top Publisher badge at any time.

Once a publisher is a Top Publisher, all its future updates and offerings must meet the previously listed requirements.

## Respond to Marketplace extension reviews

You can respond to reviews that customers leave for your extensions in the Visual Studio Marketplace. Find and select **Reply** next to a review if you have one of the following permissions: owner, creator, or contributor.

You can leave only one response. Avoid using reviews as a support forum. If you need more details, please provide a support alias for the reviewer to contact. You can then resolve their problems externally and update your reply with a resolution.

## Guidelines for publisher responses

Keep the Visual Studio Marketplace an open, inviting, respectful, and helpful place for customers to find, try, install, and review extensions. Communication plays an important role in keeping a healthy community. To help create this environment, here are guidelines for publishers responding to customer reviews. Think deeply about your customer interactions and reflect on the spirit of the customer experience that the Marketplace is trying to create.

- Reserve reviews for customer comments. Use *Reply* only to respond to a review.
- Respect all customer opinions. Treat comments as feedback without debate, criticism, or argument.
- Ensure your responses add value and are relevant to your customers' comments.
- Focus on precisely addressing questions or problems. If you need more details, ask the customer to contact you over email rather than discussing in reviews. When you resolve the problem, update your reply with the resolution. You can edit your reply like customers can edit their reviews.
- Flag any inappropriate reviews, such as spam, abusive, or offensive content, for our review.

## Request to void a review

As a publisher, you can appeal to void a review if the issue reported is because of the Marketplace or underlying platform. If the issue is valid, Marketplace admins void the rating. You can **Appeal** from ratings and review section on your extension hub page.

## Unpublish an extension

You can unpublish free extensions if you no longer want to offer them in the Marketplace.

Consider removing your extension from the Marketplace in the following scenarios:

- You developed a new extension and no longer want to offer the current one.
- Your extension has a problem, and you want to remove it from the Marketplace until you resolve the issue.
- You published your extension as public by mistake.

Certain criteria must be met for an extension to be unpublished or removed:

[+] Expand table

Action	Requirements
Unpublish	Only <b>free extensions</b> might be unpublished.
Remove	Your extension must have <b>zero (0) installs</b> to be removed.

### (i) Important

If you must remove your extension because of legal or security problems, contact [Customer Support at the Developer Community](#). We review the request and manually delete the extension.

1. Select the extension on your [publisher page](#) and choose **Unpublish** on the menu.

Your extension is unpublished immediately from the Marketplace, and new users can't install it. Ratings and reviews for your extension stay intact.

To offer your extension again in the Marketplace, select **Publish** from the menu.

If your extension has zero installs, you can choose to remove it completely from the Marketplace. To do so, select **Remove** from the menu. You can't reverse this action.

## Extension reporting hub

Once your extension is available in the Visual Studio Marketplace, you can use the **Reports** feature. With this feature, you can track and analyze how the extension is performing and take required actions. To visit the extension hub, browse to your [publisher page](#) and select the extension or select the **Reports** link on the extension details page.

## Acquisition

You can view acquisition-related data in this tab for the selected period.

- Aggregated acquisition in the selected period for overall acquisition
- Aggregated acquisition split by extension downloads Azure DevOps connected install for free extension
- Aggregated acquisition split by trials Azure DevOps connected buy for paid extension
- Daily trend of extension page views with acquisition for Azure DevOps and connected server
- Conversion percentage from page views to acquisition

For paid extensions, all transactional details for buy and trials are available with date, organization name, trial end date, and quantity. You can use the **Contact** action to communicate with your users. For more information, see the [Contact](#) section provided later in this article.

## Uninstall

You can view the following statistics:

- How many organizations uninstalled your extension
- Daily trend of uninstall extensions
- Detailed feedback shared during uninstalls
- Top uninstall reasons

You can use search for text and dates to analyze and draw more insights from the detailed feedback.

For paid extensions, you can use the **Contact** action to communicate with your users. [Contact](#) section provided later in this article for more details.

## Ratings and review

This tab gives you the following information:

- Average rating for the selected period versus overall rating
- Average rating by number of reviewers

- Daily trend of average rating

The details section provides all the reviews and your responses in transactional view.

You can **Reply** to a review or **Edit** a previous response and better manage engagement with your extension users. You can also **Appeal** to void a rating if the issue reported is because of the Marketplace or underlying platform. If the issue is valid, we void the rating.

## Manage engagement

The Q & A tab provides a snapshot of all questions from your extension users, with nonresponded queries at the top. You can reply to or edit previous responses to better manage engagement with your extension users.

## Export to Excel

All data elements available in the reports page are also available for download in XLS format to aid creating your own custom reports.

## Contact

For paid extensions, you can use the **Contact** action to communicate with your users. This feature is available only for publishers with Contributor+ access on the extension.

Marketplace brokers the first communication with the user as our privacy policy doesn't allow direct sharing of customer email addresses. Only users who opted in for communication receive the email. The last contacted date for an organization is updated after sending a communication.

### Important

Follow the guidance on transactional and promotional communication. Publishers found to be sending promotional communication or spamming users get added to a blocklist and lose access to the **Contact** feature for all their extensions.

**Transactional communication:** Emails conveying critical information necessary for the continued use of the extension or service, such as:

- Critical security notices
- Transaction confirmations
- Product recall notices
- Specific feedback requests

- Service discontinuation notices

**Promotional emails:** Emails used to market your extension, product, service, website, or event, such as:

- Invitations to events or webcasts
- Information about new marketing or partner programs
- Offers to obtain value-added content
- Newsletters containing promotional content

For more information, see the [Marketplace Publisher Agreement](#).

 [Expand table](#)

Terminology	Description
Page views	Total number of extension detail page views. Repeated views are counted.
Azure DevOps Services installs	Total number of organizations the extension is installed in. Repeated installs on the same organization get counted.
Azure DevOps Server installs	Total number of collections the extension is installed in. Repeated installs on the same collection get counted. Disconnected server data isn't available.

## Related content

- [Develop a web extension](#)
- [Explore extensibility points](#)

 **Note:** The author created this article with assistance from AI. [Learn more](#)

# Publish from the command line

04/07/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

You can use the Cross-platform CLI for Azure DevOps (tfx-cli) to publish your extension to the Visual Studio Marketplace.

For more information, see the overview of [publish, install, and share](#).

## Prerequisites

Get the TFX CLI from Node Package Manager and generate a Microsoft Entra token or a personal access token (PAT). Also, if you haven't already, set up a Publisher in the Gallery.

## Acquire the Cross-platform CLI for Azure DevOps

1. If you don't have it, download and install [NodeJS](#). During set up, ensure that you leave **Add to PATH** chosen.
2. Open a Command Prompt and enter `npm i -g tfx-cli`.

If you already have the TFX CLI installed, you can update to the latest release by running `npm up -g tfx-cli`.

## Publish with a Microsoft Entra token as a service principal

It is also possible to publish an extension as a [service principal](#).

1. Add the service principal as a member to a publisher account. You can get the service principal's ID through the REST API by logging in via the az cli and querying the service principal's profile. This can be done with the following commands:

Bash

Azure CLI

```
az login --service-principal --username <appId> --password <password> --tenant <tenant-id>
# 499b84ac-1321-427f-aa17-267ca6975798 specifies azure devops as a resource
```

```
az rest -u https://app.vssps.visualstudio.com/_apis/profile/profiles/me --  
resource 499b84ac-1321-427f-aa17-267ca6975798
```

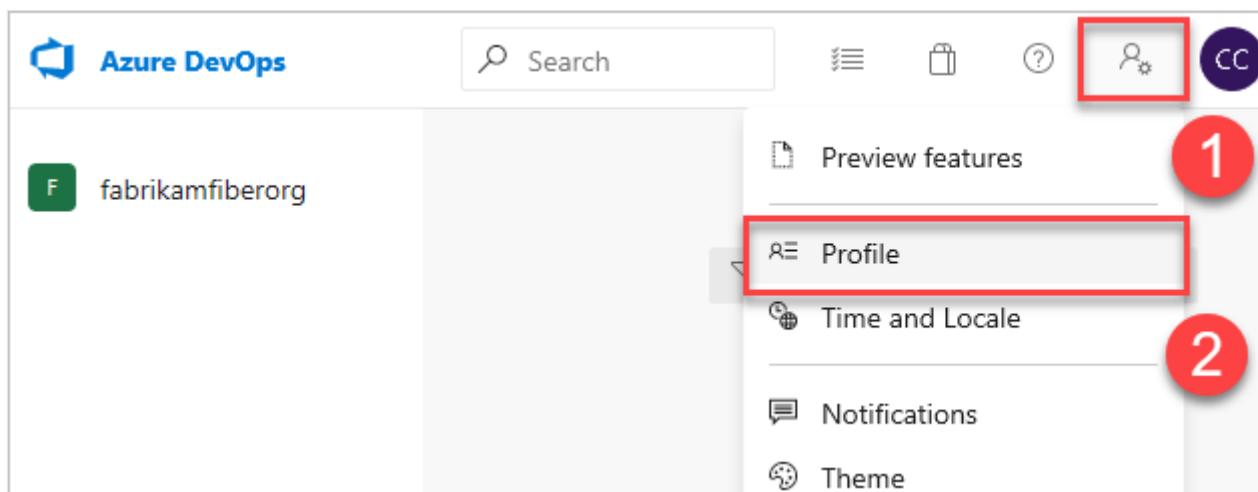
Then, you can [add the service principal as a member](#) to the publisher using the ID from the previous step.

2. Publish an extension via [TFX CLI](#) using a service principal. Execute the following [TFX CLI](#) command to use its access token:

```
tfx extension publish --publisher my-publisher --vsix my-publisher.my-extension-  
1.0.0.vsix --auth-type pat -t <ENTRA_TOKEN>
```

## Publish with a personal access token

1. Sign in to your organization (<https://dev.azure.com/{organization}>).
2. From your home page, open the user settings icon, and then select **Profile**.



3. Under **Security**, select **Personal access tokens**, and then select **New Token**.



Azure DevOps

## User settings

Account

Profile

Time and Locale

## Preferences

Notifications

Theme

Usage

## Security

Personal access tokens

SSH public keys

Alternate credentials

Authorizations

## Personal Access Tokens

These can be used instead of a password

+ New Token

Token name ↓

2

1

4. Complete the form:

- Name your token
- Select **All accessible organizations**, which is the only value that works when publishing via CLI. If you select only one organization, it results in an error, even if the PAT is valid
- Select an expiration time frame for your token. This is required because the Visual Studio Marketplace publishing APIs work outside of the context of an organization
- Set the scope of access associated with this token. Be sure to select the **Marketplace (publish)** scope. This scope limits the token to only being able to publish extensions to the Marketplace.
- Select **Create**

## Create a new personal access token

X

Name

New

Organization

All accessible organizations



Expiration (UTC)

30 days



5/6/2020



### Scopes

Authorize the scope of access associated with this token

Scopes  Full access

Custom defined

#### Identity

Identities and groups

Read     Read & manage

#### Load Test

Create and update load test runs and read metadata

Read     Read & write

#### Marketplace

Read, publish, update, and manage items and publishers

Read     Acquire     Publish     Manage

#### Member Entitlement Management

Read and manage users

Read     Read & write

#### Notifications

Read, write, manage, and publish

Show less scopes

Create

Cancel

5. Copy your generated personal access token. Make sure to keep it secret.

## Success!



You have successfully added a new personal access token. Copy the token now!

New TOKEN! token

Ilvy6hg3nrckfcqnwmn4ad7z



Warning - Make sure you copy the above token now.  
We don't store it and you will not be able to see it again.

Close

Once TFX CLI is installed and you have your token, you can use the tool to package and publish your extension.

1. Open a command prompt to the root directory of your extension.
2. Run the following command to publish your extension. When prompted, enter your token to authenticate.

### Command

```
tfx extension publish --publisher <YOUR_PUBLISHER_ID> --manifest-js  
<YOUR_EXTENSION_MANIFEST> --share-with <ACCOUNT_NAME>
```

## Potential Errors

You may receive the following error if your extension has already been published:

```
Failed Request: Internal Server Error(500) - Version number must increase each  
time an extension is published. Extension: fabrikam.my-extension Current version:  
0.1.9 Updated version: 0.1.9
```

You can add the `--rev-version` flag to automatically increment the *patch* version of your extension. This also saves the new version to your manifest.

### ! Note

All options available for `create` are available for the `publish` command.

## Example

```
C:\vso-team-calendar>tfx extension publish --publisher publishFabrikam --manifest-
js fabrikam.config.js --share-with fabrikam --rev-version
Copyright Microsoft Corporation
> Personal access token:
Checking if this extension is already published
It is, update the extension
Waiting for server to validate extension package...
Sharing extension with fabrikam.

==== Completed operation: publish extension ===
- Packaging: C:\vso-team-calendar\fabrikam.team-calendar-0.2.6.vsix
- Publishing: success
- Sharing: shared with fabrikam
```

# Package and publish extensions

07/17/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Once you [develop your extension](#), you can package and publish it to the [Visual Studio Marketplace](#). The Marketplace is a global repository for private and public extensions, integrations, and other offers from Microsoft.

## (!) Note

For information on the discovery properties available in your extension's manifest file that helps users discover and learn about your extension, see the [Extension Manifest Reference](#).

## Prerequisites

The following list of requirements must be met before you publish to the Marketplace.

[+] [Expand table](#)

Category	Requirements
Packaging tool	Install the extension packaging tool (TFX). Run <code>npm install -g tfx-cli</code> from a command prompt.
Image permissions	Ensure you have proper permissions to use any images, like icons, logos, screenshots, and so on.
Marketplace overview	Include a thorough <code>overview.md</code> file to describe your listing in the Marketplace.
Extension icon	Include an icon for your extension that represents your integration, company, or organization, at least 128x128 pixels in size (PNG or JPEG).
Microsoft product names	Use full names for Microsoft products (for example, Azure DevOps instead of AzDO or other abbreviations).
Brand names	Don't use brand names in the name of your extension.

## Create a publisher

Every extension or integration, including those from Microsoft, must have a publisher. Anyone can create a publisher and publish extensions under it. You can also share publisher access with other users, such as your development team.

1. Sign in to the [Visual Studio Marketplace Publishing Portal](#).
2. If you aren't part of an existing publisher, select **+ Create a publisher**.  
Enter a publisher name; the ID field autofills based on your entry.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

 **Note**

- Ensure your publisher name is within 16 characters for multibyte characters.
- Save the publisher ID—you need it in your extension's manifest file.

If you aren't prompted to create a publisher, scroll to **Publish extensions** under *Related sites*.

- Set a unique publisher identifier, such as `mycompany-myteam`. Use this value for the `publisher` attribute in your manifest.
- Set a display name, such as `My Team`.

3. Review the [Marketplace Publisher Agreement](#), then select **Create**.

Create Publisher | Visual Studio

https://marketplace.visualstudio.com/manage/createpublisher

Name \* ⓘ

Name of the publisher

ID \* ⓘ

Unique publisher identifier

About you ^

We will use this information to populate your publisher profile page

Description

Details of publisher

Logo ⓘ

128px X 128px

Drag and Drop file or click to upload

Company website

Ex: https://www.microsoft.com

Support

Ex: support@microsoft.com or https://www.microsoft.com

LinkedIn

Ex: https://www.linkedin.com/company/microsoft

Source code repository

Ex: https://github.com/microsoft

Twitter

Ex: https://twitter.com/microsoft

**Create** **Cancel**

After you create the publisher, you can manage items, although no items appear until you publish.

## Package your extension

To upload your extension, package it as a VSIX 2.0-compatible .vsix file. Microsoft provides a cross-platform command-line interface (CLI) to package and publish your extension.

1. Open your extension manifest file (`vss-extension.json`) and set the value of the `publisher` field to the ID of your publisher. For example:

JSON

```
{  
  ...  
  "id": "my-first-extension",  
  "publisher": "AnnetteNielsen",  
  ...  
}
```

2. From a command prompt, run the TFX tool's packaging command from your extension directory.

```
npx tfx-cli extension create
```

A message displays indicating your extension is successfully packaged:

```
==== Completed operation: create extension ===  
- VSIX: C:\my-first-extension\AnnetteNielsen.my-first-extension-1.0.0.vsix  
- Extension ID: my-first-extension  
- Extension Version: 1.0.0  
- Publisher: AnnetteNielsen
```

### ! Note

Increment the version of your extension or integration in the manifest with every update. Use the `--rev-version` command line switch. This switch increments the *patch* version number of your extension and saves the new version to your manifest.

## Check package size

Check the size of the vsix after it gets packaged. If it's greater than 50 MB, you need to optimize it. To do so, see the following considerations:

- Deduplicate common dependencies by stating them once in the extension package.
- Fetch dependencies at runtime or during install time rather than including them in the package. Consider using the tool installer library to pull tool dependencies at runtime. This approach caches the tool by version for private agents, preventing downloads for every build. The tool installer library doesn't work in disconnected scenarios (no internet), which should be mentioned in the task description or documentation.
- Use WebPack to tree shake dependencies in tasks.

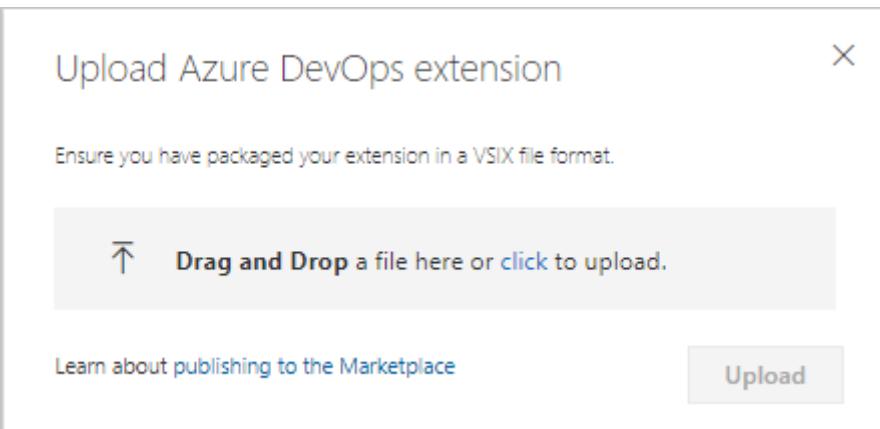
## Publish your extension

Once your extension is packaged, you can upload it to the Marketplace under a publisher. The **publisher** identifier specified in your [extension's manifest file](#) must match the identifier of the publisher the extension is uploaded under.

1. From the [management portal](#), select your publisher from the drop-down menu at the top of the page.
2. Select **New extension > Azure DevOps**.

Microsoft		Annette	+ New extension		
	Owner	Extensions	Details	Members	
	Owner	Name ↑	Visual Studio	Updated	
	Owner	Fabrikam Build Task	Visual Studio Code	2 years ago	
Microsoft Account	Owner		Azure DevOps	2 years ago	
	Owner	My first sample extension	0.1.0	2 years ago	
+ Create publisher					

3. Drag and drop your file or select it to find your VSIX file, which you created in the previous packaging step, and then choose **Upload**.



After quick validation, your extension appears in the list of published extensions. Don't worry, the extension is only visible to you.

Name ↑	Version	Updated	Availability
My First Extension	1.0.0	just now	Private (not shared)

At this point, your extension isn't visible to any accounts. To make it visible to others, you need to share the extension.

#### !**Note**

Microsoft runs a virus scan on each new and updated extension package published. Until the scan is all clear, we don't publish the extension in the Marketplace for public usage. This way we also avoid surfacing inappropriate or offensive content on the Marketplace pages.

## Share your extension

Share your extension with an organization before you can install it in Azure DevOps. To share an extension, do the following tasks:

1. From the [Marketplace management portal](#), select your extension from the list, right-click, and then choose **Share/Unshare** or **Publish/Unpublish**, depending on the extension.

The screenshot shows the 'Extensions' tab selected in the top navigation bar. Below it is a table with columns: Name, Version, and Updated. A single row is visible for 'My First Extension', which is version 1.0.0 and was updated 'just now'. To the right of this row is a context menu with the following options: Reports, View Extension, Update, Remove, Share/Unshare, and Certificate.

2. Select **Organization**, and then enter the name of your organization. Select **Enter**.

The screenshot shows the 'Shared with' panel for 'My First Extension'. It displays a list of accounts shared with the extension. A share entry for 'annette' at '.visualstudio.com' is shown, with a trash icon next to it.

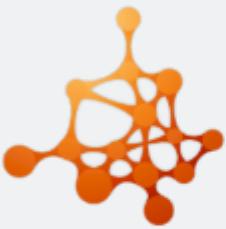
3. Close the panel.

Your extension can now be installed into this organization.

## Install your extension

To install your shared extension, do the following steps.

1. In the Marketplace, select your extension to open its overview page.



## My First Extension

| ★★★★★ (0) | Free

[Get it free](#)[Overview](#)[Q & A](#)[Rating & Review](#)

Sample Azure DevOps extension

**! Note**

Since your extension is private, only you and members of the organization it's shared with can see this page.

2. Select **Get it free** to start the installation process. Select the organization you shared the extension with from the dropdown menu.

The screenshot shows the 'My First Extension' page in the Visual Studio Marketplace. At the top, there's a note: 'Since your extension is private, only you and members of the organization it's shared with can see this page.' Below this, the main content area has a blue sidebar on the left with the extension name 'My First Extension'. The main panel starts with a 'Select a Visual Studio Team Services account' section, which includes a dropdown menu set to 'annette' and an 'Install' button. Below this, there are links for 'For Team Foundation Server' and 'Download'.

3. Select **Install**.

Congratulations! You installed your extension into an organization and you're ready to try it.

# Try your extension

1. Select **Proceed to organization** at the end of the installation wizard to go to the home page of the organization the extension was installed to (<https://dev.azure.com/{organization}>).
2. Refresh your browser.
3. Open **Organization settings**, and then select **Extensions**.

The screenshot shows the Azure DevOps Organization Settings page for the organization 'fabrikamprime'. The left sidebar includes sections for General (Overview, Projects, Users, Billing, Auditing, Global notifications, Usage, Extensions, Microsoft Entra), and a search bar. The 'Extensions' link in the sidebar is highlighted with a red box. The main content area is titled 'Extensions' and shows three tabs: 'Installed' (selected), 'Requested', and 'Shared'. Under the 'Installed' tab, there are six listed extensions:

Extension	Description
SARIF SAST Scans Tab by Microsoft	Adds a 'Scans' tab to each Build Pipeline
1ES GPT by Microsoft	1ES PT helper extension for running
Accessibility Insights for Azure DevOps	Scan accessibility issues in an Azure
Advanced Security Build Tasks by Microsoft	A set of build tasks for Advanced
Azure DevTest Labs Tasks by Microsoft	Collection of Azure Pipelines task
Build Security Monitoring by 1ES	Enables Build Security Monitoring

You should see the new extension on the **Installed** tab.

## Debug your extension

To debug the extension using Visual Studio or Browser Developer Tools, change the manifest by adding the `baseUri` property. This action speeds up the development without the need to redeploy the extension each time you change source code.

JSON

```
{  
  ...  
  "baseUri": "https://localhost:44300",  
  ...  
}
```

When you change the manifest, it loads the extension from your local web server instance. For example, IISExpress in Visual Studio. After you change the manifest, deploy and install this debugging extension only once.

 **Note**

Run your local web server in SSL mode because Azure DevOps demands that the web page is served from a secure source. Otherwise, you get an error in the browser console during the extension IFRAAME loading.

## Update your extension

To update an extension you already published, do the following steps:

 **Tip**

Update your extension instead of removing and re-uploading it. We recommend maintaining two extensions: `publisher.extension`, public in the Marketplace for customers, and `publisher.extension-dev`, private, shared only with your organization for development and testing. You don't need two copies of your source code—just maintain separate manifest files for each extension. When packaging, provide the appropriate manifest file to the tfx-cli tool. For more information, see [TFX extension commands](#).

1. Select your extension from the list of displayed items.
2. Right-click and select **Update** for the development version, such as `publisher.extension-dev`.
3. Validate your extension.
4. Apply the same updates to the production version, such as `publisher.extension`.

5. Browse to the .vsix file for your extension and upload it.

Azure DevOps automatically installs the updated version for all accounts that already have the extension. New installations also receive the latest version.

## Make your extension public

While you develop your extension or integration for the Marketplace, keep it private. This limits the visibility of the extension to specific accounts that you have shared it with.

To make your extension available publicly, set the [public flag](#) to `true` in your manifest.

## Qualifications

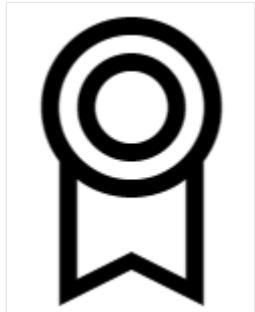
To have a public listing on the Marketplace, your integration or extension must meet the following qualifications:

- Works with or extends Azure DevOps.
- You, or your company, own, develop, and are licensed to distribute and advertise the integration or extension.
- The extension or integration is actively maintained.

Microsoft might also request a demo and to review the content planned for your Marketplace entry.

## Top Publisher

The Top Publisher program is only available for publishers with Azure DevOps extensions or integrations. It's not applicable for Visual Studio IDE and Visual Studio Code extension publishers.



The Top Publisher program recognizes publishers with commitment to their customers and the Marketplace through exemplary policies, quality, reliability, and support. Once you become a Top Publisher, all of your public offerings display the Top Publisher badge.

# Top Publisher requirements

The Top Publisher program in the Marketplace is designed to help you evaluate or acquire Azure DevOps extensions and integrations with confidence. The Top Publisher badge implies that the publisher shows commitment to their customers and the Marketplace through exemplary policies, quality, reliability, and support. It's for publishers with one or more global Azure DevOps extensions or integrations and isn't applicable for Visual Studio IDE and Visual Studio Code extension publishers.

Marketplace assigns the badge to a publisher after carefully reviewing the publisher across the following parameters:

- Privacy policy
- Licensing policy
- Support policy
- Documentation
- Q & A responsiveness
- Ratings and reviews for their offerings
- Active uptake and install count for their offerings
- Management of at least one public extension for Azure DevOps
- Public extension with more than 5,000 installs and an active install count exceeding 1000

You can expect timely support and a good overall experience when you get an extension from a Top Publisher. Check out the offerings from the Top Publishers.

For more information on adding policies to your offering, see the [extension manifest](#).

## 1. Update your publisher profile.

Through the publisher profile, you can showcase all of your offerings in one place along with key publisher-related information. To provide the information, which shows up in the profile, do the following steps:

- a. Sign in to <https://marketplace.visualstudio.com/manage/publishers> using the account with which you publish and manage your offerings in the Visual Studio Marketplace.

b. Select the publisher and complete the **About you** section in the **Details** tab.

**About you** ^

We will use this information to populate your publisher profile page

Description

Microsoft is an multinational technology company with vision to empower every person and every organization on the planet to achieve more.

Logo ⓘ

The Microsoft logo consists of four colored squares arranged in a 2x2 grid: top-left is red, top-right is green, bottom-left is blue, and bottom-right is yellow.

Company website

<https://www.microsoft.com>

Support

<https://support.microsoft.com>

LinkedIn

<https://www.linkedin.com/company/microsoft>

Source code repository

<https://github.com/microsoft>

Twitter

<https://twitter.com/microsoft>

c. Save your changes and select **View profile** to see how it appears to consumers. You can use this profile page to evangelize your offerings.

**!** Note

This program certifies the publisher, not the software, or security of their extensions and integrations. We recommend reviewing the [safety information](#) when evaluating offerings from a publisher. If you got an extension from a Top Publisher and aren't satisfied with your experience, consider engaging with the publisher first.

## Apply to be a Top Publisher

1. Sign in to [Marketplace management portal](#) using the account with which you publish and manage your offerings in Marketplace.
2. Select the publisher and go to its **Top Publisher** tab. Have one or more global Azure DevOps (Server/Service) extensions or integrations for the tab to appear.
3. If you meet the previously listed requirements and are the publisher's owner, you can apply for the program. Upon application, an email gets sent to the Marketplace team to

review your case. They respond within 10 business days with next steps, clarifying questions, or the grant of the badge.

The team likely considers other parameters, such as active uptake of your offerings, install/get started counts, and ratings & reviews across your offerings before granting the badge. Microsoft reserves the right to grant, reject, or revoke the Top Publisher badge at any time.

Once a publisher is a Top Publisher, all its future updates and offerings must meet the previously listed requirements.

## Respond to Marketplace extension reviews

You can respond to reviews that customers leave for your extensions in the Visual Studio Marketplace. Find and select **Reply** next to a review if you have one of the following permissions: owner, creator, or contributor.

You can leave only one response. Avoid using reviews as a support forum. If you need more details, please provide a support alias for the reviewer to contact. You can then resolve their problems externally and update your reply with a resolution.

## Guidelines for publisher responses

Keep the Visual Studio Marketplace an open, inviting, respectful, and helpful place for customers to find, try, install, and review extensions. Communication plays an important role in keeping a healthy community. To help create this environment, here are guidelines for publishers responding to customer reviews. Think deeply about your customer interactions and reflect on the spirit of the customer experience that the Marketplace is trying to create.

- Reserve reviews for customer comments. Use *Reply* only to respond to a review.
- Respect all customer opinions. Treat comments as feedback without debate, criticism, or argument.
- Ensure your responses add value and are relevant to your customers' comments.
- Focus on precisely addressing questions or problems. If you need more details, ask the customer to contact you over email rather than discussing in reviews. When you resolve the problem, update your reply with the resolution. You can edit your reply like customers can edit their reviews.
- Flag any inappropriate reviews, such as spam, abusive, or offensive content, for our review.

## Request to void a review

As a publisher, you can appeal to void a review if the issue reported is because of the Marketplace or underlying platform. If the issue is valid, Marketplace admins void the rating. You can **Appeal** from ratings and review section on your extension hub page.

## Unpublish an extension

You can unpublish free extensions if you no longer want to offer them in the Marketplace.

Consider removing your extension from the Marketplace in the following scenarios:

- You developed a new extension and no longer want to offer the current one.
- Your extension has a problem, and you want to remove it from the Marketplace until you resolve the issue.
- You published your extension as public by mistake.

Certain criteria must be met for an extension to be unpublished or removed:

[+] Expand table

Action	Requirements
Unpublish	Only <b>free extensions</b> might be unpublished.
Remove	Your extension must have <b>zero (0)</b> installs to be removed.

### (i) Important

If you must remove your extension because of legal or security problems, contact [Customer Support at the Developer Community](#). We review the request and manually delete the extension.

1. Select the extension on your [publisher page](#) and choose **Unpublish** on the menu.

Your extension is unpublished immediately from the Marketplace, and new users can't install it. Ratings and reviews for your extension stay intact.

To offer your extension again in the Marketplace, select **Publish** from the menu.

If your extension has zero installs, you can choose to remove it completely from the Marketplace. To do so, select **Remove** from the menu. You can't reverse this action.

## Extension reporting hub

Once your extension is available in the Visual Studio Marketplace, you can use the **Reports** feature. With this feature, you can track and analyze how the extension is performing and take required actions. To visit the extension hub, browse to your [publisher page](#) and select the extension or select the **Reports** link on the extension details page.

## Acquisition

You can view acquisition-related data in this tab for the selected period.

- Aggregated acquisition in the selected period for overall acquisition
- Aggregated acquisition split by extension downloads Azure DevOps connected install for free extension
- Aggregated acquisition split by trials Azure DevOps connected buy for paid extension
- Daily trend of extension page views with acquisition for Azure DevOps and connected server
- Conversion percentage from page views to acquisition

For paid extensions, all transactional details for buy and trials are available with date, organization name, trial end date, and quantity. You can use the **Contact** action to communicate with your users. For more information, see the [Contact](#) section provided later in this article.

## Uninstall

You can view the following statistics:

- How many organizations uninstalled your extension
- Daily trend of uninstall extensions
- Detailed feedback shared during uninstalls
- Top uninstall reasons

You can use search for text and dates to analyze and draw more insights from the detailed feedback.

For paid extensions, you can use the **Contact** action to communicate with your users. [Contact](#) section provided later in this article for more details.

## Ratings and review

This tab gives you the following information:

- Average rating for the selected period versus overall rating
- Average rating by number of reviewers

- Daily trend of average rating

The details section provides all the reviews and your responses in transactional view.

You can **Reply** to a review or **Edit** a previous response and better manage engagement with your extension users. You can also **Appeal** to void a rating if the issue reported is because of the Marketplace or underlying platform. If the issue is valid, we void the rating.

## Manage engagement

The Q & A tab provides a snapshot of all questions from your extension users, with nonresponded queries at the top. You can reply to or edit previous responses to better manage engagement with your extension users.

## Export to Excel

All data elements available in the reports page are also available for download in XLS format to aid creating your own custom reports.

## Contact

For paid extensions, you can use the **Contact** action to communicate with your users. This feature is available only for publishers with Contributor+ access on the extension.

Marketplace brokers the first communication with the user as our privacy policy doesn't allow direct sharing of customer email addresses. Only users who opted in for communication receive the email. The last contacted date for an organization is updated after sending a communication.

### Important

Follow the guidance on transactional and promotional communication. Publishers found to be sending promotional communication or spamming users get added to a blocklist and lose access to the **Contact** feature for all their extensions.

**Transactional communication:** Emails conveying critical information necessary for the continued use of the extension or service, such as:

- Critical security notices
- Transaction confirmations
- Product recall notices
- Specific feedback requests

- Service discontinuation notices

**Promotional emails:** Emails used to market your extension, product, service, website, or event, such as:

- Invitations to events or webcasts
- Information about new marketing or partner programs
- Offers to obtain value-added content
- Newsletters containing promotional content

For more information, see the [Marketplace Publisher Agreement](#).

 [Expand table](#)

Terminology	Description
Page views	Total number of extension detail page views. Repeated views are counted.
Azure DevOps Services installs	Total number of organizations the extension is installed in. Repeated installs on the same organization get counted.
Azure DevOps Server installs	Total number of collections the extension is installed in. Repeated installs on the same collection get counted. Disconnected server data isn't available.

## Related content

- [Develop a web extension](#)
- [Explore extensibility points](#)

 **Note:** The author created this article with assistance from AI. [Learn more](#)

# Extension manifest reference

05/07/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Every extension has a JSON manifest file that defines basic information about the extension. The file also defines how it can extend and enhance the experience. This article shows you how to create a manifest for your extension to Azure DevOps.

## 💡 Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

Create a file named `vss-extension.json` at the root of your extension folder. This file contains required attributes, like the extension's ID and its installation targets, where it can run. It also defines the contributions being made by your extension.

See the following example of a typical manifest:

JSON

```
{
  "manifestVersion": 1,
  "id": "tools",
  "version": "0.1.0",
  "name": "Fabrikam Tools",
  "publisher": "fabrikam",
  "description": "Awesome tools to help you and your team do great things everyday.",
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services"
    }
  ],
  "icons": {
    "default": "images/fabrikam-logo.png"
  },
  "scopes": [
    "vso.work",
    "vso.code_write",
    "vso.build_execute"
  ],
  "categories": [
    "Azure Boards"
  ],
  "branding": {
    "color": "rgb(34, 34, 34)",
    "theme": "dark"
  },
  "content": {
    "details": {
      "path": "readme.md"
    },
    "license": {
      "path": "eula.md"
    }
  },
  "links": {
    "getstarted": {
      "uri": "https://www.fabrikam-fiber-inc.com/help/getstarted"
    },
    "support": {
      "uri": "https://www.fabrikam-fiber-inc.com/support"
    }
  }
}
```

```

    }
},
"repository": {
    "type": "git",
    "uri": "https://github.com/fabrikam-fiber-inc/myextension"
},
"contributions": [
    {
        "id": "showCommits",
        "type": "ms.vss-web.action",
        "description": "Adds a menu action from builds grid to show associated items.",
        "targets": [
            "ms.vss-build-web.completed-build-menu"
        ],
        "properties": {
            "title": "View associated items",
            "uri": "launch.html"
        }
    }
],
"files": [
    {
        "path": "launch.html",
        "addressable": true
    },
    {
        "path": "node_modules/vss-web-extension-sdk/lib",
        "addressable": true,
        "packagePath": "lib"
    }
]
}
}

```

For information about inputs, see [...](#)

## Required attributes

The following properties are required:

[Expand table](#)

Property	Description	Notes
manifestVersion	A number corresponding to the version of the manifest format.	Should be <a href="#">1</a> .
ID	The extension's identifier.	The ID is a string that must be unique among extensions from the same publisher. It must start with an alphabetic or numeric character and contain 'A' through 'Z', 'a' through 'z', '0' through '9', and '-' (hyphen). Example: <a href="#">sample-extension</a> .
version	A string specifying the version of an extension.	Should be in the format <code>major.minor.patch</code> , for example <a href="#">0.1.2</a> or <a href="#">1.0.0</a> . You can also add a fourth number for the following format: <a href="#">0.1.2.3</a>
name	A short, human-readable name of the extension. Limited to 200 characters.	Example: <a href="#">"Fabrikam Agile Board Extension"</a> .
publisher	The identifier of the publisher.	This identifier must match the identifier the extension is published under. See <a href="#">Create and manage a publisher</a> .
categories	Array of strings representing the categories your extension belongs to. At least one category must be	Valid values: <a href="#">Azure Repos</a> , <a href="#">Azure Boards</a> , <a href="#">Azure Pipelines</a> , <a href="#">Azure Test Plans</a> , and <a href="#">Azure Artifacts</a> .

Property	Description	Notes
	<p><i>provided and there's no limit to how many categories you may include.</i></p>	<p>Notes:</p> <ul style="list-style-type: none"> <li>- Use version &gt;=0.6.3 of the tfx-cli if you're publishing the extension programmatically.</li> <li>- If you're using <a href="#">Azure DevOps Extension Tasks extension</a> to publish, ensure that its version is &gt;= 1.2.8. You might have to approve the extension update because of recent scope changes.</li> <li>- The categories previously mentioned are natively present in Visual Studio Marketplace and Azure DevOps Server 2019 &amp; above.</li> </ul>
targets	<p><i>The products and services supported by your integration or extension.</i></p> <p>For more information, see <a href="#">installation targets</a>.</p>	<p>An array of objects, where each object has an <code>id</code> field indicating one of the following:</p> <ul style="list-style-type: none"> <li>- <code>Microsoft.VisualStudio.Services</code> (extensions that works with Azure DevOps),</li> <li>- <code>Microsoft.TeamFoundation.Server</code> (extension that works with Azure DevOps Server),</li> <li>- <code>Microsoft.VisualStudio.Services.Integration</code>,</li> <li>- <code>Microsoft.TeamFoundation.Server.Integration</code> (integrations that work with Azure DevOps Server)</li> </ul>

## Examples of required attributes

JSON

```
{
  "manifestVersion": 1,
  "id": "tools",
  "version": "0.1.0",
  "name": "Fabrikam Tools",
  "publisher": "fabrikam",
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services"
    }
  ]
}
```

## Optional attributes

### Runtime attributes

 [Expand table](#)

Property	Description	Notes
scopes	<i>An array of authorization scopes (strings) listing permissions required by your extension.</i>	For example, <code>vso.work</code> and <code>vs.code_write</code> indicates your extension needs read-only access to work items and read/write access to source code (and related resource). Scopes are presented to the user when installing your extension. For more information, see the <a href="#">full list of scopes</a> .
demands	<i>An array of demands (strings) listing the</i>	For example, <code>api-version/3.0</code> indicates that your extension uses version 3.0 APIs, and so can't run in older products that don't support this version. For more information, see the <a href="#">full list of demands</a> .

Property	Description	Notes
	<i>capabilities required by your extension.</i>	
<b>baseUri</b>	<i>(Optional) base URL for all relative URLs specified by the extension's contributions.</i>	For example: <code>https://myapp.com/{{account.name}}/</code> . This property should be left empty if your extension's contents are packaged with your extension.
<b>contributions</b>	<i>An array of contributions to the system.</i>	
<b>contributionTypes</b>	<i>An array of contribution types defined by the extension</i>	

#### JSON

```
{
  "scopes": [
    "vso.work",
    "vso.code_write",
    "vso.build_execute"
  ],
  "demands": [
    "api-version/3.0"
  ],
  "contributions": [
    {
      "id": "showCommits",
      "type": "ms.vss-web.action",
      "description": "Adds a menu action from builds grid to show associated items.",
      "targets": [
        "ms.vss-build-web.completed-build-menu"
      ],
      "properties": {
        "title": "View associated items",
        "uri": "launch.html"
      }
    }
  ]
}
```

## Discovery attributes

The following optional properties help users discover and learn about your extension:

[Expand table](#)

Property	Description	Notes
<b>description</b>	<i>A few sentences describing the extensions. Limited to 200 characters.</i>	The description should be your extension's "elevator pitch" - a couple of lines to describe your extension in the Marketplace and make people want to install it. See the example below
<b>icons</b>	<i>Dictionary of icons representing the extension.</i>	Valid keys: <code>default</code> (128x128 pixels) of type BMP, GIF, EXIF, JPG, PNG and TIFF). Other keys such as <code>large</code> (512x512 pixels) may be supported in the future. The value of each key is the path to the icon file in the extension
<b>tags</b>	<i>Array of string tags to help users find your extension.</i>	Examples: <code>agile</code> , <code>project management</code> , <code>task timer</code> , and so on.

Property	Description	Notes
screenshots	<i>Array of images that couldn't be included in your content.</i>	Screenshots are more valuable when featured in your <b>content</b> , and should be used there to help make a quality market details page for your extension. Use <b>screenshots</b> for less important images not featured in your <b>content</b> . Each image should be 1366x768 pixels. The <code>path</code> of each item is the path to the file in the extension.
content	<i>Dictionary of content files that describe your extension to users.</i>	<i>Every extension should include solid content. This is how you'll show users what your extension can do. Make it rich, consumable, and include screenshots where necessary. Include an <code>overview.md</code> file as your base content piece. Each file is assumed to be in GitHub Flavored Markdown ↗ format. The <code>path</code> of each item is the path to the Markdown file in the extension. Valid keys: <code>details</code>. Other keys may be supported in the future.</i>
links	<i>Dictionary of links that help users learn more about your extension, get support, and move.</i>	Valid keys: <code>getstarted</code> - first steps, how to setup or use. <code>learn</code> - deeper content to help users better understand your extension or service. <code>license</code> - end-user license agreement. <code>privacypolicy</code> - privacy policy for an extension. <code>support</code> - get help and support for an extension. The value of each key is an object with a <code>uri</code> field, which is the absolute URL of the link
repository	<i>Dictionary of properties describing the source code repository for the extension</i>	Valid Keys: <code>type</code> - Type of repository. Example: <code>git</code> . <code>uri</code> - Absolute URL of the repository.
badges	<i>Array of links to external metadata badges like TravisCI, Appveyor, and so on, from the <a href="#">approved badges sites</a></i>	Valid keys: <code>href</code> - Link the user navigates to when selecting the badge. <code>uri</code> - The absolute URL of the badge image to be displayed. <code>description</code> - Description of the badge, to be displayed on hover.
branding	<i>Dictionary of brand-related properties.</i>	Valid keys: <code>color</code> - primary color of the extension or publisher; can be a hex (#ff00ff), RGB (rgb(100,200,50)), or supported HTML color names (blue). <code>theme</code> - complements the color; use <code>dark</code> for dark branding colors, or <code>light</code> for lighter branding colors.

## Mark an extension public

By default, all extensions in the [Azure DevOps Marketplace](#) are private. They are hidden from public view, and are only visible to the publisher and specific accounts shared to by the publisher. If your publisher is verified, you can make your extension public by setting the `Public` flag in your extension manifest:

JSON

```
{
  "galleryFlags": [
    "Public"
  ]
}
```

Or:

JSON

```
{
  "public": true
}
```

For more information, see [Package/Publish/Install](#).

## Mark an extension to be in preview

If your extension's ready for users on the Marketplace to try, but you're still working out a few bugs or adding function, you can mark it as `preview`:

```
JSON

{
  "galleryFlags": [
    "Preview"
  ]
}
```

## Mark an extension as paid preview

If you intend to sell your extension on the Marketplace, mark it as *paid preview*. An extension marked *free* can't be changed to *paid*.

```
JSON

{
  "galleryFlags": [
    "Paid",
    "Preview"
  ]
}
```

## Mark an extension as paid

If you want to sell your extension on the Marketplace, you can mark it with the `Paid` flag and `__BYOLENFORCED` tag (starts with two underscores):

```
JSON

{
  "galleryFlags": [
    "Paid"
  ],
  "tags": [
    "__BYOLENFORCED"
  ]
}
```

Both the `Paid` flag and `__BYOLENFORCED` tag need to be present to mark an extension as paid in the Marketplace. Bring-Your-Own-License (BYOL) means the publisher of the extension provides the billing and licensing mechanism for the extension, as it isn't provided by Microsoft for Azure DevOps extensions. All paid extensions are required to define privacy policy, support policy, and an end-user license agreement. Publishers must provide content for the pricing tab in Marketplace as follows:

```
JSON

{
  "content": {
    "details": {
      "path": "overview.md"
    },
    "pricing": {
      "path": "pricing.md"
    }
}
```

```
    }  
}
```

You also need to add a new section in your extension manifest to override paid licensing. In the future, we remove the paid licensing check and no longer require the override. For now, ensure your extension displays as expected. Each override consists of an "ID" and a "behavior." Make the "ID" match the ID of the contributions defined in the manifest.

JSON

```
"licensing": {  
    "overrides": [  
        { "id": "my-hub", "behavior": " AlwaysInclude" }  
    ]  
}
```

If your paid BYOL extension offers a trial period (we recommend so), then you can specify the length of the trial in days:

JSON

```
{  
    "galleryproperties": {  
        "trialDays": "30"  
    }  
}
```

#### ① Note

If you want to target Azure DevOps, but don't wish to surface a **Download** option for your extension, then add the `_DoNotDownload` tag (starts with two underscores) to the extension manifest. If you're moving an extension from the previously offered billing & licensing from Microsoft to the BYOL model, then contact us for suitable steps.

## Example of more properties

JSON

```
{  
    "description": "Awesome tools to help you and your team do great things everyday.",  
    "icons": {  
        "default": "images/fabrikam-logo.png"  
    },  
    "categories": [  
        "Plan and track"  
    ],  
    "tags": [  
        "working",  
        "people person",  
        "search"  
    ],  
    "content": {  
        "details": {  
            "path": "overview.md"  
        },  
        "license": {  
            "path": "license-terms.md"  
        }  
    },  
    "links": {
```

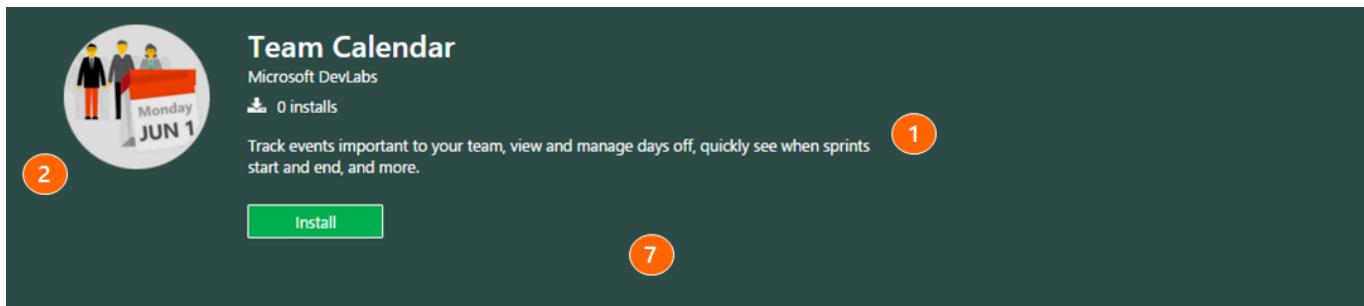
```

    "home": {
      "uri": "https://www.fabrikam-fiber-inc.com"
    },
    "getstarted": {
      "uri": "https://www.fabrikam-fiber-inc.com/help/getstarted"
    },
    "learn": {
      "uri": "https://www.fabrikam-fiber-inc.com/features"
    },
    "support": {
      "uri": "https://www.fabrikam-fiber-inc.com/support"
    },
    "repository": {
      "uri": "https://github.com/fabrikam-fiber-inc/tools"
    },
    "issues": {
      "uri": "https://github.com/fabrikam-fiber-inc/tools/issues"
    }
  },
  "repository": {
    "type": "git",
    "uri": "https://github.com/fabrikam-fiber-inc/tools"
  },
  "badges": [
    {
      "href": "https://travis.ci/fabrikam-fiber-inc/myextension",
      "uri": "https://travis.ci/fabrikam-fiber-inc/myextension.svg?branch=master",
      "description": "TravisCI build for the project"
    },
    {
      "href": "https://ci.appveyor.com/projects/fabrikam-fiber-inc/myextension",
      "uri": "https://ci.appveyor.com/api/projects/status/vlg2sgs2y7tsdxpj4c?svg=true",
      "description": "AppVeyor build for the project"
    }
  ],
  "branding": {
    "color": "rgb(34, 34, 34)",
    "theme": "dark"
  },
  "screenshots": [
    {
      "path": "screenshots/screen1.png"
    },
    {
      "path": "screenshots/screen2.png"
    }
  ]
}

```

## Details page example

- 1 - description
- 2 - icon
- 3 - categories
- 4 - screenshots
- 5 - content (details)
- 6 - links
- 7 - branding



## Stay on track

5

Team Calendar is an extension for Visual Studio Online that helps busy teams stay on track and informed about important deadlines, sprint schedules, and upcoming milestones. Team Calendar is the one place to see and manage the date important to your teams, including:

- Iterations
- Days off (for individuals or the team)
- Custom events (single or multi-day)

## Marketplace Q & A - CustomerQnASupport property

All extensions on the Visual Studio Marketplace have a Questions and Answers (Q & A) section to allow one-on-one public conversations between extension users and publishers. Publishers can choose between Marketplace Q & A, GitHub issues, or a custom Q & A URL. You can disable Q & A in the Marketplace using the `CustomerQnASupport` property in the manifest.

### Default experience (No changes to manifest are required)

- For extensions with a GitHub repository, Marketplace redirects users in the Q&A section to the associated GitHub issues.
- For extensions without a GitHub repository, Marketplace Q&A is enabled.

For a different experience than one of the default options, use the `CustomerQnASupport` property in the manifest.

```
JSON
{
  "CustomerQnASupport": {
    "enablemarketplaceqna": true,
    "url": "http://uservoice.visualstudio.com"
  }
}
```

## Properties

Properties for the Customer Q & A Support section:

- `enablemarketplaceqna` - boolean field, set to `true` for marketplace, or custom Q&A; false for disabling Q&A
- `url` - string, URL for custom Q&A

## Examples showing usage of Q & A support

### Example: Extension using custom Q & A

```
JSON

{
  "CustomerQnASupport": {
    "enablemarketplaceqna": "true",
    "url": "http://uservoice.visualstudio.com"
  }
}
```

### Example: Extension with GitHub repository but using Marketplace Q & A instead of GitHub issues

```
JSON

{
  "CustomerQnASupport": {
    "enablemarketplaceqna": "true"
  }
}
```

### Example: Extension disabling Q & A section

```
JSON

{
  "CustomerQnASupport": {
    "enablemarketplaceqna": "false"
  }
}
```

## Scopes

In your extension, you can define one or more scopes. These scopes determine which resources your extension can access and the operations permitted to perform on those resources. The scopes you specify in your extension manifest are the scopes set on access tokens issued to your extension. For more information, see [Auth and security](#).

If no scopes are specified, extensions are only provided access to user profile and extension data.

## Supported scopes

[ ] [Expand table](#)

Category	Scope	Name	High privilege	Description	Inherits from
Advanced security	vso.advsec	Advanced security (read)	Yes	Grants the ability to read alerts, result instances, and	

Category	Scope	Name	High privilege	Description	Inherits from
				analysis result instances.	
	<code>vso.advsec_write</code>	Advanced security (read and write)	Yes	Grants the ability to upload analyses in serif.	<code>vso.advsec</code>
	<code>vso.advsec_manage</code>	Advanced security (read, write, and manage)	Yes	Grants the ability to upload analyses in serif.	<code>vso.advsec_write</code>
Agent pools	<code>vso.agentpools</code>	Agent pools (read)		Grants the ability to view tasks, pools, queues, agents, and currently running or recently completed jobs for agents.	
	<code>vso.agentpools_manage</code>	Agent pools (read and manage)	Yes	Grants the ability to manage pools, queues, and agents.	<code>vso.agentpools</code>
	<code>vso.environment_manage</code>	Environment (read and manage)	Yes	Grants the ability to manage pools, queues, agents, and environments.	<code>vso.agentpools_manage</code>
Analytics	<code>vso.analytics</code>	Analytics (read)		Grants the ability to query analytics data.	
Auditing	<code>vso.auditlog</code>	Audit log (read)		Grants the ability to read the auditing log to users.	
	<code>vso.auditstreams_manage</code>	Audit streams (read)	Yes	Grants the ability to manage auditing streams to users.	<code>vso.auditlog</code>
Build	<code>vso.build</code>	Build (read)		Grants the ability to access build artifacts,	<code>vso.hooks_write</code>

Category	Scope	Name	High privilege	Description	Inherits from
				including build results, definitions, and requests, and the ability to receive notifications about build events via service hooks.	
	<code>vso.build_execute</code>	Build (read and execute)	Yes	Grants the ability to access build artifacts, including build results, definitions, and requests. Also grants the ability to queue a build, update build properties, and receive notifications about build events via service hooks.	<code>vso.build</code>
Code	<code>vso.code</code>	Code (read)		Grants the ability to read source code and metadata about commits, changesets, branches, and other version control artifacts. Also grants the ability to search code and get notified about version control events via service hooks.	<code>vso.hooks_write</code>
	<code>vso.code_write</code>	Code (read and write)	Yes	Grants the ability to read, update, and delete source code and access metadata about commits,	<code>vso.code</code>

Category	Scope	Name	High privilege	Description	Inherits from
				changesets, branches, and other version control artifacts. Also grants the ability to create and manage pull requests and code reviews and receive notifications about version control events via service hooks.	
	<code>vso.code_manage</code>	Code (read, write, and manage)	Yes	Grants the ability to read, update, and delete source code, access metadata about commits, changesets, branches, and other version control artifacts. Also grants the ability to create and manage code repositories, create and manage pull requests and code reviews, and to receive notifications about version control events via service hooks.	<code>vso.code_write</code>
	<code>vso.code_full</code>	Code (full)	Yes	Grants full access to source code, metadata about commits, changesets, branches, and other version control artifacts. Also grants the ability to	<code>vso.code_manage</code>

Category	Scope	Name	High privilege	Description	Inherits from
				create and manage code repositories, create and manage pull requests and code reviews, and receive notifications about version control events via service hooks. Also includes limited support for Client OM APIs.	
	<code>vso.code_status</code>	Code (status)		Grants the ability to read and write commit and pull-request status.	
Connected server	<code>vso.connected_server</code>	Connected server		Grants the ability to access endpoints needed from an on-premises connected server.	
Entitlements	<code>vso.entitlements</code>	Entitlements (read)		Grants read-only access to licensing entitlement endpoints to get account entitlements.	
	<code>vso.memberentitlementmanagement</code>	Member entitlement management (read)		Grants the ability to read users, their licenses, and the projects and extensions they can access.	
	<code>vso.memberentitlementmanagement_write</code>	Member entitlement management (write)	Yes	Grants the ability to manage users and their licenses and the projects	<code>vso.memberentitlementmanagement</code>

Category	Scope	Name	High privilege	Description	Inherits from
				and extensions they can access.	
Extensions	<code>vso.extension</code>	Extensions (read)		Grants the ability to read installed extensions.	<code>vso.profile</code>
	<code>vso.extension_manage</code>	Extensions (read and manage)	Yes	Grants the ability to install, uninstall, and perform other administrative actions on installed extensions.	<code>vso.extension</code>
	<code>vso.extension.data</code>	Extension data (read)		Grants the ability to read data (settings and documents) stored by installed extensions.	<code>vso.profile</code>
	<code>vso.extension.data_write</code>	Extension data (read and write)		Grants the ability to read and write data (settings and documents) stored by installed extensions.	<code>vso.extension.data</code>
GitHub connections	<code>vso.githubconnections</code>	GitHub connections (read)		Grants the ability to read GitHub connections and GitHub repositories data.	
	<code>vso.githubconnections_manage</code>	GitHub connections (read and manage)	Yes	Grants the ability to read and manage GitHub connections and GitHub repositories data.	<code>vso.githubconnections</code>
Graph and identity	<code>vso.graph</code>	Graph (read)		Grants the ability to read user, group, scope, and group	

Category	Scope	Name	High privilege	Description	Inherits from
				membership information.	
	<code>vso.graph_manage</code>	Graph (manage)	Yes	Grants the ability to read user, group, scope, and group membership information, add users and groups, and manage group memberships.	<code>vso.graph</code>
	<code>vso.identity</code>	Identity (read)		Grants the ability to read identities and groups.	
	<code>vso.identity_manage</code>	Identity (manage)	Yes	Grants the ability to read, write, and manage identities and groups.	<code>vso.identity</code>
Machine group	<code>vso.machinegroup_manage</code>	Deployment group (read, manage)	Yes	Grants the ability to manage deployment group and agent pools.	<code>vso.agentpools_manage</code>
Marketplace	<code>vso.gallery</code>	Marketplace		Grants read access to public and private items and publishers.	<code>vso.profile</code>
	<code>vso.gallery_acquire</code>	Marketplace (acquire)		Grants read access and the ability to acquire items.	<code>vso.gallery</code>
	<code>vso.gallery_publish</code>	Marketplace (publish)	Yes	Grants read access and the ability to upload, update, and share items.	<code>vso.gallery</code>
	<code>vso.gallery_manage</code>	Marketplace (manage)	Yes	Grants read access and the ability to publish and manage items and publishers.	<code>vso.gallery_publish</code>

Category	Scope	Name	High privilege	Description	Inherits from
Notifications	vso.notification	Notifications (read)		Grants read access to subscriptions and event metadata, including filterable field values.	vso.profile
	vso.notification_write	Notifications (write)		Grants read and write access to subscriptions and read access to event metadata, including filterable field values.	vso.notification
	vso.notification_manage	Notifications (manage)		Grants read, write, and management access to subscriptions and read access to event metadata, including filterable field values.	vso.notification_write
	vso.notification_diagnostics	Notifications (diagnostics)		Grants access to notification-related diagnostic logs and grants the ability to enable diagnostics for individual subscriptions.	vso.notification
Packaging	vso.packaging	Packaging (read)		Grants the ability to read feeds and packages.	vso.profile
	vso.packaging_write	Packaging (read and write)	Yes	Grants the ability to create and read feeds and packages.	vso.packaging
	vso.packaging_manage	Packaging (read, write,	Yes	Grants the ability to	vso.packaging_write

Category	Scope	Name	High privilege	Description	Inherits from
		and manage)		create, read, update, and delete feeds and packages.	
Pipeline resources	<code>vso.pipelineresources_use</code>	Pipeline resources (use)	Yes	Grants the ability to approve a pipeline's request to use a protected resource: agent pool, environment, queue, repository, secure files, service connection, and variable group.	
	<code>vso.pipelineresources_manage</code>	Pipeline resources (use and manage)	Yes	Grants the ability to manage a protected resource or a pipeline's request to use a protected resource: agent pool, environment, queue, repository, secure files, service connection, and variable group.	<code>vso.pipelineresources_use</code>
Project and team	<code>vso.project</code>	Project and team (read)		Grants the ability to read projects and teams.	
	<code>vso.project_write</code>	Project and team (read and write)		Grants the ability to read and update projects and teams.	<code>vso.project</code>
	<code>vso.project_manage</code>	Project and team (read, write, and manage)	Yes	Grants the ability to create, read, update, and delete projects and teams.	<code>vso.project_write</code>
Release	<code>vso.release</code>	Release (read)		Grants the ability to read	<code>vso.profile</code>

Category	Scope	Name	High privilege	Description	Inherits from
				release artifacts, including releases, release definitions, and release environment.	
	<code>vso.release_execute</code>	Release (read, write, and execute)	Yes	Grants the ability to read and update release artifacts, including releases, release definitions, and release environment. Also grants the ability to queue a new release.	<code>vso.release</code>
	<code>vso.release_manage</code>	Release (read, write, execute, and manage)	Yes	Grants the ability to read, update, and delete release artifacts, including releases, release definitions, and release environment. Also grants the ability to queue and approve a new release.	<code>vso.release_execute</code>
Secure files	<code>vso.securefiles_read</code>	Secure files (read)	Yes	Grants the ability to read secure files.	
	<code>vso.securefiles_write</code>	Secure files (read and create)	Yes	Grants the ability to read and create secure files.	<code>vso.securefiles_read</code>
	<code>vso.securefiles_manage</code>	Secure files (read, create, and manage)	Yes	Grants the ability to read, create, and manage secure files.	<code>vso.securefiles_write</code>
Security	<code>vso.security_manage</code>	Security (manage)	Yes	Grants the ability to read, write, and manage	

Category	Scope	Name	High privilege	Description	Inherits from
				security permissions.	
Service connections	<code>vso.serviceendpoint</code>	Service endpoints (read)	Grants the ability to read service endpoints.	<code>vso.profile</code>	
	<code>vso.serviceendpoint_query</code>	Service endpoints (read and query)	Grants the ability to read and query service endpoints.	<code>vso.serviceendpoint</code>	
	<code>vso.serviceendpoint_manage</code>	Service endpoints (read, query, and manage)	Yes	Grants the ability to read, query, and manage service endpoints.	<code>vso.serviceendpoint_query</code>
Service hooks	<code>vso.hooks</code>	Service hooks (read)	Grants the ability to read service hook subscriptions and metadata, including supported events, consumers, and actions. (No longer public.)	<code>vso.profile</code>	
	<code>vso.hooks_write</code>	Service hooks (read and write)	Grants the ability to create and update service hook subscriptions and read metadata, including supported events, consumers, and actions. (No longer public.)	<code>vso.hooks</code>	
	<code>vso.hooks_interact</code>	Service hooks (interact)	Grants the ability to interact and perform actions on events received via service hooks. (No longer public.)	<code>vso.profile</code>	

Category	Scope	Name	High privilege	Description	Inherits from
Settings	vso.settings	Settings (read)		Grants the ability to read settings.	
	vso.settings_write	Settings (read and write)		Grants the ability to read and write settings.	vso.settings
Symbols	vso.symbols	Symbols (read)		Grants the ability to read symbols.	vso.profile
	vso.symbols_write	Symbols (read and write)		Grants the ability to read and write symbols.	vso.symbols
	vso.symbols_manage	Symbols (read, write, and manage)		Grants the ability to read, write, and manage symbols.	vso.symbols_write
Task groups	vso.taskgroups_read	Task groups (read)		Grants the ability to read task groups.	
	vso.taskgroups_write	Task groups (read and create)		Grants the ability to read and create task groups.	vso.taskgroups_read
	vso.taskgroups_manage	Task groups (read, create, and manage)	Yes	Grants the ability to read, create, and manage task groups.	vso.taskgroups_write
Team dashboard	vso.dashboards	Team dashboards (read)		Grants the ability to read team dashboard information.	
	vso.dashboards_manage	Team dashboards (manage)		Grants the ability to manage team dashboard information.	vso.dashboards
Test management	vso.test	Test management (read)		Grants the ability to read test plans, cases, results, and other test management-related artifacts.	vso.profile

Category	Scope	Name	High privilege	Description	Inherits from
	<code>vso.test_write</code>	Test management (read and write)		Grants the ability to read, create, and update test plans, cases, results, and other test management-related artifacts.	<code>vso.test</code>
Threads	<code>vso.threads_full</code>	PR threads		Grants the ability to read and write to pull request comment threads.	
Tokens	<code>vso.tokens</code>	Delegated authorization tokens	Yes	Grants the ability to manage delegated authorization tokens to users.	
	<code>vso.tokenadministration</code>	Token administration	Yes	Grants the ability to manage (view and revoke) existing tokens to organization administrators.	
User profile	<code>vso.profile</code>	User profile (read)		Grants the ability to read your profile, accounts, collections, projects, teams, and other top-level organizational artifacts.	
	<code>vso.profile_write</code>	User profile (write)		Grants the ability to write to your profile.	<code>vso.profile</code>
Variable groups	<code>vso.variablegroups_read</code>	Variable groups (read)		Grants the ability to read variable groups.	
	<code>vso.variablegroups_write</code>	Variable groups (read and create)		Grants the ability to read and create	<code>vso.variablegroups_read</code>

Category	Scope	Name	High privilege	Description	Inherits from
				variable groups.	
	<code>vso.variablegroups_manage</code>	Variable groups (read, create, and manage)	Yes	Grants the ability to read, create, and manage variable groups.	<code>vso.variablegroups_write</code>
Wiki	<code>vso.wiki</code>	Wiki (read)		Grants the ability to read wikis, wiki pages, and wiki attachments. Also grants the ability to search wiki pages.	
	<code>vso.wiki_write</code>	Wiki (read and write)		Grants the ability to read, create, and update wikis, wiki pages, and wiki attachments.	<code>vso.wiki</code>
Work items	<code>vso.work</code>	Work items (read)		Grants the ability to read work items, queries, boards, area and iterations paths, and other work item tracking-related metadata. Also grants the ability to execute queries, search work items, and receive notifications about work item events via service hooks.	<code>vso.hooks_write</code>
	<code>vso.work_write</code>	Work items (read and write)		Grants the ability to read, create, and update work items and queries, update board metadata,	<code>vso.work</code>

Category	Scope	Name	High privilege	Description	Inherits from
				read area and iterations paths and other work item tracking-related metadata, execute queries, and receive notifications about work item events via service hooks.	
	<code>vso.work_full</code>	Work items (full)		Grants full access to work items, queries, backlogs, plans, and work-item tracking metadata. Also grants the ability to receive notifications about work item events via service hooks.	<code>vso.work_write</code>
User impersonation	<code>user_impersonation</code>	User impersonation	Yes	Grants full access to Visual Studio Team Services REST APIs. <i>Request or consent this scope with caution because it's very powerful.</i>	

## Changing scope of published extension

You can change the scope of a published extension. If you previously installed your extension (and authorized the previous set of scopes), authorize the new scopes before you can upgrade to the newest version.

The **Action Required** section of the Extension settings hub shows a user that, if any, installed extensions require authorization:

## Extensions

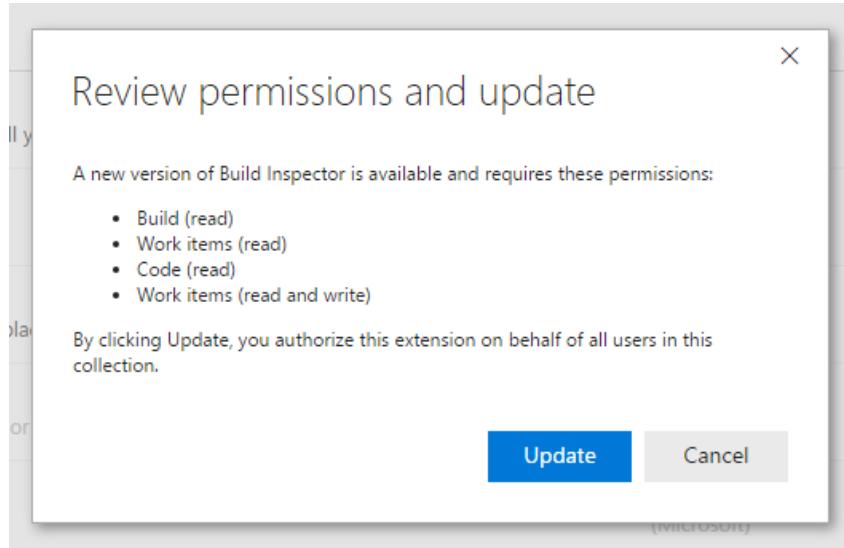
[Manage](#)   [Requested](#)

Action Required

 **Build Inspector**  
A simple extension that demonstrates how to show details for a Visual Studio...

⚠ Update available. Requires additional permissions.

An administrator can then review and authorize the new set of scopes:



## Installation targets

As the name implies, installation targets define the products and services where you can install your extension. `Microsoft.VisualStudio.Services` is the most common installation target and indicates that the extension can be installed into Azure DevOps.

The installation targets for an extension or integration are specified via the `targets` field in the manifest.

Supported identifiers for **extensions**:

- `Microsoft.VisualStudio.Services.Cloud`: installs into Azure DevOps Services
- `Microsoft.TeamFoundation.Server`: installs into Azure DevOps Server
- `Microsoft.VisualStudio.Services`: installs into both. Shortcut for `Microsoft.VisualStudio.Services.Cloud` and `Microsoft.TeamFoundation.Server` version [14.2,)

Supported identifiers for **integrations**:

- `Microsoft.VisualStudio.Services.Cloud.Integration`: integrates with Azure DevOps Services
- `Microsoft.TeamFoundation.Server.Integration`: integrates with Azure DevOps Server
- `Microsoft.VisualStudio.Services.Integration`: integrates with both. Shortcut for `Microsoft.VisualStudio.Services.Cloud.Integration` and `Microsoft.TeamFoundation.Server.Integration`

For more information, see [Extensibility points](#).

## Examples of installation targets

## Example: Extension that works with Azure DevOps

JSON

```
{  
  "targets": [  
    {  
      "id": "Microsoft.VisualStudio.Services"  
    }  
  ]  
}
```

## Example: Extension that works only with Azure DevOps Services

JSON

```
{  
  "targets": [  
    {  
      "id": "Microsoft.VisualStudio.Services.Cloud"  
    }  
  ]  
}
```

Installation targets can also be used in the manifest of integrations. For example, products, apps, or tools that work with, but don't install into Azure DevOps.

## Example: Integration that works with Azure DevOps

JSON

```
{  
  "targets": [  
    {  
      "id": "Microsoft.VisualStudio.Services.Integration"  
    }  
  ]  
}
```

## Example: Integration that only works with Azure DevOps Server

JSON

```
{  
  "targets": [  
    {  
      "id": "Microsoft.TeamFoundation.Server.Integration"  
    }  
  ]  
}
```

## Installation target versions

Some installation target identifiers, like `Microsoft.TeamFoundation.Server` and `Microsoft.TeamFoundation.Server.Integration`, support an optional version range. This optional version range further clarifies the supported releases the extension or integration is supported on.

The version or version range is specified via the `version` field on the installation target object. This value can be either:

- A specific version, for example: `15.0` (2017 RTM only)
- A range of supported versions, for example: `[14.0)` (2015 RTM and later), `[14.3,15.1]` (2015 Update 3 through 2017 Update 1). Range values are refined using:
  - `[`: minimum version inclusive
  - `]`: maximum version inclusive
  - `(`: minimum version exclusive
  - `)`: maximum version exclusive

Version numbers for Azure DevOps Server:

 Expand table

Release	Releases	Version
2010	All releases	10.0
2012	All releases	11.0
2013	RTM and updates	12.0, 12.1, 12.2, 12.3, 12.4
2015	RTM and updates	14.0, 14.1, 14.2, 14.3
2017	RTM and updates	15.0, 15.1
2018	RTM and updates	16.0
2019	RTM and updates	17.0
2020	RTM and updates	18.0

## Examples showing versions

### Example: Extension that works with Azure DevOps

JSON

```
{  
  "targets": [  
    {  
      "id": "Microsoft.VisualStudio.Services.Cloud"  
    },  
    {  
      "id": "Microsoft.TeamFoundation.Server",  
      "version": "[15.0,)"  
    }  
  ]  
}
```

## Shortcuts

`Microsoft.VisualStudio.Services` is a shortcut for Azure DevOps.

JSON

```
{  
  "targets": [  
    {  
      "id": "Microsoft.VisualStudio.Services"  
    }  
  ]  
}
```

```
{
    "id": "Microsoft.VisualStudio.Services"
}
]
```

is equivalent to:

JSON

```
{
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services.Cloud"
    },
    {
      "id": "Microsoft.TeamFoundation.Server",
      "version": "[14.2,)"
    }
  ]
}
```

## Using installation targets and demands

Installation targets and demands are used together to present users with a correct view of the products and services your extension or integration is compatible with. For example, specifying an installation target of `Microsoft.VisualStudio.Services` with a demand of `api-version/3.0` means the extension works with Azure DevOps.



For more information on REST APIs, see the [REST API Reference](#).

## Example: Extension that uses version 3.0 APIs

JSON

```
{
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services"
    }
  ],
  "demands": [
    "api-version/3.0"
  ]
}
```

Resolves to the following installation targets:

1. `Microsoft.VisualStudio.Services.Cloud`
2. `Microsoft.TeamFoundation.Server`, version: `[15.0,)`

## Example: Integration that uses version 2.0 APIs

JSON

```
{
  "targets": [
    {
      "id": "Microsoft.VisualStudio.Services.Integration"
    }
  ],
  "demands": [
    "api-version/2.0"
  ]
}
```

Resolves to the following installation targets:

1. Microsoft.VisualStudio.Services.Cloud.Integration
2. Microsoft.TeamFoundation.Server.Integration, version: [14.0,)

## Demands

Demands let you specify capabilities and other features required by your extension. You can use these demands to limit where your extension can be published or installed.

Demands get used by the Visual Studio Marketplace to list the products and environments your extension is compatible with, which helps customers understand whether your extension works with their version of Azure DevOps, for example.

See the following example of how demands get specified in the extension manifest.

JSON

```
{
  "demands": [
    "api-version/3.0",
    "contribution/ms.vss-dashboards-web.widget-catalog"
  ]
}
```

In this example, the extension demands version 3.0 of the APIs, which means it can only be installed to Azure DevOps. It also requires the `ms.vss-dashboards-web` extension (and its `widget-catalog` contribution) to be installed (and enabled) in the collection before your extension can be installed.

## Supported demands

[ ] Expand table

Type	Description	Checked at publish?	Checked at install?
<code>environment/cloud</code>	Requires running in a cloud environment	Yes	Yes
<code>environment/onprem</code>	Requires running in an on-premises environment	Yes	Yes
<code>api-version/{version}</code>	Requires a specific API version (minimum)	No	Yes
<code>extension/{id}</code>	Requires a specific extension be installed/enabled	No	Yes
<code>contribution/{id}</code>	Requires a specific contribution be available	No	Yes
<code>contributionType/{id}</code>	Requires a specific contribution type be available	No	Yes

### ⓘ Note

- Use `environment/cloud` and `environment/onprem` only when your extension has topology-related requirements that require running in that particular environment.
- `extension`, `contribution`, and `contributionType` demands are evaluated at install time and require that the specified extension is already installed and enabled in the organization or collection.

## Files

The `files` section is where you reference any files you wish to include in your extension. You can add both folders and individual files:

JSON

```
{  
  "files": [  
    {  
      "path": "hello-world.html", "addressable": true  
    },  
    {  
      "path": "scripts", "addressable": true  
    },  
    {  
      "path": "images/logo.png", "addressable": true, "packagePath": "/"  
    }  
  ]  
}
```

## Properties

Properties for the Files section:

- **path** - Path to resource on disk, which can be relative to your root directory.
- **addressable** – (optional) Set to `true` if you want your file to be URL-addressable. Defaults to `false`.
- **packagePath** – (optional) Path to the resource within the package. Defaults to the relative path on disk from your root directory.
- **contentType** – (optional) MIME type of the file. Defaults to a best guess based on the file extension and OS settings.
- **assetType** – (optional) Specify the value of the Type attribute of the asset entry in the VSIX manifest. Can also be an array of strings, in which case multiple asset entries get added for this file. Defaults to the packagePath.
- **lang** – (optional) Language of this asset. Localized files are served based on the Accept-Language header. Leave blank to signify this file is in the default (or fallback) language. Localized versions of the same file should have the same assetType.

## Contributions

Each contribution entry has the following properties:

- **id** - A reference ID (string) for the contribution. Make each contribution ID unique within an extension. See [referencing contributions and types](#).
- **type** - The ID of the contributionType of this contribution.
- **description** - (Optional) A string describing what the contribution is providing.

- **targets** - An array of contribution IDs that the contribution is targeting (contributing to). See [Targeting contributions](#).
  - **properties** - (Optional) An object that includes properties for the contribution as defined in the contribution type.

For more information, see the [contribution model overview](#).

## Contribution types

Each contribution entry has the following properties:

- **id** - A reference ID (string) for the contribution type. Make each contribution type ID unique within an extension. See [referencing contributions and types](#).
  - **name** - The friendly name of the contribution type.
  - **description** - (Optional) A string describing in more detail what the contribution type is for.
  - **properties** - (Optional) A dictionary that maps property names to property descriptions. These properties describe the required and optional properties that contributions of this type can use.

Property descriptions have the following properties:

- **description** - (Optional) A string describing what the property is used for.
  - **required** - (Optional) A boolean value, which if true indicates that the property is required for all contributions of this type.
  - **type** - The type of value that the property can have, which could be string, uri, guid, boolean, integer, double, dateTime, array, or object.

For more information, see the [contribution model overview](#).

## Referencing contributions and types

Use unique identifiers to reference contributions and contribution types. Reference types with the `type` property, and reference other contributions with the `targets` property.

- A *full* contribution reference includes the publisher identifier, extension identifier, and contribution/type identifier, separated by a dot (.). For example, `ms.vss-web.hub` is the full identifier for the contribution with identifier of `hub` in the `vss-web` extension published by the "ms" (Microsoft) publisher.
  - *Relative* contribution references might get used within an extension manifest for a contribution's reference to another contribution or contribution type within that same extension. In this case, the publisher and extension identifiers are NOT included, and the identifier is a dot (.) followed by the contribution identifier. For example, `.hub` might be used within the `vss-web` extension mentioned previously as a shortcut for `ms.vss-web.hub`.

# Targeting contributions

Some contributions act as containers targeted by other contributions.

- Hub contributions can target Hub Groups. When a page is rendered, the web UI shows all Hub contributions that target the selected hub group. Hub groups target a hub group collection, which defines a set of hub groups that show up in a given navigational area, for example, project-level admin pages.
  - Different types of contributions can target menus: action, hyperlink-action, and action-provider. Actions and hyperlink-actions provide single menu item entries. An action-provider can provide multiple dynamic menu items. For a given menu, items are aggregated across all contributions (of any of these types) that target that specific menu contribution.

## Adding a hub icon

For information on adding an icon to your hub, check out the [hub icon guidance](#).

## Supported badge services

The Marketplace only supports badges from the following trusted services:

- api.travis-ci.org/
- badge.fury.io/
- badges.frapsoft.com/
- badges.gitter.im/
- badges.greenkeeper.io/
- cdn.travis-ci.org/
- ci.appveyor.com/
- codeclimate.com/
- codecov.io/
- coveralls.io/
- david-dm.org/
- gemnasium.com/
- img.shields.io/
- isitmaintained.com/
- marketplace.visualstudio.com/
- snyk.io/
- travis-ci.com/
- travis-ci.org/
- vsmarketplacebadges.dev/
- bithound.io/
- deepscan.io/
- githost.io/
- gitlab.com/
- opencollective.co/

ⓘ Note

Replace `vsmarketplacebadge.apphb.com` with `vsmarketplacebadges.dev`.

To show a badge from another service, contact [Customer Support at the Developer Community](#).

## Example manifest

The following extension contributes an action to the completed builds context menu and a hub to the Build hub group:

JSON

```
{  
  "manifestVersion": 1,  
  "id": "tools",  
  "version": "0.1.0",  
  "name": "Fabrikam Tools",  
  "publisher": "fabrikam",  
  "description": "Awesome tools to help you and your team do great things everyday.",  
  "targets": [  
    {  
      "group": "Build",  
      "label": "Fabrikam Tools",  
      "uri": "https://vsmarketplacebadges.dev/badge/extensionId/tools",  
      "icon": "https://vsmarketplacebadges.dev/icon/extensionId/tools",  
      "type": "hub"  
    }  
  ]  
}
```

```
{
    "id": "Microsoft.VisualStudio.Services"
}
],
"demands": [
    "api-version/3.0"
],
"icons": {
    "default": "images/fabrikam-logo.png"
},
"scopes": [
    "vso.work",
    "vso.code_write"
],
"categories": [
    "Plan and track"
],
"tags": [
    "working",
    "people person",
    "search"
],
"branding": {
    "color": "rgb(34, 34, 34)",
    "theme": "dark"
},
"screenshots": [
    {
        "path": "screenshots/screen1.png"
    },
    {
        "path": "screenshots/screen2.png"
    }
],
"content": {
    "details": {
        "path": "overview.md"
    },
    "license": {
        "path": "eula.md"
    }
},
"links": {
    "home": {
        "uri": "https://www.fabrikam-fiber-inc.com"
    },
    "getstarted": {
        "uri": "https://www.fabrikam-fiber-inc.com/help/getstarted"
    },
    "learn": {
        "uri": "https://www.fabrikam-fiber-inc.com/features"
    },
    "support": {
        "uri": "https://www.fabrikam-fiber-inc.com/support"
    },
    "repository": {
        "uri": "https://github.com/fabrikam-fiber-inc/tools"
    },
    "issues": {
        "uri": "https://github.com/fabrikam-fiber-inc/tools/issues"
    }
},
"repository": {
    "type": "git",
    "uri": "https://github.com/fabrikam-fiber-inc/myextension"
},
"badges": [
    {
        "href": "https://travis.ci/fabrikam-fiber-inc/myextension",
        "label": "Build Status"
    }
]
```

```
        "uri": "https://travis.ci/fabrikam-fiber-inc/myextension.svg?branch=master",
        "description": "TravisCI build for the project"
    },
    {
        "href": "https://ci.appveyor.com/projects/fabrikam-fiber-inc/myextension",
        "uri": "https://ci.appveyor.com/api/projects/status/vlg2sgs2y7tsdxpj4c?svg=true",
        "description": "AppVeyor build for the project"
    }
],
"contributions": [
{
    "id": "showCommits",
    "type": "ms.vss-web.action",
    "description": "Adds a menu action from builds grid to show associated items.",
    "targets": [
        "ms.vss-build-web.completed-build-menu"
    ],
    "properties": {
        "title": "View associated items",
        "uri": "launch.html"
    }
}
]
}
```

# Integrate custom build pipeline tasks with extensions

Article • 04/14/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Use this reference when you want to create and integrate custom build pipeline tasks with extensions in Azure DevOps.

## Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Custom tasks

Tasks are the building blocks for defining automation in a build or release pipeline. To learn more, see [Task types & usage](#).

We offer many [built-in tasks](#) to enable fundamental build and deployment scenarios. We recommend that you review these scenarios before building your own. If the existing tasks don't satisfy your needs, you can build a [custom task](#).

## Custom build task JSON

When you create a custom build or release task with an extension, your extension includes a *task.json* file for each build or release task. The *task.json* file describes the build or release task. The build system uses it to render configuration options to the user and to know which scripts to execute at build time.

To learn more, see the [build and release task SDK documentation](#) on GitHub. Specifically, you might be interested in the [task.json schema](#).

## Bundle multiple versions of build or release tasks within one extension

You can include multiple versions of a build or release task within your extension. Roll out future versions of your extension without interrupting service of users on older versions. The following information shows the layout for having multiple versions in one extension.

## Traditional extension layout

- extensionManifest.json
- extensionIcon.png
- Task1
  - task.json
  - icon.png
  - taskScript.ps1

### ⚠ Note

When you're setting up a task icon, ensure the following is true.

- The icon name is *icon.png*
- The icon size is 32x32 pixels
- The icon is in the same location as the *task.json* file

## Multiple versions layout

### ⚠ Note

The code looks for the *task.json* file inside the task folder and then one level deeper. If one isn't found in either level, you see an error message.

- extensionManifest.json
- extensionIcon.png
- Task1
  - Task1V1
    - task.json
    - icon.png
    - taskScript.ps1
  - Task1V2
    - task.json
    - icon.png
    - taskScript.ps1
- Task2
  - Task2V1
    - task.json
    - icon.png

- taskScript.ps1
- Task2V2
  - task.json
  - icon.png
  - taskScript.ps1

 Tip

To ensure the `_build/Tasks/ssrsfilesdeploy` folder contains the V1 and V2 contents, set `matchCopy(item, srcPath, destPath, { noRecurse: true })` to `false` in the `make-util.js` file.

## Related content

- [Add a custom pipelines task extension](#)
- [Server Task Authoring ↗](#)
- [Build and release task examples ↗](#)

# Pipeline decorator expression context

Article • 11/08/2024

## Azure DevOps Services

Pipeline decorators have access to context about the pipeline in which they run. As a pipeline decorator author, you can use this context to make decisions about the decorator's behavior. The information available in context is different for pipelines and for release. Also, decorators run after task names are resolved to task globally unique identifiers (GUIDs). When your decorator wants to reference a task, it should use the GUID rather than the name or keyword.

### 💡 Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

### ⓘ Note

Pipeline decorators are used when building [web extensions](#). These examples are not designed to work in YAML pipelines.

## Resources

Pipeline resources are available on the `resources` object.

## Repositories

Currently, there's only one key: `repositories`. `repositories` is a map from repo ID to information about the repository.

In a designer build, the primary repo alias is `__designer_repo`. In a YAML pipeline, the primary repo is called `self`. In a release pipeline, repositories aren't available. [Release artifact variables](#) are available.

For example, to print the name of the `self` repo in a YAML pipeline:

```
steps:
- script: echo ${resources.repositories['self'].name}
```

Repositories contain these properties:

JavaScript

```
resources['repositories']['self'] =
{
    "alias": "self",
    "id": "<repo guid>",
    "type": "Git",
    "version": "<commit hash>",
    "name": "<repo name>",
    "project": "<project guid>",
    "defaultBranch": "<default ref of repo, like 'refs/heads/main'>",
    "ref": "<current pipeline ref, like 'refs/heads/topic'>",
    "versionInfo": {
        "author": "<author of tip commit>",
        "message": "<commit message of tip commit>"
    },
    "checkoutOptions": {}
}
```

## Job

Job details are available on the `job` object.

The data looks similar to:

JavaScript

```
job =
{
    "steps": [
        {
            "environment": null,
            "inputs": {
                "script": "echo hi"
            },
            "type": "Task",
            "task": {
                "id": "d9bafed4-0b18-4f58-968d-86655b4d2ce9",
                "name": "CmdLine",
                "version": "2.146.1"
            },
            "condition": null,
            "continueOnError": false,
            "timeoutInMinutes": 0,
            "script": "echo hi"
        }
    ]
}
```

```
        "id": "5c09f0b5-9bc3-401f-8cfb-09c716403f48",
        "name": "CmdLine",
        "displayName": "CmdLine",
        "enabled": true
    }
]
}
```

For instance, to conditionally add a task only if it doesn't already exist:

YAML

```
- ${{ if not(containsValue(job.steps.*.task.id, 'f3ab91e7-bed6-436a-b651-
399a66fe6c2a')) }}:
  - script: echo conditionally inserted
```

## Variables

[Pipeline variables](#) are also available.

For instance, if the pipeline had a variable called `myVar`, its value would be available to the decorator as `variables['myVar']`.

For example, to give a decorator an opt-out, we could look for a variable. Pipeline authors who wish to opt out of the decorator can set this variable, and the decorator isn't injected. If the variable isn't present, then the decorator is injected as usual.

### my-decorator.yml

YAML

```
- ${{ if ne(variables['skipInjecting'], 'true') }}:
  - script: echo Injected the decorator
```

Then, in a pipeline in the organization, the author can request the decorator not to inject itself.

### pipeline-with-opt-out.yml

YAML

```
variables:
  skipInjecting: true
```

```
steps:  
- script: echo This is the only step. No decorator is added.
```

## Task names and GUIDs

Decorators run after tasks already turned into GUIDs. Consider the following YAML:

YAML

```
steps:  
- checkout: self  
- bash: echo This is the Bash task  
- task: PowerShell@2  
  inputs:  
    targetType: inline  
    script: Write-Host This is the PowerShell task
```

Each of those steps maps to a task. Each task has a unique GUID. Task names and keywords map to task GUIDs before decorators run. If a decorator wants to check for the existence of another task, it must search by task GUID rather than by name or keyword.

For normal tasks (which you specify with the `task` keyword), you can look at the task's `task.json` to determine its GUID. For special keywords like `checkout` and `bash` in the previous example, you can use the following GUIDs:

[Expand table](#)

Keyword	GUID	Task Name
checkout	6D15AF64-176C-496D-B583-FD2AE21D4DF4	n/a, see note
bash	6C731C3C-3C68-459A-A5C9-BDE6E6595B5B	Bash
script	D9BAFED4-0B18-4F58-968D-86655B4D2CE9	CmdLine
powershell	E213FF0F-5D5C-4791-802D-52EA3E7BE1F1	PowerShell
pwsh	E213FF0F-5D5C-4791-802D-52EA3E7BE1F1	PowerShell
publish	ECDC45F6-832D-4AD9-B52B-EE49E94659BE	PublishPipelineArtifact
download	30f35852-3f7e-4c0c-9a88-e127b4f97211	DownloadPipelineArtifact

After task names and keywords resolve, the previous YAML becomes:

## YAML

```
steps:
- task: 6D15AF64-176C-496D-B583-FD2AE21D4DF4@1
  inputs:
    repository: self
- task: 6C731C3C-3C68-459A-A5C9-BDE6E6595B5B@3
  inputs:
    targetType: inline
    script: echo This is the Bash task
- task: E213FF0F-5D5C-4791-802D-52EA3E7BE1F1@2
  inputs:
    targetType: inline
    script: Write-Host This is the PowerShell task
```

### 💡 Tip

You can find each of these GUIDs in the `task.json` for the corresponding [in-box task](#). The only exception is `checkout`, which is a native capability of the agent. Its GUID is built into the Azure Pipelines service and agent.

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

# Service endpoint authentication schemes

Article • 01/09/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Learn how to set the credentials in the HTTP request header when you're calling the external endpoint. Azure DevOps can then connect to the external service using the credentials. Azure DevOps supports a closed set of authentication schemes utilized by a custom service endpoint type. Azure DevOps interprets the authentication scheme that's used in any custom endpoint & support connection to the external service.

See the following authentication schemes that are part of the closed set.

## Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Basic authentication

As a security measure, we recommend using service principals & managed identities over basic authentication. For more information, see [Use service principals & managed identities](#).

## Token-based authentication

This scheme takes one input - API Token (confidential)

Default authentication header used is: {{endpoint.apitoken}}

JSON

```
{  
  "id": "endpoint-auth-scheme-token",  
  "description": "i18n:Token based endpoint authentication scheme",  
  "type": "ms.vss-endpoint.service-endpoint-type",  
  "targets": [  
    "ms.vss-endpoint.endpoint-types"  
  ],  
  "properties": {
```

```

    "name": "Token",
    "displayName": "i18n:Token Based Authentication",
    "authenticationSchemes": [
        {
            "type": "ms.vss-endpoint.endpoint-auth-scheme-token",
            "headers": [
                {
                    "name": "Authorization",
                    "value": "{{endpoint.apitoken}}"
                }
            ],
            "inputDescriptors": [
                {
                    "id": "apitoken",
                    "name": "i18n:API Token",
                    "description": "i18n:API Token for connection to endpoint",
                    "inputMode": "textbox",
                    "isConfidential": true,
                    "validation": {
                        "isRequired": true,
                        "dataType": "string",
                        "maxLength": 300
                    }
                }
            ]
        }
    ]
}

```

## Certificate-based authentication

This scheme takes one input - Certificate (confidential)

The value of certificate has to be provided in the text area.

JSON

```
{
    "id": "endpoint-auth-scheme-cert",
    "description": "i18n:Creates a certificate-based endpoint authentication scheme",
    "type": "ms.vss-endpoint.service-endpoint-type",
    "targets": [
        "ms.vss-endpoint.endpoint-types"
    ],
    "properties": {
        "name": "Certificate",
        "displayName": "i18n:Certificate Based",
        "authenticationSchemes": [

```

```
{  
    "type": "ms.vss-endpoint.endpoint-auth-scheme-cert",  
    "inputDescriptors": [  
        {  
            "id": "certificate",  
            "name": "i18n:Certificate",  
            "description": "Content of the certificate",  
            "inputMode": "TextArea",  
            "isConfidential": true,  
            "validation": {  
                "isRequired": true,  
                "dataType": "string"  
            }  
        }  
    ]  
}  
}
```

## No authentication

This scheme is used when an endpoint type doesn't require to take any input. For example, external services that support anonymous access to its resources.

JSON

```
{  
    "id": "endpoint-auth-scheme-none",  
    "description": "i18n:Creates an endpoint authentication scheme with no authentication.",  
    "type": "ms.vss-endpoint.endpoint-auth-scheme-none",  
    "targets": [  
        "ms.vss-endpoint.endpoint-auth-schemes"  
    ],  
    "properties": {  
        "name": "None",  
        "displayName": "i18n:No Authentication"  
    }  
}
```

## Feedback

Was this page helpful?

 Yes

 No

Provide product feedback ↗

# Static content hosting

Article • 10/04/2022

## Azure DevOps Services

Choose to host static content for your extension, like HTML, CSS, and JavaScript files, on your own service, on a third-party hosting service, like Azure or Heroku, or on Azure DevOps Services directly.

### ⓘ Important

If your extension needs to create a custom table in the TFS database, do not create it using the 'dbo' schema. Instead, custom tables should be created in a separate schema. For example, 'YourExtensionName'.

### ⓘ Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Host on Azure DevOps Services

In this model, static content is packaged with your extension's .vsix file and is served from a public endpoint at <https://publisher.gallerycdn.vsassets.io>.

Your extension's static content is useful when you're enhancing or decorating data from Azure DevOps Services. The extension pub doesn't require you (the extension publisher) to set up, manage, or pay for hosting services for your extension

## Steps

1. In your extension manifest file, specify the files you want to include via the `files` property:

JSON

```
{  
  "files": [  
    {  
      "path": "scripts", "addressable": true  
    }  
  ]  
}
```

```
    },
    {
      "path": "images/extr/icon1.png", "addressable": true
    }
  ]
}
```

2. Remove the `baseUri` property (if set) from your extension manifest.
3. Package your extension ([steps](#))
4. Publish (or republish) your extension ([steps](#))

> [!IMPORTANT] > Make sure to increment the version of your extension when you make changes to static content files included in your .vsix.

Keep in mind:

- The value specified by the `path` attribute can be a folder or individual file. If a folder, the entire folder (and any subfolders) is included.
- The `addressable` attribute is important and is what tells Visual Studio Codespaces to make the file(s) URL addressable.
- All `addressable` asset requests are case-sensitive. If the request for an asset has a different case than the actual uploaded asset, it results in an HTTP 404 (Not found) error.
- Not specifying a `baseUri` or setting an empty value tells Visual Studio Codespaces at runtime to calculate the base URI as if your static content's hosted by Azure DevOps Services.

## Host on your own service (or a third-party service)

In this model, static content is served from your own service and not included in your extension's .vsix file.

### Steps

1. Set the `baseUri` property in your extension manifest For example, assuming a value of `https://myservice.net/extension` and this hub contribution:

JSON

```
"baseUri": "https://myservice.net/extension",
"contributions": [
```

```
{  
  "id": "Fabrikam.HelloWorld",  
  "type": "ms.vss-web.hub",  
  "targets": [  
    "ms.vss-work-web.work-hub-group"  
  ],  
  "properties": {  
    "name": "Hello",  
    "uri": "hello-world.html"  
  }  
}  
]
```

Azure DevOps Services loads the contents of this hub when it's rendered at

<https://myservice.net/extension/hello-world.html>.

## Next steps

[Package, publish, and install extensions](#)

# Create modal dialogs in extensions

07/02/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Modal dialogs provide a powerful way to create focused user experiences in Azure DevOps extensions. The `HostDialogService` lets you present a modal dialog that blocks user interaction with the entire Azure DevOps interface until the dialog gets dismissed. This action ensures that users complete important tasks or provide required information.

Use modal dialogs in your extensions to:

- Collect user input through forms
- Display confirmation messages for critical actions
- Show detailed information that requires user attention
- Guide users through multi-step processes

## Important

When you create modal dialogs with `HostDialogService`, they block interaction with the entire Azure DevOps page, not just your extension. This approach provides a true modal experience but you should use it thoughtfully to avoid disrupting the user workflow.

## Prerequisites

Before you can create modal dialogs in your Azure DevOps extension, ensure you have the following:

 Expand table

Category	Requirement	Details
Extension setup	Working extension project	A valid <code>vss-extension.json</code> manifest file
	Marketplace registration	Extension registered with Visual Studio Marketplace for testing and deployment
	Development knowledge	Understanding of <a href="#">Azure DevOps extension development basics</a>
Development environment	Node.js and npm	Node.js version 14 or later with npm installed

Category	Requirement	Details
	Code editor	Visual Studio Code or other IDE recommended
	Azure DevOps access	Access to an Azure DevOps organization for testing
Required packages	Extension SDK	Install: <code>npm install azure-devops-extension-sdk</code>
	Extension API	Install: <code>npm install azure-devops-extension-api</code>
Extension permissions	Manifest scopes	Include appropriate scopes in <code>vss-extension.json</code> , for example: <code>"vso.work", "vso.project"</code>
SDK imports	Required modules	Import SDK and services: <code>import * as SDK from "azure-devops-extension-sdk"</code> and <code>import { CommonServiceIds, IHostDialogService } from "azure-devops-extension-api"</code>

## Dialog contents

To start, declare a contribution of type `ms.vss-web.control` in your extension manifest. This contribution represents the content displayed within the dialog.

### JSON

```
{
  "id": "registration-form",
  "type": "ms.vss-web.control",
  "description": "The content to be displayed in the dialog",
  "targets": [],
  "properties": {
    "uri": "registration-form.html"
  }
}
```

The `uri` property references a page that is rendered within the content area of the dialog:

### HTML

```
<!DOCTYPE html>
<html>
  <head>
    <script src="node_modules/azure-devops-extension-sdk/lib/SDK.js"></script>
  </head>
  <body>
    <h2 id="header">Register now</h2>
    <p>
      <label>Name:</label>
    </p>
  </body>
</html>
```

```
<input id="inpName" />
</p>
<p>
    <label>Date of birth:</label>
    <input id="inpDob" />
</p>
<p>
    <label>Email address:</label>
    <input id="inpEmail" />
</p>
<script type="module">
    import * as SDK from "azure-devops-extension-sdk";

    SDK.init();
    const registrationForm = (function() {
        const callbacks = [];

        function inputChanged() {
            // Execute registered callbacks
            for(let i = 0; i < callbacks.length; i++) {
                callbacks[i](isValid());
            }
        }

        function isValid() {
            // Check whether form is valid or not
            return !(name.value) && !(dateOfBirth.value) && !!(email.value);
        }

        function getFormData() {
            // Get form values
            return {
                name: name.value,
                dateOfBirth: dateOfBirth.value,
                email: email.value
            };
        }

        const name = document.getElementById("inpName");
        const dateOfBirth = document.getElementById("inpDob");
        const email = document.getElementById("inpEmail");

        name.addEventListener("change", inputChanged);
        dateOfBirth.addEventListener("change", inputChanged);
        email.addEventListener("change", inputChanged);

        return {
            isFormValid: function() {
                return isValid();
            },
            getFormData: function() {
                return getFormData();
            },
            attachFormChanged: function(cb) {

```

```

        callbacks.push(cb);
    }
};

))();

// Register form object to be used across this extension
SDK.register("registration-form", registrationForm);
</script>
</body>
</html>

```

## Show the dialog

To show the dialog (for example, when a user selects an action on a toolbar or menu), call the `openDialog` function on an instance of the `IHostDialogService`, passing the fully qualified identifier of the dialog content, for example `my-publisher.my-extension.registration-form` and any dialog options:

JavaScript

```

import * as SDK from "azure-devops-extension-sdk";

SDK.getService<IHostDialogService>
(CommonServiceIds.HostDialogService).then((dialogService) => {
    const extensionCtx = SDK.getExtensionContext();
    // Build absolute contribution ID for dialogContent
    const contributionId =
` ${extensionCtx.publisherId}.${extensionCtx.extensionId}.registration-form`;

    // Show dialog
    const dialogOptions = {
        title: "My Dialog",
        width: 800,
        height: 600
    };

    dialogService.openDialog(contributionId, dialogOptions);
});

```

## Advanced dialog features

A function can be called when the OK button is selected. This function is specified by `getDialogResult` in the options you provide when showing the dialog.

If a call to `getDialogResult` returns a non-null value, this value is then passed to the function specified by `okCallback` (also in the options) and the dialog is closed.

In this example, the `attachFormChanged` callback gets called when inputs on the form change. Based on whether the form is valid or not, the OK button is enabled or disabled.

JavaScript

```
import * as SDK from "azure-devops-extension-sdk";
import { CommonServiceIds, IHostDialogService } from "azure-devops-extension-api";

SDK.getService<IHostDialogService>
(CommonServiceIds.HostDialogService).then((dialogService) => {
    let registrationForm: any;
    const extensionCtx = SDK.getExtensionContext();
    const contributionId =
` ${extensionCtx.publisherId}.${extensionCtx.extensionId}.registration-form`;

    const dialogOptions = {
        title: "Registration Form",
        width: 800,
        height: 600,
        getDialogResult: () => {
            // Get the result from registrationForm object
            return registrationForm ? registrationForm.getFormData() : null;
        },
        okCallback: (result: any) => {
            // Log the result to the console
            console.log(JSON.stringify(result));
        }
    };
    dialogService.openDialog(contributionId, dialogOptions).then((dialog) => {
        // Get registrationForm instance which is registered in
registrationFormContent.html
        dialog.getContributionInstance("registration-
form").then((registrationFormInstance) => {

            // Keep a reference of registration form instance (to be used
previously in dialog options)
            registrationForm = registrationFormInstance;

            // Subscribe to form input changes and update the Ok enabled state
            registrationForm.attachFormChanged((isValid: boolean) => {
                dialog.updateOkButton(isValid);
            });

            // Set the initial ok enabled state
            registrationForm.isFormValid().then((isValid: boolean) => {
                dialog.updateOkButton(isValid);
            });
        });
    });
});
```

# Control the OK button

Initially, the OK button is disabled. However, you can enable/disable this button by calling the `updateOkButton` method on the dialog:

JavaScript

```
dialogService.openDialog(contributionId, dialogOptions).then((dialog) => {
    // Set true/false to enable/disable ok button
    dialog.updateOkButton(true);
});
```

# Pass values to the dialog

It's possible to pass initial values to dialog content when it is opened in the host dialog.

JSON

```
{
    "id": "registration-form",
    "type": "ms.vss-web.control",
    "description": "The content displayed in the dialog",
    "targets": [],
    "properties": {
        "uri": "registration-form.html?id={{myId}}"
    }
}
```

When the dialog is opened, following options need to be specified to pass `myId`:

JavaScript

```
const dialogOptions = {
    title: "My Dialog Title",
    width: 800,
    height: 600,
    urlReplacementObject: { myId: new Date().getTime() }
};
```

# Customize dialog buttons

The `okText` and `cancelText` attributes can be used to specify alternate titles for the OK and Cancel buttons:

```
JavaScript
```

```
const dialogOptions = {
    title: "My Dialog Title",
    width: 800,
    height: 600,
    okText: "Yes",
    cancelText: "No"
};
```

To not show any buttons on the dialog, you can set the `buttons` attribute to `[]`:

```
JavaScript
```

```
const dialogOptions = {
    title: "My Dialog Title",
    width: 800,
    height: 600,
    buttons: []
};
```

## Related resources

If you have a question or are looking for more information, consider going to one of the following areas:

- [Azure DevOps on Stack Overflow ↗](#)
- [Developer Community ↗](#)

ⓘ **Note:** The author created this article with assistance from AI. [Learn more](#)

# Host page navigation

Article • 10/04/2022

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

The HostNavigationService provides APIs for interacting with the parent host frame, including refreshing it and accessing the hash of the URL:



## Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

## Get the current hash value

JavaScript

```
// Get navigation service

VSS.getService(VSS.ServiceIds.Navigation).then(function(navigationService) {
    // Get current hash value from host url
    navigationService.getHash().then(function (hash) {
        console.log("Host hash value: " + hash);
    });
});
```

## Get notified when the hash value changes

JavaScript

```
// Get navigation service

VSS.getService(VSS.ServiceIds.Navigation).then(function(navigationService) {
    navigationService.onHashChanged(function (hash) {
        // Adding #something to the end of browser url executes this
        // handler with the hash value "something"
        console.log("Hash changed to : " + hash);
    });
});
```

## Change the hash value

Two methods are available for changing the hash value of the host page URL:

- `setHash` adds a new entry to the browser history
- `replaceHash` does **not** add a new entry to the browser history

JavaScript

```
// Get navigation service

VSS.getService(VSS.ServiceIds.Navigation).then(function(navigationService) {
    // Adds a new entry to browser history
    navigationService.setHash("new-hash-value");
});
```

## Refresh the host page

Following code piece shows how host page can be reloaded.

JavaScript

```
// Get navigation service

VSS.getService(VSS.ServiceIds.Navigation).then(function(navigationService) {
    // Reload whole page
    navigationService.reload();
});
```

# Basic styles for your widgets

Article • 02/20/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

We recommend you use the basic styles provided via the Widget SDK. Using these styles helps you quickly and easily create a widget that's consistent with the rest of the widgets on the dashboard. To use these styles, add the below line inside the `vss.require` block in the JavaScript code for your widget.

## 💡 Tip

Check out our newest documentation on extension development using the [Azure DevOps Extension SDK](#).

JavaScript

```
WidgetHelpers.IncludeWidgetStyles();
```

This pulls a stylesheet by the name sdk-widget.css and include it in the iframe for your widget. It includes styles for font-family, font-size, margin and paddings for your widget. It also includes styles for headings (h1, h2, h3 etc.), links, and more.

Similarly, to use common styles in the widget configuration, include the line below inside the `vss.require` block in the JavaScript code for your widget configuration.

JavaScript

```
WidgetHelpers.IncludeWidgetConfigurationStyles();
```

This pulls a stylesheet by the name sdk-widget-configuration.css and include it in the iframe for your widget configuration. It includes styles for font-family, font-size and styles for common form elements like input, textarea, and select.

## ⚠ Note

For these styles to apply to your widget, you need to add a "widget" class on the HTML element that contains your widget. All styles from the sdk-widgets.css are scoped to this class. Similarly, add a "widget-configuration" class on the HTML

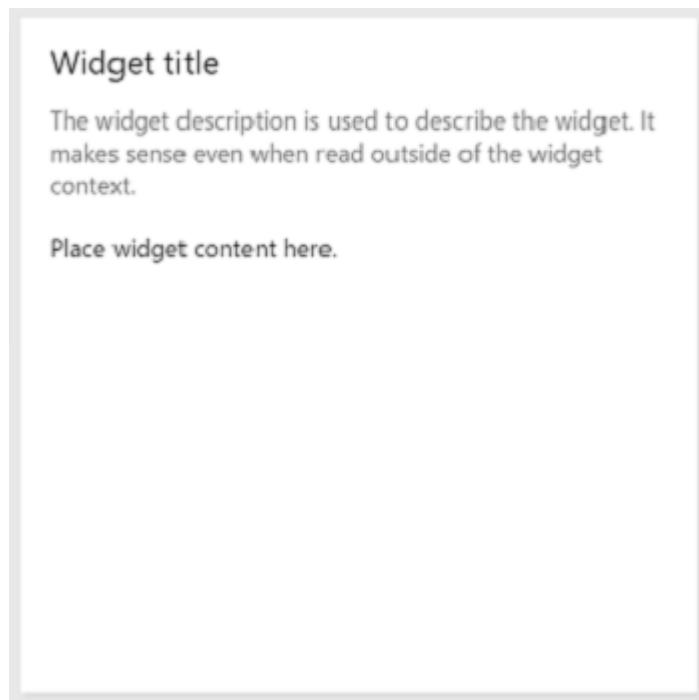
element that contains your widget configuration. All styles from the sdk-widget-configuration.css are scoped to this class.

Download the [extension sample](#).

## Widget body, title and description

By adding the class "widget" on the HTML element that contains your widget, you automatically get padding, font, and color for widget contents.

You should always have a title for your widget. This helps the user identify your widget and its functionality at a glance. Use `<h2>` with class "title". This also helps people using screen readers to quickly identify the different widgets on the dashboard.



**Design principle:** Widgets should have a title. Use the `<h2>` tag with the "title" class.

Sometimes you might want to provide a small description about your widget or how to use it. In such cases, use the class "description" on the HTML element you wish to use for widget description.

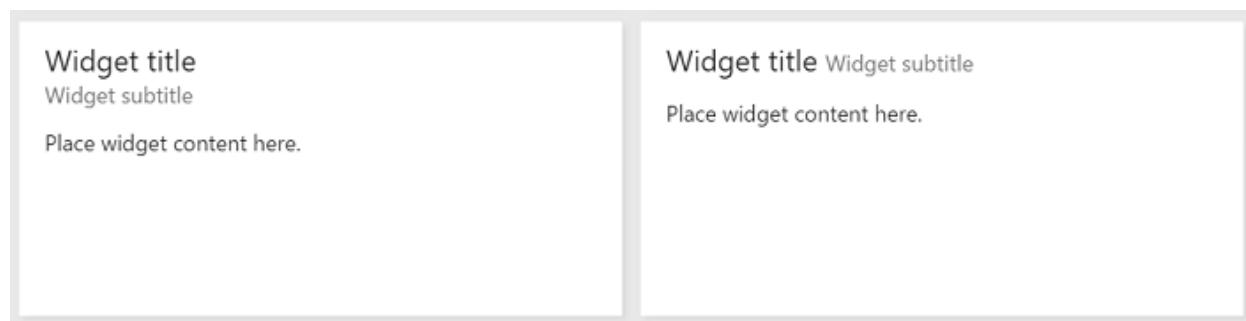
**Design principle:** Use the "description" class for the widget description. Descriptions should make sense even when read out of the widget context.

HTML

```
<div class="widget">
  <h2 class="title">Widget title</h2>
  <div class="description">The widget description is used to describe
the widget. It makes sense even when read outside of the widget context.
</div>
  <p>Place widget content here.</p>
</div>
```

## Widget titles and subtitles

Subtitles are text that supplement the title. They may not always make sense when read out of context without reading the title.



**Design principle:** Use the "subtitle" class to provide more information about the widget. It may not make sense out of the widget context.

Use the below structure and classes "title", "inner-title" and "subtitle" to get the right font, color and margins for a title and subtitle combination. The title gets a greater font-size than the subtitle. The subtitle has a subdued color relative to the title or rest of the widget.

### HTML

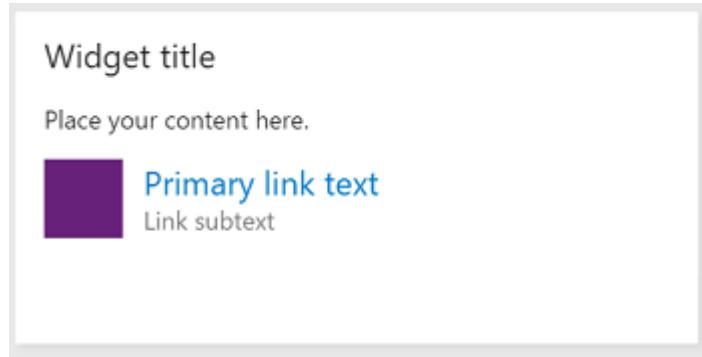
```
<div class="widget">
  <h2 class="title">
    <div class="inner-title">Widget title</div>
    <div class="subtitle">Widget subtitle</div>
  </h2>
  <div class="content">
    Place widget content here.
  </div>
</div>
```

You can use any html element for the title and subtitle combination. Here are some tips:

- When you need the subtitle to appear in the same line as the title, use an inline element like `<span>`
- When you need the subtitle to appear in the next line from the title, use a block element like `<div>`

## Display hyperlinks, icons, text, and subtext in a widget

Some widgets have links which have an icon, text and subtext per link.



**Design principle:** Use links with an icon and subtext to make the purpose of the link obvious to the user. Ensure that the icon symbolizes the link's target.

To get the same look and feel, use the below HTML structure and classes.

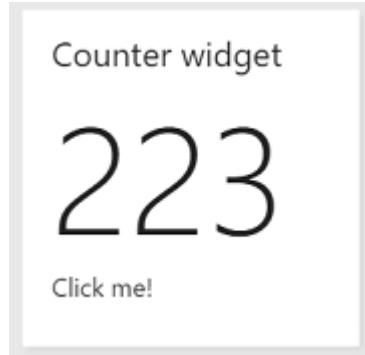
```
HTML

<div class="widget">
    <h2 class="title">Widget title</h2>
    <div class="content">
        <p>Place your content here.</p>
        <a class="link-with-icon-text" href="http://bing.com"
target="_blank">
            <span class="icon-container" style="background-color:
#68217A"></span>
            <div class="title">
                Primary link text
                <div class="subtitle">Link subtext</div>
            </div>
        </a>
    </div>
</div>
```

## Display counters in a widget

The primary purpose of some widgets is to display the count of some data. The Query Tile and the Code Tile widgets are examples in this category of widgets. To use the same

styles as these widgets, add the "big-count" class on the HTML element holding the number to get the big font that is used by the Query Tile and the Code Tile widgets.



**Design principle:** Use the "big-count" class to present the user with numbers in large font. It should not be used with non-numeric characters.

#### HTML

```
<div class="widget">
  <h2 class="title">Counter widget</h2>
  <div class="big-count">223</div>
  <div>Additional text</div>
</div>
```

## Make a widget a hyperlink

Clicking anywhere on some widgets redirects the user to another page. To have your widget do the same, you can:

- Add an anchor tag as a child to the HTML element that acts as your widget container.
- Put all your widget content inside the anchor tag.
- Since your widget is hosted in an iframe, add the attribute "target" with value "\_blank" to the anchor tag so that the link opens in a new tab/window instead of inside the same iframe.
- In addition to the "widget" class, add the "clickable" class to the widget container.

Your widget content gets the correct colors even though they are inside an anchor tag. Without the "clickable" class, the default blue color is forced on all text inside the widget. The widget also gets a custom visual cue on focus to help users who use the keyboard to navigate the dashboard.

**Design principle:** Use the "clickable" class and the `<a>` tag to make the entire widget clickable. This is ideal when your widget is a summary of data available on

another page.

#### HTML

```
<div class="widget clickable">
  <a href="https://bing.com" target="_blank">
    <h2 class="title">Counter widget</h2>
    <div class="big-count">223</div>
    <div>Click me!</div>
  </a>
</div>
```

## Styles for common form elements in widget configuration

To use basic styles from the widget sdk for common form elements in widget configuration, follow these guidelines:

Form element	Wrapping element	Guidelines
Simple text box	div with class "single-line-text-input".	Use a <code>label</code> element to add text next to the text box. Use the <code>input</code> element to create a text box. Use the <code>placeholder</code> attribute to provide placeholder text.
Checkbox	fieldset with class "checkbox"	Use a <code>label</code> element to add text next to each checkbox. Use a <code>legend</code> element to caption the group of checkboxes. Use the <code>for</code> attribute on each <code>label</code> element to help screen readers understand the form element.
Radio button	fieldset with class "radio"	Use a <code>label</code> element to add text next to each radio button. Use a <code>legend</code> element to caption the group of radio buttons. Use the <code>for</code> attribute on each <code>label</code> element to help screen readers understand the form element.
Dropdown	div with class "dropdown"	Use a <code>label</code> element to add text next to the dropdown. If you want a dropdown occupying half the width, add class "half" to the wrapping <code>div</code> element. If you want to use the standard arrow icon from the sdk instead of the one provided by the browser, wrap the <code>select</code> element with another <code>div</code> with class "wrapper".
Multi-line text box	div with class "multi-line-text-input".	Use <code>label</code> element to label the <code>textarea</code> element used as multi-line text box.

The example below uses each of the form elements listed in the table.

## Configuration

Your name

Drop down

▼

Select results to display

- Query ID
- Query Name
- Created By

Display as

- Ordered List
- Unordered List

Comments

Save Close

## HTML

```
<div class="widget-configuration">

    <div class="single-line-text-input" id="name-input">
        <label>Your name</label>
        <input type="text" value="Contoso"></input>
    </div>

    <div class="dropdown" id="query-path-dropdown">
        <label>Drop down</label>
        <div class="wrapper">
            <select>
                <option value="Shared Queries/Feedback">Shared
                Queries/Feedback</option>
                <option value="Shared Queries/My Bugs">Shared Queries/My
                Bugs</option>
                <option value="Shared Queries/My Tasks">Shared Queries/My
                Tasks</option>
            </select>
        </div>

        <fieldset class="checkbox" id="select-results">
            <legend>Select results to display</legend>
            <input type="checkbox" id="check-option1" value="id"
            name="check" checked="true">
                <label for="check-option1">Query ID</label><br/>
            <input type="checkbox" id="check-option2" value="name"
            name="check" checked="true">
                <label for="check-option2">Query Name</label><br/>
            <input type="checkbox" id="check-option3" value="createdBy"
            name="check" checked="true">
                <label for="check-option3">Created By</label><br/>
        </fieldset>

        <fieldset class="radio" id="display-options">
            <legend>Display as </legend>
            <input type="radio" id="radio-option1" value="ordered"
            name="radio" checked="true">
                <label for="radio-option1">Ordered List</label><br/>
            <input type="radio" id="radio-option2" value="unordered"
            name="radio">
                <label for="radio-option2">Unordered List</label><br/>
        </fieldset>

        <div class="multi-line-text-input">
            <label>Comments</label>
            <textarea></textarea>
        </div>
    </div>
</div>
```

## Display validation errors below a form element

We recommend providing validation errors below the relevant form elements. To display these messages in a manner consistent with 1st party widgets, add the following code snippet under each form element for which you want to show the error message.

HTML

```
<span class="validation-error">
    <span class="icon-error-exclamation"></span>
    <span class="validation-error-text"></span>
</span>
```

The previous code snippet has the visibility hidden by default. Whenever you want to display an error message, find the corresponding "validation-error-text", add text to it and set `visibility:visible` on its parent.

Example: There is a simple text box where the user needs to type in a string. You need to show an error message if the text box is empty.

A screenshot of a web form. At the top, there is a label "Your name" followed by a text input field. Below the input field, a red exclamation mark icon is followed by the text "Please enter your name." The entire form is enclosed in a light gray border.

The html for this would be:

HTML

```
<div class="widget-configuration">
    <div class="single-line-text-input">
        <label>Your name</label>
        <input type="text">Type Here</input>

        <span class="validation-error">
            <span class="icon-error-exclamation"></span>
            <span class="validation-error-text"></span>
        </span>
    </div>
</div>
```

And the JavaScript code behind this would be:

JavaScript

```
var $singleLineInput = $(".single-line-text-input input");
var $errorSingleLineInput = $(".single-line-text-input input .validation-
```

```
error-text");

$singleLineInput.on("input", function(){
    if ($singleLineInput.val() == ""){
        $errorSingleLineInput.text("Please enter your name.");
        $errorSingleLineInput.parent().css("visibility", "visible");
        return;
    }
    $errorSingleLineInput.parent().css("visibility", "hidden");
});
```

# Deploy web content to Azure

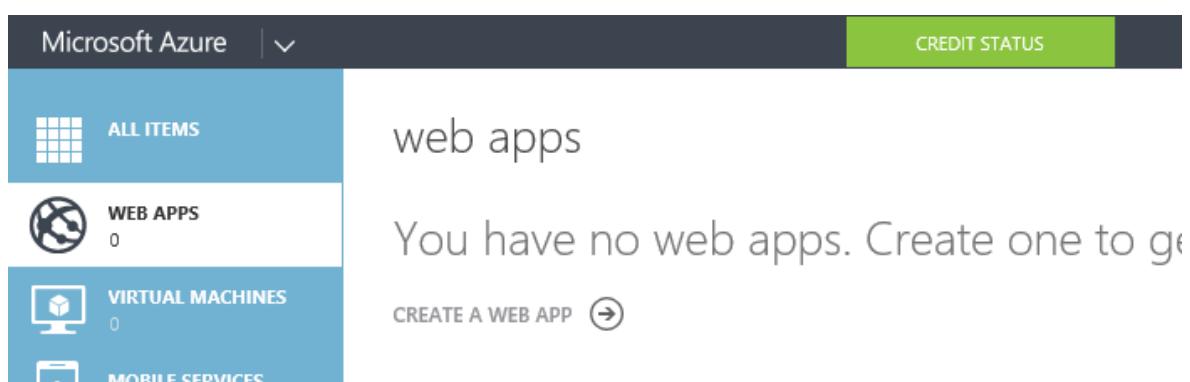
Article • 10/04/2022

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

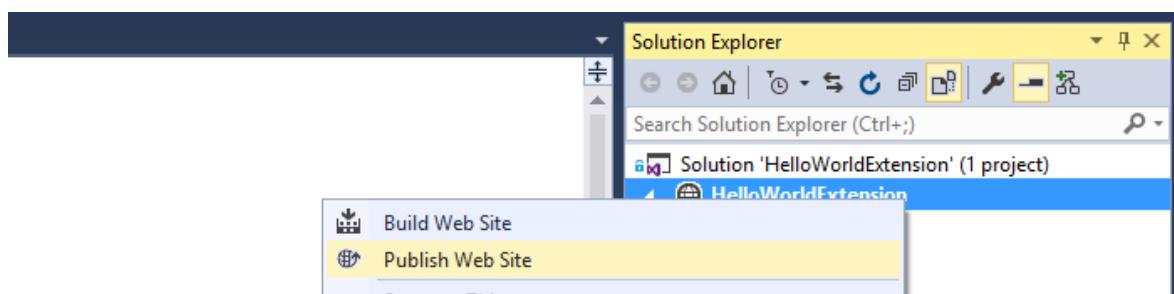
If your extension doesn't require any back-end logic, consider hosting your static content directly on Azure DevOps. See [content hosting](#).

After you've [created an extension](#), you can publish it to Azure so that it's available in the cloud.

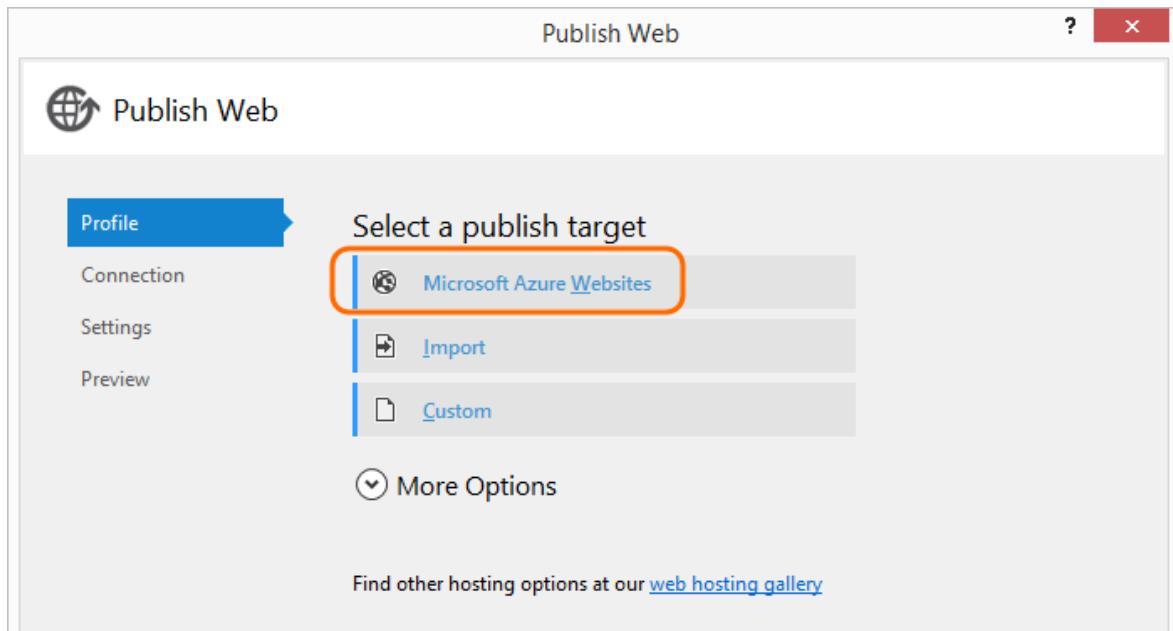
1. If you don't have an Azure subscription, create one. You can use the [free trial](#).
2. Create a web app in Microsoft Azure to host your extension.



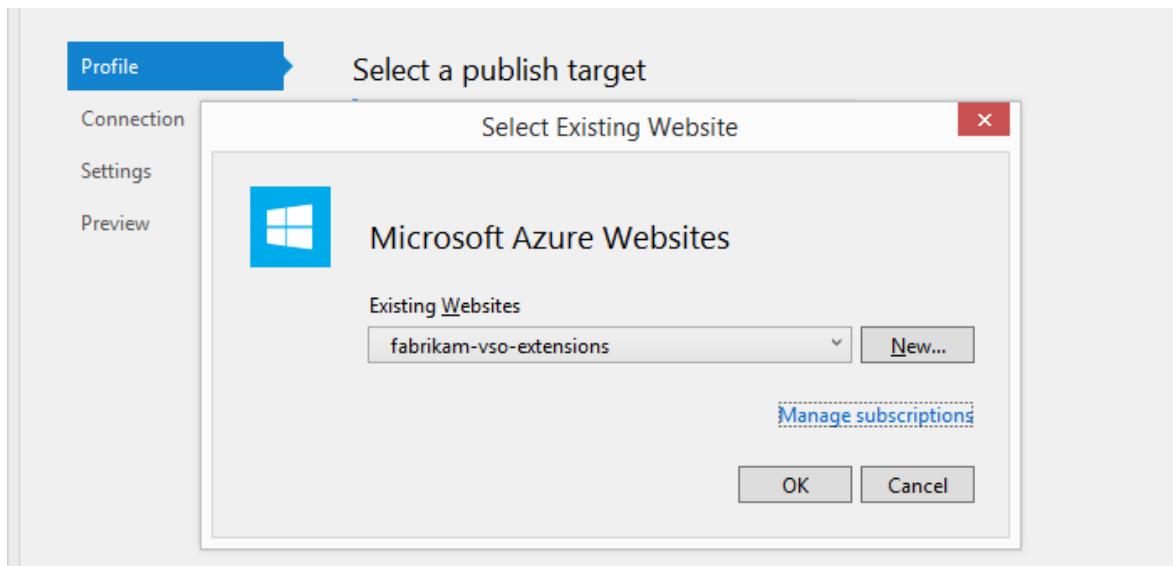
3. Publish your web site from the solution explorer.



4. Publish to Azure.

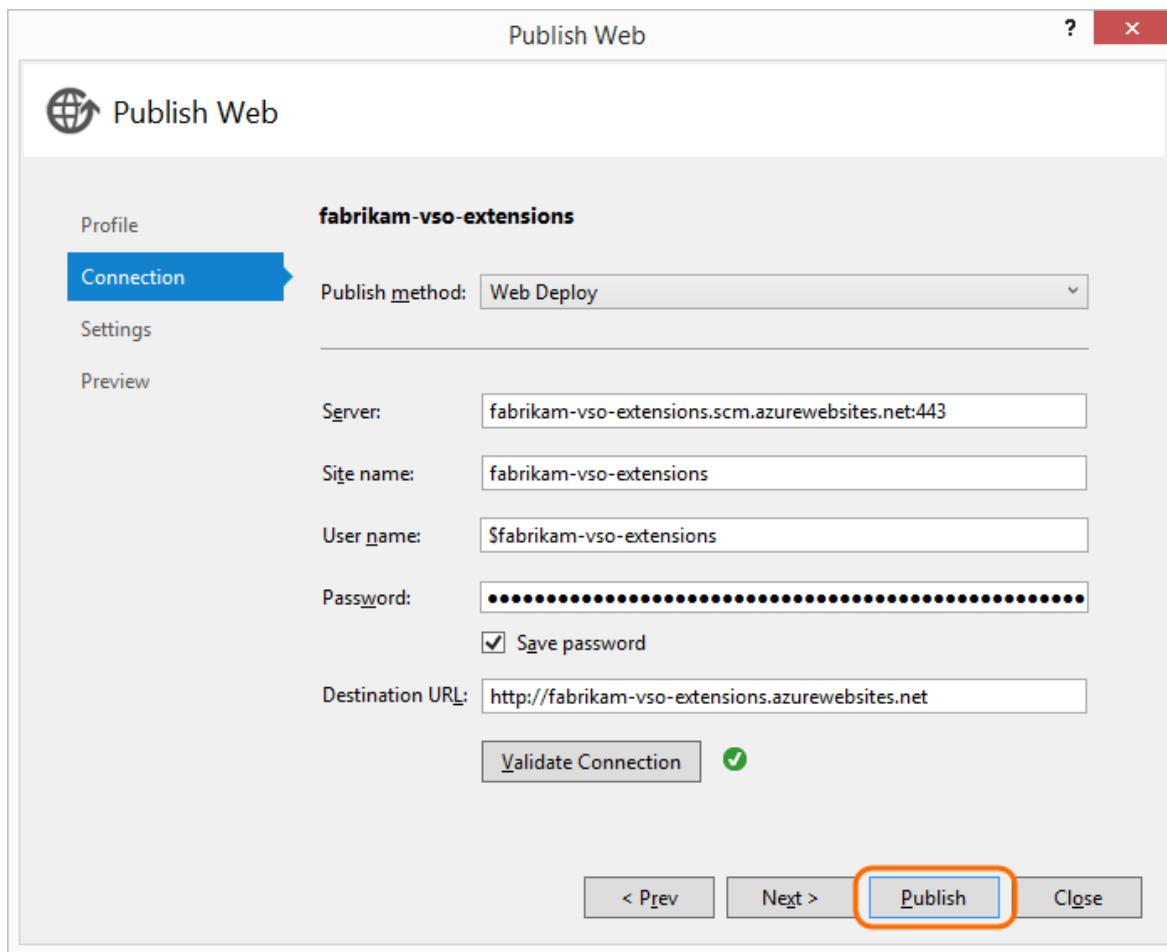


5. Pick the web app that you set up to host your extension.



If your web site doesn't show up, use the **Manage subscriptions** dialog to connect your Visual Studio organization to your Microsoft Azure subscription.

6. Publish your extension.



7. Change your extension manifest to use your Microsoft Azure web app instead of localhost.

JSON

```
"baseUri": "https://fabrikam-vso-extensions.azurewebsites.net/",
```

8. [Install](#) your extension again and try it out.

# Integrate with service hooks

07/15/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

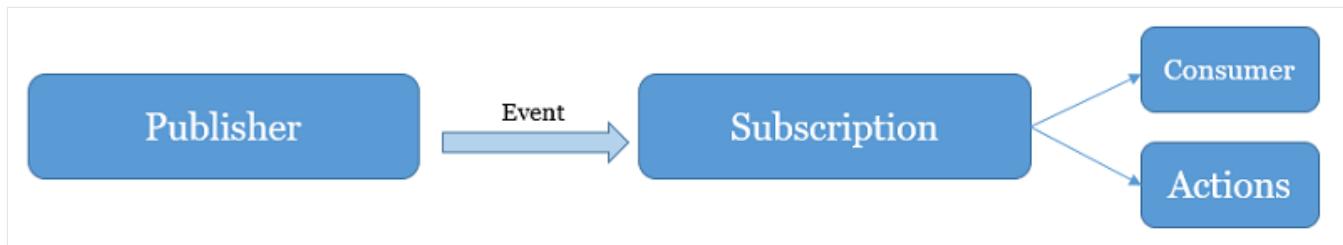
You can use service hooks to run tasks on other services when events happen in your Azure DevOps project.

For example, you can use a service hook to automatically create a card in Trello when a work item gets created in Azure DevOps. Or you can automatically send a push notification to your team's mobile devices when a build fails. You can also use service hooks in custom apps and services as a more efficient way to drive activities when events happen in your projects.

## How do service hooks work?

Service hook **publishers** define a set of *events* that you can subscribe to. **Subscriptions** listen for these events and define **actions** to take based on events.

Subscriptions also target **consumers**, which are external services that can run their own actions when events occur.



### ! Note

To use service hooks, you must allow specific IP address ranges for inbound connections to service endpoints. A service endpoint is a set of properties provided to a service hook. For more information, see [IP addresses and range restrictions](#).

## Available services

The following services are available as targets of service hooks. For more information about other apps and services that integrate with Azure DevOps, see the [Visual Studio Marketplace](#).

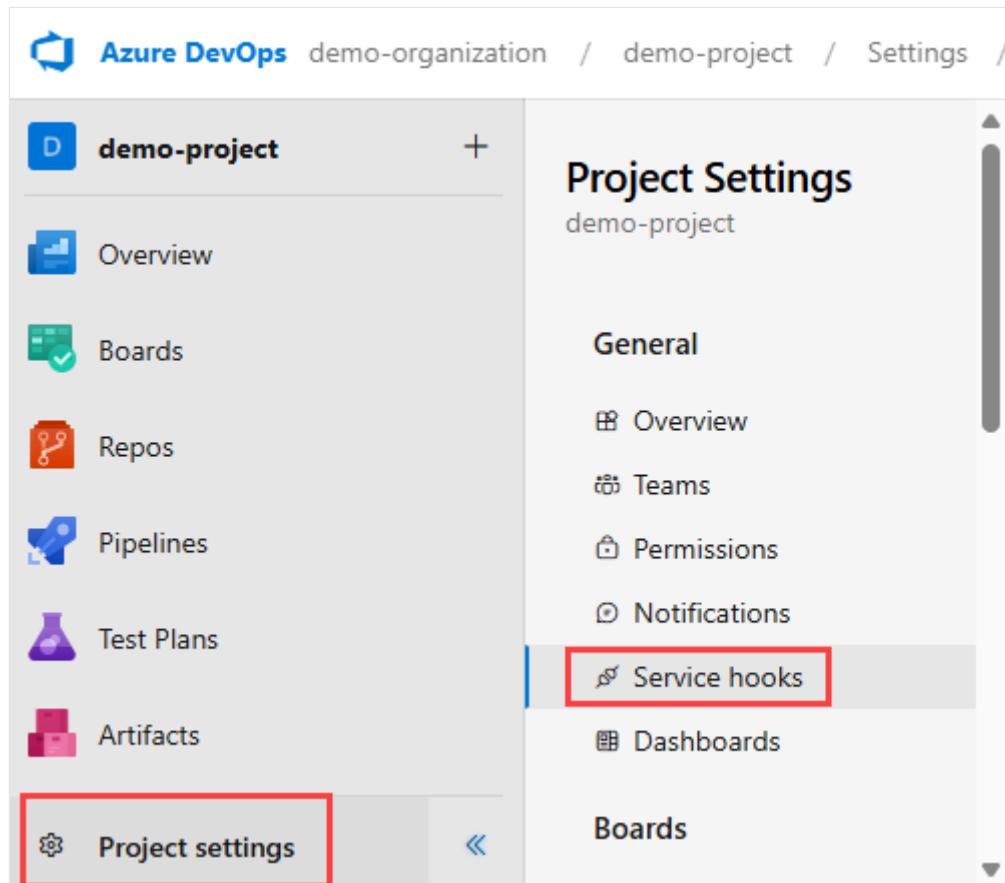
<b>Service</b>	<b>Supported events</b>	<b>Supported actions</b>	<b>Subscription management</b>
Visual Studio App Center	Work item updated	Send a notification	In App Center
AppVeyor ↗	Code pushed	Trigger an AppVeyor build	In AppVeyor
Azuqua ↗	All	Post an event to a flow language object (FLO)	In Azuqua
Azure App Service ↗	Code pushed	Deploy a web app	In App Service
Azure Service Bus	All	Send a message to Azure Notification Hubs, a Service Bus queue, or a Service Bus topic	In Azure DevOps
Azure Storage	All	Insert a message in a Storage queue	In Azure DevOps
Bamboo ↗	Build completed, code pushed	Queue a build	In Azure DevOps
Datadog	All	Post an event in Datadog	In Azure DevOps
Grafana	Release deployment completed	Add an annotation to a Grafana database	In Azure DevOps
Jenkins	Build completed, code pushed, PR merge attempted	Trigger a generic or Git build	In Azure DevOps
Microsoft Teams	All	Post a message to a channel	In Teams
MyGet ↗	Build completed, code pushed	Publish a NuGet package to MyGet, trigger a MyGet build	In MyGet
Office 365	All	Post a message to a group	In Office 365
Slack	All	Post a message to a channel	In Azure DevOps
Trello	All	Create a card or list	In Azure DevOps
UserVoice ↗	Work item created, work item updated	Send a linked work item event	In UserVoice
Webhooks	All	Post a request via HTTP	In Azure DevOps
Workplace messaging apps	All	Send a notification	In workplace messaging apps
Zapier ↗	All	Send a notification	In Zapier

Service	Supported events	Supported actions	Subscription management
Zendesk	Work item commented on	Create a private comment in a ticket	In Azure DevOps

## Create a subscription

To integrate one of these services with Azure DevOps, you create a subscription. In many cases, you also need to configure the target service. For detailed information, see the documentation for the service that you want to integrate.

1. Go to your project, select **Project settings**, and then select **Service hooks**.



2. Select **Create subscription**.

The screenshot shows the Azure DevOps interface for a project named 'demo-project'. The left sidebar includes links for Overview, Boards, Repos, Pipelines, Test Plans, Artifacts, and Project settings. The 'Project Settings' section is expanded, showing options like General, Overview, Teams, Permissions, Notifications, Service hooks (which is selected and highlighted in blue), Dashboards, and Boards. On the right, the 'Service Hooks' section is displayed with the sub-instruction: 'Integrate with your favorite services by notifying them when events happen in your project.' A prominent blue button labeled '+ Create subscription' is visible.

3. Select the service that you want to integrate.

The screenshot shows the 'NEW SERVICE HOOKS SUBSCRIPTION' dialog. The title bar says 'NEW SERVICE HOOKS SUBSCRIPTION'. The main area is titled 'Service' with the sub-instruction 'Select a service to integrate with. [Discover more integrations](#)'. A vertical list of services on the left includes: Azure App Service, Azure Service Bus, Azure Storage, Bamboo, Datadog, Jenkins, Microsoft Teams, MyGet, Office 365, Slack, Trello (which is highlighted in a blue box), UserVoice, Web Hooks, and Workplace Messaging Apps. To the right of the list, details for 'Trello' are shown: 'Provides integration with Trello', 'Supported events: All events', 'Supported actions: Create a card, Create a list', and a link 'Learn more about this service'. At the bottom are navigation buttons: Previous, Next (which is highlighted in a blue box), Test, Finish, and Cancel.

4. Select the event to trigger on and any applicable filters.

# Trigger

Select an event to trigger on and configure any filters.

Trigger on this type of event

Code pushed

**i** Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.

## FILTERS

Repository [i](#)

optional

Fabrikam-Fiber-Git



Branch [i](#)

optional

[Any]



Pushed by a member of group: [i](#)

optional

[demo-project]\Build Administrators



Previous

Next

Test

Finish

Cancel



5. Select an action to run on the target service.

**!** Note

The available actions depend on the type of event that you select.

# Action

Select and configure the action to perform.

Perform this action

**Create a card**

This action creates a card on an existing list in Trello. A card can represent a task, issue, event, or just about anything. A card's state is typically determined by what list it is on. [Learn More](#)

## SETTINGS

User token (need one? [Get it now](#)) i required

!

Board i required

!

List i required

!

Labels i optional

Create at top of list i

Previous

Next

Test

Finish

Cancel

6. To confirm the settings are correct, test the subscription and then finish the wizard.

**TEST NOTIFICATION**

x

**Trello (Create a card)**[Summary](#)   [Request](#)   [Response](#)   [Event](#)**Succeeded**

Sent at: Tuesday, May 13, 2025 8:12:55 PM

**Message**

Jamal Hartnett pushed updates to Fabrikam-Fiber-Git:main.

[Close](#)**TEST NOTIFICATION**

x

**Trello (Create a card)**[Summary](#)   **Request**   [Response](#)   [Event](#)

Method: POST  
URI: [https://api.trello.com/1/cards?key=7d6630fd03ac2b6fc9fde2f2ef0c4096&token=\\*\\*\\*\\*\\*](https://api.trello.com/1/cards?key=7d6630fd03ac2b6fc9fde2f2ef0c4096&token=*****)  
HTTP Version: 1.1  
Headers:  
{  
  Content-Type: application/json; charset=utf-8  
}  
Content:  
{  
  "name": "Jamal Hartnett pushed updates to Fabrikam-Fiber-Git:main.",  
  "desc": "Jamal Hartnett pushed a commit to [Fabrikam-Fiber-Git](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_git/Fabrikam-Fiber-Git/):[main](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_git/Fabrikam-Fiber-Git/#version=GBmain).\\n\* Fixed bug in web.config file [33b55f7c](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_git/Fabrikam-Fiber-Git/commit/33b55f7cb7e7e245323987634f960cf4a6e6bc74)",  
  "pos": "top",  
  "due": null,  
  "labels": "",  
  "idList": "6823a747bc1033e7b21928f5"  
}

[Close](#)

# FAQs

## Q: What permissions do I need to set up a subscription?

A: You need the *Edit subscriptions* and *View subscriptions* permissions. By default, only project administrators have these permissions. To grant them to other users directly, you can use a [command-line tool](#) or the [Security REST API](#).

To grant the *View* permission to a group, see [Set View permission for a group in service hooks](#).

## Q: What are the security implications of granting *Edit subscriptions* and *View subscriptions* permissions?

A: A user who has the *Edit subscriptions* and *View subscriptions* permissions can:

- See all subscriptions in the project.
- See the notification history for all subscriptions in the project.
- Create any type of service hook subscription in the project.

If the user sets up a subscription for a resource that they don't otherwise have permission to access, the subscription doesn't get triggered.

For example, suppose you create a subscription to send a notification when a work item in a specific area path gets updated. If you don't have access to the work items in that area path, the notifications don't get sent. However, if other users have access to the work items, you can see the notification history for subscriptions that alert them about updates.

## Q: Can I create service hook subscriptions for a project programmatically?

A: Yes. For more information, see [Create a service hooks subscription programmatically](#).

## Q: Can I remove an app's access to my organization after I authorize it?

A: Yes. You can revoke authorizations from your profile.

1. Go to <https://visualstudio.microsoft.com>. Select your profile photo, and then select **Visual Studio profile**.

Make sure you start from the Visual Studio site, <https://visualstudio.microsoft.com>, when you manage authorizations. From that site, you can access the correct implementation of the authorizations management feature. Don't start from your organization (<https://dev.azure.com/{organization-name}>).

2. Select Manage authorizations.



Dakota Sanchez [Edit profile](#)

dakota@fabrikam.com

Microsoft

United States

dakota@fabrikam.com

---

**Visual Studio Dev Essentials**

Get everything you need to build and deploy your app on any platform.

[Use your benefits](#)

**Authorizations**

You have authorized **3 applications**

[Manage authorizations](#)

3. Revoke any authorizations you no longer want to allow.

## Authorizations

X

List of OAuth-compliant applications and providers that you have given permissions to access your resources

### OpenPublishing.Build.ProdTest (Microsoft)

Grants the ability to read, create, and update work items and queries, update board metadata, read area and iterations paths other work item tracking related metadata, execute queries, and to receive notifications about work item events via service hooks.

Grants the ability to access build artifacts, including build results, definitions, and requests, and the ability to queue a build, update build properties, and the ability to receive notifications about build events via service hooks.

Grants the ability to read, update, and delete source code, access metadata about commits, changesets, branches, and other version control artifacts. Also grants the ability to create and manage code repositories, create and manage pull requests and code reviews, and to receive notifications about version control events via service hooks.

[Revoke](#)

Grants the ability to access rooms and view, post, and update messages. Also grants the ability to manage rooms and users and to receive notifications about new messages via service hooks.

Grants the ability to read, create, and update test plans, cases, results and other test management related artifacts.

[Close](#)

## Q: Why can't I set up service hooks for HipChat anymore?

A: Atlassian no longer supports HipChat. For more information, see [Atlassian Frequently Asked Questions ↗](#).

## Related content

- [Troubleshoot service hooks](#)
- [Visual Studio Marketplace ↗](#)
- [Billing overview](#)

# Workplace messaging apps

Article • 10/04/2022

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

The following integrations help users receive notifications in response to events in Azure DevOps within their workplace messaging apps such as [Microsoft Teams](#) and [Slack](#).

Along with receiving notifications, below integrations also help users to complete workflows on Azure DevOps, such as allowing users to approve release deployments and creating work items from their channels. You can use these apps only with a project hosted on Azure DevOps Services.

## Microsoft Teams

- [Azure Boards app for Microsoft Teams](#)
- [Azure Pipelines app for Microsoft Teams](#)
- [Azure Repos app for Microsoft Teams](#)

## Slack

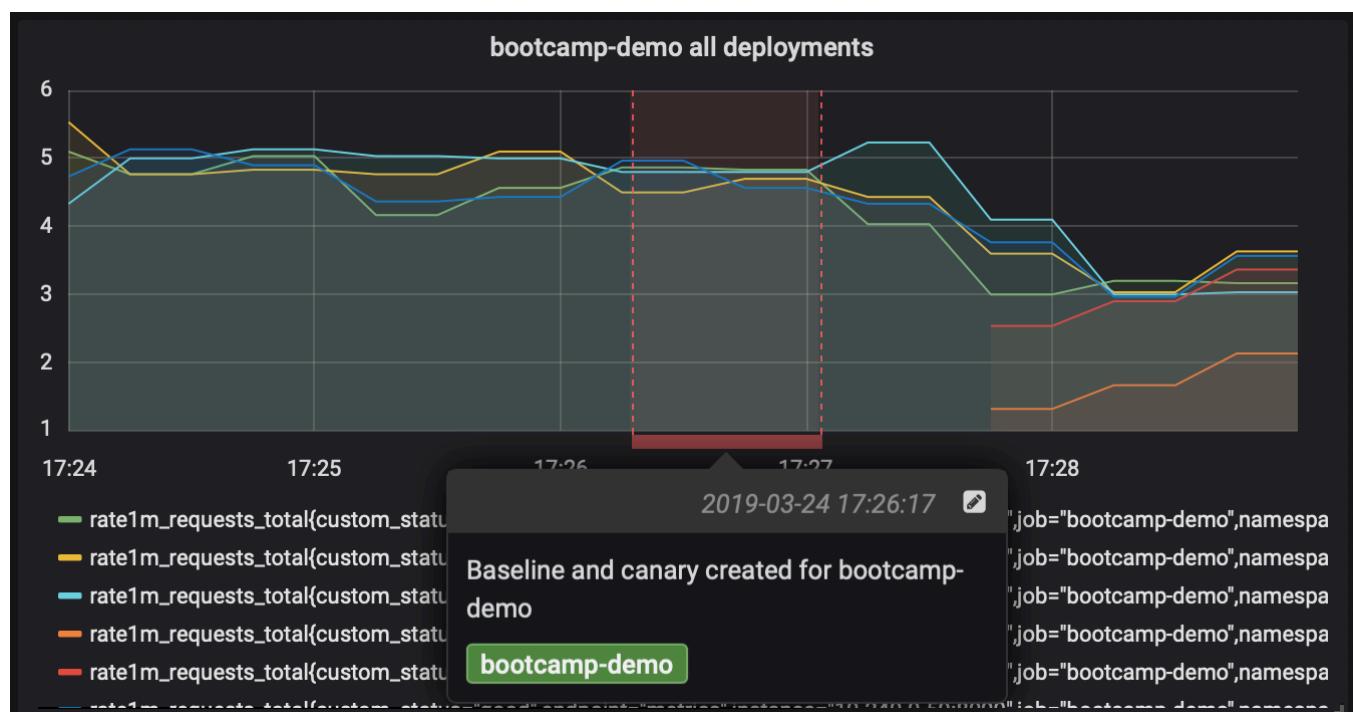
- [Azure Boards app for Slack](#)
- [Azure Pipelines app for Slack](#)
- [Azure Repos app for Slack](#)

# Create a service hook for Azure DevOps with Grafana

09/04/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Grafana is an open source, feature-rich metrics dashboard, and graph editor. It supports Graphite, Elasticsearch, OpenTSDB, Prometheus, and InfluxDB analytics services. Use the Grafana service hook to annotate Grafana dashboards upon completion of Azure Pipelines deployments.



## Prerequisites

[ ] Expand table

Category	Requirements
Permissions	<ul style="list-style-type: none"><li>- Member of the <a href="#">Project Collection Administrators group</a>. Organization owners are automatically members of this group.</li><li>- <b>Edit subscriptions</b> and <b>View subscriptions</b> permissions set to <b>Allow</b>. By default, only project administrators have these permissions. To grant the permissions to other users, you can use the command-line tool or the <a href="#">Security REST API</a>.</li></ul>
Tools	Grafana

# Create a service hook subscription

1. Navigate to project settings:

```
https://dev.azure.com/{orgName}/{project_name}/_settings/serviceHooks
```

The screenshot shows the Azure DevOps Project Settings interface for the 'fabrikamprime / Fabrikam Fiber' project. On the left, a sidebar lists 'Project Settings' (General, Overview, Teams, Permissions, Notifications, Service hooks), 'Dashboards', 'Boards', and 'Project configuration'. The 'Service hooks' item is highlighted with a red box. A second red box highlights the gear icon at the bottom of the sidebar. The main area is titled 'Service Hooks' with the sub-instruction 'Integrate with your favorite services by notifying them when events happen in your project.' Below this is a table with columns: Consumer ↑, Event ↑, Event Filter, and Action. The table lists several Web Hook subscriptions:

Consumer ↑	Event ↑	Event Filter	Action
Web Hooks	... Build completed	Any completed build.	Post via HTTP
Web Hooks	Code pushed	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request comment...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request created	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request merge att...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request updated	Any branch on any repository.	Post via HTTP
Web Hooks	Repository created	Any branch on any repository.	Post via HTTP
Web Hooks	Repository deleted	Repository [Any]	Post via HTTP

Select **Create Subscription**.

2. Choose **Grafana** among the list of services and choose **Next**

# Service

Select a service to integrate with. [Discover more integrations](#)

App Center

AppVeyor

Azuqua

Azure App Service

Azure Service Bus

Azure Storage

Bamboo

Campfire

Datadog

Flowdock

Grafana

HipChat

HockeyApp

Jenkins

## Grafana

Grafana is an open source, feature rich metrics dashboard and graph editor for Graphite, Elasticsearch, OpenTSDB, Prometheus and InfluxDB.

### **Supported events:**

Release deployment completed

### **Supported actions:**

Add annotation to Grafana database

[Learn more about this service](#)

Previous

Next

Test

Finish

Cancel

3. For **Release deployment completed** event, configure the optional filters: **Release pipeline name**, **Stage name**, and **Status**

# Trigger

Select an event to trigger on and configure any filters.

Trigger on this type of event

Release deployment completed

**Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.**

## FILTERS

Release pipeline name [i](#)

optional

bootcamp-demo



Stage name [i](#)

optional

DeployCanary



Status [i](#)

optional

Succeeded



Previous

Next

Test

Finish

Cancel

4. Provide the Grafana URL and [Grafana API token](#) required for Azure DevOps to post annotations.

- If you check the **Annotate deployment duration window**, the annotation added corresponds to the duration of the deployment (start and end timestamps of deployment).
- If not checked, annotation corresponds to only the completion timestamp of the deployment. The dashboard input can be used to choose a specific dashboard as the target of annotations rather than the default behavior of annotating all dashboards.

5. Choose **Test** to verify that Azure DevOps can use the provided configuration and successfully create a subscription.

6. Once verified, choose **Finish** to complete the creation of subscription.

# Action

Select and configure the action to perform.

Perform this action

Add annotation to Grafana database

Create an annotation in Grafana. Note that annotations will light up in Grafana dashboard only upon adding and saving annotation queries to filter by the tags that are specified in the service hook creation form. [Learn more](#)

## SETTINGS

Grafana url <small>i</small>	required
<input type="text" value="https://grafana.com/api"/>	✓
API token. (need one? <a href="#">Create a Grafana api token</a> ) <small>i</small>	required
<input type="text" value="XXXXXXXXXX"/>	✓
Tags <small>i</small>	required
<input type="text" value="bootcamp-demo"/>	✓
<input checked="" type="checkbox"/> Annotate deployment duration window <small>i</small>	
Text <small>i</small>	optional
<input type="text" value="Baseline and canary created for bootcamp-demo"/>	✓
Dashboard <small>i</small>	optional
<input type="text" value="Bootcamp Demo"/>	✓

Previous

Next

Test

Finish

Cancel

# Create a service hook for Azure DevOps with Slack

07/15/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

In this article, learn how to post messages to [Slack](#) in response to events in your Azure DevOps organization, such as completed builds, code changes, pull requests, releases, work items changes, and more.

## ! Note

For Azure DevOps Services, we recommend you use the following suite of apps which offer features to integrate with Slack.

## Azure Boards app for Slack

The [Azure Boards app for Slack](#) helps to easily create and monitor work items from your Slack channels. You can create work items using a slash command, or use message actions to convert conversations in the channel into work items. You can also set up and manage subscriptions to get notifications in their channel whenever work items are created or updated.

## Azure Pipelines app for Slack

The [Azure Pipelines app for Slack](#) helps to easily monitor the events in your pipelines. You can set up and manage subscriptions for completed builds, releases, pending approvals and more from the app and get notifications for these events in their channels. You can also approve release deployments from your channels.

## Azure Repos app for Slack

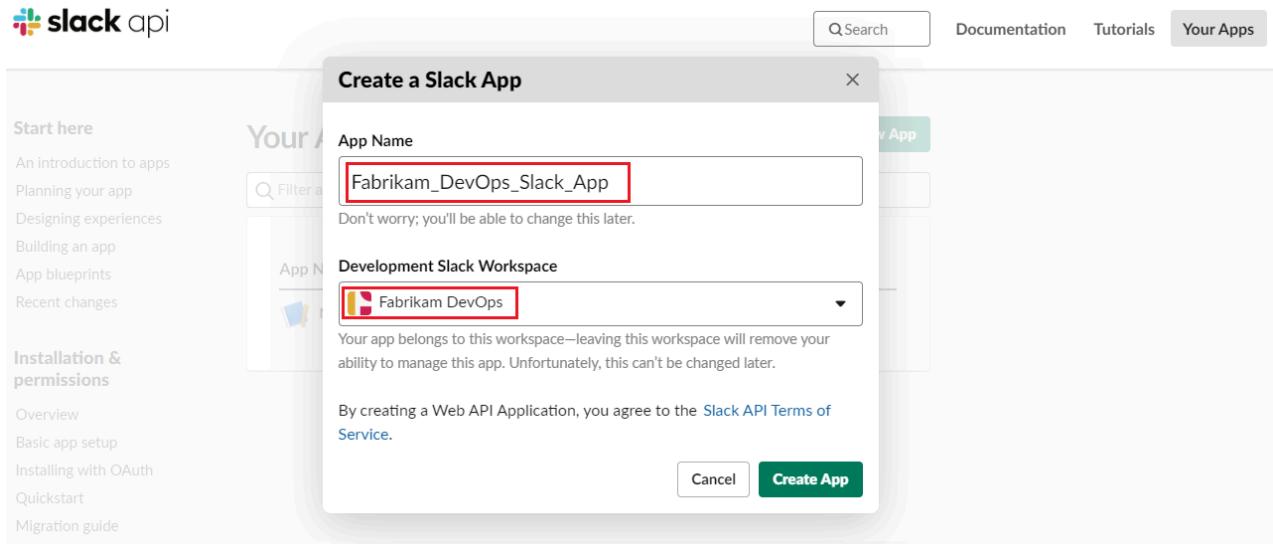
The [Azure Repos app for Slack](#) helps to easily monitor the events in your repositories. You can set up and manage subscriptions for code commits, PR creation and updates, and more, and get notifications for these events in their channels.

## Prerequisites

Refer to the [Slack documentation, Sending messages using Incoming Webhooks](#) to understand the process of using Web Hooks to push information to a Slack channel.

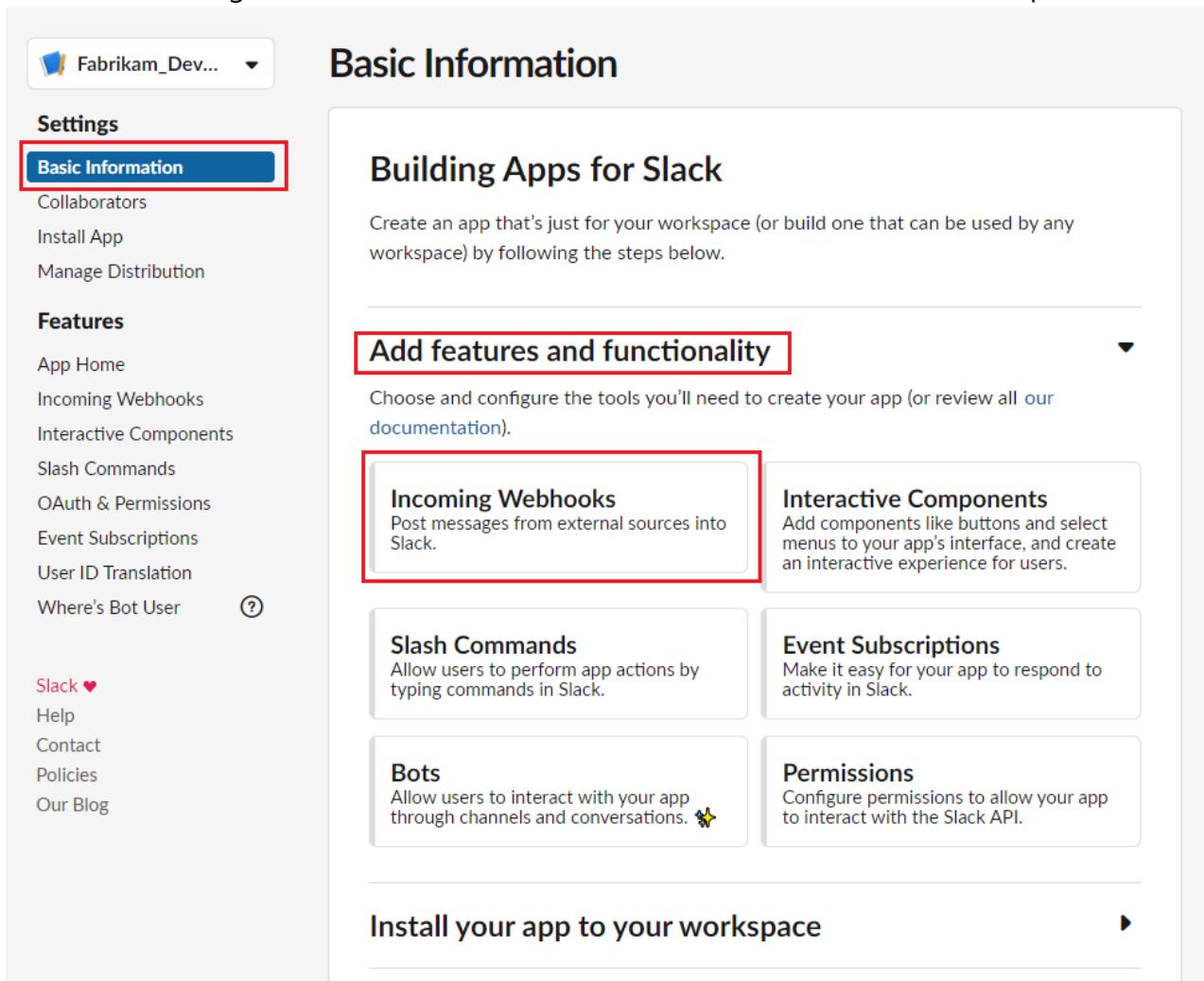
## Create a custom app in Slack

### 1. Create a new [Slack app](#)



The screenshot shows the Slack API interface for creating a new app. The 'Create a Slack App' dialog is open, prompting for an 'App Name' (Fabrikam\_DevOps\_Slack\_App) and a 'Development Slack Workspace' (Fabrikam DevOps). The 'Create App' button is highlighted in green.

### 2. Activate incoming Web Hook and add a new Web Hook to the desired workspace



The screenshot shows the 'Basic Information' tab selected in the Slack app settings. The 'Incoming Webhooks' section is highlighted with a red box. Other sections like 'Features' and 'Add features and functionality' are also visible.

Fabrikam\_Dev... ▾

## Incoming Webhooks

**Settings**

- Basic Information
- Collaborators
- Install App
- Manage Distribution

**Features**

- App Home
- Incoming Webhooks**
- Interactive Components
- Slash Commands
- OAuth & Permissions
- Event Subscriptions
- User ID Translation
- Where's Bot User ?

**Slack ❤**

Help

Contact

Policies

Our Blog

### Activate Incoming Webhooks

**On**

Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include message attachments to display richly-formatted messages.

Adding incoming webhooks requires a bot user. If your app doesn't have a bot user, we'll add one for you.

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to [Revoke All OAuth Tokens](#).

---

### Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your message in JSON as the body of an `application/json` POST request.

Add this webhook to your workspace below to activate this curl example.

Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}' YOUR_WEBHOOK_URL_HERE
```

Webhook URL	Channel	Added By
No webhooks have been added yet.		

**Add New Webhook to Workspace**

3. Select the channel for which Web Hook must be created.

This app was created by a member of your workspace, Fabrikam DevOps.

## Fabrikam\_DevOps\_Slack\_App is requesting permission to access the Fabrikam DevOps Slack workspace



Where should Fabrikam\_DevOps\_Slack\_App post?

- # Fabrikam\_DevOps\_Slack\_App requires a channel to post to as an app

# devops



Cancel

Allow

4. Copy the Web Hook URL and go to Azure DevOps.

## Create a service hook subscription in your organization

1. Go to your project Service Hooks page.

[https://{{orgName}}/{{project\\_name}}/\\_settings/serviceHooks](https://{{orgName}}/{{project_name}}/_settings/serviceHooks)

The screenshot shows the 'Project Settings' page for 'Fabrikam Fiber'. On the left, there's a sidebar with icons for General, Overview, Teams, Permissions, Notifications, Service hooks (which is highlighted with a red box), Dashboards, Boards, and Project configuration. The main area is titled 'Service Hooks' with the sub-instruction 'Integrate with your favorite services by notifying them when events happen in your project.' Below this is a table with columns: Consumer ↑, Event ↑, Event Filter, and Action. The table lists several Web Hook entries:

Consumer ↑	Event ↑	Event Filter	Action
Web Hooks	... Build completed	Any completed build.	Post via HTTP
Web Hooks	Code pushed	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request comment...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request created	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request merge att...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request updated	Any branch on any repository.	Post via HTTP
Web Hooks	Repository created	Any branch on any repository.	Post via HTTP
Web Hooks	Repository deleted	Repository [Any]	Post via HTTP

Select **Create Subscription**.

2. Choose the types of events you want to appear in your Slack channel.

You can filter each of the triggers in specific ways. For example, filter the *pull request created* trigger on the repository in which the pull request occurs, the target branch it applies to, and the team members required or invited to review the request.

3. Paste the Web Hook URL from the Slack integration that you created and select **Finish**.

The screenshot shows the 'New Service Hooks Subscription' configuration page. It has a header 'NEW SERVICE HOOKS SUBSCRIPTION' and a section title 'Action'. Below it is the instruction 'Select and configure the action to perform.' There are two options under 'Perform this action': 'Post a message to a channel' (selected) and 'Post a message about the event to a Slack channel'. Under 'SETTINGS', there's a field for 'Slack Webhook URL' with the value 'https://hooks.slack.com/services' and a green checkmark indicating it's required.

When the event occurs in your project, a notification appears in your team's Slack channel.

## FAQs

## **Q: Why don't I have the pull request events as an option when I configure my trigger?**

A: Pull requests are only available with projects that use Git. If your project uses TFVC, pull event triggers aren't available, and your code event is called "Code checked in" instead of "Code pushed."

## **Q: How can I get multiple events to show up in my Slack channel?**

A: Create a new subscription for each type of event you want. For example, if you want to see build failures and new work items in your Slack channel, create two more subscriptions.

# Create a service hook for Jenkins

Article • 02/08/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

This article describes how to set up a service hook for your Azure DevOps project to trigger a Jenkins build. If you use Jenkins to build your apps, you can store your code in a Git repository and use Jenkins for your continuous integration builds. You can trigger a Jenkins build when you push code to your Git repository or when you check in code to a [Team Foundation Version Control \(TFVC\)](#) repository.

You can build part of your app in Azure Pipelines and part in Jenkins. You can trigger a Jenkins build when your Azure Pipelines build is completed so that you use both systems to build your app.

Azure DevOps doesn't charge for setting up service hooks or integrating with external services, and Jenkins is fully open-source and free to use.

## Prerequisites

[+] [Expand table](#)

Category	Requirements
Permissions	<ul style="list-style-type: none"><li>- Member of the <a href="#">Project Collection Administrators group</a>. Organization owners are automatically members of this group.</li><li>- <b>Edit subscriptions</b> and <b>View subscriptions</b> permissions set to <b>Allow</b>. By default, only project administrators have these permissions. To grant the permissions to other users, you can use the command-line tool or the <a href="#">Security REST API</a>.</li></ul>
Tools	<p>Jenkins <a href="#">server</a>. If you set up Jenkins on-premises, <a href="#">enable HTTPS</a>. In your <i>jenkins.xml</i> configuration file, set the <a href="#">hudson.plugins.git.GitStatus.NOTIFY_COMMIT_ACCESS_CONTROL system property</a> to <code>disabled</code> by adding or updating the following line in the <code>&lt;arguments&gt;</code> tag, before the <code>-jar</code> parameter:</p> <pre>-Dhudson.plugins.git.GitStatus.NOTIFY_COMMIT_ACCESS_CONTROL=disabled.</pre>

## Create a Jenkins service hook

Do the following steps to create a Jenkins service hook.

# Set up a Jenkins build

1. In Jenkins, create a new item.

The screenshot shows the Jenkins dashboard. At the top left is the Jenkins logo. To its right is a search bar with the placeholder "Search (CTRL+K)". Further right are a help icon, a shield icon, the Jenkins dropdown menu, and a "log out" link. Below the header is a navigation bar with links: "Dashboard" (selected), "People", "Build History", "Manage Jenkins", and "My Views". A red box highlights the "+ New Item" button. To the right of the button is an "Add description" link. The main content area features a "Welcome to Jenkins!" message and a brief description: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this is a "Start building your software project" button.

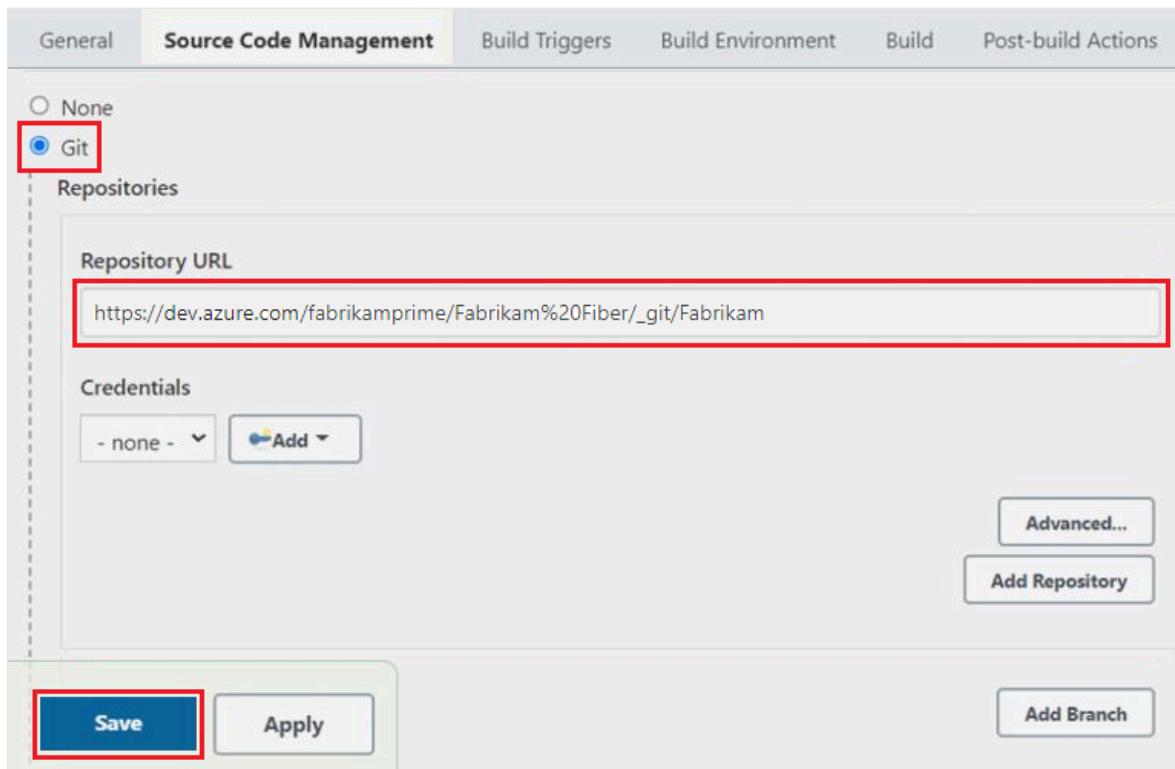
2. Select the type of build that's appropriate for your project.

The screenshot shows the "Enter an item name" dialog. A red box highlights the input field containing "Jenkins Pipeline Project". Below the input field is a note: "» Required field". The dialog lists three project types:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

A red box highlights the "OK" button at the bottom of the dialog.

3. Enter the URL for your Git repository.



## Set up the Jenkins service hook

1. In your Azure DevOps project, go to **Project settings > Service hooks** at [https://<organization-name>/<project-name>/\\_settings/serviceHooks](https://<organization-name>/<project-name>/_settings/serviceHooks).

The screenshot shows the Azure DevOps Project Settings Service Hooks page. On the left, there is a sidebar with icons for Project Settings, General, Teams, Permissions, Notifications, Service hooks (which is highlighted with a red box), Dashboards, Boards, and Project configuration. The main area is titled 'Service Hooks' and contains the sub-instruction: 'Integrate with your favorite services by notifying them when events happen in your project.' Below this is a table of existing service hooks:

Consumer	Event	Event Filter	Action
Web Hooks	Build completed	Any completed build.	Post via HTTP
Web Hooks	Code pushed	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request comment...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request created	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request merge att...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request updated	Any branch on any repository.	Post via HTTP
Web Hooks	Repository created	Any branch on any repository.	Post via HTTP
Web Hooks	Repository deleted	Repository [Any]	Post via HTTP

2. On the Service Hooks page, select the + icon or **Create subscription**.

The screenshot shows the 'Project Settings' interface for 'Fabrikam Fiber'. On the left, there's a sidebar with icons for General, Overview, Teams, Permissions, Notifications, Service hooks (which is selected and highlighted in blue), and Dashboards. The main area is titled 'Service Hooks' with the sub-instruction 'Integrate with your favorite services by notifying them'. A red box highlights the 'Create a new subscription...' button, which is a black button with a white plus sign. Below it is a table with columns for 'Event ↑', '...', and service names. The first row shows 'Build completed' with 'Web Hooks'. Subsequent rows show 'Code pushed', 'Pull request comment...', 'Pull request created', and 'Pull request merge att...'. Each row has a 'Web Hooks' entry.

3. On the Service screen, select Jenkins, and then select Next.

The screenshot shows the 'NEW SERVICE HOOKS SUBSCRIPTION' dialog. The title bar says 'NEW SERVICE HOOKS SUBSCRIPTION' with a close button. The main area is titled 'Service' with the sub-instruction 'Select a service to integrate with. Discover more integrations'. On the left is a list of services: App Center, AppVeyor, Azuqua, Azure App Service, Azure Service Bus, Azure Storage, Bamboo, Datadog, Grafana, Jenkins (which is selected and highlighted with a red box), Microsoft Teams, MyGet, Office 365, and Slack. To the right of Jenkins, there's a detailed description: 'Jenkins is an open source continuous integration service. Install the [Azure DevOps Server Plugin](#) on Jenkins to enhance traceability and integration.' Below this are sections for 'Supported events:' (Build completed, Code pushed, Pull request merge attempted, Release deployment completed) and 'Supported actions:' (Trigger generic build, Trigger Git build). At the bottom, there are buttons for 'Previous', 'Next' (which is highlighted with a red box), 'Test', 'Finish', and 'Cancel'.

4. On the Trigger screen, select and configure the Azure DevOps event you want to trigger a Jenkins build, and then select Next.

# Trigger

Select an event to trigger on and configure any filters.

Trigger on this type of event

Code pushed

**i** Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.

## FILTERS

Repository **i**

optional

Fabrikam

Branch **i**

optional

main

Pushed by a member of group: **i**

optional

[Any]

Previous

Next

Test

Finish

Cancel

5. On the **Action** screen, configure the Jenkins action to take when the event occurs.

6. Select **Test** to test the service hook, and **Finish** to finish the configuration.

Now when the event occurs in the Git repository, it triggers a Jenkins build.

### 💡 Tip

You can also create a service hook subscription programmatically with REST APIs.

For more information, see [Create a service hook subscription programmatically](#).

## Related content

- [Integrate with service hooks](#)
- [Service hooks events](#)
- [Create a service hook subscription programmatically](#)

---

# Feedback

Was this page helpful?

 Yes

 No

Provide product feedback ↗

# Create a service hook for Azure DevOps Services and TFS with Trello

03/25/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

Create cards and lists in Trello in response to events from Azure DevOps. For example, when code is pushed, or a build occurs.

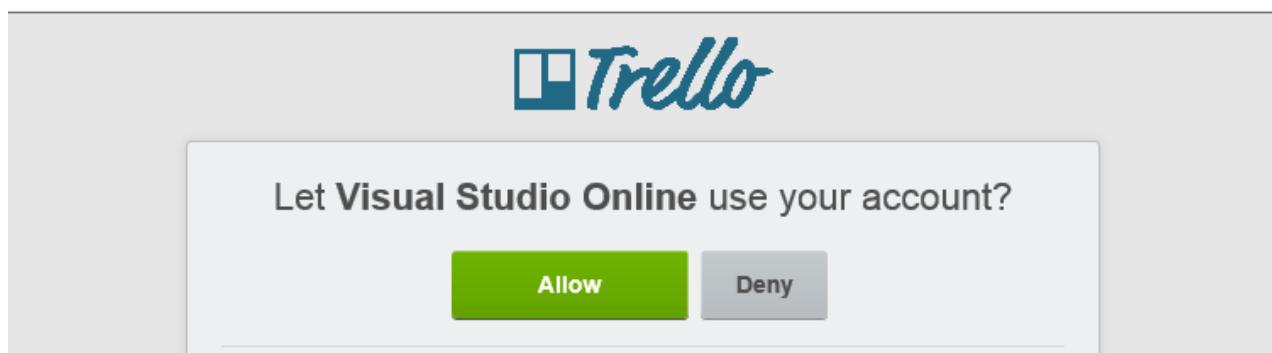
## Prerequisites

[ ] Expand table

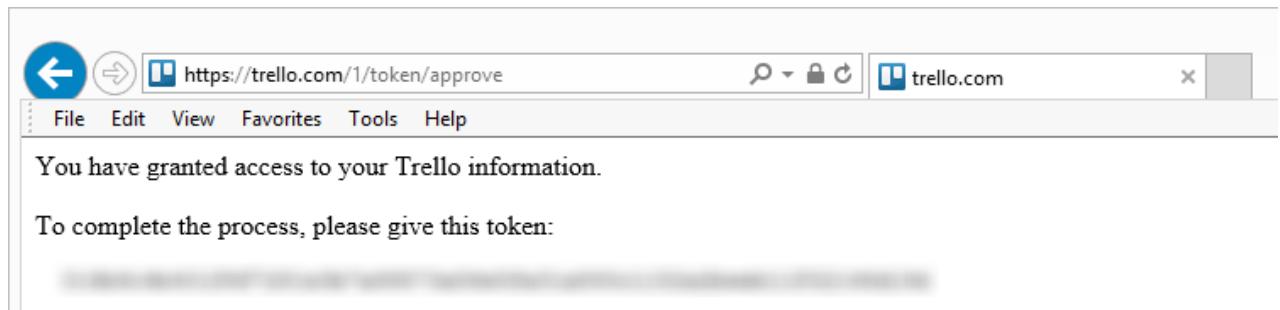
Category	Requirements
Permissions	<ul style="list-style-type: none"><li>- Member of the <a href="#">Project Collection Administrators group</a>. Organization owners are automatically members of this group.</li><li>- <b>Edit subscriptions</b> and <b>View subscriptions</b> permissions set to <b>Allow</b>. By default, only project administrators have these permissions. To grant the permissions to other users, you can use the command-line tool or the <a href="#">Security REST API</a>.</li></ul>
Tools	Trello

## Get a Trello authorization token

1. If you don't have a Trello account, sign up at [Trello](#).
2. Go to the [Authorize Azure DevOps Services for Trello page](#), and sign in with your Trello credentials.
3. Allow Azure DevOps to use your Trello account.



4. Copy the authorization token.



Create a Trello card or list from an Azure DevOps Services event.

1. Go to your project Service Hooks page:

[https://{{orgName}}/{{project\\_name}}/\\_settings/serviceHooks](https://{{orgName}}/{{project_name}}/_settings/serviceHooks)

A screenshot of the Azure DevOps Project Settings Service Hooks page. The left sidebar shows "Project Settings" for "Fabrikam Fiber" with sections for General, Notifications, and Boards. The "Service hooks" section is highlighted with a red box. The main area shows a table of existing service hook subscriptions:

Consumer	Event	Event Filter	Action
Web Hooks	... Build completed	Any completed build.	Post via HTTP
Web Hooks	Code pushed	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request comment...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request created	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request merge att...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request updated	Any branch on any repository.	Post via HTTP
Web Hooks	Repository created	Any branch on any repository.	Post via HTTP
Web Hooks	Repository deleted	Repository [Any]	Post via HTTP

Select **Create Subscription**.

2. Pick the Trello service.

**brikam-Fiber** NEW SERVICE HOOKS SUBSCRIPTION x

Service

Select a service to integrate with. [Discover more integrations](#)

ing them whe

ct

AppVeyor

Azuqua

Azure Service Bus

Azure Storage

Campfire

Flowdock

HipChat

Hubot

Jenkins

Kato

MyGet

Slack

**Trello**

UserVoice

**Trello**

Provides integration with Trello

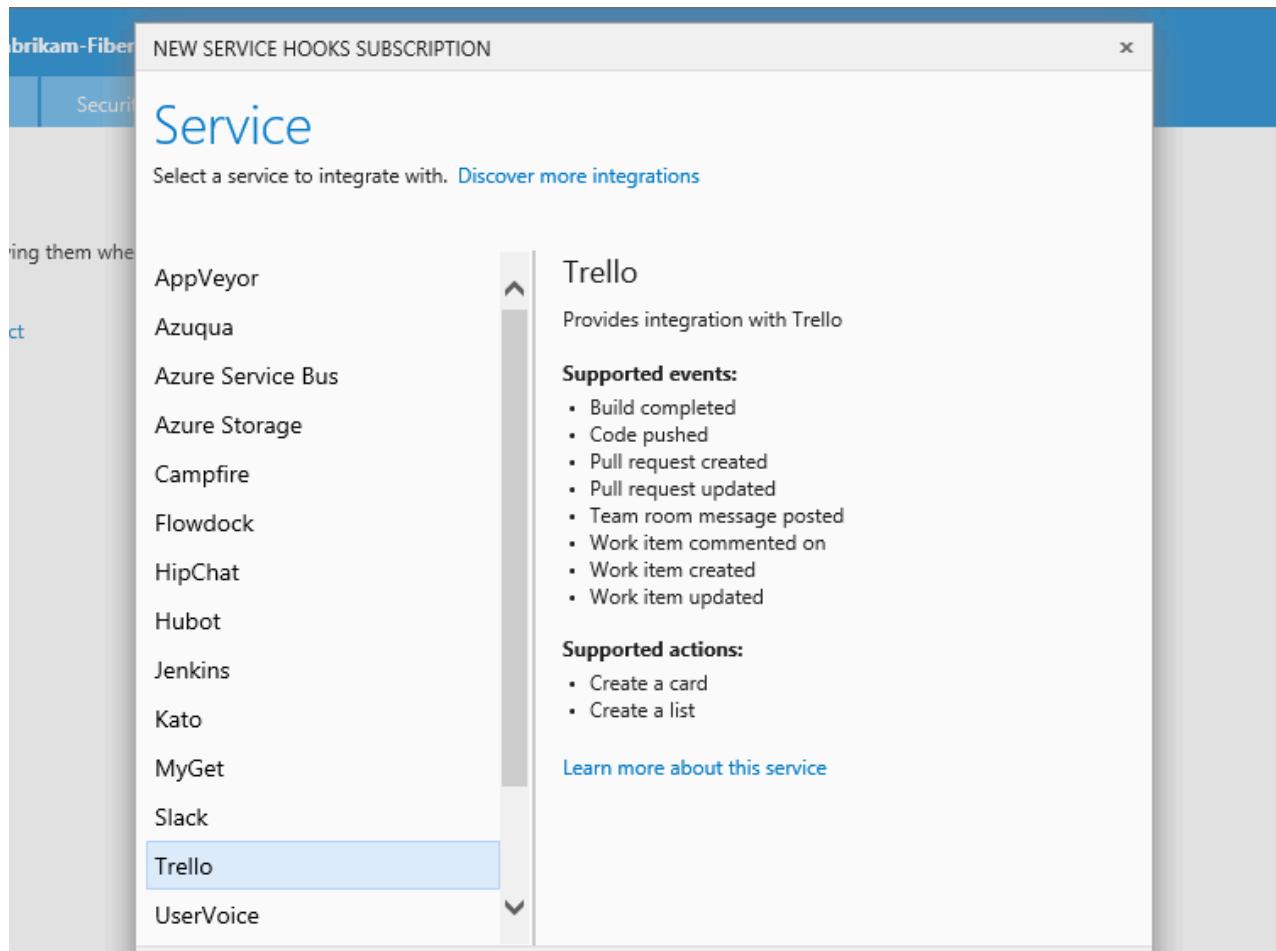
**Supported events:**

- Build completed
- Code pushed
- Pull request created
- Pull request updated
- Team room message posted
- Work item commented on
- Work item created
- Work item updated

**Supported actions:**

- Create a card
- Create a list

[Learn more about this service](#)



3. Configure the triggering Visual Studio event. In this case, we're going to have the subscription respond when a work item is created.

**brikam-Fiber** NEW SERVICE HOOKS SUBSCRIPTION x

Trigger

Select an event to trigger on and configure any filters.

ing them whe

ct

Trigger on this type of event

Work item created

i Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.

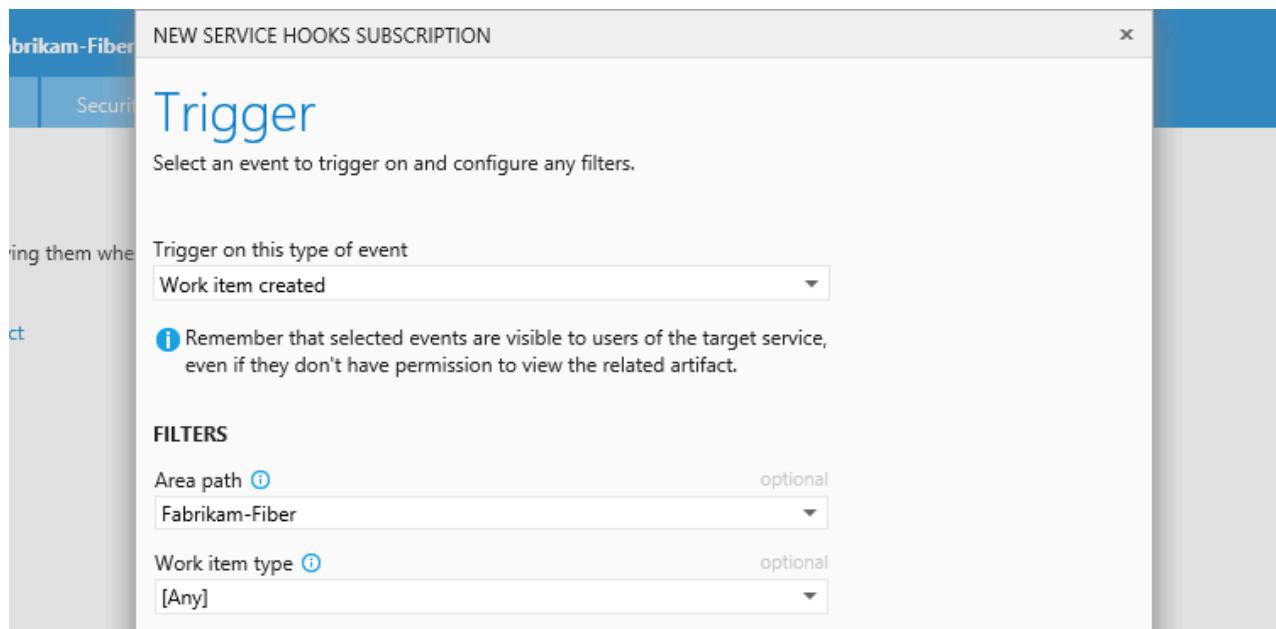
**FILTERS**

Area path i optional

Fabrikam-Fiber

Work item type i optional

[Any]



4. Set up the action that Trello will take in response to the trigger - either create a card or a list.

**brikam-Fiber**

**NEW SERVICE HOOKS SUBSCRIPTION**

**Action**

Select and configure the action to perform.

Perform this action

Create a card

This action creates a card on an existing list in Trello. A card can represent a task, issue, event, or just about anything. A card's state is typically determined by what list it is on. [Learn More](#)

**SETTINGS**

User token (need one? [Get it now](#)) required

Board i required  
Welcome Board

List i required  
Basics

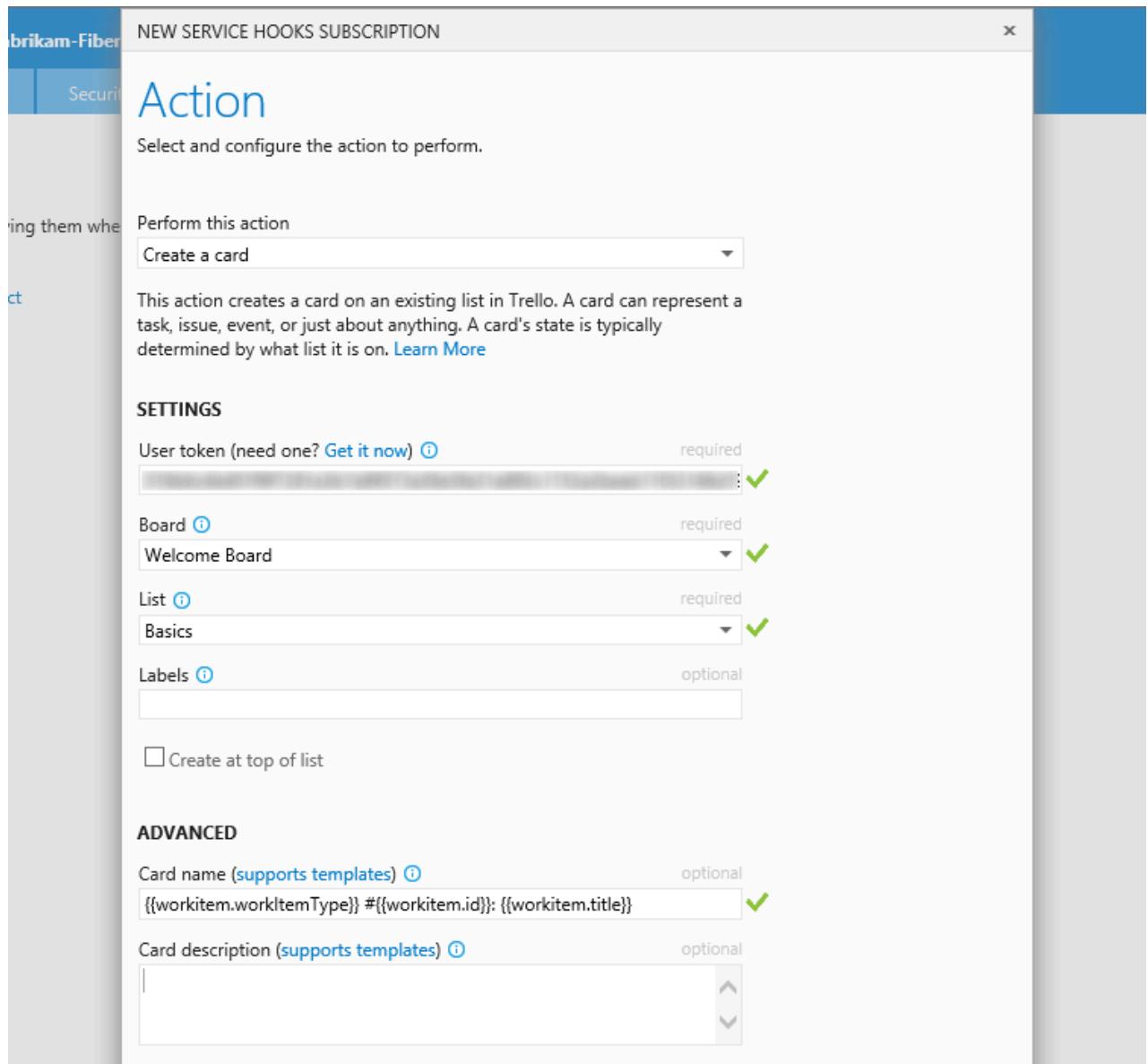
Labels i optional

Create at top of list

**ADVANCED**

Card name (supports templates) optional  
`{{workitem.workItemTypeId}} #{{workitem.id}}: {{workitem.title}}`

Card description (supports templates) optional  
`|`



You can use placeholders to insert content from the event into the name and description of the cards or lists that the subscription creates. For example, when we run the test notification, the card that gets created is named "Bug #5: Some great new idea!" because the test work item is a bug (ID=5) with the title "Some great new idea!".

## 5. Test the service hook subscription and finish the wizard.

**TEST NOTIFICATION**

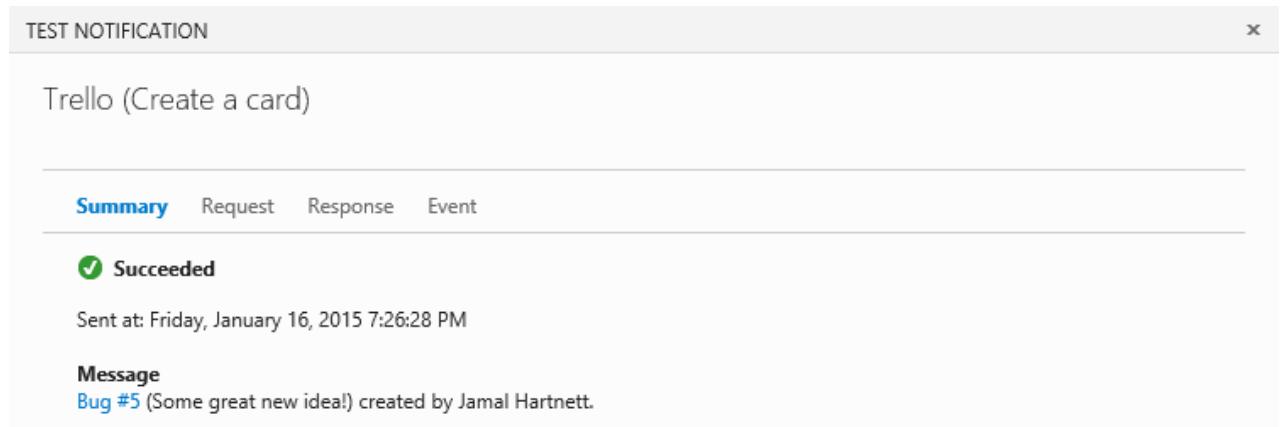
Trello (Create a card)

**Summary** Request Response Event

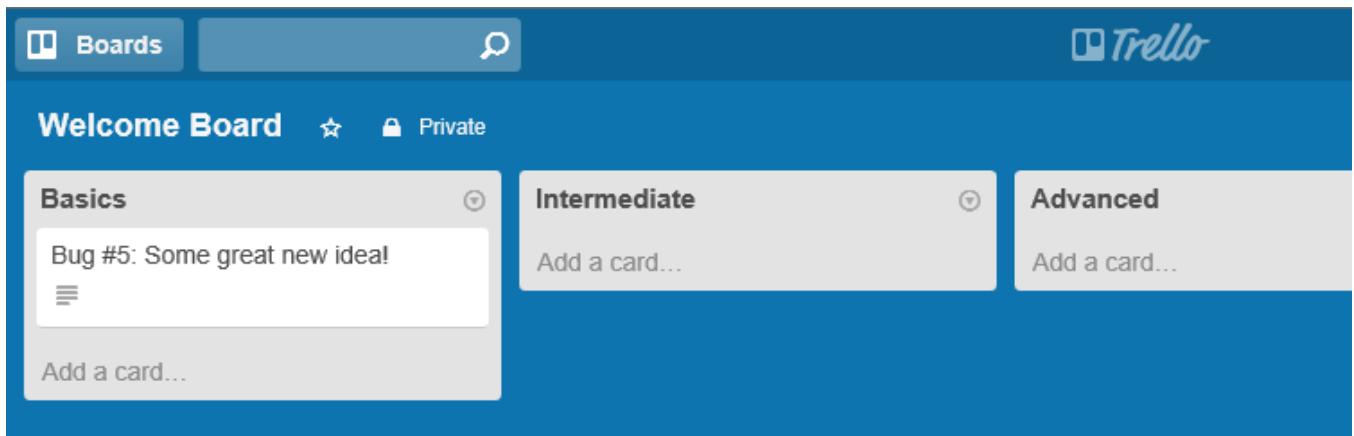
**Succeeded**

Sent at: Friday, January 16, 2015 7:26:28 PM

**Message**  
Bug #5 (Some great new idea!) created by Jamal Hartnett.



Now it's set up. Go to Trello and see the cards appear.



## Placeholders

You can insert placeholders in the name of a list or the name or description of a card to be created by your subscription. When a list or card is created, the placeholders will be replaced by values from the event that was raised. The example we used for the card name in this page uses:

```
 {{workitem.workItemType}} #{{workitem.id}}: {{workitem.title}}
```

So when a bug is created with the ID 5 and title "Some great new idea!", the card name is "Bug #5: Some great new idea!".

The basic form of the placeholder is  `{{resource.field}}` where resource is the name of the resource raising the event (work item, build, and so on) and field is a field within the resource section of the event, like ID. So, if the subscription is for a completed build, it might be something like

```
Build {{build.id}} completed at {{build.finishTime}}
```

Where build is the name of the resource and ID, finishTime are the fields available in this resource. To understand the event types, fields and resources available to use, look at the [events reference](#).

## Work item fields

Work item fields appear in the event in the fields array, like this example:

```

    " fields": {
        " System.AreaPath": "Fabrikam-Fiber-Git",
        " System.TeamProject": "Fabrikam-Fiber-Git",
        " System.IterationPath": "Fabrikam-Fiber-Git",
        " System.WorkItemType": "Product Backlog Item",
        " System.State": "New",
        " System.Reason": "New backlog item",
        " System.CreatedDate": "2014-11-05T21:11:28.85Z",
        " System.CreatedBy": "Normal Paulk",
        " System.ChangedDate": "2014-11-05T21:11:29.23Z",
        " System.ChangedBy": "Normal Paulk",
        " System.Title": "Test PBI",
        " Microsoft.Azure DevOps Services.Common.BacklogPriority": 999999999,
        " WEF_6CB513B6E70E43499D9FC94E5BBFB784_Kanban.Column": "New"
    },

```

Working directly from the event definition, we would have created our card name like this example:

```

{{workitem.fields["System.workItemType"]}} #{{workitem.fields["System.id"]}}:
{{workitem.fields["System.title"]}}

```

As a shortcut, you can reference any fields in the System or Microsoft.Azure DevOps Services.Common namespaces as if they were fields of the resource. So

`{{workitem.fields["System.workItemType"]}}` becomes  `{{workitem.workItemType}}`.

## Placeholder expressions

These placeholders use a [Handlebars templates](#) implementation that is largely compatible with [Mustache](#). Some useful expressions include

[ ] [Expand table](#)

Type of expression	examples
basic expressions	<code> {{workitem.name}}</code>
array expressions	<code> {{pullRequest.reviewers.[0].displayName}}</code>
Mustache sections	<code> {{#workitem.assignedTo}} This WI is assigned {{/workitem.assignedTo}}</code>
Mustache Inverted Sections	<code> {{^workitem.assignedTo}} This WI isn't assigned {{/workitem.assignedTo}}</code>

Type of expression	examples
Handlebars block helpers	with if/else unless each
Handlebars paths	... this For example, <code>{{../comment/id}}</code> or <code>{{this/title}}</code>
Template comments	<code>{{!-- this is a handlebar comment --}}</code>

## Pricing

Azure DevOps doesn't charge for the framework for integrating with external services. Check out the specific service's site for pricing related to their services.

## Q & A

### Q: Can I programmatically create subscriptions?

A: Yes, see details [here](#).

### Q: Can I get more information about Trello?

A: Yes, [trello.com](https://trello.com).

# Create a service hook for Azure DevOps with Datadog

06/26/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

You can create events and metrics in Datadog in response to events in Azure DevOps. In Datadog, you can use these metrics and events to create dashboards, troubleshoot issues, and create monitors to alert you to critical issues. Datadog accepts all Azure DevOps event types.

This article shows you how to use service hooks to send Azure DevOps events to Datadog.

## Prerequisites

[ ] Expand table

Category	Requirements
Permissions	<ul style="list-style-type: none"><li>- Member of the <a href="#">Project Collection Administrators group</a>. Organization owners are automatically members of this group.</li><li>- <b>Edit subscriptions</b> and <b>View subscriptions</b> permissions set to <b>Allow</b>. By default, only project administrators have these permissions. To grant the permissions to other users, you can use the command-line tool or the <a href="#">Security REST API</a>.</li></ul>
Tools	<a href="#">Datadog</a> . In the Datadog application, go to your profile and then select <b>Organization Settings &gt; API Keys</b> . Create a new key or select an existing one, and then copy the key to your clipboard.

## Send Azure DevOps events to Datadog

To send Azure DevOps events to Datadog, you set up a subscription for each type of event.

### Create a subscription for an event

1. Go to your Azure DevOps project, select **Project settings**, and then select **Service hooks**. Alternately, go to [https://{{organization-name}}/{{project-name}}/\\_settings/serviceHooks](https://{{organization-name}}/{{project-name}}/_settings/serviceHooks).
2. Select **Create subscription**.

The screenshot shows the Azure DevOps interface for a project named 'demo-project'. The left sidebar includes links for Overview, Boards, Repos, Pipelines, Test Plans, Artifacts, and Project settings. The main content area is titled 'Project Settings' for 'demo-project' and shows the 'Service hooks' section. A sub-menu on the right lists General, Overview, Teams, Permissions, Notifications, Service hooks (which is selected and highlighted in blue), Dashboards, and Boards. A large button labeled '+ Create subscription' is visible.

3. In the list of services, select **Datadog**, and then select **Next**.

The screenshot shows the 'NEW SERVICE HOOKS SUBSCRIPTION' dialog. On the left, a list of services includes App Center, AppVeyor, Azuqua, Azure App Service, Azure Service Bus, Azure Storage, Bamboo, **Datadog** (which is highlighted with a red box), Jenkins, Microsoft Teams, MyGet, Office 365, Slack, and Trello. On the right, details for the selected 'Datadog' service are displayed: 'A monitoring and analytics platform for modern cloud environments.', 'Supported events: All events', 'Supported actions: Post an event in Datadog', and a link to 'Learn more about this service'. At the bottom, there are buttons for Previous, Next (which is highlighted with a red box), Test, Finish, and Cancel.

4. Select an event to trigger on, configure any filters that you want to use, and then select **Next**.

# Trigger

Select an event to trigger on and configure any filters.

Trigger on this type of event

Run job state changed



- i** Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.

## FILTERS

Pipeline [\(i\)](#)

optional

[Any]

Stage [\(i\)](#)

optional

[Any]

Job [\(i\)](#)

optional

[Any]

State [\(i\)](#)

optional

Completed



Result [\(i\)](#)

optional

Succeeded



Previous

Next

Test

Finish

Cancel

5. Configure the action to perform when the event happens:

- Under **Datadog API Key**, enter your Datadog API key.
- Under **Datadog Account Type**, select your account type. You can determine your account type from the hostname of the URL that your Datadog account uses.

Expand table

URL hostname	Account type
app.datadoghq.com	US
app.datadoghq.eu	EU
us3.datadoghq.com	US3

URL hostname	Account type
us5.datadoghq.com	US5
ap1.datadoghq.com	AP1
app.dog-gov.com	GOV

6. To verify that Azure DevOps can use your configuration settings and successfully create a subscription, select **Test**.

7. To finish creating the subscription, select **Finish**.

**NEW SERVICE HOOKS SUBSCRIPTION**

## Action

Select and configure the action to perform.

Perform this action

Post an event in Datadog

Create an event and corresponding metric(s) in Datadog whenever this service hook is triggered.

### SETTINGS

Datadog API Key (i) required  
AA11BB22CC33DD44EE55FF66AA77BB88

Datadog Account Type (i) required  
US Datadog Account

**Previous** **Next** **Test** **Finish** **Cancel**

## Add subscriptions for other events

Repeat the steps in [Create a subscription for an event](#) for each event type you want to send to Datadog. Datadog accepts and encourages users to send all event types.

# Use your data in Datadog

As events occur and their data and metrics start to flow into Datadog, you can set up dashboards and monitors. To get started, go to [Datadog](#).

## FAQs

### Q: Can I create service hook subscriptions programmatically?

A: Yes. For more information, see [Create a service hook subscription programmatically](#). Your Datadog account type determines the endpoint that your subscription should submit requests to. Use one of the following endpoints:

[ ] [Expand table](#)

Account type	Endpoint
US	<code>https://app.datadoghq.com/intake/webhook/azuredevops?api_key=&lt;API-key&gt;</code>
EU	<code>https://app.datadoghq.eu/intake/webhook/azuredevops?api_key=&lt;API-key&gt;</code>
US3	<code>https://us3.datadoghq.com/intake/webhook/azuredevops?api_key=&lt;API-key&gt;</code>
US5	<code>https://us5.datadoghq.com/intake/webhook/azuredevops?api_key=&lt;API-key&gt;</code>
AP1	<code>https://ap1.datadoghq.com/intake/webhook/azuredevops?api_key=&lt;API-key&gt;</code>
Gov	<code>https://app.ddog-gov.com/intake/webhook/azuredevops?api_key=&lt;API-key&gt;</code>

### Q: How can I use these events in Datadog?

A: Azure DevOps events that are sent to Datadog are useful for creating dashboards, setting up monitors, and finding correlations during troubleshooting. You can also use event data to get insights into how processes in your developer operations affect application performance.

### Q: What event types can I send to Datadog?

A: Datadog accepts all event types.

### Q: Can I get more general information about Datadog?

A: Yes, see [datadoghq.com](#).

# Azure DevOps integration with Microsoft Teams

Article • 02/08/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

This article describes how you can integrate your Azure DevOps project activities directly into your Microsoft Teams channels. For example, you can see the following Azure DevOps events in your Teams channels:

- Work item updates
- Pull requests
- Code commits
- Builds
- Release deployments and approvals

## Requirements and limitations

[ ] Expand table

Category	Requirements
Permissions	<ul style="list-style-type: none"><li>- To create Azure DevOps subscriptions in Teams: Member of the <b>Project Administrators</b> group.</li><li>- To receive Azure DevOps notifications in Teams: <b>Third party application access via OAuth</b> enabled in your Azure DevOps organization settings.</li></ul>
Microsoft Entra	Only Azure DevOps organizations in the same Microsoft Entra tenant as your Microsoft Teams account can be connected. Even if your email addresses are the same for Azure DevOps and Microsoft Teams, they can't be linked if they're backed by different tenants. For more information, see <a href="#">Access via Microsoft Entra FAQs</a> .

### ⓘ Note

Office 365 Connectors within Teams are retired. For more information, see [Retirement of Office 365 connectors within Microsoft Teams](#). Features that provide similar functionality to Office 365 Connectors with better scalability and security include [Workflows](#) or the Azure DevOps Services apps for Teams.

# Azure DevOps Services apps for Teams

## ⓘ Note

This feature is only available on Azure DevOps Services. Typically, new features are introduced in the cloud service first, and then made available on-premises in the next major version or update of Azure DevOps Server. For more information, see [Azure DevOps Feature Timeline](#).

You can use the following apps that offer rich features to integrate with Microsoft Teams.

## Azure Boards app for Teams

The Azure Boards app for Teams helps you easily create and monitor work items from your Teams channels. You can create work items by using a command, or use message actions to convert conversations in the channel into work items. You can also set up and manage subscriptions to get notifications in your channel whenever work items are created or updated. For more information, see [Use the Azure Boards app in Microsoft Teams](#).

## Azure Pipelines app for Teams

The Azure Pipelines app for Teams helps you easily monitor events in your pipelines from your Teams channels. You can set up and manage subscriptions for completed builds, releases, and pending approvals, and get notifications for these events in your channels. You can also approve builds and release deployments from your channels. For more information, see [Integrate Azure Pipelines with Microsoft Teams](#).

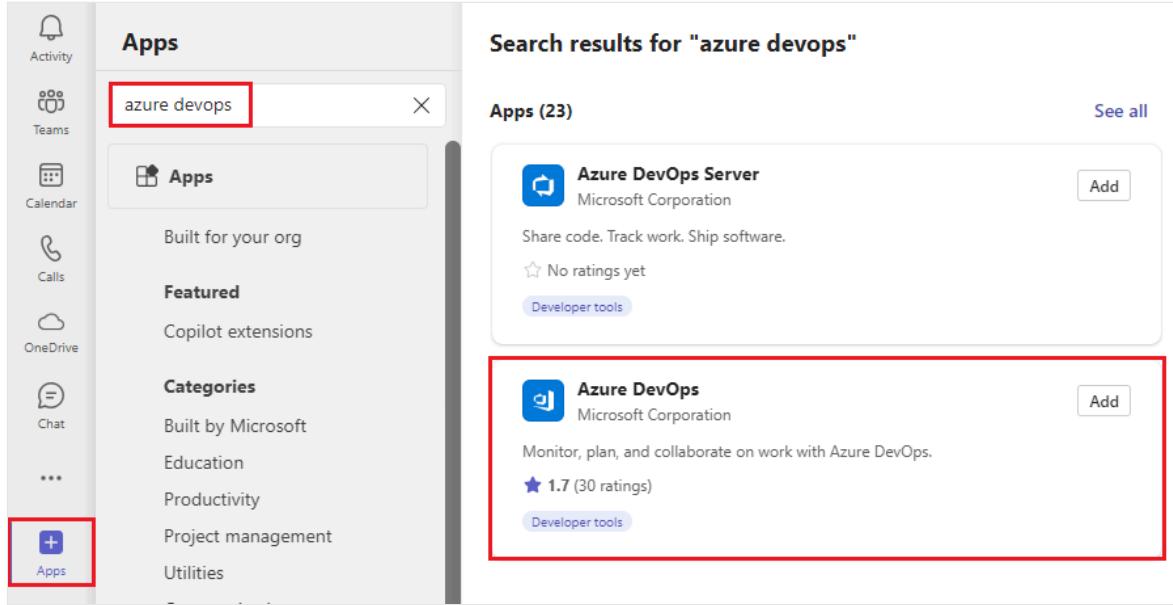
## Azure Repos app for Teams

The Azure Repos app for Teams helps you easily monitor events in your repositories from your Teams channels. You can set up and manage subscriptions for code commits, pull request (PR) creation, and PR updates, and get notifications for these events in your channels. For more information, see [Use Azure Repos with Microsoft Teams](#).

## Add and configure the Azure DevOps tab in Teams

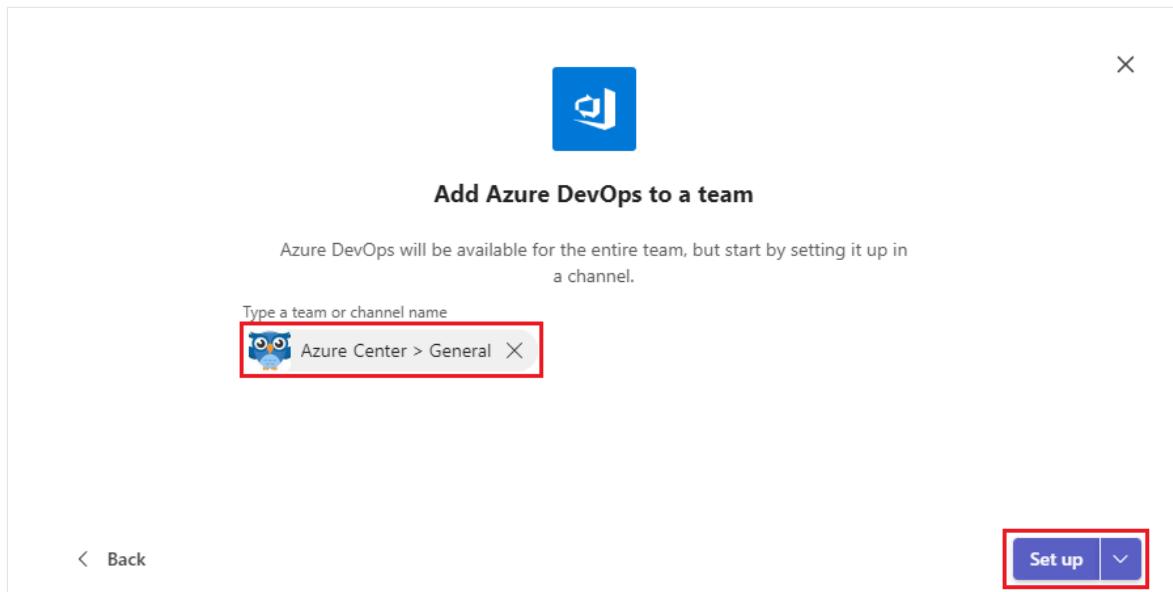
To bring your project dashboard or Kanban board into Teams, you can install the Azure DevOps app in a tab in your Teams channel. The Azure DevOps app lets you insert content from the app in messages, and get notifications from the app in your channels.

1. In Teams, select Apps from the left menu and then search for **Azure DevOps**.

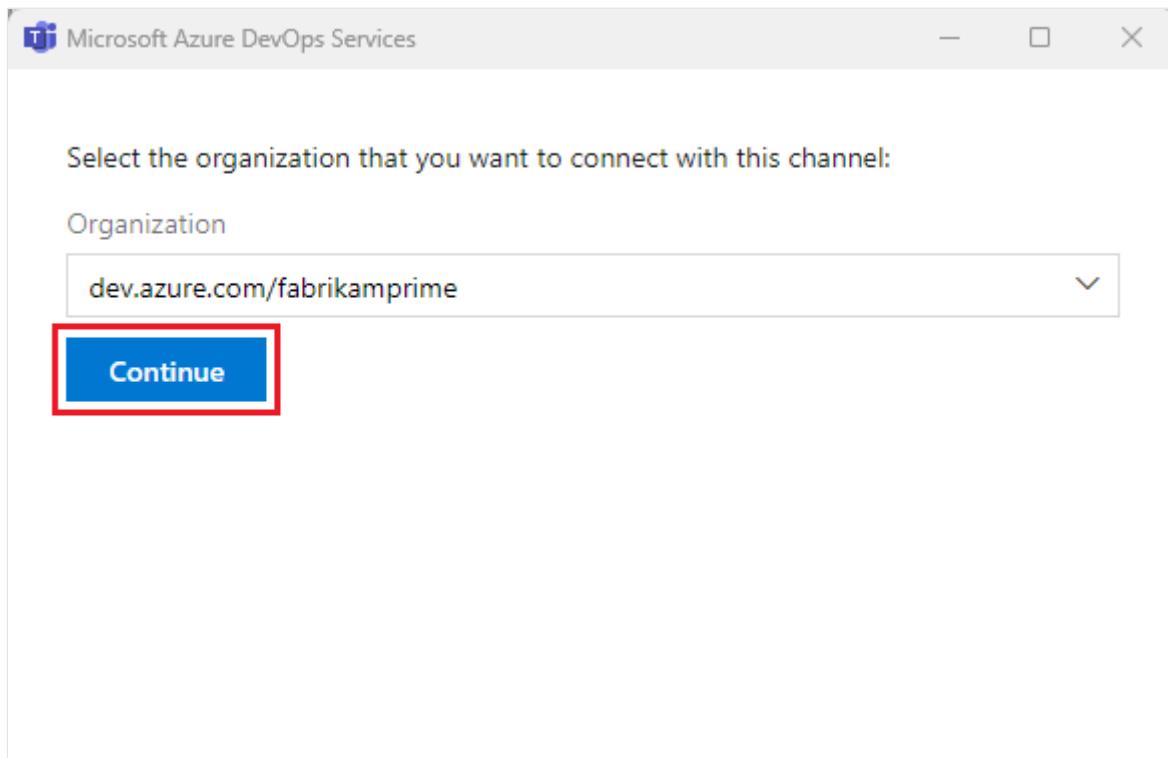


2. Select **Azure DevOps**, and then select **Add to a team**.

3. Select the Teams channel to add to the app to, and then select **Set up**.



4. Select **Select organization**, select your organization, and then select **Continue**.



5. On the Azure DevOps screen, select a **Project**, and whether to add a **Dashboard** or a **Kanban board** to the tab. Select other configurations depending on your choice, and select whether you want to post to the channel about adding the tab.



Connected to account: https://dev.azure.com/fabrikamprime/

Add a Dashboard

Add a Kanban board

Dashboards provide easy-to-read, easy access, real-time information. [Learn more about Dashboards](#)

Project

Fabrikam Fiber

Dashboard type

Team

Team

Fabrikam Fiber Team

Dashboard

Overview



Post to the channel about this tab

Back

Save

6. Select **Save**. The new tab and board appear in your channel.

The screenshot shows the Microsoft Teams General channel dashboard. At the top, there's a navigation bar with icons for Home, Posts, Files, and a search bar containing 'Fabrikam Fiber Team ...'. A red box highlights the search bar. Below the navigation, there are several cards:

- Agile Lead Time**: Last 30 days. Shows a message: 'Last time you checked there were 0 results' and an icon of hands with magnifying glasses over a cloud.
- Fabrikam Fiber Team Lead Time**: Last 30 days. Shows a message: 'Last time you checked there were 0 results' and an icon of hands with magnifying glasses over a cloud.
- Future sprint**: June 14 - June 28. Shows a grey progress bar.
- New Work Item**: Fields for 'Enter title' (placeholder 'Bug') and a dropdown menu. A 'Create' button is at the bottom right.
- Visual Studio**: Buttons for 'Open in' (Requires Visual Studio) and 'Get Visual Studio' (See Visual Studio)
- Work in Progress**: Shows 178 work items.
- Fabrikam Fiber Team Velocity**: Last 6 iterations. Shows a chart with a single data point at 4.
- Count of work items**: Average Velocity 0
- New Work Item**: Fields for 'Enter title' (placeholder 'Bug')

To manage your Teams apps and tabs, select **Apps** in the Teams menu and then select **Manage your apps** at the bottom of the **Apps** panel.

## Related content

- Use the Azure Boards app in Microsoft Teams
- Integrate Azure Pipelines with Microsoft Teams
- Use Azure Repos with Microsoft Teams
- Install the Workflows app in Microsoft Teams

## Feedback

Was this page helpful?

Yes

No

Provide product feedback ↗

# Webhooks

07/15/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

This article describes webhooks and how to set them up for your Azure DevOps project. Webhooks provide a way to send a JSON representation of an Azure DevOps event to any service that has a public endpoint.

## ⓘ Note

Azure DevOps doesn't charge for setting up service hooks or integrating with external services. Refer to the specific service's site for pricing related to their services.

## Prerequisites

  Expand table

Category	Requirements
Permissions	- Member of the <a href="#">Project Collection Administrators group</a> . Organization owners are automatically members of this group.
Project and service	- A project in the organization - A service with a public HTTPS endpoint to which you want to send Azure DevOps events.

## ⓘ Important

- Use only HTTPS endpoints. HTTP has the potential to send private data, including authentication headers, unencrypted in the event payload. You must use HTTPS for basic authentication on a webhook.
- If you're connecting to a service behind a virtual private network, ensure that Azure DevOps IP addresses are allowed for inbound connections. See [Inbound Connections](#).

## Send JSON representation to a service

1. In your Azure DevOps project, go to **Project settings > Service hooks** at

[https://<organization-name>/<project-name>/\\_settings/serviceHooks](https://<organization-name>/<project-name>/_settings/serviceHooks).

The screenshot shows the 'Service Hooks' page within the 'Project Settings' of an Azure DevOps project named 'Fabrikam Fiber'. The left sidebar has a 'Notifications' section with 'Service hooks' highlighted by a red box. The main area displays a table of service hook subscriptions:

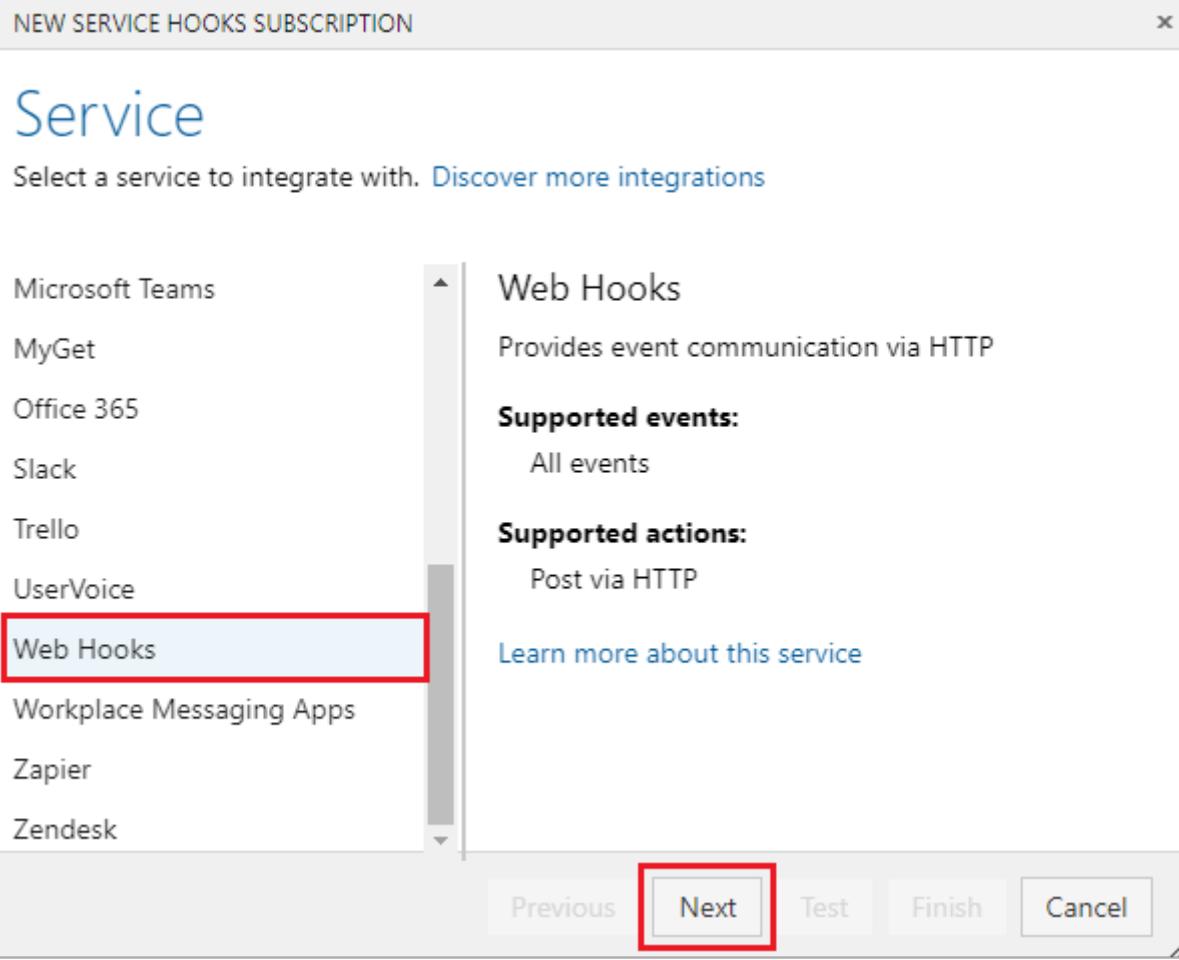
Consumer ↑	Event ↑	Event Filter	Action
Web Hooks	... Build completed	Any completed build.	Post via HTTP
Web Hooks	Code pushed	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request comment...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request created	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request merge att...	Any branch on any repository.	Post via HTTP
Web Hooks	Pull request updated	Any branch on any repository.	Post via HTTP
Web Hooks	Repository created	Any branch on any repository.	Post via HTTP
Web Hooks	Repository deleted	Repository [Any]	Post via HTTP

2. On the **Service Hooks** page, select the **+** icon or **Create subscription**.

The screenshot shows the 'Service Hooks' page within the 'Project Settings' of the same 'Fabrikam Fiber' project. A red box highlights the '+' icon in the top toolbar, which is also shown in a larger callout box with the text 'Create a new subscription...'. The main area displays a table of service hook subscriptions:

Event ↑	Action
... Build completed	Post via HTTP
Code pushed	Post via HTTP
Pull request comment...	Post via HTTP
Pull request created	Post via HTTP
Pull request merge att...	Post via HTTP

3. On the **Service** screen, select **Web Hooks** and then select **Next**.



4. On the **Trigger** screen, select and configure the Azure DevOps event you want to trigger the webhook on, and then select **Next**. For more information about the JSON payloads posted by the Azure DevOps consumer, see [Service hooks events](#).

# Trigger

Select an event to trigger on and configure any filters.

Trigger on this type of event

Code pushed

i Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.

## FILTERS

Repository i

optional

Fabrikam

Branch i

optional

[Any]

Previous

Next

Test

Finish

Cancel

5. On the **Action** screen, configure the target URL and what to do when the event occurs. For more information about what to enter in **Resource details to send**, **Messages to send**, and **Detailed messages to send**, see [Resource details to send](#).

! Note

Webhooks can't target localhost (loopback) or special range [IPv4 ↴](#) / [IPv6 ↴](#) addresses.

6. Select **Test** to test the service hook subscription.

## NEW SERVICE HOOKS SUBSCRIPTION

### Action

Select and configure the action to perform.

Perform this action

Post via HTTP

This action posts a JSON object representation of the event to the specified URL.

It's recommended that you only use HTTPS endpoints due to the potential for private data including any authentication headers in the event payload. [Learn more about Webhooks](#)

#### SETTINGS

URL (i)

required

`https://requestb.in/00000000`

Previous

Next

**Test**

Finish

Cancel

- When the test completes successfully, close the notification screen, and then select **Finish** on the **Action** screen.

## TEST NOTIFICATION

Web Hooks (Post via HTTP)

[Summary](#) Request Response Event

✓ **Succeeded**

Sent at: Tuesday, August 6, 2024 8:45:00 PM

#### Message

Tester pushed updates to Fabrikam-Fiber-Git:main.

**Close**

- Now the webhook is set up. Go to the target service to view the JSON representation.

The screenshot shows a RequestBin interface. At the top, there's a blue circular icon with a green arrow pointing right, followed by the text "RequestBin". To the right is another blue circular icon with a white dot, and the URL "https://requestb.in/00000000". Below this is a table with the following data:

<a href="https://requestb.in">https://requestb.in</a>	</> application/json; charset=utf-8	5m ago
<b>POST</b> /00000000	3.22 KB	From 000.000.00.00, 000.000.00.00

Below the table, there are two sections: "FORM/POST PARAMETERS" and "HEADERS". The "FORM/POST PARAMETERS" section contains "None". The "HEADERS" section lists numerous HTTP headers, including X-Request-Id, Total-Route-Time, Connect-Time, Cf-LpCountry, Host, Connection, Accept-Encoding, Cf-Visitor, Content-Type, Via, Content-Length, Cf-Connecting-IP, and Cf-Ray.

Under "RAW BODY", the JSON payload is shown:

```
{"subscriptionId": "00000000-0000-0000-0000-000000000000", "notificationId": 1, "id": "00000000-0000-0000-0000-000000000000", "eventType": "git.push"}
```

### 💡 Tip

You can also create a webhook programmatically. For more information, see [Create a service hook subscription programmatically](#).

## Resource details to send

The **Resource details to send**, **Messages to send**, and **Detailed messages to send** settings in the **Action** pane control the size of the JSON payload to send. **Resource details to send** controls how much of the resource to send. The default is **All**, but you can also choose to send **Minimal**, which sends only key fields like URL and ID, or **None**.

**None** and **Minimal** are useful in scenarios where the caller doesn't need much or any information about the resource, because it relies on the message or detailed message itself. **None** and **Minimal** are also useful for security reasons. The caller must call back into Azure DevOps Services and go through normal security and permission checks to get more details about the resource.

The following sample JSON shows minimal details about the resource:

JSON

```
{
  "eventType": "git.push",
  ...
  "messages": {
    "text": "...",
    ...
  }
}
```

```
        "html": "...",
        "markdown": "..."

    },
    "detailedMessage": {
        "text": "...",
        "html": "...",
        "markdown": "..."

    },
    "resource": {
        "id": "...",
        "url": "https://...",
        "name": "...",
        "field1": "..."

    }
}
```

## Related content

- [Integrate with service hooks](#)
- [Service hooks events](#)
- [Create a service hook subscription programmatically](#)

# Manage authorization of services to access Azure DevOps

07/10/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

When you integrate a service with Azure DevOps, you can grant the service access to your Azure DevOps resources, such as work items, source code, and build results.

Azure DevOps uses OpenID Connect (OIDC)-based authentication to grant the service access to your resources.

- Authorizations are bound to your credentials, so the service can use an authorization to access your resources in Azure DevOps.
- You use your Microsoft account or your work account to authorize the service.
- The service that you authorize doesn't have access to your Azure DevOps credentials.

## Prerequisites

 Expand table

Category	Requirement
Project access	<a href="#">Project member</a> .
Access levels	At least <b>Basic</b> access.
Permissions	Member of the <a href="#">Project Collection Administrators group</a> . Organization owners are automatically members of this group.

## Authentication frameworks

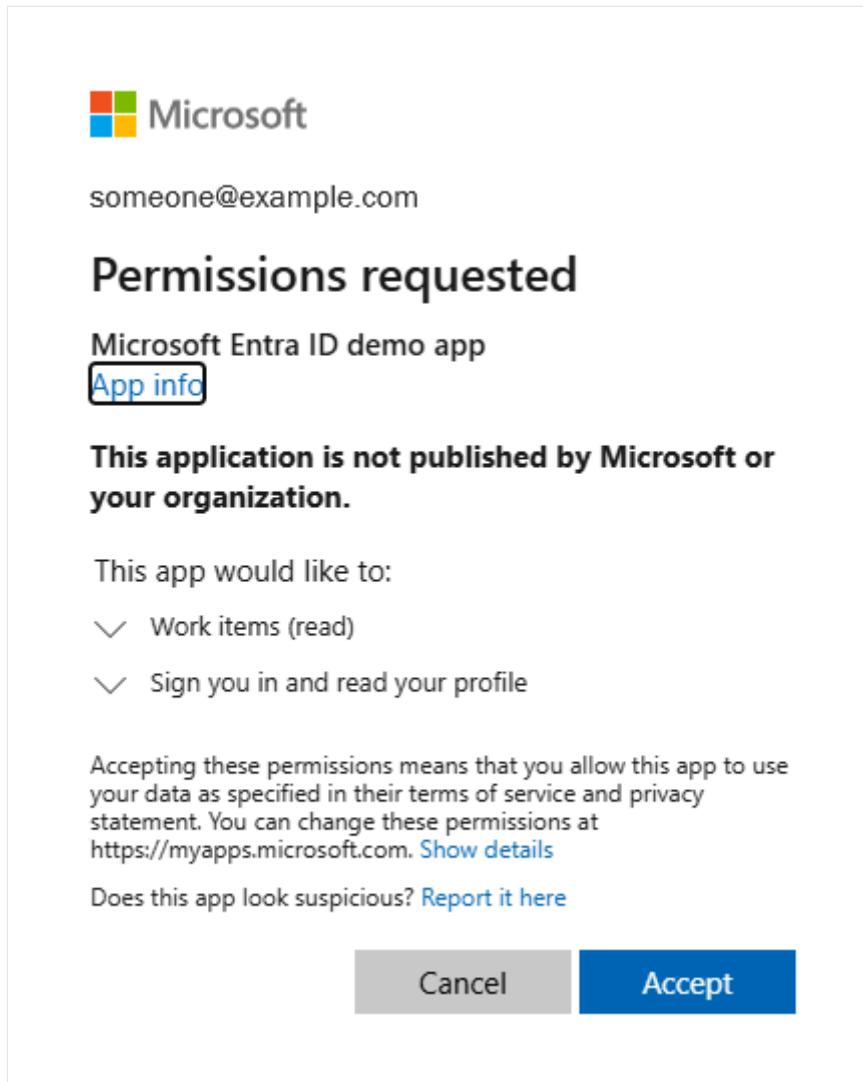
When you build an application on top of Azure DevOps REST APIs, you can use OIDC-based authentication by [registering your application in Microsoft Entra ID](#). For more information, see [What is the Microsoft identity platform?](#).

Some older apps use an implementation of OAuth 2.0 to get access tokens for Azure DevOps resources. Registrations of these Azure DevOps OAuth applications are no longer supported, because Azure DevOps OAuth is slated for deprecation in 2026. For more information, see [No new Azure DevOps OAuth apps beginning April 2025 ↗](#).

# Authorize a service

A typical authorization flow might include the following steps:

1. You use a service that uses Azure DevOps resources, so the service requests authorization.
2. To initiate the authentication process for the service, the registered app opens a Microsoft Entra ID website that prompts you to select an account.
3. After you select an account, the authorization approval page appears.



4. You review the request and approve the authorization.
5. The authorized service uses that authorization to access resources in your Azure DevOps organization.

To ensure an authorization request is legitimate, take the following precautions:

- Pay attention to any HTTPS-related security warnings in your browser.
- Don't give your credentials to other services directly. Enter your credentials only through the authorization approval page in Azure DevOps.

# Manage authorizations

When you register an app in Microsoft Entra ID, the app can request tokens from the Microsoft identity platform. An authenticated service can then use a token to access specific protected resources. The lifetime of each token is at most 90 minutes. After a token expires, the service's access to the resources is revoked. For more information, see [Token lifetime](#).

In contrast, apps that are registered with Azure DevOps OAuth can authorize services to access Azure DevOps resources for longer periods of time. For a list of services that are currently authorized to access your account, go to <https://app.vssps.visualstudio.com/Profile/View> and select **Manage authorizations**.

The screenshot shows a modal window titled "AUTHORIZATIONS". It displays a list of OAuth-compliant applications and providers that have been granted permissions to access the user's resources. The first entry is for "Zapier (Zapier)", which has a "Revoke" link next to it. The second entry is for another "Zapier" application, also with a "Revoke" link. A "Close" button is located in the bottom right corner of the modal.

Application	Description	Action
Zapier (Zapier)	Provides this application the ability to act on your behalf with full access (read and write) to work items, source code, builds, and other resources in all Visual Studio Online accounts you can access.	Revoke
Zapier (Zapier)	Provides this application the ability to access your MSDN subscription information including your level and expiration date and interact with Microsoft Developer Services on your behalf.	Revoke

You can use this page to revoke authorizations so that services can't access your account on your behalf.

# Service hook consumers

06/30/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

You can use a service hook to perform an action on a consumer service when an event occurs in an Azure DevOps project. For example, a service hook can notify a consumer when a build fails.

To configure a service hook, you create a subscription that specifies the event, the consumer, and the action. With some consumers, you create a subscription in the consumer service, not in Azure DevOps. This article provides information about the consumer settings you use when you [programmatically create a subscription](#) in Azure DevOps.

You can select from the following consumers when you create a subscription in Azure DevOps:

- [Azure Service Bus](#)
- [Azure Storage](#)
- [Bamboo](#)
- [Datadog](#)
- [Grafana](#)
- [Jenkins](#)
- [Slack](#)
- [Trello](#)
- [Webhooks](#)
- [Zendesk](#)

For information about these consumers and others you can integrate with service hooks, see [Available services](#).

## Azure Service Bus

Service Bus is a messaging service that facilitates asynchronous communication between applications and services. As a service hook consumer, it provides integration with Service Bus queues and topics and also with Azure Notification Hubs.

## Send a message to a notification hub

This action sends a generic, template notification to a specified instance of Notification Hubs.

- Consumer ID: `azureServiceBus`
- Action ID: `serviceBusNotificationHubSend`
- Supported events: All events

- Settings:

[+] Expand table

Input ID	Name	Description	Data type	Required
<code>connectionString</code>	SAS connection string	The shared access signature (SAS) connection string to use to connect with Service Bus. This connection string is available in the Azure portal.	<code>string</code>	Yes
<code>notificationHubName</code>	Notification hub name	The name of the notification hub to send the notification to. The name can contain only letters, numbers, periods, hyphens, forward slashes, and underscores. The name must start and end with a letter or number. The hub should already exist.	<code>string</code>	Yes
<code>tagsExpression</code>	Tags	A tag expression that targets a specific set of devices. For more information, see <a href="#">Routing and tag expressions</a> .	<code>string</code>	No

## Send a message to a Service Bus queue

This action sends a JSON string representation of an event to a specified Service Bus queue. For more information, see [Service Bus queues, topics, and subscriptions](#).

- Consumer ID: `azureServiceBus`
- Action ID: `serviceBusQueueSend`
- Supported events: All events
- Settings:

[+] Expand table

Input ID	Name	Description	Data type	Required
<code>AuthenticationMechanismInputId</code>	Authentication mode	The authentication mode to use: either a connection string or a service connection.	<code>string</code>	No
<code>ServiceConnectionInputId</code>	Azure Resource Manager connections	The ID of a service connection.	<code>string</code>	No

Input ID	Name	Description	Data type	Required
ServiceBusHostNameInputId	Service Bus host name	The host name in the Azure portal, in the format sb://<Service-Bus-name>.servicebus.windows.net.	string	No
connectionString	SAS connection string	The SAS connection string to use to connect with Service Bus. This connection string is available in the Azure portal.	string	No
queueName	Queue name	The name of the queue to send the message to. The name can contain only letters, numbers, periods, hyphens, forward slashes, and underscores. The name must start and end with a letter or number. If the queue doesn't exist, it's created if the specified connection string has the necessary permissions.	string	Yes
bypassSerializer	Send as nonserialized string	An option for sending messages to Service Bus as nonserialized strings instead of as .NET serialized strings. Select this setting when the receiver isn't a .NET client, for instance, when the client uses Azure Client Library for Node.	boolean	No
resourceDetailsToSend	Resource details to send	The number of resource fields to send to the queue. Possibilities are all fields, a minimum number, and none.	string	No
messagesToSend	Messages to send	The types of messages to send to the queue.	string	No
detailedMessagesToSend	Detailed messages to send	The types of detailed messages to send to the queue.	string	No

## Send a message to a Service Bus topic

This action sends a JSON string representation of an event to a specified Service Bus topic. For more information, see [Use the Azure portal to create a Service Bus topic and subscriptions to the topic](#).

- Consumer ID: `azureServiceBus`
- Action ID: `serviceBusTopicSend`
- Supported events: All events
- Settings:

[\[+\] Expand table](#)

Input ID	Name	Description	Data type	Required
<code>AuthenticationMechanismInputId</code>	Authentication mode	The authentication mode to use.	<code>string</code>	No
<code>ServiceConnectionInputId</code>	Azure Resource Manager connections	The ID of a service connection.	<code>string</code>	No
<code>ServiceBusHostNameInputId</code>	Service Bus host name	The host name in the Azure portal, in the format <code>sb://{Service-Bus-name}.servicebus.windows.net</code> .	<code>string</code>	No
<code>connectionString</code>	SAS connection string	The SAS connection string to use to connect with Service Bus. This connection string is available in the Azure portal.	<code>string</code>	No
<code>topicName</code>	Topic name	The name of the topic to send the message to. The name can contain only letters, numbers, periods, hyphens, forward slashes, and underscores. The name must start and end with a letter or number. If the topic doesn't exist, it's created if the specified connection string has the necessary permissions.	<code>string</code>	Yes
<code>bypassSerializer</code>	Send as nonserialized string	An option for sending messages to Service Bus as nonserialized strings instead of as .NET serialized strings. Select this setting when the receiver isn't a .NET client, for instance, when the client uses Azure Client Library for Node.	<code>boolean</code>	No

<b>Input ID</b>	<b>Name</b>	<b>Description</b>	<b>Data type</b>	<b>Required</b>
<code>resourceDetailsToSend</code>	Resource details to send	The number of resource fields to send to the topic. Possibilities are all fields, a minimum number, and none.	<code>string</code>	No
<code>messagesToSend</code>	Messages to send	The types of messages to send to the topic.	<code>string</code>	No
<code>detailedMessagesToSend</code>	Detailed messages to send	The types of detailed messages to send to the topic.	<code>string</code>	No

## Azure Storage

Storage is a cloud storage solution for various types of data. Azure Queue Storage is a part of Storage that provides messaging queues that can act as service hooks consumers.

### Insert a message into a Storage queue

This action inserts a JSON string representation of an event into a specified Storage queue. For more information, see [What is Azure Queue Storage?](#).

- Consumer ID: `azureStorageQueue`
- Action ID: `enqueue`
- Supported events: All events
- Settings:

[ ] [Expand table](#)

<b>Input ID</b>	<b>Name</b>	<b>Description</b>	<b>Data type</b>	<b>Required</b>
<code>AuthenticationMechanismInputId</code>	Authentication mode	The authentication mode to use.	<code>string</code>	No
<code>ServiceConnectionInputId</code>	Azure Resource Manager connections	The ID of a service connection.	<code>string</code>	No
<code>accountName</code>	Storage account name	The name associated with your Storage account. This	<code>string</code>	Yes

<b>Input ID</b>	<b>Name</b>	<b>Description</b>	<b>Data type</b>	<b>Required</b>
		name is available in the Azure portal.		
accountKey	Storage account key	The key associated with your Storage account.	string	No
queueName	Queue name	The lowercase-only name of the queue to use within Storage. A queue with this name gets created if it doesn't already exist.	string	Yes
visiTimeout	Message visibility timeout	The visibility timeout value, in seconds, for the enqueued message, relative to the server time. The value must be greater than or equal to 0 and can't be greater than seven days, or 604,800 seconds. The visibility timeout must be set to a value that's less than the message's time-to-live value.	number	Yes
ttl	Message time-to-live	The time-to-live interval for the queue message, in seconds. The maximum value you can use is seven days, or 604,800 seconds.	number	Yes
resourceDetailsToSend	Resource details to send	The number of resource fields to send to the queue. Possibilities are all fields, a minimum number, and none.	string	No
messagesToSend	Messages to send	The types of messages to send to the queue.	string	No
detailedMessagesToSend	Detailed messages to send	The types of detailed messages to send to the queue.	string	No

## Bamboo

Bamboo is a continuous integration server from Atlassian.

## Queue a build

This action queues a Bamboo build.

- Consumer ID: `bamboo`
- Action ID: `queueBuild`
- Supported events: `git.push`, `build.complete`, `tfvc.checkin`
- Settings:

[+] Expand table

Input ID	Name	Description	Data type	Required type
<code>serverBaseUrl</code>	Bamboo base URL	The URI that contains the hostname of the Bamboo server.	<code>uri</code>	Yes
<code>basicAuthCredentials</code>	Basic authentication credentials	Standard credentials to use to authenticate to the Bamboo server. To avoid sending credentials in plain text, use the HTTPS protocol to encrypt the credentials via Transport Layer Security (TLS). We recommend using <a href="#">service principals and managed identities in Azure DevOps</a> .	<code>string</code>	Yes
<code>planName</code>	Plan	The name of the plan to queue.	<code>string</code>	Yes

## Datadog

Datadog is a monitoring and analytics platform for cloud environments.

### Post an event in Datadog

This action creates an event and corresponding metrics in Datadog.

- Consumer ID: `datadog`
- Action ID: `postEventInDatadog`
- Supported events: All events
- Settings:

[+] Expand table

Input ID	Name	Description	Data type	Required
apiKey	Datadog API Key	The access key for your Datadog account. You can find your API key in the Datadog portal. Go to your profile, and then select <b>Organization Settings &gt; API Keys</b> .	string	Yes
accountType	Datadog Account Type	The type of your Datadog account. You can determine your account type from the hostname of the URL that your Datadog account uses: - app.datadoghq.com: US - app.datadoghq.eu: EU - us3.datadoghq.com: US3 - us5.datadoghq.com: US5 - ap1.datadoghq.com: AP1 - app.dog-gov.com: GOV	string	Yes

## Grafana

Grafana is an open-source dashboard and graph editor.

### Add an annotation to a Grafana database

This action creates an annotation in Grafana.

- Consumer ID: grafana
- Action ID: addAnnotation
- Supported events: ms.vss-release.deployment-completed-event
- Settings:

[ ] Expand table

Input ID	Name	Description	Data type	Required
url	Grafana URL	The URL to use to post an annotation in Grafana.	uri	Yes
apiToken	API token	The access token for posting annotations to a Grafana database. For information about creating a token, see <a href="#">Create Service Account tokens</a>	string	Yes

<b>Input ID</b>	<b>Name</b>	<b>Description</b>	<b>Data type</b>	<b>Required</b>
		and dashboards for an organization <a href="#">↗</a> .		
<code>tags</code>	Tags	The comma-separated list of tags to use for adding annotations.	<code>string</code>	Yes
<code>annotationDeploymentDurationWindow</code>	Annotate deployment duration window	An option for configuring the duration window of an annotation. - When selected, the annotation applies to the time between the start and the completion of deployment. - When not selected, the annotation applies to the completion of the deployment.	<code>boolean</code>	No
<code>text</code>	Text	A custom description for an annotation. When not specified, this setting describes the release and status. This setting can include links, such as <code>&lt;a href="https://www.contoso.com" target="_blank"&gt;Contoso&lt;/a&gt;</code> .	<code>string</code>	No
<code>dashboardId</code>	Dashboard	The ID of the dashboard to add the annotation to. When an ID isn't specified, the annotation is added at the global level.	<code>string</code>	No

## Jenkins

Jenkins is a continuous integration server that you can use to build and test software projects continuously.

## Trigger a Git build

This action uses the [Jenkins Git plug-in ↗](#) to trigger a build in a Git repo.

- Consumer ID: `jenkins`
- Action ID: `triggerGitBuild`
- Supported events: `git.push`, `git.pullrequest.merged`

- Settings:

[+] [Expand table](#)

Input ID	Name	Description	Data type	Required
<code>serverBaseUrl1</code>	Jenkins base URL	The URI that contains the hostname of the Jenkins server.	<code>uri</code>	Yes
<code>basicAuthCredentials</code>	Basic authentication credentials	Standard HTTP authentication credentials. To avoid sending credentials in plain text, use the HTTPS protocol to encrypt the credentials via TLS. We recommend using <a href="#">service principals and managed identities in Azure DevOps</a> .	<code>string</code>	Yes
<code>useTfsPlugin</code>	Integration level	The selected integration level, which is one of two values: - The built-in Jenkins API - The extended integration provided by the Azure DevOps Server plug-in if that plug-in is installed on the Jenkins server	<code>string</code>	No

## Trigger a generic build

This action triggers a generic Jenkins build that invokes the Jenkins build URL.

- Consumer ID: `jenkins`
- Action ID: `triggerGenericBuild`
- Supported events: `git.push`, `git.pullrequest.merged`, `build.complete`, `tfvc.checkin`, `ms.vss-release.deployment-completed-event`
- Settings:

[+] [Expand table](#)

Input ID	Name	Description	Data type	Required
<code>serverBaseUrl1</code>	Jenkins base URL	The URI that contains the hostname of the Jenkins server.	<code>uri</code>	Yes

Input ID	Name	Description	Data type	Required
<code>basicAuthCredentials</code>	Basic authentication credentials	Standard HTTP authentication credentials. To avoid sending credentials in plain text, use the HTTPS protocol to encrypt the credentials via TLS. We recommend using <a href="#">service principals and managed identities in Azure DevOps</a> .	<code>string</code>	Yes
<code>buildName</code>	Build	The name of the build to trigger.	<code>string</code>	Yes
<code>useTfsPlugin</code>	Integration level	The selected integration level, which is one of two values: - The built-in Jenkins API - The extended integration provided by the Azure DevOps Server plug-in if that plug-in is installed on the Jenkins server	<code>string</code>	No
<code>buildAuthToken</code>	Build token	An authorization token for a build. Only users who know the token can remotely trigger builds.	<code>string</code>	No
<code>buildParameterized</code>	Accepts parameters	An option that specifies whether the build accepts parameters.	<code>boolean</code>	No
<code>buildParams</code>	Build parameters	Build parameters in the form of name-value pairs. In each pair, the name and value are separated by a colon, such as <code>&lt;parameter&gt;:&lt;value&gt;</code> . Each name-value pair appears on its own line.	<code>string</code>	No

## Slack

Slack is a searchable platform for team communication.

### Post a message to a channel

This action posts a message about an event to a Slack channel. For more information, see [Create a service hook for Azure DevOps with Slack](#).

- Consumer ID: `slack`
- Action ID: `postMessageToChannel`
- Supported events: All events

- Settings:

[+] Expand table

Input	Name	Description	Data type	Required
ID			type	
url	Slack webhook URL	The webhook URL provided by Slack to send HTTP POST requests to.	uri	Yes

## Trello

Trello is a project management tool that uses boards, lists, and cards to help teams track workflows.

### Create a card

This action creates a card on an existing list in Trello. A card can represent a task, an issue, an event, or other project-related items. For more information, see [Create a service hook for Azure DevOps Services and TFS with Trello](#).

- Consumer ID: `trello`
- Action ID: `createCard`
- Supported events: All events
- Settings:

[+] Expand table

Input ID	Name	Description	Data type	Required
			type	
userToken	User token	A user token that provides access to Trello resources. To get a token, go to the <a href="#">Trello authorization page</a> .	string	Yes
boardId	Board	The name of the board on which the Trello card gets created.	string	Yes
listId	List	The name of the list on which the Trello card gets created.	string	Yes
labels	Labels	A comma-separated list of label colors to apply to the created card. Valid label color names are <code>red</code> , <code>orange</code> , <code>yellow</code> , <code>green</code> , <code>blue</code> , and <code>purple</code> .	string	No
addToTop	Create at beginning of	An option that indicates whether to create the card at the beginning or end of the Trello list.	boolean	No

<b>Input ID</b>	<b>Name</b>	<b>Description</b>	<b>Data type</b>	<b>Required</b>
	list	When this field is <code>true</code> , the card is created at the beginning.		
<code>cardName</code>	Card name	The name for the new card. By default, the text description of the event is used as the name. You can use placeholders to insert content from the event into the name. For more information, see <a href="#">Create a service hook for Azure DevOps Services and TFS with Trello</a> .	string	No
<code>cardDescription</code>	Card description	The description for the new card. By default, the detailed Markdown description of the event is used as the description. You can use placeholders to insert content from the event into the description. For more information, see <a href="#">Create a service hook for Azure DevOps Services and TFS with Trello</a> .	string	No

## Create a list

This action creates a list on an existing board in Trello. A list is used to organize cards on a board and typically represents a state. For more information, see [Create a service hook for Azure DevOps Services and TFS with Trello](#).

- Consumer ID: `trell0`
- Action ID: `createList`
- Supported events: All events
- Settings:

[Expand table](#)

<b>Input ID</b>	<b>Name</b>	<b>Description</b>	<b>Data type</b>	<b>Required</b>
<code>userToken</code>	User token	A user token that provides access to Trello resources. To get a token, go to the <a href="#">Trello authorization page</a> .	string	Yes
<code>boardId</code>	Board	The name of the board on which the Trello list gets created.	string	Yes
<code>addToBottom</code>	Create at bottom of board	An option that indicates whether to create the card at the beginning or end of the board. When this field is <code>true</code> , the card is created at the end.	boolean	No

Input ID	Name	Description	Data type	Required
<code>listName</code>	List name	The name for the new list. By default, the text description of the event is used as the name. You can use placeholders to insert content from the event into the name. For more information, see <a href="#">Create a service hook for Azure DevOps Services and TFS with Trello</a> .	string	No

## Webhooks

Webhooks provide a way to send a JSON representation of an Azure DevOps event to any service that has a public endpoint.

### Post via HTTP

This action posts a JSON object representation of an event to a specified URL. HTTPS endpoints are recommended due to the potential for private data in the event payload. For more information, see [Webhooks](#).

- Consumer ID: `webHooks`
- Action ID: `httpRequest`
- Supported events: All events
- Settings:

[Expand table](#)

Input ID	Name	Description	Data type	Required
<code>url</code>	URL	The URL to send an HTTP POST to.	uri	Yes
<code>acceptUntrustedCerts</code>	Accept untrusted SSL certificates	An option for not requiring a trusted Secure Sockets Layer (SSL) certificate for an endpoint. Use this option only during development and testing.	boolean	No
<code>basicAuthCredentials</code>	Basic authentication credentials	Standard HTTP authentication credentials. To avoid sending credentials in plain text, use the HTTPS protocol to encrypt the credentials via TLS. We recommend using <a href="#">service principals and</a>	string	Yes

Input ID	Name	Description	Data type	Required
		managed identities in Azure DevOps.		
httpHeaders	HTTP headers	HTTP header keys and values in the form of key-value pairs. In each pair, the key and value are separated by a colon, such as <key>:<value>. Each key-value pair appears on its own line. These values are viewable by anyone who has access to the service hook subscription.	string	No
resourceDetailsToSend	Resource details to send	The number of resource fields to send to the queue. Possibilities are all fields, a minimum number, and none.	string	No
messagesToSend	Messages to send	The types of messages to send to the queue.	string	No
detailedMessagesToSend	Detailed messages to send	The types of detailed messages to send to the queue.	string	No

## Zendesk

Zendesk is a software as a service (SaaS) suite that offers help-desk ticketing, issue tracking, and customer-service support.

### Create a private comment in a ticket

This action creates a private comment in a Zendesk ticket.

- Consumer ID: `zendesk`
- Action ID: `createPrivateComment`
- Supported events: `workitemcommented`
- Settings:

[ ] Expand table

Input ID	Name	Description	Data type	Required
accountName	Account name	The Zendesk account name. You can find the account name in the URL of your Zendesk account, which has the format <code>https://&lt;account-name&gt;.zendesk.com</code> .	string	Yes
username	User name	The user name of the Zendesk user who updates tickets.	string	Yes
apiToken	API token	The Zendesk API token. To find the token, go to the Zendesk app, and then select <b>Admin &gt; Channels &gt; API</b> .	string	Yes

## Related content

- [Manage authorization of services to access Azure DevOps](#)
- [Integrate with service hooks](#)

# Service hook events

07/10/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

You can use service hooks to run tasks on other services when events happen in your Azure DevOps project. This article provides information about the Azure DevOps events that a service hook can trigger on.

For each event, the article lists the ID values and settings that you use when you create a subscription for the event programmatically. Each event section also provides an example of a payload that's sent when the service hook for the event is triggered.

## Available event types

The following types of events are available for use in service hooks. For a list of the events that each target service supports, see [Available services](#).

- **Build and release**
  - [Build completed](#)
  - [Release abandoned](#)
  - [Release created](#)
  - [Release deployment approval completed](#)
  - [Release deployment approval pending](#)
  - [Release deployment completed](#)
  - [Release deployment started](#)
- **Pipeline**
  - [Check updated](#)
  - [Elastic agent pool resized](#)
  - [Manual intervention pending](#)
  - [Project-level agent pool created](#)
  - [Project-level agent pool updated](#)
  - [Run state changed](#)
  - [Run stage state changed](#)
  - [Run stage waiting for approval](#)
  - [Run stage approval completed](#)
  - [Run job state changed](#)
- **Code**
  - [Code checked in](#)

- Code pushed
- Pull request created
- Pull request merge attempted
- Pull request updated
- Pull request commented on
- Repository created
- Repository deleted
- Repository forked
- Repository renamed
- Repository status changed

- Service connection
  - Service connection created
  - Service connection updated
- Work item
  - Work item created
  - Work item deleted
  - Work item restored
  - Work item updated
  - Work item commented on
- Advanced security
  - Advanced security alert created
  - Advanced security alert state changed
  - Advanced security alert updated

① Note

The [NuGet WebHooks Receivers package](#) provides support for receiving webhook notifications from Azure DevOps.

## Build and release

The following build and release events are available for use in service hooks.

### Build completed

Event: A build finishes.

- Publisher ID: tfs

- Event ID: `build.complete`
- Resource name: `build`

## Settings

- `definitionName`: Include only events for completed builds for a specific pipeline.
- `buildStatus`: Include only events for completed builds that have a specific completion status.
  - Valid values:
    - `Succeeded`
    - `PartiallySucceeded`
    - `Failed`
    - `Stopped`

## Sample payload

JSON

```
{
  "subscriptionId": "aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e",
  "notificationId": 1,
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "build.complete",
  "publisherId": "tfs",
  "message": {
    "text": "Build 20241202.1 succeeded",
    "html": "Build <a href=\"https://dev.azure.com/FabrikamFiber/web/build.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-devops%3a%2f%2fBuild%2fBuild%2f2727068\">20241202.1</a> succeeded",
    "markdown": "Build [20241202.1](https://dev.azure.com/FabrikamFiber/web/build.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-devops%3a%2f%2fBuild%2fBuild%2f2727068) succeeded"
  },
  "detailedMessage": {
    "text": "Build 20241202.1 succeeded",
    "html": "Build <a href=\"https://dev.azure.com/FabrikamFiber/web/build.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-devops%3a%2f%2fBuild%2fBuild%2f2727068\">20241202.1</a> succeeded",
    "markdown": "Build [20241202.1](https://dev.azure.com/FabrikamFiber/web/build.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-devops%3a%2f%2fBuild%2fBuild%2f2727068) succeeded"
  },
  "resource": {
    "id": 2727068,
    "buildNumber": "20241202.1",
    "status": "completed",
  }
}
```

```
"result": "succeeded",
"url": "https://dev.azure.com/FabrikamFiber/web/build.aspx?pcguid=e4e4e4e4-
ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-
devops%3a%2f%2f%2fBuild%2fBuild%2f2727068",
"definition": {
  "id": 1,
  "name": "FabrikamFiber CI"
},
"project": {
  "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
  "name": "FabrikamFiber"
}
},
"createdDate": "2024-12-02T12:21:13.8866607Z"
}{

  "subscriptionId": "aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e",
  "notificationId": 1,
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "build.complete",
  "publisherId": "azure-devops",
  "message": {
    "text": "Build 20241202.1 succeeded",
    "html": "Build <a href=\"https://dev.azure.com/FabrikamFiber/web/build.aspx?
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-
devops%3a%2f%2f%2fBuild%2fBuild%2f2727068\">20241202.1</a> succeeded",
    "markdown": "Build [20241202.1]
(https://dev.azure.com/FabrikamFiber/web/build.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-devops%3a%2f%2f%2fBuild%2fBuild%2f2727068)
succeeded"
  },
  "detailedMessage": {
    "text": "Build 20241202.1 succeeded",
    "html": "Build <a href=\"https://dev.azure.com/FabrikamFiber/web/build.aspx?
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-
devops%3a%2f%2f%2fBuild%2fBuild%2f2727068\">20241202.1</a> succeeded",
    "markdown": "Build [20241202.1]
(https://dev.azure.com/FabrikamFiber/web/build.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&builduri=azure-devops%3a%2f%2f%2fBuild%2fBuild%2f2727068)
succeeded"
  },
  "resource": {
    "_links": {
      "self": {
        "href": "https://dev.azure.com/FabrikamFiber/e4e4e4e4-ffff-aaaa-bbbb-
c5c5c5c5c5c5/_apis/build/Builds/2727068"
      },
      "web": {
        "href": "https://dev.azure.com/FabrikamFiber/e4e4e4e4-ffff-aaaa-bbbb-
c5c5c5c5c5c5/_build/results?buildId=2727068"
      },
      "sourceVersionDisplayUri": {
        "href": "https://dev.azure.com/FabrikamFiber/e4e4e4e4-ffff-aaaa-bbbb-
c5c5c5c5c5c5/_apis/build/builds/2727068/sources"
      },
      "timeline": {
        "href": "https://dev.azure.com/FabrikamFiber/e4e4e4e4-ffff-aaaa-bbbb-
c5c5c5c5c5c5/_apis/build/builds/2727068/timeline"
      }
    }
  }
}
```

```
        "href": "https://dev.azure.com/FabrikamFiber/e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5/_apis/build/builds/2727068/Timeline"
    },
    "badge": {
        "href": "https://dev.azure.com/FabrikamFiber/e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5/_apis/build/status/4658"
    }
},
"properties": {},
"tags": [],
"validationResults": [],
"plans": [
    {
        "planId": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa"
    }
],
"triggerInfo": {},
"id": 2727068,
"buildNumber": "20241202.1",
"status": "completed",
"result": "succeeded",
"queueTime": "2024-12-02T12:18:45.7367977Z",
"startTime": "2024-12-02T12:18:56.6205723Z",
"finishTime": "2024-12-02T12:21:08.520904Z",
"url": "https://dev.azure.com/FabrikamFiber/33dd33dd-ee44-ff55-aa66-77bb77bb77bb/_apis/build/Builds/2727068",
"definition": {
    "drafts": [],
    "id": 4658,
    "name": "MainRepo (1)",
    "url": "https://dev.azure.com/FabrikamFiber/33dd33dd-ee44-ff55-aa66-77bb77bb77bb/_apis/build/Definitions/4658?revision=1",
    "uri": "azure-devops:///Build/Definition/4658",
    "path": "\\",
    "type": "build",
    "queueStatus": "enabled",
    "revision": 1,
    "project": {
        "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
        "name": "FabrikamFiberChat",
        "url": "https://dev.azure.com/FabrikamFiber/_apis/projects/22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
        "state": "wellFormed",
        "revision": 876,
        "visibility": "organization",
        "lastUpdateTime": "2024-04-06T14:51:16.337Z"
    }
},
"buildNumberRevision": 1,
"project": {
    "id": "33dd33dd-ee44-ff55-aa66-77bb77bb77bb",
    "name": "FabrikamFiberChat",
    "url": "https://dev.azure.com/FabrikamFiber/_apis/projects/33dd33dd-ee44-ff55-aa66-77bb77bb77bb",
    "state": "wellFormed",

```

```
        "revision": 876,
        "visibility": "organization",
        "lastUpdateTime": "2024-04-06T14:51:16.337Z"
    },
    "uri": "azure-devops:///Build/Build/2727068",
    "sourceBranch": "refs/heads/main",
    "sourceVersion": "a6a6a6a6-bbbb-cccc-dddd-e7e7e7e7e7e7",
    "queue": {
        "id": 4799,
        "name": "Azure Pipelines",
        "pool": {
            "id": 112,
            "name": "Azure Pipelines",
            "isHosted": true
        }
    },
    "priority": "normal",
    "reason": "manual",
    "requestedFor": {
        "displayName": "Fabrikam Fiber",
        "url": "https://spsprodwus22.vssps.visualstudio.com/fffff5f5f-aa6a-bb7b-cc8c-ddddd9d9d9d/_apis/Identities/33dd33dd-ee44-ff55-aa66-77bb77bb77bb",
        "_links": {
            "avatar": {
                "href": "https://dev.azure.com/FabrikamFiber/_apis/GraphProfile/MemberAvatars/aad.NTdhNWQ3OTQtOTc3My03YzMyLQJiYjYtNTUwNTg1Njk1MTE5"
            }
        },
        "id": "66aa66aa-bb77-cc88-dd99-00ee00ee00ee",
        "uniqueName": "chuck@FabrikamFiber.com",
        "imageUrl": "https://dev.azure.com/FabrikamFiber/_apis/GraphProfile/MemberAvatars/aad.NTdhNWQ3OTQtOTc1My03YzMyLWJiYjYtNTUwNTg1Njk1MTE5",
        "descriptor": "aad.NTdhNWQ3OTQtOTc6My03YzMyLWJiYjYtNTUwNTg1Njk1MTE5"
    },
    "requestedBy": {
        "displayName": "Chuck Reinhart",
        "url": "https://spsprodwus22.vssps.visualstudio.com/fffff5f5f-aa6a-bb7b-cc8c-ddddd9d9d9d/_apis/Identities/33dd33dd-ee44-ff55-aa66-77bb77bb77bb",
        "_links": {
            "avatar": {
                "href": "https://dev.azure.com/FabrikamFiber/_apis/GraphProfile/MemberAvatars/aad.NTdhNWQ3OTQtOTc3My03YzMyLQJiYjYtNTUwNTg1Njk1MTE5"
            }
        },
        "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
        "uniqueName": "chuck@FabrikamFiber.com",
        "imageUrl": "https://dev.azure.com/FabrikamFiber/_apis/GraphProfile/MemberAvatars/aad.NTdhNWQ3OTQtOTc1My03YzMyLWJiYjYtNTUwNTg1Njk1MTE5",
        "descriptor": "aad.NTdhNWQ3OTQtOTc6My03YzMyLWJiYjYtNTUwNTg1Njk1MTE5"
    },
    "lastChangedDate": "2024-12-02T12:21:08.96Z",
```

```
"lastChangedBy": {
    "displayName": "Microsoft.VisualStudio.Services.TFS",
    "url": "https://spspodwus22.vssps.visualstudio.com/fffff5f5f-aa6a-bb7b-cc8c-ddddd9d9d9d/_apis/Identities/11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "_links": {
        "avatar": {
            "href": "https://dev.azure.com/FabrikamFiber/_apis/GraphProfile/MemberAvatars/s2s.MDAwMDAwMDItMDAwMC020Dg4LTgwMDAtMDAwMDAwMDAwMDAwQDJj0Dk1OTA4LTA0ZTAtNDk1Mi040WZkLTU0YjAwNDZkNjI40A"
        }
    },
    "id": "44ee44ee-ff55-aa66-bb77-88cc88cc88cc",
    "uniqueName": "fabrikamfiber16@hotmail.com",
    "imageUrl": "https://dev.azure.com/FabrikamFiber/_apis/GraphProfile/MemberAvatars/s2s.MDAwMDAwMDItMDAwMC040Dg4LTgwMDAtMDAwMDAwMDAwMDAwMDAwQDJj0Dk1OTA4LTA2ZTAtNDk1Mi040WZkLTU0YjAwNDZkNjI40A",
    "descriptor": "s2s.MDAwMDAwMDItMDAwMC040Dg4LTgwMDAtMDAwMDAwMDAwMDAwQDJj0Dk1OTA4LTA2ZTAtNDk1Mi040WZkLTU0YjAwNDZkNjI40A"
},
"orchestrationPlan": {
    "planId": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4b4"
},
"logs": {
    "id": 0,
    "type": "Container",
    "url": "https://dev.azure.com/FabrikamFiber/_apis/build/builds/f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6/logs"
},
"repository": {
    "id": "a6a6a6a6-bbbb-cccc-dddd-e7e7e7e7e7e7",
    "type": "TfsGit",
    "name": "MainRepo",
    "url": "https://dev.azure.com/FabrikamFiber/_git/FabrikamFiberChat",
    "clean": null,
    "checkoutSubmodules": false
},
"retainedByRelease": false,
"triggeredByBuild": null,
"appendCommitMessageToRunName": true
},
"resourceVersion": "2.0",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2",
        "baseUrl": "https://dev.azure.com/FabrikamFiber/"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f",
        "baseUrl": "https://dev.azure.com/FabrikamFiber/"
    },
    "project": {
```

```
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee",
        "baseUrl": "https://dev.azure.com/FabrikamFiber/"
    },
},
"createdDate": "2024-12-02T12:21:13.8866607Z"
}
```

## Release abandoned

Event: A release is abandoned.

- Publisher ID: `rm`
- Event ID: `ms.azure-devops-release.release-abandoned-event`
- Resource name: `resource`

## Settings

- `releaseDefinitionId`: Include only events for completed deployments for a specific pipeline.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.azure-devops-release.release-abandoned-event",
  "publisherId": "rm",
  "scope": "all",
  "resource": {
    "release": {
      "id": "release-id",
      "name": "release-name",
      "status": "abandoned",
      "releaseDefinition": {
        "id": "release-definition-id",
        "name": "release-definition-name"
      }
    },
    "project": {
      "id": "project-id",
      "name": "project-name"
    }
  },
  "createdDate": "2024-12-02T12:21:13.8866607Z"
}
```

## Release created

Event: A release is created.

- Publisher ID: `rm`
- Event ID: `ms.azure-devops-release.release-created-event`
- Resource name: `resource`

## Settings

- `releaseDefinitionId`: Include only events for completed deployments for a specific pipeline.

## Sample payload

JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "ms.azure-devops-release.release-created-event",  
  "publisherId": "rm",  
  "scope": "all",  
  "message": {  
    "text": "Release Release-1 created.",  
    "html": "<a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/5'>Release-1</a> created.",  
    "markdown": "Release [Release-1]  
(http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/5) created."  
  },  
  "detailedMessage": {  
    "text": "Release Release-1 created from release pipeline  
Fabrikam.CD.\r\nRelease description: QFE release for fixing  
title\r\nContinuousIntegration Requested for Chuck Reinhart\r\nBuild:  
fabrikam.Bd.2016.04.10 & 2 more<\li>",  
    "html": "Release <a  
href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/5'>Release-1</a> created from release pipeline <a  
href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releasedefinitions/1'>Fabrikam.CD</a>.\r\n- Release  
description: QFE release for fixing title<br>\r\n- ContinuousIntegration  
Requested for Chuck Reinhart<br>\r\n- Build: fabrikam.Bd.2016.04.10 & 2  
more<\li>",  
    "markdown": "Release [Release-1]  
(http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/5) created from release pipeline [Fabrikam.CD]  
(http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releasedefinitions/1).\\r\\n- Release description: QFE release  
for fixing title<br>\\r\\n- ContinuousIntegrationRequested for Chuck
```

Reinhart</br>\r\n- Build: fabrikam.Bd.2016.04.10 & 2 more<\li>"

```
},
"resource": {
  "release": {
    "id": 4,
    "name": "Release-1",
    "status": "active",
    "createdOn": "2016-01-21T08:19:17.26Z",
    "modifiedOn": "2016-01-21T08:19:17.26Z",
    "modifiedBy": {
      "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
      "displayName": "Chuck Reinhart"
    },
    "createdBy": {
      "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
      "displayName": "Chuck Reinhart"
    },
    "environments": [
      {
        "id": 5,
        "releaseId": 0,
        "name": "Dev",
        "status": "succeeded",
        "variables": {},
        "preDeployApprovals": [],
        "postDeployApprovals": [],
        "preApprovalsSnapshot": {
          "approvals": [],
          "approvalOptions": {
            "requiredApproverCount": 0,
            "releaseCreatorCanBeApprover": true
          }
        },
        "postApprovalsSnapshot": {
          "approvals": []
        },
        "deploySteps": [],
        "rank": 1,
        "definitionEnvironmentId": 1,
        "queueId": 1,
        "environmentOptions": {
          "emailNotificationType": "OnlyOnFailure",
          "emailRecipients": "release.environment.owner;release.creator",
          "skipArtifactsDownload": false,
          "timeoutInMinutes": 0,
          "enableAccessToken": false
        },
        "demands": [],
        "conditions": [],
        "modifiedOn": "2016-01-21T08:19:17.26Z",
        "workflowTasks": [
          {
            "taskId": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2",
            "version": "*",
            "name": "Deploy Website to Azure",
            "id": 1
          }
        ]
      }
    ]
  }
}
```

```
        "enabled": true,
        "alwaysRun": false,
        "continueOnError": false,
        "timeoutInMinutes": 0,
        "definitionType": null,
        "inputs": {
            "ConnectedServiceName": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
            "WebSiteName": "$(webAppName)",
            "WebSiteLocation": "Southeast Asia",
            "Slot": "",
            "Package": "$(System.DefaultWorkingDirectory)\\**\\*.zip"
        }
    }
],
"deployPhasesSnapshot": [],
"owner": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "Chuck Reinhart"
},
"scheduledDeploymentTime": "2016-01-21T08:19:17.26Z",
"schedules": [],
"release": {
    "id": 5,
    "name": "Release-1",
    "url": "http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/5"
}
},
],
"variables": {},
"artifacts": [
{
    "sourceId": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4:1",
    "type": "Build",
    "alias": "Fabrikam.CI",
    "definitionReference": {
        "Definition": {
            "id": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5",
            "name": "Fabrikam.CI"
        },
        "Project": {
            "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b46",
            "name": "Fabrikam"
        }
    },
    "isPrimary": true
}
],
"releaseDefinition": {
    "id": 1,
    "name": "Fabrikam.CD",
    "url": "http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/definitions/1"
},
"description": "QFE release for fixing title",
```

```

    "reason": "continuousIntegration",
    "releaseNameFormat": "Release-$(rev:r)",
    "keepForever": false,
    "definitionSnapshotRevision": 0,
    "comment": "",
    "logsContainerUrl": null,
    "_links": {}
},
"project": {
    "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4b46",
    "name": "Fabrikam"
}
},
"resourceVersion": "3.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-ffffff5f5f5f"
    }
},
"createdDate": "2024-09-19T13:03:27.6570261Z"
}

```

## Release deployment approval completed

Event: A deployment approval is completed.

- Publisher ID: `rm`
- Event ID: `ms.azure-devops-release.deployment-approval-completed-event`
- Resource name: `resource`

## Settings

- `releaseApprovalStatus`: Include only events for deployments with an approval of a specific status.
  - Valid values:
    - `2` - Approved
    - `4` - Rejected
- `releaseApprovalType`: Include only events for deployments for which an approval of a specific type is requested.
  - Valid values:
    - `1` - Predeployment
    - `2` - Post-deployment

- `releaseEnvironmentId`: Include only events for completed deployments for a specific environment.
- `releaseDefinitionId`: Include only events for completed deployments for a specific pipeline.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.azure-devops-release.deployment-approval-completed-event",
  "publisherId": "rm",
  "scope": "all",
  "message": {
    "text": "Pre Deployment approval for deployment of release Release-1 on environment Dev Succeeded.",
    "html": "Pre Deployment approval for release <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/1'>Release-1</a> on environment <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/definitions/1'>Dev</a> Succeeded.",
    "markdown": "Pre Deployment approval for deployment of release [Release-1] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apis/Release/releases/1) on environment [Dev] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apis/Release/definitions/1) Succeeded."
  },
  "detailedMessage": {
    "text": "Pre Deployment approval for release Release-1 on environment Dev Succeeded.\r\nApprover: Chuck Reinhart\r\nComment: Approving",
    "html": "Pre Deployment approval for release <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/1'>Release-1</a> on environment <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/definitions/1'>Dev</a> Succeeded.<br>Approver: Chuck Reinhart<br>Comment: Approving",
    "markdown": "Pre Deployment approval for release [Release-1] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apis/Release/releases/1) on environment [Dev] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apis/Release/definitions/1) Succeeded.\r\nApprover: Chuck Reinhart\r\nComment: Approving"
  },
  "resource": {
    "approval": {
      "id": 1,
      "revision": 1,
      "approvalType": "preDeploy",
      "status": "approved",
      "createdOn": "2024-12-02T12:21:13.8866607Z",
      "lastModifiedOn": "2024-12-02T12:21:13.8866607Z"
    }
  }
}
```

```

    "modifiedOn": "2024-12-02T12:21:13.8866607Z",
    "comments": "Approving",
    "isAutomated": false,
    "isNotificationOn": false,
    "trialNumber": 1,
    "attempt": 1,
    "approver": {
        "displayName": "Chuck Reinhart",
        "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff"
    },
},
"environment": {
    "id": 1,
    "name": "Dev"
},
"release": {
    "id": 1,
    "name": "Release-1",
    "releaseDefinition": {
        "id": 1,
        "name": "Release Definition"
    },
    "project": {
        "id": "project-id",
        "name": "project-name"
    }
},
"createdDate": "2024-12-02T12:21:13.8866607Z"
}

```

## Release deployment approval pending

Event: A deployment approval is requested.

- Publisher ID: `rm`
- Event ID: `ms.azure-devops-release.deployment-approval-pending-event`
- Resource name: `resource`

## Settings

- `releaseApprovalType`: Include only events for deployments for which an approval of a specific type is requested.
  - Valid values:
    - `1` - Predeployment
    - `2` - Post-deployment

- `releaseEnvironmentId`: Include only events for completed deployments for a specific environment.
- `releaseDefinitionId`: Include only events for completed deployments for a specific pipeline.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.azure-devops-release.deployment-approval-pending-event",
  "publisherId": "rm",
  "scope": "all",
  "message": {
    "text": "Pre deployment approval pending for release Release-1 on environment Dev.",
    "html": "Pre deployment approval pending for release <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/1'>Release-1</a> on environment <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?_a=environment-summary&definitionEnvironmentId=8&definitionId=1'>Dev</a>.",
    "markdown": "Pre deployment approval pending for release [Release-1] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apis/Release/releases/1) on environment [Dev] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?\_a=environment-summary&definitionEnvironmentId=8&definitionId=1)."
  },
  "detailedMessage": {
    "text": "Pre deployment approval pending for release Release-1 on environment Dev.\r\nPending on: Chuck Reinhart\r\nPending since: 09 May 2016 12:09:29 (UTC)",
    "html": "Pre deployment approval pending for release <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/1'>Release-1</a> on environment <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?_a=environment-summary&definitionEnvironmentId=8&definitionId=1'>Dev</a>.<br>Pending on: Chuck Reinhart<br>Pending since: 09 May 2016 12:09:29 (UTC)",
    "markdown": "Pre deployment approval pending for release [Release-1] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apis/Release/releases/1) on environment [Dev] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?\_a=environment-summary&definitionEnvironmentId=8&definitionId=1).\r\nPending on: Chuck Reinhart\r\nPending since: 09 May 2016 12:09:29 (UTC)"
  },
  "resource": {
    "approval": {
      "id": 1,
      "status": "PENDING"
    }
  }
}
```

```

    "revision": 1,
    "approvalType": "preDeploy",
    "status": "pending",
    "createdOn": "2016-05-09T12:09:29Z",
    "modifiedOn": "2016-05-09T12:09:29Z",
    "isAutomated": false,
    "isNotificationOn": false,
    "trialNumber": 1,
    "attempt": 1,
    "approver": {
        "displayName": "Chuck Reinhart",
        "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff"
    },
},
"environment": {
    "id": 8,
    "name": "Dev"
},
"release": {
    "id": 1,
    "name": "Release-1",
    "releaseDefinition": {
        "id": 1,
        "name": "Release Definition"
    },
    "project": {
        "id": "project-id",
        "name": "project-name"
    }
},
"createdDate": "2016-05-09T12:09:29Z"
}{

    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "ms.azure-devops-release.deployment-approval-pending-event",
    "publisherId": "rm",
    "scope": "all",
    "message": {
        "text": "Pre deployment approval pending for release Release-1 on environment Dev.",
        "html": "Pre deployment approval pending for release <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/1'>Release-1</a> on environment <a href='http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?_a=environment-summary&definitionEnvironmentId=8&definitionId=1'>Dev</a>.",
        "markdown": "Pre deployment approval pending for release [Release-1] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apis/Release/releases/1) on environment [Dev] (http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/\_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?\_a=environment-summary&definitionEnvironmentId=8&definitionId=1)."
    },
    "detailedMessage": {
        "text": "Pre deployment approval pending for release Release-1 on environment"
}

```



```
"modifiedOn": "2016-01-21T08:19:17.26Z",
"modifiedBy": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "Chuck Reinhart"
},
"createdBy": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "Chuck Reinhart"
},
"environments": [
{
    "id": 5,
    "releaseId": 0,
    "name": "Dev",
    "status": "succeeded",
    "variables": {},
    "preDeployApprovals": [],
    "postDeployApprovals": [],
    "preApprovalsSnapshot": {
        "approvals": [],
        "approvalOptions": {
            "requiredApproverCount": 0,
            "releaseCreatorCanBeApprover": true
        }
    },
    "postApprovalsSnapshot": {
        "approvals": []
    },
    "deploySteps": [],
    "rank": 1,
    "definitionEnvironmentId": 1,
    "queueId": 1,
    "environmentOptions": {
        "emailNotificationType": "OnlyOnFailure",
        "emailRecipients": "release.environment.owner;release.creator",
        "skipArtifactsDownload": false,
        "timeoutInMinutes": 0,
        "enableAccessToken": false
    },
    "demands": [],
    "conditions": [],
    "modifiedOn": "2016-01-21T08:19:17.26Z",
    "workflowTasks": [
{
        "taskId": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2",
        "version": "*",
        "name": "Deploy Website to Azure",
        "enabled": true,
        "alwaysRun": false,
        "continueOnError": false,
        "timeoutInMinutes": 0,
        "definitionType": null,
        "inputs": {
            "ConnectedServiceName": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
            "WebSiteName": "$(webAppName)",
            "WebSitePath": "$(webAppPath)"
        }
    }
]
```

```
        "WebSiteLocation": "Southeast Asia",
        "Slot": "",
        "Package": "$(System.DefaultWorkingDirectory)\\**\\*.zip"
    }
}
],
"deployPhasesSnapshot": [],
"owner": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "Chuck Reinhart"
},
"scheduledDeploymentTime": "2016-01-21T08:19:17.26Z",
"schedules": [],
"release": {
    "id": 1,
    "name": "Release-1",
    "url": "http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/releases/1"
},
},
],
"variables": {},
"artifacts": [
{
    "sourceId": "31419848-1780-4137-b7e3-62092e986fd6:1",
    "type": "Build",
    "alias": "Fabrikam.CI",
    "definitionReference": {
        "Definition": {
            "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4b4",
            "name": "Fabrikam.CI"
        },
        "Project": {
            "id": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5",
            "name": "Fabrikam"
        }
    },
    "isPrimary": true
}
],
"releaseDefinition": {
    "id": 1,
    "name": "Fabrikam.CD",
    "url": "http://dev.azure.com/fabfiber/DefaultCollection/Fabrikam-Fiber-Git/_apis/Release/definitions/1"
},
"description": "QFE release for fixing title",
"reason": "continuousIntegration",
"releaseNameFormat": "Release-$(rev:r)",
"keepForever": false,
"definitionSnapshotRevision": 0,
"comment": "",
"logsContainerUrl": null,
"_links": {}
},
```

```
"project": {
    "id": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5",
    "name": "Fabrikam"
},
"resourceVersion": "3.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    }
},
"createdDate": "2024-09-19T13:03:28.0320509Z"
}
```

## Release deployment completed

Event: A deployment finishes.

- Publisher ID: `rm`
- Event ID: `ms.azure-devops-release.deployment-completed-event`
- Resource name: `resource`

## Settings

- `releaseEnvironmentId`: Include only events for completed deployments for a specific environment.
- `releaseDefinitionId`: Include only events for completed deployments for a specific pipeline.
- `releaseEnvironmentStatus`: Include only events for completed deployments with a specific status.
  - Valid values:
    - `4` - Succeeded
    - `8` - Canceled
    - `16` - Rejected
    - `128` - Partially succeeded

## Sample payload

JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "ms.azure-devops-release.deployment-completed-event",  
  "publisherId": "rm",  
  "scope": "all",  
  "message": {  
    "text": "Deployment of release Release-1 on environment Dev Succeeded.",  
    "html": "Deployment on environment <a  
href='http://fabfiber.visualstudio.com/DefaultCollection/Fabrikam-Fiber-  
Git/_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=8&definitionId=1'>Dev</a> Succeeded.",  
    "markdown": "Deployment on environment [Dev]  
(http://fabfiber.visualstudio.com/DefaultCollection/Fabrikam-Fiber-  
Git/_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=8&definitionId=1) Succeeded."  
  },  
  "detailedMessage": {  
    "text": "Deployment of release Release-1 on environment Dev Succeeded. Time to  
deploy: 0.11 minutes.",  
    "html": "Deployment on environment <a  
href='http://fabfiber.visualstudio.com/DefaultCollection/Fabrikam-Fiber-  
Git/_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=8&definitionId=1'>Dev</a> Succeeded. Time to  
deploy: 0.11 minutes.",  
    "markdown": "Deployment on environment [Dev]  
(http://fabfiber.visualstudio.com/DefaultCollection/Fabrikam-Fiber-  
Git/_apps/hub/ms.azure-devops-releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=8&definitionId=1) Succeeded. Time to deploy: 0.11  
minutes."  
  },  
  "resource": {  
    "deployment": {  
      "id": 1,  
      "status": "succeeded",  
      "release": {  
        "id": 1,  
        "name": "Release-1",  
        "releaseDefinition": {  
          "id": 1,  
          "name": "Release Definition"  
        },  
        "project": {  
          "id": "project-id",  
          "name": "project-name"  
        }  
      },  
      "environment": {  
        "id": 8,  
        "name": "Dev"  
      }  
    }  
  }  
},
```

```
"createdDate": "2024-12-02T12:21:13.8866607Z"  
}
```

## Release deployment started

Event: A deployment starts.

- Publisher ID: `rm`
- Event ID: `ms.azure-devops-release.deployment-started-event`
- Resource name: `resource`

## Settings

- `releaseEnvironmentId`: Include only events for deployments in a specific environment.
- `releaseDefinitionId`: Include only events for deployments for a specific pipeline.

## Sample payload

JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "ms.azure-devops-release.deployment-started-event",  
  "publisherId": "rm",  
  "message": {  
    "text": "Deployment of release Release-5 to stage Dev started.",  
    "html": "Deployment on stage <a  
href='http://fabfiber.visualstudio.com/Fabrikam-Fiber-Git/_apps/hub/ms.azure-  
devops-releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=1&definitionId=4'>Dev</a> started.",  
    "markdown": "Deployment on stage [Dev]  
(https://fabfiber.visualstudio.com/Fabrikam-Fiber-Git/_apps/hub/ms.azure-devops-  
releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=1&definitionId=4) started."  
  },  
  "detailedMessage": {  
    "text": "Deployment of release Release-5 on stage Dev started.\r\nTrigger:  
Manual",  
    "html": "Deployment on stage <a  
href='Dev'>http://fabfiber.visualstudio.com/Fabrikam-Fiber-Git/_apps/hub/ms.azure-  
devops-releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=1&definitionId=4</a> started.<br>Trigger: Manual",  
    "markdown": "Deployment on stage [Release-1]  
(https://fabfiber.visualstudio.com/Fabrikam-Fiber-Git/_apps/hub/ms.azure-devops-  
releaseManagement-web.hub-explorer?_a=environment-  
summary&definitionEnvironmentId=1&definitionId=4) started.\r\nTrigger: Dev"  
  },  
}
```

```
"resource": {
    "environment": {
        "id": 5,
        "releaseId": 0,
        "name": "Dev",
        "status": "queued",
        "variables": {},
        "variableGroups": [],
        "preDeployApprovals": [],
        "postDeployApprovals": [],
        "preApprovalsSnapshot": {
            "approvals": [],
            "approvalOptions": {
                "requiredApproverCount": 0,
                "releaseCreatorCanBeApprover": true,
                "autoTriggeredAndPreviousEnvironmentApprovedCanBeSkipped": false,
                "enforceIdentityRevalidation": false,
                "timeoutInMinutes": 0,
                "executionOrder": "beforeGates"
            }
        },
        "postApprovalsSnapshot": {
            "approvals": []
        },
        "deploySteps": [],
        "rank": 1,
        "definitionEnvironmentId": 1,
        "queueId": 1,
        "environmentOptions": {
            "emailNotificationType": "OnlyOnFailure",
            "emailRecipients": "release.environment.owner;release.creator",
            "skipArtifactsDownload": false,
            "timeoutInMinutes": 0,
            "enableAccessToken": false,
            "publishDeploymentStatus": false,
            "badgeEnabled": false,
            "autoLinkWorkItems": false,
            "pullRequestDeploymentEnabled": false
        },
        "demands": [],
        "conditions": [],
        "modifiedOn": "2016-01-21T08:19:17.26Z",
        "workflowTasks": [],
        "deployPhasesSnapshot": [],
        "owner": {
            "displayName": "Chuck Reinhart",
            "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff"
        },
        "scheduledDeploymentTime": "2016-01-21T08:19:17.26Z",
        "schedules": [],
        "release": {
            "id": 5,
            "name": "Release-5",
            "_links": {
                "self": {
                    "href": "https://api.tfs.com/tfs/DefaultCollection/_apis/releases/5"
                }
            }
        }
    }
}
```

```
        "web": {
            "href": "https://fabfiber.visualstudio.com/Fabrikam-Fiber-Git/_release?releaseId=1&_a=release-summary"
        }
    },
    "preDeploymentGatesSnapshot": {
        "id": 0,
        "gatesOptions": null,
        "gates": []
    },
    "postDeploymentGatesSnapshot": {
        "id": 0,
        "gatesOptions": null,
        "gates": []
    }
},
"release": {
    "id": 0,
    "name": null,
    "status": "undefined",
    "createdOn": "0001-01-01T00:00:00",
    "modifiedOn": "0001-01-01T00:00:00",
    "modifiedBy": null,
    "createdBy": null,
    "environments": [],
    "variables": {},
    "variableGroups": [],
    "artifacts": [],
    "releaseDefinition": {
        "id": 1,
        "name": "Fabrikam.CD",
        "projectReference": null,
        "_links": {}
    },
    "releaseDefinitionRevision": 0,
    "reason": "none",
    "releaseNameFormat": null,
    "keepForever": false,
    "definitionSnapshotRevision": 0,
    "logsContainerUrl": null,
    "_links": {},
    "tags": [],
    "triggeringArtifactAlias": null,
    "projectReference": null
},
"project": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "name": "Fabrikam"
}
},
"resourceVersion": "3.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
```

```
        },
        "account": {
            "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
        },
        "project": {
            "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
        }
    },
    "createdDate": "2019-10-10T17:49:39.157Z"
}
```

# Pipeline

The following pipeline events are available for use in service hooks.

## Check updated

Event: A check is updated.

- Publisher ID: `pipelines`
- Event ID: `ms.vss-pipelinechecks-events.check-updated-event`
- Resource name: `check`

## Settings

- `resourceType`: Include only events for checks updated for a specific resource type.

## Sample payload

JSON

```
{
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "ms.vss-pipelinechecks-events.check-updated-event",
    "publisherId": "pipelines",
    "message": {
        "text": "Check with configuration ID 1 updated for resource queue:1",
        "html": "Check with configuration ID 1 updated for resource queue:1",
        "markdown": "Check with configuration ID 1 updated for resource queue:1"
    },
    "detailedMessage": {
        "text": "Check with configuration ID 1 updated for resource queue:1",
        "html": "Check with configuration ID 1 updated for resource queue:1",
        "markdown": "Check with configuration ID 1 updated for resource queue:1"
    },
    "resource": {
```

```

    "resource": {
        "type": "queue",
        "id": "1"
    },
    "checkConfigurationId": 1,
    "projectId": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
},
"resourceVersion": "1.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-12T18:52:30.863Z"
}

```

## Elastic agent pool resized

Event: An elastic agent pool is resized.

- Publisher ID: `distributedtask`
- Event ID: `elasticagentpool.resized`
- Resource name: `elasticagentpool`

## Settings

- `poolId`: Include only events for an elastic agent pool with a specific ID.

## Sample payload

JSON

```
{
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "elasticagentpool.resized",
    "publisherId": "distributedtask",
    "message": {
        "text": "\"Resizing pool Sample pool from 1 to 5 VMs.",
        "html": "\"Resizing pool Sample pool from 1 to 5 VMs.",
        "markdown": "\"Resizing pool Sample pool from 1 to 5 VMs."
    },
    "detailedMessage": {

```

```

    "text": "\"Resizing pool Sample pool from 1 to 5 VMs.",
    "html": "\"Resizing pool Sample pool from 1 to 5 VMs.",
    "markdown": "\"Resizing pool Sample pool from 1 to 5 VMs."
},
"resource": {
    "poolId": 1,
    "poolName": "Sample pool",
    "resourceId": "VM Scale Set Id",
    "previousSize": 1,
    "newSize": 5
},
"resourceVersion": "1.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-12T19:13:58.458Z"
}

```

## Manual intervention pending

Event: A pipeline run starts waiting for manual intervention.

- Publisher ID: `rm`
- Event ID: `manualintervention.pending`
- Resource name: `manualintervention`

## Settings

- `project`: Include only events for manual interventions pending in a specific project.
- `interventionName`: Include only events for manual interventions with a specific name pattern.
- `status`: Include only events for manual interventions with a specific status.

## Sample payload

JSON

```
{
    "publisherId": "rm",

```

```
"eventId": "manualintervention.pending",
"resource": {
    "manualIntervention": {
        "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
        "name": "intervention-name",
        "status": "pending",
        "project": {
            "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
            "name": "project-name"
        }
    }
},
"createdDate": "2024-07-17T21:34:22.338Z"
}
```

## Project-level agent pool created

Event: A project-level agent pool is created.

- Publisher ID: `distributedtask`
- Event ID: `agentqueue.created`
- Resource name: `projectlevelagentpool`

## Settings

- `project`: Include only events for project-level agent pools created in a specific project.

## Sample payload

JSON

```
{
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "agentqueue.created",
    "publisherId": "distributedtask",
    "message": {
        "text": "Creating project-level agent pool Sample pool with id 1.",
        "html": "Creating project-level agent pool Sample pool with id 1.",
        "markdown": "Creating project-level agent pool Sample pool with id 1."
    },
    "detailedMessage": {
        "text": "Creating project-level agent pool Sample pool with id 1.",
        "html": "Creating project-level agent pool Sample pool with id 1.",
        "markdown": "Creating project-level agent pool Sample pool with id 1."
    },
    "resource": {
        "queueId": 1,
        "queueName": "Sample pool",
    }
}
```

```

    "projectId": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
},
"resourceVersion": "1.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-12T19:25:19.515Z"
}

```

## Project-level agent pool updated

Event: A project-level agent pool is updated.

- Publisher ID: `distributedtask`
- Event ID: `agentqueue.updated`
- Resource name: `projectlevelagentpool`

## Settings

- `queueId`: Include only events for project-level agent pools with a specific ID.

## Sample payload

JSON

```
{
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "agentqueue.updated",
    "publisherId": "distributedtask",
    "message": {
        "text": "Updating project-level agent pool Sample pool with id 1.",
        "html": "Updating project-level agent pool Sample pool with id 1.",
        "markdown": "Updating project-level agent pool Sample pool with id 1."
    },
    "detailedMessage": {
        "text": "Updating project-level agent pool Sample pool with id 1.",
        "html": "Updating project-level agent pool Sample pool with id 1.",
        "markdown": "Updating project-level agent pool Sample pool with id 1."
    },
    "resource": {

```

```
        "queueId": 1,
        "queueName": "Sample pool",
        "projectId": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1"
    },
    "resourceVersion": "1.0-preview.1",
    "resourceContainers": {
        "collection": {
            "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
        },
        "account": {
            "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
        },
        "project": {
            "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
        }
    },
    "createdDate": "2025-06-12T19:30:24.500Z"
}
```

## Run state changed

Event: The overall status of a pipeline run changes. A new run starts, or a run transitions to a canceling, canceled, failed, partially succeeded, or succeeded state.

- Publisher ID: `pipelines`
- Event ID: `ms.vss-pipelines.run-state-changed-event`
- Resource name: `resource`

## Settings

- `pipelineId`: Include only events for a specific pipeline.
- `runStateId`: Include only events for runs with a specific new state.
  - Valid values:
    - `InProgress`
    - `Canceled`
    - `Completed`
- `runResultId`: Include only events for runs with a specific result.
  - Valid values:
    - `Failed`
    - `Succeeded`

## Sample payload

## JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.vss-pipelines.run-state-changed-event",
  "publisherId": "pipelines",
  "message": {
    "text": "Run 11 succeeded.",
    "html": "Run <a href=\"https://codedev.ms/org/11bb11bb-cc22-dd33-ee44-55ff55ff55ff/_build/results?buildId=11\">11</a> succeeded.",
    "markdown": "Run [11](https://codedev.ms/org/11bb11bb-cc22-dd33-ee44-55ff55ff55ff/_build/results?buildId=11) succeeded."
  },
  "detailedMessage": {
    "text": "Run 11 succeeded.",
    "html": "Run <a href=\"https://codedev.ms/org/11bb11bb-cc22-dd33-ee44-55ff55ff55ff/_build/results?buildId=11\">11</a> succeeded.",
    "markdown": "Run [11](https://codedev.ms/org/11bb11bb-cc22-dd33-ee44-55ff55ff55ff/_build/results?buildId=11) succeeded."
  },
  "resource": {
    "run": {
      "_links": {
        "self": {
          "href": "https://codedev.ms/org/11bb11bb-cc22-dd33-ee44-55ff55ff55ff/_apis/Pipelines/1/runs/11"
        }
      },
      "web": {
        "href": "https://codedev.ms/org/11bb11bb-cc22-dd33-ee44-55ff55ff55ff/_build/results?buildId=11"
      }
    },
    "pipeline": {
      "id": 1,
      "name": "Pipeline-Name"
    },
    "state": "completed",
    "result": "succeeded",
    "createdDate": "2024-07-17T21:34:22.338Z",
    "finishedDate": "2024-07-17T21:45:22.338Z",
    "url": "https://codedev.ms/org/11bb11bb-cc22-dd33-ee44-55ff55ff55ff/_apis/Pipelines/1/runs/11"
  }
},
"createdDate": "2024-07-17T21:34:22.338Z"
}
```

## Run stage state changed

Event: A new stage starts in a pipeline run, or a stage transitions to a canceling, canceled, failed, partially succeeded, or succeeded state.

- Publisher ID: pipelines
- Event ID: ms.vss-pipelines.stage-state-changed-event
- Resource name: resource

## Settings

- pipelineId: Include only events for a specific pipeline.
- stageNameId: Include only events for a specific stage name.
- stageStateId: Include only events for a stage in a specific new state.
  - Valid values:
    - NotStarted
    - Waiting
    - Running
    - Completed
- stageResultId: Include only events for stages with a specific result.
  - Valid values:
    - Canceled
    - Failed
    - Rejected
    - Skipped
    - Succeeded

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.vss-pipelines.stage-state-changed-event",
  "publisherId": "pipelines",
  "message": {
    "text": "Run 2 stage __default succeeded.",
    "html": "Run 2 stage <a href=\"https://codedev.ms/org/b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/_build/results?buildId=2\">__default</a> succeeded.",
    "markdown": "Run 2 stage [__default](https://codedev.ms/org/b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/_build/results?buildId=2) succeeded."
  },
  "detailedMessage": {
    "text": "Run 2 stage __default succeeded.",
    "html": "Run 2 stage <a href=\"https://codedev.ms/org/b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/_build/results?buildId=2\">__default</a> succeeded.",
    "markdown": "Run 2 stage [__default](https://codedev.ms/org/b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/_build/results?buildId=2) succeeded."
  }
},
```

```
"resource": {
  "stage": {
    "_links": {
      "web": {
        "href": "https://codedev.ms/org/b1b1b1b1-cccc-dddd-eaaa-f2f2f2f2f2f2/_build/results?buildId=2"
      },
      "pipeline.web": {
        "href": "https://codedev.ms/org/b1b1b1b1-cccc-dddd-eaaa-f2f2f2f2f2f2/_build/definition?definitionId=2"
      }
    },
    "id": "c2c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3a3",
    "name": "__default",
    "displayName": null,
    "state": "completed",
    "result": "succeeded"
  },
  "run": {
    "pipeline": {
      "url": "https://codedev.ms/org/d3d3d3d3-eaaa-ffff-aaaa-b4b4b4b4b4/_apis/Pipelines/2?revision=2",
      "id": 2,
      "revision": 2,
      "name": "TEST-CI",
      "folder": "\\"
    },
    "state": "completed",
    "result": "succeeded",
    "createdDate": "2019-12-13T06:10:10.164Z",
    "finishedDate": "2019-12-13T06:10:10.164Z",
    "id": 2,
    "name": "2"
  },
  "pipeline": {
    "url": "https://codedev.ms/org/d3d3d3d3-eaaa-ffff-aaaa-b4b4b4b4b4/_apis/Pipelines/2?revision=2",
    "id": 2,
    "revision": 2,
    "name": "TEST-CI",
    "folder": "\\"
  },
  "repositories": [
    {
      "type": "Git",
      "change": {
        "author": {
          "name": "Himani Maharjan",
          "email": "himani@fabrikamfiber.com",
          "date": "2024-11-11T15:09:21Z"
        },
        "committer": {
          "name": "Himani Maharjan",
          "email": "himani@fabrikamfiber.com",
          "date": "2024-11-11T15:09:21Z"
        }
      }
    }
  ]
}
```

```

        "name": "Himani Maharjan",
        "email": "himani@fabrikamfiber.com",
        "date": "2024-11-11T15:09:21Z"
    },
    "message": "Added Viva support"
},
"url": "https://fabrikamfiber@dev.azure.com/fabrikamfiber/fabrikamfiber-viva/_git/fabrikamfiber"
}
],
},
"resourceVersion": "5.1-preview.1",
"resourceContainers": {
    "collection": {
        "id": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2019-12-13T06:10:10.186Z"
}

```

## Run stage waiting for approval

Event: An approval is created for a stage in a pipeline run.

- Publisher ID: `pipelines`
- Event ID: `ms.vss-pipelinechecks-events.approval-pending`
- Resource name: `resource`

## Settings

- `pipelineId`: Include only events for a pipeline with a specific ID.
- `stageName`: Include only events for deployment approvals for a specific stage name.
- `environmentName`: Include only events for deployment approvals in a specific environment.

## Sample payload

JSON

```
{
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "ms.vss-pipelinechecks-events.approval-pending",
    "stageName": "Deploy to Production"
}
```

```
"publisherId": "pipelines",
"message": {
    "text": "Approval pending for deployment of pipeline run1 to environment env1.",
    "html": "Approval pending for deployment of pipeline <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results> run1 </a> to environment <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources>env1</a>.",
    "markdown": "Approval pending for deployment of pipeline [https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results](run1) to environment [env1] (https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources)"
},
"detailedMessage": {
    "text": "Approval pending for deployment of pipeline run1 to environment env1.",
    "html": "Approval pending for deployment of pipeline <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results> run1 </a> to environment <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources>env1</a>.",
    "markdown": "Approval pending for deployment of pipeline [https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results](run1) to environment [env1] (https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources)"
},
"resource": {
    "approval": {
        "id": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6",
        "steps": [
            {
                "assignedApprover": {
                    "displayName": null,
                    "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3a3"
                },
                "status": "pending",
                "comment": "Sample comment",
                "initiatedOn": "2019-12-13T06:14:11.642Z"
            }
        ],
        "status": "pending",
        "createdOn": "2019-12-13T06:14:11.642Z",
        "lastModifiedOn": "2019-12-13T06:14:11.642Z",
        "instructions": "Instructions",
        "minRequiredApprovers": 2,
        "blockedApprovers": [
            {
                "displayName": null,
                "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4b4"
            }
        ],
        "_links": {}
    }
}
```

```
 },
  "projectId": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5",
  "pipeline": null,
  "stage": null,
  "run": null,
  "resource": null,
  "id": 0,
  "url": null,
  "stageName": null,
  "attemptId": 0
},
"resourceVersion": "5.1-preview.1",
"resourceContainers": {
  "collection": {
    "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
  },
  "account": {
    "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
  },
  "project": {
    "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
  }
},
"createdDate": "2019-12-13T06:14:11.671Z"
}
```

## Run stage approval completed

Event: An approval is completed for a stage in a pipeline run.

- Publisher ID: `pipelines`
- Event ID: `ms.vss-pipelinechecks-events.approval-completed`
- Resource name: `resource`

## Settings

- `pipelineId`: Include only events for a pipeline with a specific ID.
- `stageName`: Include only events for a specific stage name.
- `environmentName`: Include only events for deployment approvals in a specific environment.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.vss-pipelinechecks-events.approval-completed",
```

```
"publisherId": "pipelines",
"message": {
    "text": "Approval completed for deployment of pipeline run1 to environment env1.",
    "html": "Approval completed for deployment of pipeline <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results> run1 </a> to environment <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources>env1</a>.",
    "markdown": "Approval completed for deployment of pipeline [https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results](run1) to environment [env1] (https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources)"
},
"detailedMessage": {
    "text": "Approval completed for deployment of pipeline run1 to environment env1.",
    "html": "Approval completed for deployment of pipeline <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results> run1 </a> to environment <a href=https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources>env1</a>.",
    "markdown": "Approval completed for deployment of pipeline [https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_build/results?buildId=2&view=results](run1) to environment [env1] (https://dev.azure.com/fabfiber/Fabrikam-Fiber-Git/_environments/1?view=resources)"
},
"resource": {
    "approval": {
        "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3a3",
        "steps": [
            {
                "assignedApprover": {
                    "displayName": null,
                    "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4b4"
                },
                "status": "approved",
                "comment": "Sample comment",
                "initiatedOn": "2019-12-13T06:18:22.460Z"
            }
        ],
        "status": "approved",
        "createdOn": "2019-12-13T06:18:22.460Z",
        "lastModifiedOn": "2019-12-13T06:18:22.460Z",
        "instructions": "Instructions",
        "minRequiredApprovers": 2,
        "blockedApprovers": [
            {
                "displayName": null,
                "id": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5"
            }
        ],
        "_links": {}
    }
}
```

```
 },
  "projectId": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6",
  "pipeline": null,
  "stage": null,
  "run": null,
  "resource": null,
  "id": 0,
  "url": null,
  "stageName": null,
  "attemptId": 0
},
"resourceVersion": "5.1-preview.1",
"resourceContainers": {
  "collection": {
    "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
  },
  "account": {
    "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
  },
  "project": {
    "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
  }
},
"createdDate": "2019-12-13T06:18:22.487Z"
}
```

## Run job state changed

Event: A new job starts running, finishes, or starts waiting for an agent.

- Publisher ID: `pipelines`
- Event ID: `ms.vss-pipelines.job-state-changed-event`
- Resource name: `resource`

## Settings

- `pipelineId`: Include only events for a specific pipeline.
- `stageNameId`: Include only events for a specific stage name.
- `jobNameId`: Include only events for a specific job name.
- `jobStateId`: Include only events for a job in a specific state.
  - Valid values:
    - `Waiting`
    - `Running`
    - `Completed`
- `jobResultId`: Include only events for a job that has a specific result.
  - Valid values:

- Succeeded
- Skipped
- Rejected
- Failed
- Canceled

## Sample payload

JSON

```
{
  "subscriptionId": "aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e",
  "notificationId": 3,
  "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3a3",
  "eventType": "ms.vss-pipelines.job-state-changed-event",
  "publisherId": "pipelines",
  "message": {
    "text": "Run 20241121.5 stage Build job Compile succeeded.",
    "html": "Run 20241121.5 stage Build job <a href=\"https://dev.azure.com/fabrikamfiber/fabrikamfiber-viva/_build/results?buildId=2710088\">Compile</a> succeeded.",
    "markdown": "Run 20241121.5 stage Build job [Compile](https://dev.azure.com/fabrikamfiber/fabrikamfiber-viva/_build/results?buildId=2710088) succeeded."
  },
  "detailedMessage": {
    "text": "Run 20241121.5 stage Build job Compile succeeded.",
    "html": "Run 20241121.5 stage Build job <a href=\"https://dev.azure.com/fabrikamfiber/fabrikamfiber-viva/_build/results?buildId=2710088\">Compile</a> succeeded.",
    "markdown": "Run 20241121.5 stage Build job [Compile](https://dev.azure.com/fabrikamfiber/fabrikamfiber-viva/_build/results?buildId=2710088) succeeded."
  },
  "resource": {
    "job": {
      "_links": {
        "web": {
          "href": "https://dev.azure.com/fabrikamfiber/fabrikamfiber-viva/_build/results?buildId=2"
        },
        "pipeline.web": {
          "href": "https://dev.azure.com/fabrikamfiber/fabrikamfiber-viva/_build/definition?definitionId=2"
        }
      }
    }
  }
}
```

```
        }
    },
    "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4b4",
    "name": "__default",
    "state": "completed",
    "result": "succeeded",
    "startTime": "2024-11-21T16:42:52.7761408Z",
    "finishTime": "2024-11-21T16:42:52.7761408Z"
},
"stage":
{
    "id": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5",
    "name": "__default",
    "displayName": null,
    "state": "completed",
    "result": "succeeded",
    "startTime": null,
    "finishTime": null
},
"run":
{
    "pipeline":
    {
        "url": "https://codedev.ms/org/f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6/_apis/Pipelines/2?revision=2",
        "id": 2,
        "revision": 2,
        "name": "TEST-CI",
        "folder": "\\"
    },
    "state": "completed",
    "result": "succeeded",
    "createdDate": "2024-11-21T16:42:52.7761408Z",
    "finishedDate": "2024-11-21T16:42:52.7761408Z",
    "id": 2,
    "name": "2"
},
"pipeline":
{
    "url": "https://codedev.ms/org/f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6/_apis/Pipelines/2?revision=2",
    "id": 2,
    "revision": 2,
    "name": "TEST-CI",
    "folder": "\\"
},
"repositories":
[
    {
        "type": "Git",
        "change":
        {
            "author":
            {
                "name": "Himani Maharjan",
                "email": "himani.maharjan@example.com"
            }
        }
    }
]
```

```
        "email": "himani@fabrikamfiber.com",
        "date": "2024-11-11T15:09:21Z"
    },
    "committer":
    {
        "name": "Himani Maharjan",
        "email": "himani@fabrikamfiber.com",
        "date": "2024-11-11T15:09:21Z"
    },
    "message": "Added Viva support"
},
"url":
"https://fabrikamfiber@dev.azure.com/fabrikamfiber/fabrikamfiber-
viva/_git/fabrikamfiber"
},
{
    "type": "GitHub",
    "change":
    {
        "author":
        {
            "name": "Himani Maharjan",
            "email": "himani@github.com",
            "date": "2024-08-11T15:05:20Z"
        },
        "committer":
        {
            "name": "Himani Maharjan",
            "email": "himani@github.com",
            "date": "2024-08-11T15:05:20Z"
        },
        "message": "Added Viva open source REST API library"
},
    "url": "https://api.github.com/repos/FabrikamFiber/Viva"
}
]
},
"resourceVersion": "5.1-preview.1",
"resourceContainers":
{
    "collection":
    {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account":
    {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project":
    {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2024-11-21T16:42:53.5254422Z"
}
```

# Code

The following code events are available for use in service hooks.

## Code checked in

Event: A changeset is checked into Team Foundation Version Control (TFVC).

- Publisher ID: `tfs`
- Event ID: `tfvc.checkin`
- Resource name: `changeset`

## Settings

- `path`: Include only events for check-ins that change files under a specific path.
  - Required

## Sample payload

JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "tfvc.checkin",  
  "publisherId": "tfs",  
  "scope": "all",  
  "message": {  
    "text": "Normal Paulk checked in changeset 18: Dropping in new Java sample",  
    "html": "Normal Paulk checked in changeset <a  
href=\"https://dev.azure.com/fabrikam-fiber-inc/web/cs.aspx?pcguid=c2c2c2c2-dddd-  
eeee-ffff-a3a3a3a3a3a3\">18</a>: Dropping in new Java sample",  
    "markdown": "Chuck Reinhart checked in changeset [18]  
(https://dev.azure.com/fabrikam-fiber-inc/web/cs.aspx?pcguid=c2c2c2c2-dddd-eeee-  
ffff-a3a3a3a3a3a3): Dropping in new Java sample"  
  },  
  "detailedMessage": {  
    "text": "Chuck Reinhart checked in changeset 18: Dropping in new Java sample",  
    "html": "Chuck Reinhart checked in changeset <a  
href=\"https://dev.azure.com/fabrikam-fiber-inc/web/cs.aspx?pcguid=c2c2c2c2-dddd-  
eeee-ffff-a3a3a3a3a3&cs=18\">18</a>: Dropping in new Java sample",  
    "markdown": "Chuck Reinhart checked in changeset [18]  
(https://dev.azure.com/fabrikam-fiber-inc/web/cs.aspx?pcguid=c2c2c2c2-dddd-eeee-  
ffff-a3a3a3a3a3a3): Dropping in new Java sample"  
  },  
  "resource": {
```

```
"changesetId": 18,
  "url": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/tfvc/changesets/18",
  "author": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "Chuck Reinhart",
    "uniqueName": "fabrikamfiber16@hotmail.com"
  },
  "checkedInBy": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "Chuck Reinhart",
    "uniqueName": "fabrikamfiber16@hotmail.com"
  },
  "createdDate": "2014-05-12T22:41:16Z",
  "comment": "Dropping in new Java sample"
},
"resourceVersion": "1.0",
"resourceContainers": {
  "collection": {
    "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
  },
  "account": {
    "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
  }
},
"createdDate": "2024-09-19T13:03:26.2056408Z"
}
```

## Code pushed

Event: Code is pushed to a Git repository.

- Publisher ID: `tfs`
- Event ID: `git.push`
- Resource name: `push`

## Settings

- `branch`: Include only events for code pushes to a specific branch.
- `pushedBy`: Include only events for code pushes by users in a specific group.
- `repository`: Include only events for code pushes to a specific repository.
  - Data type: `guid`

## Sample payload

JSON

```
{  
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
    "eventType": "git.push",  
    "publisherId": "tfs",  
    "scope": "all",  
    "message": {  
        "text": "Jamal Hartnett pushed updates to branch main of repository Fabrikam-Fiber-Git.",  
        "html": "Jamal Hartnett pushed updates to branch main of repository Fabrikam-Fiber-Git.",  
        "markdown": "Jamal Hartnett pushed updates to branch `main` of repository `Fabrikam-Fiber-Git`."  
    },  
    "detailedMessage": {  
        "text": "Jamal Hartnett pushed 1 commit to branch main of repository Fabrikam-Fiber-Git.\n- Fixed bug in web.config file 33b55f7c",  
        "html": "Jamal Hartnett pushed 1 commit to branch <a href=\"https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_git/Fabrikam-Fiber-Git/#version=GBmain\">main</a> of repository <a href=\"https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_git/Fabrikam-Fiber-Git/\">Fabrikam-Fiber-Git</a>. \n- Fixed bug in web.config file <a href=\"https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_git/Fabrikam-Fiber-Git/commit/4444eeee455ff5aaaaabb66ccccccccc7777cccc\">33b55f7c</a>\n",  
        "markdown": "Jamal Hartnett pushed 1 commit to branch [main] (https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/\_git/Fabrikam-Fiber-Git/#version=GBmain) of repository [Fabrikam-Fiber-Git] (https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/\_git/Fabrikam-Fiber-Git/).\n* Fixed bug in web.config file [33b55f7c](https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/\_git/Fabrikam-Fiber-Git/commit/4444eeee455ff5aaaaabb66ccccccccc7777cccc)"  
    },  
    "resource": {  
        "commits": [  
            {  
                "commitId": "4444eeee455ff5aaaaabb66ccccccccc7777cccc",  
                "author": {  
                    "name": "Jamal Hartnett",  
                    "email": "fabrikamfiber4@hotmail.com",  
                    "date": "2024-02-25T19:01:00Z"  
                },  
                "committer": {  
                    "name": "Jamal Hartnett",  
                    "email": "fabrikamfiber4@hotmail.com",  
                    "date": "2024-02-25T19:01:00Z"  
                },  
                "comment": "Fixed bug in web.config file",  
                "url": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_git/Fabrikam-Fiber-Git/commit/4444eeee455ff5aaaaabb66ccccccccc7777cccc"  
            }  
        ],  
        "refUpdates": [  
            {  
                "name": "refs/heads/main",  
                "oldValue": null,  
                "newValue": "4444eeee455ff5aaaaabb66ccccccccc7777cccc"  
            }  
        ]  
    }  
}
```

```

        "oldObjectId": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4",
        "newObjectId": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5"
    }
],
"repository": {
    "id": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6",
    "name": "Fabrikam-Fiber-Git",
    "url": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/repos/git/repositories/f5f5f5f5-aaaa-bbbb-cccc-
d6d6d6d6d6",
    "project": {
        "id": "a6a6a6a6-bbbb-cccc-dddd-e7e7e7e7e7",
        "name": "Fabrikam-Fiber-Git",
        "url": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/projects/a6a6a6a6-bbbb-cccc-dddd-e7e7e7e7e7",
        "state": "wellFormed"
    },
    "defaultBranch": "refs/heads/main",
    "remoteUrl": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_git/Fabrikam-Fiber-Git"
},
"pushedBy": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff@Live.com",
    "displayName": "Jamal Hartnett",
    "uniqueName": "Windows Live ID\fabrikamfiber4@hotmail.com"
},
"pushId": 14,
"date": "2014-05-02T19:17:13.3309587Z",
"url": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/repos/git/repositories/f5f5f5f5-aaaa-bbbb-cccc-
d6d6d6d6d6/pushes/14"
},
"resourceVersion": "1.0",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffffff5f5f5"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2024-09-19T13:03:27.0379153Z"
}

```

## Pull request created

Event: A pull request is created in a Git repository.

- Publisher ID: tfs

- Event ID: `git.pullrequest.created`
- Resource name: `pullrequest`

## Settings

- `repository`: Include only events for pull requests in a specific repository.
  - Data type: `guid`
- `pullrequestCreatedBy`: Include only events for pull requests created by users in a specific group.
- `pullrequestReviewersContains`: Include only events for pull requests with reviewers in a specific group.
- `branch`: Include only events for pull requests in a specific branch.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "git.pullrequest.created",
  "publisherId": "tfs",
  "scope": "all",
  "message": {
    "text": "Jamal Hartnett created a new pull request",
    "html": "Jamal Hartnett created a new pull request",
    "markdown": "Jamal Hartnett created a new pull request"
  },
  "detailedMessage": {
    "text": "Jamal Hartnett created a new pull request\r\n\r\n- Merge status: Succeeded\r\n- Merge commit: 6666aa(https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/commits/6666aaaa677bb7ccccdd88eeeeee999999eeee)\r\n",
    "html": "Jamal Hartnett created a new pull request\r\n<ul>\r\n- Merge status: Succeeded<br>\r\n- Merge commit: <a href=\"https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/commits/6666aaaa677bb7ccccdd88eeeeee999999eeee\">6666aa</a><br>\r</ul>",
    "markdown": "Jamal Hartnett created a new pull request\r\n\r\n- Merge status: Succeeded\r\n- Merge commit: [6666aa]\r\n(https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/commits/6666aaaa677bb7ccccdd88eeeeee999999eeee)\r\n"
  },
  "resource": {
    "repository": {
      "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    }
  }
}
```

```
"name": "Fabrikam",
"url":
"https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2",
"project": {
    "id": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6",
    "name": "Fabrikam",
    "url":
"https://dev.azure.com/fabrikam/DefaultCollection/_apis/projects/abcd-1234-efgh-5678",
    "state": "wellFormed"
},
"defaultBranch": "refs/heads/main",
"remoteUrl":
"https://dev.azure.com/fabrikam/DefaultCollection/_git/Fabrikam"
},
"pullRequestId": 1,
"status": "active",
"createdBy": {
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "Jamal Hartnett",
    "uniqueName": "fabrikamfiber4@hotmail.com",
    "url": "https://dev.azure.com/fabrikam/_apis/Identities/11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "imageUrl":
"https://dev.azure.com/fabrikam/DefaultCollection/_api/_common/identityImage?id=11bb11bb-cc22-dd33-ee44-55ff55ff55ff"
},
"creationDate": "2024-06-17T11:22:33.456789Z",
"title": "my first pull request",
"description": " - test2\r\n",
"sourceRefName": "refs/heads/mytopic",
"targetRefName": "refs/heads/main",
"mergeStatus": "succeeded",
"mergeId": "a6a6a6a6-bbbb-cccc-dddd-e7e7e7e7e7",
"lastMergeSourceCommit": {
    "commitId": "4444eeee455ff5aaaabb66cccccccc7777cccc",
    "url":
"https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/commits/4444eeee455ff5aaaabb66cccccccc7777cccc"
},
"lastMergeTargetCommit": {
    "commitId": "5555ffff566aa6bbbbbcc77ddddddd8888888ddd",
    "url":
"https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/commits/5555ffff566aa6bbbbbcc77ddddddd8888888ddd"
},
"lastMergeCommit": {
    "commitId": "6666aaaa677bb7ccccdd88eeeeeee999999eeee",
    "url":
"https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1-cccc-dddd-eeee-f2f2f2f2f2/commits/6666aaaa677bb7ccccdd88eeeeeee999999eeee"
```

```

},
"reviewers": [
  {
    "reviewerUrl": null,
    "vote": 0,
    "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
    "displayName": "[Mobile]\\Mobile Team",
    "uniqueName": "azure-devops:///Classification/TeamProject/22cc22cc-dd33-ee44-ff55-66aa66aa66aa\\Mobile Team",
    "url": "https://dev.azure.com/fabrikam/_apis/Identities/22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
    "imageUrl":
"https://dev.azure.com/fabrikam/DefaultCollection/_api/_common/identityImage?
id=22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
    "isContainer": true
  }
],
"url":
"https://dev.azure.com/fabrikam/DefaultCollection/_apis/repos/git/repositories/b1b1b1cccc-dddd-eeee-f2f2f2f2f2f2/pullRequests/1"
},
"resourceVersion": "1.0",
"resourceContainers": {
  "collection": {
    "id": "b1b1b1b1cccc-dddd-eeee-f2f2f2f2f2f2"
  },
  "account": {
    "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
  },
  "project": {
    "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
  }
},
"createdDate": "2024-09-19T13:03:27.2879096Z"
}

```

## Pull request merge attempted

Event: A pull request merge is attempted in a Git repository.

- Publisher ID: `tfs`
- Event ID: `git.pullrequest.merged`
- Resource name: `pullrequest`

## Settings

- `repository`: Include only events for pull requests in a specific repository.
  - Data type: `guid`

- `pullrequestCreatedBy`: Include only events for pull requests created by users in a specific group.
- `pullrequestReviewersContains`: Include only events for pull requests with reviewers in a specific group.
- `branch`: Include only events for pull requests in a specific branch.
- `mergeResult`: Include only events for pull requests with a specific merge result.
  - Valid values:
    - `Succeeded`
    - `Unsuccessful`
    - `Conflicts`
    - `Failure`
    - `RejectedByPolicy`

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "git.pullrequest.merged",
  "publisherId": "tfs",
  "message": {
    "text": "Jamal Hartnett has created a pull request merge commit",
    "html": "Jamal Hartnett has created a pull request merge commit",
    "markdown": "Jamal Hartnett has created a pull request merge commit"
  },
  "detailedMessage": {
    "text": "Jamal Hartnett has created a pull request merge commit\r\n\r\nMerge status: Succeeded\r\n- Merge commit:  
4444ee(https://fabrikam.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaaabb66cccccccc7777cccc)\r\n",
    "html": "Jamal Hartnett has created a pull request merge  
commit\r\n

- Merge status: Succeeded
- Merge commit: <a href=\"https://fabrikam.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaaabb66cccccccc7777cccc\">4444ee</a>

  
",
    "markdown": "Jamal Hartnett has created a pull request merge  
commit\r\n+ Merge status: Succeeded\r\n+ Merge commit: [4444ee]  
(https://fabrikam.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaaabb66cccccccc7777cccc)\r\n"
  },
  "resource": {
    "repository": {
      "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
      "name": "Fabrikam",
    }
  }
}
```

```
"url":  
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2  
c2-dddd-eaaa-ffff-a3a3a3a3a3",  
    "project": {  
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee",  
        "name": "Fabrikam",  
        "url":  
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/projects/00aa00aa-bb11-  
cc22-dd33-44ee44ee44ee",  
            "state": "wellFormed",  
            "visibility": "unchanged",  
            "lastUpdateTime": "2001-01-01T00:00:00"  
    },  
    "defaultBranch": "refs/heads/main",  
    "remoteUrl":  
"https://fabrikam.visualstudio.com/DefaultCollection/_git/Fabrikam"  
},  
    "pullRequestId": 1,  
    "status": "completed",  
    "createdBy": {  
        "displayName": "Jamal Hartnett",  
        "url":  
"https://fabrikam.vssps.visualstudio.com/_apis/Identities/22cc22cc-dd33-ee44-ff55-  
66aa66aa66aa",  
        "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",  
        "uniqueName": "fabrikamfiber4@hotmail.com",  
        "imageUrl":  
"https://fabrikam.visualstudio.com/DefaultCollection/_api/_common/identityImage?  
id=22cc22cc-dd33-ee44-ff55-66aa66aa66aa"  
    },  
    "creationDate": "2014-06-17T16:55:46.589889Z",  
    "closedDate": "2014-06-30T18:59:12.3660573Z",  
    "title": "my first pull request",  
    "description": " - test2\r\n",  
    "sourceRefName": "refs/heads/mytopic",  
    "targetRefName": "refs/heads/main",  
    "mergeStatus": "succeeded",  
    "mergeId": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6",  
    "lastMergeSourceCommit": {  
        "commitId": "6666aaaa677bb7ccccdd88eeeeeee999999eeee",  
        "url":  
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2  
c2-dddd-eaaa-ffff-a3a3a3a3a3/commits/6666aaaa677bb7ccccdd88eeeeeee999999eeee"  
    },  
    "lastMergeTargetCommit": {  
        "commitId": "5555ffff566aa6bbbbbcc77dddddd888888ddd",  
        "url":  
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2  
c2-dddd-eaaa-ffff-a3a3a3a3a3/commits/5555ffff566aa6bbbbbcc77dddddd888888ddd"  
    },  
    "lastMergeCommit": {  
        "commitId": "4444eeee455ff5aaaaabb6cccccccc7777cccc",  
        "url":  
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2  
c2-dddd-eaaa-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaaabb6cccccccc7777cccc"
```

```
        },
        "reviewers": [
            {
                "reviewerUrl": "https://fabrikam.visualstudio.com/\_apis/git/repositories/c2c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3/pullRequests/1/reviewers/11bb11bb-cc22-dd33-ee44-55ff55ff55ff" ,
                    "vote": 0,
                    "displayName": "[Mobile]\\Mobile Team",
                    "url": "https://fabrikam.vssps.visualstudio.com/\_apis/Identities/11bb11bb-cc22-dd33-ee44-55ff55ff55ff" ,
                        "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
                        "uniqueName": "vstfs:///Classification/TeamProject/00aa00aa-bb11-cc22-dd33-44ee44ee44ee\\Mobile Team",
                        "imageUrl": "https://fabrikam.visualstudio.com/DefaultCollection/\_api/\_common/identityImage?id=11bb11bb-cc22-dd33-ee44-55ff55ff55ff" ,
                            "isContainer": true
                }
            ],
            "commits": [
                {
                    "commitId": "4444eeee455ff5aaaaabb66cccccccc7777cccc",
                    "url": "https://fabrikam.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3a3/commits/4444eeee455ff5aaaaabb66cccccccc7777cccc" 
                }
            ],
            "url": "https://fabrikam.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3a3/pullRequests/1" ,
                "_links": {
                    "web": {
                        "href": "https://fabrikam.visualstudio.com/DefaultCollection/\_git/Fabrikam/pullrequest/1#view=discussion" 
                    },
                    "statuses": {
                        "href": "https://fabrikam.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3a3/pullRequests/1/statuses" 
                    }
                }
            },
            "resourceVersion": "1.0-preview.1",
            "resourceContainers": {
                "collection": {
                    "id": "b1b1b1b1-cccc-dddd-eaaa-f2f2f2f2f2"
                },
                "account": {
                    "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
                },
                "project": {
                    "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
                }
            }
        }
    }
}
```

```
  },
  "createdDate": "2025-06-12T19:57:30.694Z"
}
```

## Pull request updated

Event: A pull request is updated: the status, review list, or a reviewer vote changes, or the source branch is updated by a push.

- Publisher ID: `tfs`
- Event ID: `git.pullrequest.updated`
- Resource name: `pullrequest`

## Settings

- `notificationType`: Include only events for pull requests with a specific change.
  - Valid values:
    - `PushNotification` - The source branch is updated.
    - `ReviewersUpdateNotification` - The reviewers change.
    - `StatusUpdateNotification` - The status changes.
    - `ReviewerVoteNotification` - The votes score changes.
- `repository`: Include only events for pull requests in a specific repository.
  - Data type: `guid`
- `pullrequestCreatedBy`: Include only events for pull requests created by users in a specific group.
- `pullrequestReviewersContains`: Include only events for pull requests with reviewers in a specific group.
- `branch`: Include only events for pull requests in a specific branch.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "git.pullrequest.updated",
  "publisherId": "tfs",
  "scope": "all",
  "message": {
    "text": "Jamal Hartnett marked the pull request as completed",
    "html": "Jamal Hartnett marked the pull request as completed",
    "markdown": "Jamal Hartnett marked the pull request as completed"
  }
}
```

```
},
  "detailedMessage": {
    "text": "Jamal Hartnett marked the pull request as completed\r\n\r\n- Merge status: Succeeded\r\n- Merge commit:  
4444ee(https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaaabb66cccccccc7777cccc)\r\n",
    "html": "Jamal Hartnett marked the pull request as completed\r\n<ul>\r\n- Merge status: Succeeded<br>\r\n- Merge commit: <a href=\"https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaaabb66cccccccc7777cccc\">4444ee</a><br>\r</ul>",
    "markdown": "Jamal Hartnett marked the pull request as completed\r\n\r\n- Merge status: Succeeded\r\n- Merge commit: [4444ee](https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaaabb66cccccccc7777cccc)\r\n"
  },
  "resource": {
    "repository": {
      "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
      "name": "Fabrikam",
      "url": "https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
      "project": {
        "id": "d3d3d3d3-eccc-ffff-aaaa-b4b4b4b4b4",
        "name": "Fabrikam",
        "url": "https://dev.azure.com/fabrikam/DefaultCollection/\_apis/projects/d3d3d3d3-eccc-ffff-aaaa-b4b4b4b4b4",
        "state": "wellFormed"
      },
      "defaultBranch": "refs/heads/main",
      "remoteUrl": "https://dev.azure.com/fabrikam/DefaultCollection/\_git/Fabrikam"
    },
    "pullRequestId": 1,
    "status": "completed",
    "createdBy": {
      "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
      "displayName": "Jamal Hartnett",
      "uniqueName": "fabrikamfiber4@hotmail.com",
      "url": "https://dev.azure.com/fabrikam/\_apis/Identities/22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
      "imageUrl": "https://dev.azure.com/fabrikam/DefaultCollection/\_api/\_common/identityImage?id=22cc22cc-dd33-ee44-ff55-66aa66aa66aa"
    },
    "creationDate": "2024-06-17T16:55:46.589889Z",
    "closedDate": "2024-06-30T18:59:12.3660573Z",
    "title": "my first pull request",
    "description": "- test2\r\n",
    "sourceRefName": "refs/heads/mytopic",
```

```
"targetRefName": "refs/heads/main",
"mergeStatus": "succeeded",
"mergeId": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6",
"lastMergeSourceCommit": {
    "commitId": "5555ffff566aa6bbbbbcc77dddddd888888ddd",
    "url":
https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3/commits/5555ffff566aa6bbbbbcc77dddddd888888ddd
},
"lastMergeTargetCommit": {
    "commitId": "6666aaaa677bb7ccccdd88eeeeee999999eeee",
    "url":
https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3/commits/6666aaaa677bb7ccccdd88eeeeee999999eeee
},
"lastMergeCommit": {
    "commitId": "4444eeee455ff5aaaabb66ccccccccc7777cccc",
    "url":
https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaabb66ccccccccc7777cccc
},
"reviewers": [
{
    "reviewerUrl": null,
    "vote": 0,
    "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "displayName": "[Mobile]\\Mobile Team",
    "uniqueName": "azure-devops:///Classification/TeamProject/d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4\\Mobile Team",
    "url": "https://dev.azure.com/fabrikam/_apis/Identities/11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
    "imageUrl":
https://dev.azure.com/fabrikam/DefaultCollection/\_api/\_common/identityImage?id=11bb11bb-cc22-dd33-ee44-55ff55ff55ff
}
],
"commits": [
{
    "commitId": "4444eeee455ff5aaaabb66ccccccccc7777cccc",
    "url":
https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaabb66ccccccccc7777cccc
}
],
"url":
https://dev.azure.com/fabrikam/DefaultCollection/\_apis/repos/git/repositories/c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3/pullRequests/1
},
"resourceVersion": "1.0",
"resourceContainers": {
```

```
    "collection": {
      "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
      "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffffff5f5f5f"
    },
    "project": {
      "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
  },
  "createdDate": "2024-09-19T13:03:27.2813828Z"
}
```

## Pull request commented on

Event: A pull request is commented on.

- Publisher ID: `tfs`
- Event ID: `ms.vss-code.git-pullrequest-comment-event`
- Resource name: `pullrequest`

## Settings

- `repository`: Include only events for pull requests in a specific repository.
  - Data type: `guid`
- `branch`: Include only events for pull requests in a specific branch.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.vss-code.git-pullrequest-comment-event",
  "publisherId": "tfs",
  "message": {
    "text": "Jamal Hartnett has edited a pull request comment",
    "html": "Jamal Hartnett has <a href=\"https://fabrikam.visualstudio.com/DefaultCollection/_git/Fabrikam/pullrequest/1?discussionId=5\">edited</a> a pull request comment",
    "markdown": "Jamal Hartnett has [edited](https://fabrikam.visualstudio.com/DefaultCollection/_git/Fabrikam/pullrequest/1?discussionId=5) a pull request comment"
  },
  "detailedMessage": {
    "text": "Jamal Hartnett has edited a pull request comment\r\nThis is my comment.\r\n"
  }
}
```

```
        "html": "Jamal Hartnett has <a href=\"https://fabrikam.visualstudio.com/DefaultCollection/_git/Fabrikam/pullrequest/1?discussionId=5\">edited</a> a pull request comment<p>This is my comment.</p>",
        "markdown": "Jamal Hartnett has [edited](https://fabrikam.visualstudio.com/DefaultCollection/_git/Fabrikam/pullrequest/1?discussionId=5) a pull request comment\r\nThis is my comment.\r\n",
    },
    "resource": {
        "comment": {
            "id": 2,
            "parentCommentId": 1,
            "author": {
                "displayName": "Jamal Hartnett",
                "url":
"https://fabrikam.vssps.visualstudio.com/_apis/Identities/11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
                "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
                "uniqueName": "fabrikamfiber4@hotmail.com",
                "imageUrl":
"https://fabrikam.visualstudio.com/DefaultCollection/_api/_common/identityImage?id=11bb11bb-cc22-dd33-ee44-55ff55ff55ff"
            },
            "content": "This is my comment.",
            "publishedDate": "2024-06-17T11:22:33.456789Z",
            "lastUpdatedDate": "2024-06-17T16:58:33.123889Z",
            "lastContentUpdatedDate": "2024-06-17T16:58:33.123889Z",
            "commentType": "text",
            "_links": {
                "self": {
                    "href":
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/pullRequests/1/threads/5/comments/2"
                },
                "repository": {
                    "href": "http://joscol12/DefaultCollection/ebed510c-62eb-474b-965f-fd151ebb82e4/_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3"
                },
                "threads": {
                    "href":
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3/pullRequests/1/threads/5"
                }
            }
        },
        "pullRequest": {
            "repository": {
                "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
                "name": "Fabrikam",
                "url":
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
                "project": {
                    "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4",
                    "name": "Fabrikam",
                }
            }
        }
    }
}
```

```
        "url":  
        "https://fabrikam.visualstudio.com/DefaultCollection/_apis/projects/d3d3d3d3-eeee-  
        ffff-aaaa-b4b4b4b4b4b4",  
            "state": "wellFormed",  
            "visibility": "unchanged",  
            "lastUpdateTime": "2001-01-01T00:00:00"  
        },  
        "defaultBranch": "refs/heads/main",  
        "remoteUrl":  
        "https://fabrikam.visualstudio.com/DefaultCollection/_git/Fabrikam"  
    },  
    "pullRequestId": 1,  
    "status": "active",  
    "createdBy": {  
        "displayName": "Jamal Hartnett",  
        "url":  
        "https://fabrikam.vssps.visualstudio.com/_apis/Identities/11bb11bb-cc22-dd33-ee44-  
        55ff55ff55ff",  
            "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",  
            "uniqueName": "fabrikamfiber4@hotmail.com",  
            "imageUrl":  
            "https://fabrikam.visualstudio.com/DefaultCollection/_api/_common/identityImage?  
            id=11bb11bb-cc22-dd33-ee44-55ff55ff55ff"  
        },  
        "creationDate": "2024-06-17T11:22:33.456789Z",  
        "title": "my first pull request",  
        "description": " - test2\r\n",  
        "sourceRefName": "refs/heads/mytopic",  
        "targetRefName": "refs/heads/main",  
        "mergeStatus": "succeeded",  
        "mergeId": "e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5",  
        "lastMergeSourceCommit": {  
            "commitId": "4444eeee455ff5aaaabb66cccccccc7777cccc",  
            "url":  
            "https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2  
            c2-dddd-eeee-ffff-a3a3a3a3a3/commits/4444eeee455ff5aaaabb66cccccccc7777cccc"  
        },  
        "lastMergeTargetCommit": {  
            "commitId": "5555ffff566aa6bbbbbcc77dddddd888888ddd",  
            "url":  
            "https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2  
            c2-dddd-eeee-ffff-a3a3a3a3a3/commits/5555ffff566aa6bbbbbcc77dddddd888888ddd"  
        },  
        "lastMergeCommit": {  
            "commitId": "6666aaaa677bb7ccccdd88eeeeeee999999eeee",  
            "url":  
            "https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2  
            c2-dddd-eeee-ffff-a3a3a3a3a3/commits/6666aaaa677bb7ccccdd88eeeeeee999999eeee"  
        },  
        "reviewers": [  
            {  
                "reviewerUrl": null,  
                "vote": 0,  
                "displayName": "[Mobile]\\Mobile Team",  
                "url":
```

```

"https://fabrikam.vssps.visualstudio.com/_apis/Identities/22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
    "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
    "uniqueName": "azure-
devops:///Classification/TeamProject/22cc22cc-dd33-ee44-ff55-66aa66aa66aa\\Mobile
Team",
        "imageUrl":
"https://fabrikam.visualstudio.com/DefaultCollection/_api/_common/identityImage?
id=22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
            "isContainer": true
        },
    ],
    "commits": [
        {
            "commitId": "6666aaaa677bb7ccccdd88eeeeeee999999eeee",
            "url":
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2
c2-dddd-eeee-ffff-a3a3a3a3a3/commits/6666aaaa677bb7ccccdd88eeeeeee999999eeee"
        }
    ],
    "url":
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2
c2-dddd-eeee-ffff-a3a3a3a3a3/pullRequests/1",
        "_links": {
            "web": {
                "href":
"https://fabrikam.visualstudio.com/DefaultCollection/_git/Fabrikam/pullrequest/1#v
iew=discussion"
            },
            "statuses": {
                "href":
"https://fabrikam.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2
c2-dddd-eeee-ffff-a3a3a3a3a3/pullRequests/1/statuses"
            }
        }
    },
    "resourceVersion": "2.0",
    "resourceContainers": {
        "collection": {
            "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
        },
        "account": {
            "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
        },
        "project": {
            "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
        }
    },
    "createdDate": "2024-07-17T21:34:22.338Z"
}

```

## Repository created

Event: A repository is created.

- Publisher ID: tfs
- Event ID: git.repo.created
- Resource name: repository

## Settings

- projectId: Include only events for pull requests in a specific project.

## Sample payload

JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "git.repo.created",  
  "publisherId": "tfs",  
  "message": {  
    "text": "A new Git repository was created with name Fabrikam-Fiber-Git and  
ID c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3.",  
    "html": "A new Git repository was created with name <a  
href=\"https://fabrikam-fiber-  
inc.visualstudio.com/DefaultCollection/_git/Fabrikam-Fiber-Git/\">Fabrikam-Fiber-  
Git</a> and ID c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3.",  
    "markdown": "A new Git repository was created with name [Fabrikam-Fiber-  
Git](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/_git/Fabrikam-  
Fiber-Git/) and ID `c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3`."  
  },  
  "detailedMessage": {  
    "text": "A new Git repository was created with name Fabrikam-Fiber-Git and  
ID c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3.",  
    "html": "A new Git repository was created with name <a  
href=\"https://fabrikam-fiber-  
inc.visualstudio.com/DefaultCollection/_git/Fabrikam-Fiber-Git/\">Fabrikam-Fiber-  
Git</a> and ID c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3.",  
    "markdown": "A new Git repository was created with name [Fabrikam-Fiber-  
Git](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/_git/Fabrikam-  
Fiber-Git/) and ID `c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3`."  
  },  
  "resource": {  
    "repository": {  
      "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",  
      "name": "Fabrikam-Fiber-Git",  
      "url": "https://fabrikam-fiber-  
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-  
ffff-a3a3a3a3a3",  
      "project": {  
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee",  
        "name": "Fabrikam-Fiber-Git",  
      }  
    }  
  }  
}
```

```

        "url": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/projects/00aa00aa-bb11-cc22-dd33-
44ee44ee44ee",
        "state": "wellFormed",
        "revision": 11,
        "visibility": "private",
        "lastUpdateTime": "2025-06-12T20:22:53.7494088+00:00"
    },
    "defaultBranch": "refs/heads/main",
    "size": 728,
    "remoteUrl": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_git/Fabrikam-Fiber-Git",
    "sshUrl": "ssh://git@ssh.fabrikam-fiber-
inc.visualstudio.com/v3/DefaultCollection/Fabrikam-Fiber-Git",
    "isDisabled": false
},
"initiatedBy": {
    "displayName": "Ivan Yurev",
    "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
    "uniqueName": "user@fabrikamfiber.com"
},
"utcTimestamp": "2022-12-12T12:34:56.5498459Z"
},
"resourceVersion": "1.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-12T20:22:53.818Z"
}

```

## Repository deleted

Event: A repository is deleted.

- Publisher ID: `tfs`
- Event ID: `git.repo.deleted`
- Resource name: `repository`

## Settings

- **repository**: Include only events for pull requests in repositories with a specific name pattern.
  - Data type: `guid`

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "git.repo.deleted",
  "publisherId": "tfs",
  "message": {
    "text": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was deleted.",
    "html": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was deleted.",
    "markdown": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was deleted."
  },
  "detailedMessage": {
    "text": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was deleted.\r\nProject name: Contoso\r\n\r\nRepository name: Fabrikam-Fiber-Git\r\n\r\nRepository can be restored: true\r\n",
    "html": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was deleted.

Project name: Contoso



Repository name: Fabrikam-Fiber-Git



Repository can be restored: true

",
    "markdown": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was deleted.\r\nProject name: Contoso\r\n\r\nRepository name: Fabrikam-Fiber-Git\r\n\r\nRepository can be restored: true\r\n"
  },
  "resource": {
    "project": {
      "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee",
      "name": "Contoso",
      "url": "https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/_apis/projects/00aa00aa-bb11-cc22-dd33-44ee44ee44ee",
      "state": "wellFormed",
      "revision": 11,
      "visibility": "private",
      "lastUpdateTime": "2025-06-12T20:33:32.4370396+00:00"
    },
    "repositoryId": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
    "repositoryName": "Fabrikam-Fiber-Git",
    "isHardDelete": false,
    "initiatedBy": {
      "displayName": "Himani Maharjan",
      "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4",
      "uniqueName": "himani@fabrikamfiber.com"
    },
    "utcTimestamp": "2022-12-12T12:34:56.5498459Z"
  }
}
```

```
},
  "resourceVersion": "1.0-preview.1",
  "resourceContainers": {
    "collection": {
      "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
      "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
      "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
  },
  "createdDate": "2025-06-12T20:33:32.512Z"
}
```

## Repository forked

Event: A repository is forked.

- Publisher ID: `tfs`
- Event ID: `git.repo.forked`
- Resource name: `repository`

## Settings

- `repository`: Include only events for pull requests in repositories with a specific name pattern.
  - Data type: `guid`

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "git.repo.forked",
  "publisherId": "tfs",
  "message": {
    "text": "Git repository Fabrikam-Fiber-Git was forked by Ivan Yurev.",
    "html": "Git repository <a href=\"https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_git/Fabrikam-Fiber-Git/\\">Fabrikam-Fiber-
Git</a> was forked by Ivan Yurev.",
    "markdown": "Git repository [Fabrikam-Fiber-Git](https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_git/Fabrikam-Fiber-Git/) was forked by
Ivan Yurev."
  },
}
```

```
"detailedMessage": {
    "text": "Git repository Fabrikam-Fiber-Git was forked by Ivan Yurev.\r\nSource Repository\r\n\r\nProject name: Fabrikam-Fiber-Git\r\nRepository Id: c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3\r\nRepository Name: Fabrikam-Fiber-Git\r\nDefault branch: refs/heads/main\r\nSize: 729\r\nRepository link(https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3)\r\n\r\nFork\r\nProject name: Forked-fiber-inc\r\nRepository Id: d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4\r\nRepository Name: Forked-fiber-inc\r\nDefault branch: refs/heads/main\r\nRepository link(https://forked-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4)",
    "html": "Git repository <a href=\"https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_git/Fabrikam-Fiber-Git\">Fabrikam-Fiber-Git</a> was forked by Ivan Yurev.\r\n

### Source Repository

\r\n

Project name: Fabrikam-Fiber-Git



Repository Id: c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3



Repository Name: Fabrikam-Fiber-Git



Default branch: refs/heads/main



Size: 729



https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3\" data-bbox=\"111 491 894 568\" style="color: inherit; text-decoration: none;">Repository link



Fork



Project name: Another-Great-Project



Repository Id: d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4



Repository Name: Forked-fiber-inc



Default branch: refs/heads/main



https://forked-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4\" data-bbox=\"111 491 894 568\" style="color: inherit; text-decoration: none;">Repository link


",
    "markdown": "Git repository [Fabrikam-Fiber-Git](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3) was forked by Ivan Yurev.\r\n### Source Repository\r\nProject name: Fabrikam-Fiber-Git\r\nRepository Id: c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3\r\nRepository Name: Fabrikam-Fiber-Git\r\nDefault branch: refs/heads/main\r\nSize: 729\r\n[Repository link](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3)\r\n\r\n### Fork\r\nProject name: Another-Great-Project\r\nRepository Id: d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4\r\nRepository Name: Forked-Fiber-Git\r\nDefault branch: refs/heads/main\r\n[Repository link](https://forked-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4)"
},
"resource": {
    "targetRepository": {
        "id": "d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4",
        "name": "Forked-Fiber-Git",
        "url": "https://forked-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4",
        "project": {
            "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
            "name": "Another-Great-Project",
            "url": "https://another-great-project.visualstudio.com/DefaultCollection/\_apis/projects/11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
            "state": "wellFormed",
            "revision": 11,
            "visibility": "private"
        }
    }
}
```

```
        "lastUpdateTime": "2025-06-12T20:39:05.1186682+00:00"
    },
    "defaultBranch": "refs/heads/main",
    "size": 728,
    "remoteUrl": "https://another-great-
inc.visualstudio.com/DefaultCollection/_git/Forked-Fiber-Git",
    "sshUrl": "ssh://git@ssh.another-great-
inc.visualstudio.com/v3/DefaultCollection/Forked-Fiber-Git",
    "isDisabled": false
},
"sourceRepository": {
    "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
    "name": "Fabrikam-Fiber-Git",
    "url": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-
ffff-a3a3a3a3a3",
    "project": {
        "id": "22cc22cc-dd33-ee44-ff55-66aa66aa66aa",
        "name": "Fabrikam-Fiber-Git",
        "url": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/projects/22cc22cc-dd33-ee44-ff55-
66aa66aa66aa",
        "state": "wellFormed",
        "revision": 11,
        "visibility": "private",
        "lastUpdateTime": "2025-06-12T20:39:05.1186682+00:00"
    },
    "defaultBranch": "refs/heads/main",
    "size": 728,
    "remoteUrl": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_git/Fabrikam-Fiber-Git",
    "sshUrl": "ssh://git@ssh.fabrikam-fiber-
inc.visualstudio.com/v3/DefaultCollection/Fabrikam-Fiber-Git",
    "isDisabled": false
},
"initiatedBy": {
    "displayName": "Ivan Yurev",
    "id": "66aa66aa-bb77-cc88-dd99-00ee00ee00ee",
    "uniqueName": "user@fabrikamfiber.com"
},
"utcTimestamp": "2023-01-25T12:34:56.5498459Z"
},
"resourceVersion": "1.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f5"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
}
},
```

```
        "createdDate": "2025-06-12T20:39:05.382Z"  
    }  

```

## Repository renamed

Event: A repository is renamed.

- Publisher ID: `tfs`
- Event ID: `git.repo.renamed`
- Resource name: `repository`

## Settings

- `repository`: Include only events for pull requests in repositories with a specific name pattern.
  - Data type: `guid`

## Sample payload

JSON

```
{  
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1e1",  
    "eventType": "git.repo.renamed",  
    "publisherId": "tfs",  
    "message": {  
        "text": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was renamed  
to Fabrikam-Fiber-Git.",  
        "html": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was renamed  
to <a href=\"https://fabrikam-fiber-  
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-  
ffff-a3a3a3a3a3\">Fabrikam-Fiber-Git</a>.",  
        "markdown": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was  
renamed to [Fabrikam-Fiber-Git](https://fabrikam-fiber-  
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-  
ffff-a3a3a3a3a3)."  
    },  
    "detailedMessage": {  
        "text": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was renamed  
to Fabrikam-Fiber-Git.\r\nProject name: Contoso\r\n\r\nRepository name before  
renaming: Diber-Git\r\n\r\nDefault branch: refs/heads/main\r\n\r\nRepository  
link(https://fabrikam-fiber-  
inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-  
ffff-a3a3a3a3a3a3)\r\n",  
        "html": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 was renamed  
to <a href=\"https://fabrikam-fiber-  
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-  
ffff-a3a3a3a3a3a3\">Fabrikam-Fiber-Git</a>."  
    }  
}
```

```
ffff-a3a3a3a3a3a3\>Fabrikam-Fiber-Git</a>.<p>Project name: Contoso</p>
<p>Repository name before renaming: Diber-Git</p><p>Default branch:
refs/heads/main</p><p><a href=\"https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eaaa-
ffff-a3a3a3a3a3a3\>Repository link</a></p>",
    "markdown": "Git repository c2c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3 was
renamed to [Fabrikam-Fiber-Git](https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eaaa-
ffff-a3a3a3a3a3a3).\r\nProject name: Contoso\r\n\r\nRepository name before
renaming: Diber-Git\r\n\r\nDefault branch: refs/heads/main\r\n\r\n[Repository
link](https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eaaa-
ffff-a3a3a3a3a3a3)\r\n"
},
"resource": {
    "oldName": "Diber-Git",
    "newName": "Fabrikam-Fiber-Git",
    "repository": {
        "id": "c2c2c2c2-dddd-eaaa-ffff-a3a3a3a3a3",
        "name": "Fabrikam-Fiber-Git",
        "url": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eaaa-
ffff-a3a3a3a3a3a3",
        "project": {
            "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
            "name": "Contoso",
            "url": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_apis/projects/11bb11bb-cc22-dd33-ee44-
55ff55ff55ff",
            "state": "wellFormed",
            "revision": 11,
            "visibility": "private",
            "lastUpdateTime": "2025-06-12T20:48:38.8174565+00:00"
        },
        "defaultBranch": "refs/heads/main",
        "size": 728,
        "remoteUrl": "https://fabrikam-fiber-
inc.visualstudio.com/DefaultCollection/_git/Fabrikam-Fiber-Git",
        "sshUrl": "ssh://git@ssh.fabrikam-fiber-
inc.visualstudio.com/v3/DefaultCollection/Fabrikam-Fiber-Git",
        "isDisabled": false
    },
    "initiatedBy": {
        "displayName": "Himani Maharjan",
        "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
        "uniqueName": "himani@fabrikamfiber.com"
    },
    "utcTimestamp": "2022-12-12T12:34:56.5498459Z"
},
"resourceVersion": "1.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eaaa-f2f2f2f2f2"
    },
    "account": {
```

```
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f",
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-12T20:48:38.859Z"
}
```

## Repository status changed

Event: A repository status changes.

- Publisher ID: `tfs`
- Event ID: `git.repo.statuschanged`
- Resource name: `repository`

## Settings

- `repository`: Include only events for repositories with a specific name pattern.
  - Data type: `guid`

## Sample payload

JSON

```
{
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "git.repo.statuschanged",
    "publisherId": "tfs",
    "message": {
        "text": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 with name Fabrikam-Fiber-Git status was changed: enabled.",
        "html": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 with name <a href=\"https://fabrikam-fiber- inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3\">Fabrikam-Fiber-Git</a> status was changed: enabled.",
        "markdown": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 with name [Fabrikam-Fiber-Git](https://fabrikam-fiber- inc.visualstudio.com/DefaultCollection/_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3) status was changed: enabled."
    },
    "detailedMessage": {
        "text": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 with name Fabrikam-Fiber-Git status was changed: enabled.\r\nProject name: Contoso\r\n\r\nRepository name: Fabrikam-Fiber-Git\r\n\r\nDefault branch: refs/heads/main\r\n\r\nIsFork: false\r\n\r\nIsDisabled: "
    }
}
```

```
false\r\n\r\n\r\nIsInMaintenance: false\r\n\r\n\r\nClone link(https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3)\r\n",
    "html": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 with name <a href=\"https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3\">Fabrikam-Fiber-Git</a> status was changed: enabled.<p>Project name: Contoso</p><p>Repository name: Fabrikam-Fiber-Git</p><p>Default branch: refs/heads/main</p><p>IsFork: false</p><p>IsDisabled: false</p><p>IsInMaintenance: false</p><p><a href=\"https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3\">Clone link</a></p>",
    "markdown": "Git repository c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3 with name [Fabrikam-Fiber-Git](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3) status was changed: enabled.\r\nProject name: Contoso\r\n\r\nRepository name: Fabrikam-Fiber-Git\r\n\r\nDefault branch: refs/heads/main\r\n\r\nIsFork: false\r\n\r\nIsDisabled: false\r\n\r\nIsInMaintenance: false\r\n\r\n[Clone link](https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3)\r\n",
},
"resource": {
    "disabled": false,
    "repository": {
        "id": "c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
        "name": "Fabrikam-Fiber-Git",
        "url": "https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/git/repositories/c2c2c2c2-dddd-eeee-ffff-a3a3a3a3a3",
        "project": {
            "id": "11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
            "name": "Fabrikam-Fiber-Git",
            "url": "https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_apis/projects/11bb11bb-cc22-dd33-ee44-55ff55ff55ff",
            "state": "wellFormed",
            "revision": 11,
            "visibility": "private",
            "lastUpdateTime": "2025-06-12T20:55:07.6222336+00:00"
        },
        "defaultBranch": "refs/heads/main",
        "size": 728,
        "remoteUrl": "https://fabrikam-fiber-inc.visualstudio.com/DefaultCollection/\_git/Fabrikam-Fiber-Git",
        "sshUrl": "ssh://git@ssh.fabrikam-fiber-inc.visualstudio.com/v3/DefaultCollection/Fabrikam-Fiber-Git",
        "isDisabled": false
    },
    "initiatedBy": {
        "displayName": "Himani Maharjan",
        "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
        "uniqueName": "himani@fabrikamfiber.com"
    },
    "utcTimestamp": "2022-12-12T12:34:56.5498459Z"
```

```
},
"resourceVersion": "1.0-preview.1",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-12T20:55:07.812Z"
}
```

## Service connection

The following service connection events are available for use in service hooks.

### Service connection created

Event: A service connection is created.

- Publisher ID: `tfs`
- Event ID: `ms.vss-endpoint.endpoint-created`
- Resource name: `serviceendpoint`

### Settings

- `project`: Include only events for service connections created in a specific project.

### Sample payload

JSON

```
{
    "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
    "eventType": "ms.vss-endpoint.endpoint-created",
    "publisherId": "tfs",
    "message": {
        "text": "Generic service connection created: Sample service connection",
        "html": "Generic service connection created: Sample service connection",
        "markdown": "Generic service connection created: Sample service connection"
    },
}
```

```

  "detailedMessage": {
    "text": "Generic service connection created: Sample service connection",
    "html": "Generic service connection created: Sample service connection",
    "markdown": "Generic service connection created: Sample service
connection"
  },
  "resource": {
    "id": "a6a6a6a6-bbbb-cccc-dddd-e7e7e7e7e7",
    "name": "Sample service connection",
    "type": "Generic",
    "authorization": null,
    "projectIds": []
  },
  "resourceVersion": "1.0-preview.1",
  "resourceContainers": {
    "collection": {
      "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
      "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
      "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
  },
  "createdDate": "2025-06-12T20:59:01.867Z"
}

```

## Service connection updated

Event: A service connection is updated.

- Publisher ID: `tfs`
- Event ID: `ms.vss-endpoint.endpoint-updated`
- Resource name: `serviceendpoint`

## Settings

- `project`: Include only events for service connections updated in a specific project.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.vss-endpoint.endpoint-updated",
  "publisherId": "tfs",
}
```

```
  "message": {
    "text": "Generic service connection updated: Sample service connection",
    "html": "Generic service connection updated: Sample service connection",
    "markdown": "Generic service connection updated: Sample service
connection"
  },
  "detailedMessage": {
    "text": "Generic service connection updated: Sample service connection",
    "html": "Generic service connection updated: Sample service connection",
    "markdown": "Generic service connection updated: Sample service
connection"
  },
  "resource": {
    "id": "f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6",
    "name": "Sample service connection",
    "type": "Generic",
    "authorization": null,
    "projectIds": []
  },
  "resourceVersion": "1.0-preview.1",
  "resourceContainers": {
    "collection": {
      "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
      "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
      "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
  },
  "createdDate": "2025-06-13T12:58:59.908Z"
}
```

## Work item

The following work item events are available for use in service hooks.

### Work item created

Event: A work item is created.

- Publisher ID: `tfs`
- Event ID: `workitem.created`
- Resource name: `workitem`

## Settings

- `areaPath`: Include only events for work items under a specific area path.
- `workItemType`: Include only events for work items of a specific type.
- `linksChanged`: Include only events for work items with one or more links added or removed.
- `tag`: Include only events for work items that contain a specific tag.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "workitem.created",
  "publisherId": "tfs",
  "scope": "all",
  "message": {
    "text": "Bug #5 (Some great new idea!) created by Jamal Hartnett.\r\n(\r\n

```

```
"System.State": "New",
"System.Reason": "New defect reported",
"System.CreatedDate": "2014-07-15T17:42:44.663Z",
"System.CreatedBy": "Jamal Hartnett",
"System.ChangedDate": "2014-07-15T17:42:44.663Z",
"System.ChangedBy": "Jamal Hartnett",
"System.Title": "Some great new idea!",
"Microsoft.Azure.DevOps.Services.Common.Severity": "3 - Medium",
"WEF_EB329F44FE5F4A94ACB1DA153FDF38BA_Kanban.Column": "New"
},
"_links": {
  "self": {
    "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5"
  },
  "workItemUpdates": {
    "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5/updates"
  },
  "workItemRevisions": {
    "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5/revisions"
  },
  "workItemType": {
    "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/d3d3d3d3-eeee-ffff-aaaa-
b4b4b4b4b4b4/workItemTypes/Bug"
  },
  "fields": {
    "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/fields"
  }
},
"url": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5"
},
"resourceVersion": "1.0",
"resourceContainers": {
  "collection": {
    "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
  },
  "account": {
    "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
  },
  "project": {
    "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
  }
},
"createdDate": "2024-09-19T13:03:29.7688022Z"
}
```

## Work item deleted

Event: A work item is deleted.

- Publisher ID: tfs
- Event ID: workitem.deleted
- Resource name: resource

## Settings

- areaPath: Include only events for work items under a specific area path.
- workItemType: Include only events for work items of a specific type.
- tag: Include only events for work items that contain a specific tag.

## Sample payload

JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "workitem.deleted",  
  "publisherId": "tfs",  
  "scope": "all",  
  "message": {  
    "text": "Bug #5 (Some great new idea!) deleted by Jamal Hartnett.",  
    "html": "Bug #5 (Some great new idea!) deleted by Jamal Hartnett.",  
    "markdown": "[Bug #5](Some great new idea!) deleted by Jamal Hartnett."  
  },  
  "detailedMessage": {  
    "text": "Bug #5 (Some great new idea!) deleted by Jamal Hartnett.\r\n\r\n- Area: FabrikamCloud\r\n- Iteration: FabrikamCloud\\Release 1\\Sprint 1\r\n- State: New\r\n",  
    "html": "Bug #5 (Some great new idea!) deleted by Jamal Hartnett.<ul>\r\n- Area: FabrikamCloud</br>\r\n- Iteration: FabrikamCloud\\Release 1\\Sprint 1</br>\r\n- State: New</br></ul>",  
    "markdown": "[Bug #5](Some great new idea!) deleted by Jamal Hartnett.\r\n\r\n* Area: FabrikamCloud\r\n* Iteration: FabrikamCloud\\Release 1\\Sprint 1\r\n* State: New\r\n"  
  },  
  "resource": {  
    "id": 5,  
    "rev": 1,  
    "fields": {  
      "System.AreaPath": "FabrikamCloud",  
      "System.TeamProject": "FabrikamCloud",  
      "System.IterationPath": "FabrikamCloud\\Release 1\\Sprint 1",  
      "System.WorkItemType": "Bug",  
      "System.State": "New",  
      "System.Reason": "New defect reported",  
      "System.CreatedDate": "2014-07-15T17:42:44.663Z",  
      "System.CreatedBy": "Jamal Hartnett",  
    }  
  }  
}
```

```

    "System.ChangedDate": "2014-07-15T17:42:44.663Z",
    "System.ChangedBy": "Jamal Hartnett",
    "System.Title": "Some great new idea!",
    "Microsoft.Azure.DevOps.Services.Common.Severity": "3 - Medium",
    "WEF_EB329F44FE5F4A94ACB1DA153FDF38BA_Kanban.Column": "New"
},
"_links": {
    "self": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/recyclebin/5"
    },
    "workItemType": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/d3d3d3d3-eeee-ffff-aaaa-
b4b4b4b4b4/workItemTypes/Bug"
    },
    "fields": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/fields"
    }
},
"url": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/recyclebin/5"
},
"resourceVersion": "1.0",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2024-09-19T13:03:30.0657064Z"
}

```

## Work item restored

Event: A work item is restored.

- Publisher ID: `tfs`
- Event ID: `workitem.restored`
- Resource name: `resource`

## Settings

- `areaPath`: Include only events for work items under a specific area path.

- `workItemType`: Include only events for work items of a specific type.
- `tag`: Include only events for work items that contain a specific tag.

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "workitem.restored",
  "publisherId": "tfs",
  "scope": "all",
  "message": {
    "text": "Bug #5 (Some great new idea!) restored by Jamal Hartnett.\r\n(\r\nhttps://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5)",

    "html": "<a href=\"https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5&id=5\">Bug #5</a> (Some great new idea!) restored by Jamal Hartnett.",

    "markdown": "[Bug #5](https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5) (Some great new idea!) restored by Jamal Hartnett."
  },
  "detailedMessage": {
    "text": "Bug #5 (Some great new idea!) restored by Jamal Hartnett.\r\n(\r\nhttps://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5)\r\n\r\n- Area: FabrikamCloud\r\n- Iteration: FabrikamCloud\\Release 1\\Sprint 1\r\n- State: New\r\n- Severity: 3 - Medium\r\n",

    "html": "<a href=\"https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&id=5\">Bug #5</a> (Some great new idea!) restored by Jamal Hartnett.<ul>\r\n- Area: FabrikamCloud<br>\r\n- Iteration: FabrikamCloud\\Release 1\\Sprint 1<br>\r\n- State: New<br>Severity: 3 - Medium<br></ul>",

    "markdown": "[Bug #5](https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5) (Some great new idea!) restored by Jamal Hartnett.\r\n\r\n* Area: FabrikamCloud\r\n* Iteration: FabrikamCloud\\Release 1\\Sprint 1\r\n* State: New\r\n* Severity: 3 - Medium"
  },
  "resource": {
    "id": 5,
    "rev": 1,
    "fields": {
      "System.AreaPath": "FabrikamCloud",
      "System.TeamProject": "FabrikamCloud",
      "System.IterationPath": "FabrikamCloud\\Release 1\\Sprint 1",
      "System.WorkItemType": "Bug",
      "System.State": "New",
      "System.Reason": "New defect reported",
      "System.CreatedDate": "2014-07-15T17:42:44.663Z",
      "System.CreatedBy": "Jamal Hartnett",
      "System.ChangedDate": "2014-07-15T17:42:44.663Z",
      "System.ChangedBy": "Jamal Hartnett"
    }
  }
}
```

```
"System.ChangedBy": "Jamal Hartnett",
"System.Title": "Some great new idea!",
"Microsoft.Azure.DevOps.Services.Common.Severity": "3 - Medium",
"WEF_EB329F44FE5F4A94ACB1DA153FDF38BA_Kanban.Column": "New"
},
"_links": {
  "self": {
    "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5"
  },
  "workItemUpdates": {
    "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5/updates"
  },
  "workItemRevisions": {
    "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5/revisions"
  },
  "workItemType": {
    "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6/workItemTypes/Bug"
  },
  "fields": {
    "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/fields"
  },
  "html": {
    "href": "https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4&id=5"
  },
  "workItemHistory": {
    "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5/history"
  }
},
"url": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5"
},
"resourceVersion": "1.0",
"resourceContainers": {
  "collection": {
    "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
  },
  "account": {
    "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
  },
  "project": {
    "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
  }
},
"createdDate": "2024-09-19T13:30:30.1456784Z"
}
```

# Work item updated

## Event: A work item changes.

- Publisher ID: tfs
  - Event ID: workitem.updated
  - Resource name: workitem

# Settings

- `areaPath`: Include only events for work items under a specific area path.
  - `changedFields`: Include only events for work items with a change in a specific field.
  - `workItemType`: Include only events for work items of a specific type.
  - `linksChanged`: Include only events for work items with one or more links added or removed.
  - `tag`: Include only events for work items that contain a specific tag.

## Sample payload

## JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "workitem.updated",  
  "publisherId": "tfs",  
  "scope": "all",  
  "message": {  
    "text": "Bug #5 (Some great new idea!) updated by Jamal  
Hartnett.\r\n(https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5)",  
    "html": "<a href=\"https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5&id=5\">Bug #5</a> (Some great new  
idea!) updated by Jamal Hartnett.",  
    "markdown": "[Bug #5](https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5) (Some great new idea!) updated by  
Jamal Hartnett."  
,  
    "detailedMessage": {  
      "text": "Bug #5 (Some great new idea!) updated by Jamal  
Hartnett.\r\n(https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5)\r\n- Area: FabrikamCloud\r\n- Iteration:  
FabrikamCloud\\Release 1\\Sprint 1\r\n- State: New\r\n- Severity: 3 -  
Medium\r\n",  
      "html": "<a href=\"https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&id=5\">Bug #5</a> (Some great new  
idea!) updated by Jamal Hartnett.<ul>\r\n- Area: FabrikamCloud<br>\r\n- Iteration:  
FabrikamCloud\\Release 1\\Sprint 1<br>\r\n- State: New<br>Severity: 3 -  
Medium</ul>",  
      "markdown": "[Bug #5](https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5c5&id=5) (Some great new idea!) updated by  
Jamal Hartnett.

- Area: FabrikamCloud
- Iteration: FabrikamCloud\\Release 1\\Sprint 1
- State: New
- Severity: 3 - Medium

"  
    }  
  }  
}
```

```
- Medium</br></ul>",
  "markdown": "[Bug #5](https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5) (Some great new idea!) updated by Jamal Hartnett.\r\n\r\n* Area: FabrikamCloud\r\n* Iteration: FabrikamCloud\\Release 1\\Sprint 1\r\n* State: New\r\n* Severity: 3 - Medium\r\n",
},
"resource": {
  "id": 5,
  "rev": 1,
  "fields": {
    "System.AreaPath": "FabrikamCloud",
    "System.TeamProject": "FabrikamCloud",
    "System.IterationPath": "FabrikamCloud\\Release 1\\Sprint 1",
    "System.WorkItemType": "Bug",
    "System.State": "New",
    "System.Reason": "New defect reported",
    "System.CreatedDate": "2014-07-15T17:42:44.663Z",
    "System.CreatedBy": "Jamal Hartnett",
    "System.ChangedDate": "2014-07-15T17:42:44.663Z",
    "System.ChangedBy": "Jamal Hartnett",
    "System.Title": "Some great new idea!",
    "Microsoft.Azure.DevOps.Services.Common.Severity": "3 - Medium",
    "WEF_EB329F44FE5F4A94ACB1DA153FDF38BA_Kanban.Column": "New"
  },
  "_links": {
    "self": {
      "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5"
    },
    "workItemUpdates": {
      "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5/updates"
    },
    "workItemRevisions": {
      "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5/revisions"
    },
    "workItemType": {
      "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/f5f5f5f5-aaaa-bbbb-cccc-d6d6d6d6d6d6/workItemTypes/Bug"
    },
    "fields": {
      "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/fields"
    },
    "html": {
      "href": "https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?pcguid=d3d3d3d3-eeee-ffff-aaaa-b4b4b4b4b4&id=5"
    },
    "workItemHistory": {
      "href": "https://dev.azure.com/fabrikam-fiber-inc/DefaultCollection/_apis/wit/workItems/5/history"
    }
  }
},
```

```
        "url": "https://dev.azure.com/fabrikam-fiber-  
inc/DefaultCollection/_apis/wit/workItems/5"  
    },  
    "resourceVersion": "1.0",  
    "resourceContainers": {  
        "collection": {  
            "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"  
        },  
        "account": {  
            "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"  
        },  
        "project": {  
            "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"  
        }  
    },  
    "createdDate": "2024-09-19T13:03:30.1456784Z"  
}
```

## Filter on multiple work item fields

If you want to trigger on a change in more than one work item field, you have a few possibilities:

- To trigger on a change in any field, remove the `changedFields` filter.
- To trigger on multiple fields, but not all fields, use one of the following options:
  - Use the Azure DevOps Services REST API to create a custom payload for each field of interest. For more information, see [Subscriptions - Create](#).
  - To create a service hook subscription for each field of interest, take the following steps for each field:
    1. Go to your project, select **Project settings**, and then select **Service hooks**.
    2. Select **Create subscription**.
    3. Select **Web Hooks**, and then select **Next**.
    4. Under **Trigger on this type of event**, select **Work item updated**.
    5. Under **Field**, select a field you want to trigger on. For example, if you want to track changes in the work item state, select **State**.
    6. Configure any other filters you want to use by specifying an area path, a work item type, or a tag, and then select **Next**.
    7. In the **Action** dialog, configure the settings, and then select **Test** or **Finish**.

# Work item commented on

Event: A work item is commented on.

- Publisher ID: tfs
- Event ID: workitemcommented
- Resource name: workitem

## Settings

- areaPath: Include only events for work items under a specific area path.
- commentPattern: Include only events for work items with a comment that contains a specific string.
- workItemType: Include only events for work items of a specific type.
- tag: Include only events for work items that contain a specific tag.

## Sample payload

JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "workitemcommented",  
  "publisherId": "tfs",  
  "scope": "all",  
  "message": {  
    "text": "Bug #5 (Some great new idea!) commented on by Jamal  
Hartnett.\r\n(\nhttps://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5)",  
    "html": "<a href=\"https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5&id=5\">Bug #5</a> (Some great new  
idea!) commented on by Jamal Hartnett.",  
    "markdown": "[Bug #5](https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5) (Some great new idea!) commented on  
by Jamal Hartnett."  
  },  
  "detailedMessage": {  
    "text": "Bug #5 (Some great new idea!) commented on by Jamal  
Hartnett.\r\n(\nhttps://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5)\r\nThis is a great new idea",  
    "html": "<a href=\"https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5&id=5\">Bug #5</a> (Some great new  
idea!) commented on by Jamal Hartnett.<br/>This is a great new idea",  
    "markdown": "[Bug #5](https://dev.azure.com/fabrikam-fiber-inc/web/wi.aspx?  
pcguid=e4e4e4e4-ffff-aaaa-bbbb-c5c5c5c5c5) (Some great new idea!) commented on  
by Jamal Hartnett.\r\nThis is a great new idea"  
  },  
  "resource": {
```

```
"id": 5,
"rev": 4,
"fields": {
    "System.AreaPath": "FabrikamCloud",
    "System.TeamProject": "FabrikamCloud",
    "System.IterationPath": "FabrikamCloud\\Release 1\\Sprint 1",
    "System.WorkItemType": "Bug",
    "System.State": "New",
    "System.Reason": "New defect reported",
    "System.CreatedDate": "2014-07-15T17:42:44.663Z",
    "System.CreatedBy": "Jamal Hartnett",
    "System.ChangedDate": "2014-07-15T17:42:44.663Z",
    "System.ChangedBy": "Jamal Hartnett",
    "System.Title": "Some great new idea!",
    "Microsoft.Azure DevOps Services.Common.Severity": "3 - Medium",
    "WEF_EB329F44FE5F4A94ACB1DA153FDF38BA_Kanban.Column": "New",
    "System.History": "This is a great new idea"
},
"_links": {
    "self": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5"
    },
    "workItemUpdates": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5/updates"
    },
    "workItemRevisions": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5/revisions"
    },
    "workItemType": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/f5f5f5f5-aaaa-bbbb-cccc-
d6d6d6d6d6d6/workItemTypes/Bug"
    },
    "fields": {
        "href": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/fields"
    }
},
"url": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection/_apis/wit/workItems/5"
},
"resourceVersion": "1.0",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
}
```

```
 },
  "createdDate": "2024-09-19T13:03:28.9695265Z"
}
```

## Advanced security

The following advanced security events are available for use in service hooks that you create by using the UI.

### Advanced security alert created

Event: An advanced security alert is created.

- Publisher ID: `advsec`
- Event ID: `ms.vss-alerts.alert-created-event`
- Resource name: `resource`

### Settings

- `repository`: Include only events for alerts that are created in a specific repository.
- `branch`: Include only events for alerts that are created in a specific branch.
- `alertType`: Include only events for alerts of a specific type.
  - Valid values:
    - `Unknown`
    - `Dependency`
    - `Secret`
    - `Code`
- `severity`: Include only events for alerts with a specific severity.
  - Valid values:
    - `Low`
    - `Medium`
    - `High`
    - `Critical`
    - `Note`
    - `Warning`
    - `Error`
    - `Undefined`

### Sample payload

## JSON

```
{  
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",  
  "eventType": "ms.vss-alerts.alert-created-event",  
  "publisherId": "advsec",  
  "message": {  
    "text": "New alert is created",  
    "html": "New alert is created",  
    "markdown": "New alert is created"  
  },  
  "detailedMessage": {  
    "text": "New alert is created\r\n\r\n- Alert status: Created\r\n",  
    "html": "New alert is created\r\n\r\n- Alert status: Created\r\n",  
    "markdown": "New alert is created\r\n\r\n- Alert status: Created\r\n"  
  },  
  "resource": {  
    "alertId": 1,  
    "severity": "critical",  
    "title": "Alert title",  
    "tools": [  
      {  
        "name": "codeql",  
        "rules": [  
          {  
            "opaqueId": null,  
            "friendlyName": "codeql rule",  
            "description": null,  
            "resources": null,  
            "helpMessage": "update the version",  
            "tags": null,  
            "additionalProperties": null  
          }  
        ]  
      }  
    ]  
  },  
  "dismissal": {  
    "dismissalId": 1,  
    "message": "Fixed",  
    "stateChangedBy": "66aa66aa-bb77-cc88-dd99-00ee00ee00ee",  
    "stateChangedByIdentity": null,  
    "requestedOn": null,  
    "dismissalType": "fixed"  
  },  
  "repositoryUrl": "https://dev.azure.com/test/test/_git/test",  
  "gitRef": "testRef",  
  "alertType": "code",  
  "firstSeenDate": null,  
  "lastSeenDate": null,  
  "fixedDate": null,  
  "introducedDate": null,  
  "state": "fixed",  
  "physicalLocations": null,  
  "logicalLocations": null
```

```
 },
 "resourceVersion": "1.0",
 "resourceContainers": {
   "collection": {
     "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2"
   },
   "account": {
     "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffff5f5f5f"
   },
   "project": {
     "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
   }
 },
 "createdDate": "2025-06-09T18:22:39.862Z"
}
```

## Advanced security alert state changed

Event: The state of an advanced security alert changes.

- Publisher ID: `advsec`
- Event ID: `ms.vss-alerts.alert-state-changed-event`
- Resource name: `resource`

## Settings

- `repository`: Include only events for alerts that are associated with a specific repository.
- `branch`: Include only events for alerts that are associated with a specific branch.
- `alertType`: Include only events for alerts of a specific type.
  - Valid values:
    - `Unknown`
    - `Dependency`
    - `Secret`
    - `Code`
- `severity`: Include only events for alerts with a specific severity.
  - Valid values:
    - `Low`
    - `Medium`
    - `High`
    - `Critical`
    - `Note`
    - `Warning`
    - `Error`

- `Undefined`
- `state`: Include only events for alerts with a specific new state.
  - Valid values:
    - `Unknown`
    - `Active`
    - `Dismissed`
    - `Fixed`
    - `AutoDismissed`

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.vss-alerts.alert-state-changed-event",
  "publisherId": "advsec",
  "message": {
    "text": "Alert's state is changed",
    "html": "Alert's state is changed",
    "markdown": "Alert's state is changed"
  },
  "detailedMessage": {
    "text": "Alert's state is changed\r\n\r\n- Alert status: State Changed\r\n",
    "html": "Alert's state is changed\r\n\r\n- Alert status: State Changed\r\n",
    "markdown": "Alert's state is changed\r\n\r\n- Alert status: State Changed\r\n"
  },
  "resource": {
    "alertId": 1,
    "severity": "critical",
    "title": "Alert title",
    "tools": [
      {
        "name": "codeql",
        "rules": [
          {
            "opaqueId": null,
            "friendlyName": "codeql rule",
            "description": null,
            "resources": null,
            "helpMessage": "update the version",
            "tags": null,
            "additionalProperties": null
          }
        ]
      }
    ],
  }
},
```

```

    "dismissal": {
        "dismissalId": 1,
        "message": "Fixed",
        "stateChangedBy": "66aa66aa-bb77-cc88-dd99-00ee00ee00ee",
        "stateChangedByIdentity": null,
        "requestedOn": null,
        "dismissalType": "fixed"
    },
    "repositoryUrl": "https://dev.azure.com/test/test/_git/test",
    "gitRef": "testRef",
    "alertType": "code",
    "firstSeenDate": null,
    "lastSeenDate": null,
    "fixedDate": null,
    "introducedDate": null,
    "state": "fixed",
    "physicalLocations": null,
    "logicalLocations": null
},
"resourceVersion": "1.0",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-09T18:26:56.647Z"
}

```

## Advanced security alert updated

Event: An advanced security alert is updated.

- Publisher ID: `advsec`
- Event ID: `ms.vss-alerts.alert-updated-event`
- Resource name: `resource`

## Settings

- `repository`: Include only events for alerts associated with a specific repository.
- `branch`: Include only events for alerts associated with a specific branch.
- `alertType`: Include only events for alerts of a specific type.
  - Valid values:

- Unknown
  - Dependency
  - Secret
  - Code
- `severity`: Include only events for alerts with a specific severity.
  - Valid values:
    - Low
    - Medium
    - High
    - Critical
    - Note
    - Warning
    - Error
    - Undefined

## Sample payload

JSON

```
{
  "id": "a0a0a0a0-bbbb-cccc-dddd-e1e1e1e1e1",
  "eventType": "ms.vss-alerts.alert-updated-event",
  "publisherId": "advsec",
  "message": {
    "text": "New alert is updated",
    "html": "New alert is updated",
    "markdown": "New alert is updated"
  },
  "detailedMessage": {
    "text": "New alert is updated\r\n\r\n- Alert status: Updated\r\n",
    "html": "New alert is updated\r\n\r\n- Alert status: Updated\r\n",
    "markdown": "New alert is updated\r\n\r\n- Alert status: Updated\r\n"
  },
  "resource": {
    "alertId": 1,
    "severity": "critical",
    "title": "Alert title",
    "tools": [
      {
        "name": "codeql",
        "rules": [
          {
            "opaqueId": null,
            "friendlyName": "codeql rule",
            "description": null,
            "resources": null,
            "helpMessage": "update the version",
            "version": "1.0.0"
          }
        ]
      }
    ]
  }
}
```

```

        "tags": null,
        "additionalProperties": null
    }
]
}
],
"dismissal": {
    "dismissalId": 1,
    "message": "Fixed",
    "stateChangedBy": "66aa66aa-bb77-cc88-dd99-00ee00ee00ee",
    "stateChangedByIdentity": null,
    "requestedOn": null,
    "dismissalType": "fixed"
},
"repositoryUrl": "https://dev.azure.com/test/test/_git/test",
"gitRef": "testRef",
"alertType": "code",
"firstSeenDate": null,
"lastSeenDate": null,
"fixedDate": null,
"introducedDate": null,
"state": "fixed",
"physicalLocations": null,
"logicalLocations": null
},
"resourceVersion": "1.0",
"resourceContainers": {
    "collection": {
        "id": "b1b1b1b1-cccc-dddd-eeee-f2f2f2f2f2f2"
    },
    "account": {
        "id": "bbbb1b1b-cc2c-dd3d-ee4e-fffffff5f5f5f"
    },
    "project": {
        "id": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee"
    }
},
"createdDate": "2025-06-09T18:31:56.933Z"
}

```

## Resource containers

Each event payload contains a `resourceContainers` dictionary that includes the IDs of the project, collection, account, or server where the event originates.

In some products and environments, the dictionary also includes a `baseUrl` field for each entry. That field provides the full URL to the container. You can use this URL to create a connection to the container to make REST API calls.

## Related content

- [Integrate with service hooks](#)
- [Create a service hook subscription programmatically](#)
- [Service hook consumers](#)

ⓘ **Note:** The author created this article with assistance from AI. [Learn more](#)

# Troubleshoot service hooks

07/10/2025

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

This article offers general troubleshooting guidance for Azure DevOps service hooks. It also provides answers to frequently asked questions (FAQs).

## View activity and debug problems

The **Service Hooks** page in the web access admin summarizes activity from the last seven days for each subscription. The page also shows whether each subscription is enabled, disabled, or restricted.

For each subscription, you can access a detailed history that includes the complete request and response data for each event. This information can help you debug a problematic service or subscription.

1. To view the activity and status of your subscriptions, go to the **Service Hooks** page.

The screenshot shows the 'Service Hooks' page in the Azure DevOps web interface. The left sidebar has 'Project Settings' selected. The main area displays a table of service hook subscriptions:

Consumer	Event	Action	Action Description	Owner	7 Day Status ↑	State
Trello	Work item created	Create a card	List a0a0a0a0bbb...	Nils Aunins	3 succeeded	Enabled
Azure Service Bus	Repository forked	Repository [Any]	Send a message to a S...	Nils Aunins	2 succeeded, 1 failed	Enabled
Azure Storage	Code pushed	Any branch on any repository.	Insert a message in a S...	Nils Aunins	2 succeeded	Enabled
Web Hooks	... Work item created	Any work item	Post via HTTP	To host my-app@...	1 succeeded, 1 failed	Enabled (restricted)

2. To view detailed activity for a subscription, including full request, response, and event payload data, select a subscription in the table, and then select **History**.

The screenshot shows the 'History' details for a specific service hook entry. The top bar indicates the event was a 'Work item created > Web Hooks (Post via HTTP)'. The entry was 'Created by: Nils Aunins (an hour ago)' and 'Modified by: Nils Aunins (an hour ago)'. The 'Detailed logging: On' option is selected.

The history table shows two entries:

Status	When ↑	Summary	Request	Response	Event
Succeeded	an hour ago	✓ Failed			
Failed	53 minutes ago				

Details for the failed entry:

- Summary:** Failed
- Message:** Task #1024 (Add AI functionality) created by Nils Aunins
- Error Message:** Bad Request (400)

# Subscription failures and probation (restricted)

If the HTTP response to a notification request indicates an error, the severity of the failure determines how Azure DevOps responds. Certain types of failures can disable subscriptions or put them on probation.

## Failure types

Failures from service hook notifications are grouped into the following categories:

- Terminal failures
- Transient failures
- Enduring failures

The error code from the HTTP response determines how Azure DevOps categorizes the failure.

### Terminal failures

The only HTTP status code that's categorized as a terminal failure is 410 (Gone).

When a terminal failure occurs in a subscription, the subscription is automatically disabled no matter its prior state.

### Transient failures

HTTP responses with the following status codes are categorized as transient failures:

- 408 (Request Timeout)
- 502 (Bad Gateway)
- 503 (Service Unavailable)
- 504 (Gateway Timeout)

When a transient failure occurs in a subscription, Azure DevOps attempts to resend the notification up to eight times, with an increasing delay between each attempt.

The following table lists information about retries that are attempted after a transient failure occurs. Included is the approximate *backoff* time, or the time to wait before attempting to resend a notification. The maximum backoff time is 60 seconds. The table also shows the total delay for each retry.

 Expand table

Retry number	Backoff time in seconds	Total delay in seconds
1	1	1
2	2	3
3	4	7
4	8	15
5	16	31
6	32	63
7	60	123
8	60	183

If all retries for a notification are exhausted and each attempt results in a transient failure, the notification is no longer sent. Instead, it's categorized as an enduring failure.

## Enduring failures

All other HTTP failure codes, for example, 404 (Not Found) and 500 (Internal Server Error), result in enduring failures.

When an enduring failure occurs in a subscription, the subscription is placed on [probation](#).

## Probation

When a subscription is on probation, any new events are lost. The system makes a limited number of attempts to resend the failed notification.

The following table lists approximate backoff times and total probation times for retries that are attempted during probation. At most seven retries are attempted, and the maximum backoff time for a probation retry is 15 hours.

[ ] [Expand table](#)

Retry number	Backoff time	Total probation time in hours
1	20 minutes	0.33
2	40 minutes	1
3	1 hour 20 minutes	2.33

Retry number	Backoff time	Total probation time in hours
4	2 hours 40 minutes	5
5	5 hours 20 minutes	10.33
6	10 hours 40 minutes	21
7	15 hours	36

If the subscription receives a successful response while on probation, it gets restored to a fully enabled state, and events are published again. If all seven retries fail, the subscription state gets set to *DisabledBySystem*.

## FAQs

### Q: What is the payload limit of a service hook?

A: The payload limit is 2 MB. Larger payloads cause degradation in performance and reliability. As a best practice, service hooks should limit the payload to 2 MB.

### Q: What does the state Enabled (restricted) mean?

A: A subscription becomes restricted if too many failures occur. Being in the *Enabled (restricted)* state is the same as being on probation.

### Q: What does the state Disabled (due to failures) mean?

A: A subscription is automatically disabled in the following cases:

- A terminal failure is encountered.
- A series of consecutive failures occurs over a prolonged period.

Notifications that result in transient failures are retried several times before being declared enduring failures. Enduring failure notifications are retried a limited number of times during [probation](#). If all probation retries fail, the subscription gets disabled.

The following status codes provide examples of each type of failure:

- Transient: 408 (Request Timeout), 502 (Bad Gateway), 503 (Service Unavailable), 504 (Gateway Timeout)
- Terminal: 410 (Gone)
- Enduring: All failures that aren't transient or terminal

## **Q: What does the state Disabled (user left project) mean?**

A: The user who created the subscription is no longer a member of the team.

## **Q: What should I try if a service hook isn't working?**

A: Check the following items:

- Confirm the subscription is enabled.
- Confirm the subscription settings are correct. Check event filters and actions.
- Look at the history, especially if there are failures.

## **Q: Can I grant a regular project user the ability to view and manage service hook subscriptions for a project?**

A: By default, only project administrators have these permissions. To grant them to other users directly, you can use the [command-line tool](#) or the [Security](#) REST API.

## **Q: Can I programmatically create subscriptions?**

A: Yes, use [REST APIs](#).

# Use Azure Boards with Slack

10/10/2025

## Azure DevOps Services

If you use [Slack](#), you can use the [Azure Boards app for Slack](#) to create work items and monitor work item activity in your Azure Boards project from your Slack channel.

The Azure Boards app for Slack allows users to set up and manage subscriptions in their Slack channel. They can manage subscriptions for create, update, and other work item events. Users can also get notifications for these events in their Slack channel. Conversations in the Slack channel can be used to create work items. Previews for work item URLs help users to start discussions around work.

Only visible to you

**Azure Boards** APP 10:03 PM  
Successfully created Feature 26 : Design fault tolerant network monitors

**Azure Boards** APP 11:35 PM  
Turing created a new Feature.  
Feature 27 : Geo replication of checksum data  
State New Assigned to Alice

**Azure Boards** APP 11:40 PM  
Turing updated a Bug.  
Bug 28 : Network switch reboot is causing service timeouts  
Assigned to : Alice  
State : Active  
Area path : Smart 360 Vision  
Priority : 1  
Severity : 1 - Critical  
Discussion : We have seen 2% service degradation in the last 72 hours

1 Create work items from your channel

2 Get notified when a work item is created

3 Monitor changes to work items and much more

### ! Note

Notifications are sent to channels—they don't appear in direct messages.

## Prerequisites

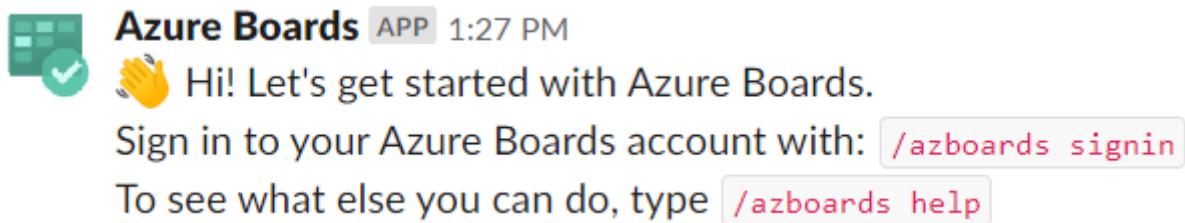
[Expand table](#)

Category	Requirements
Platform	Azure Boards and the Azure Boards Slack app work only with Azure DevOps Services (cloud); they're not supported on Azure DevOps Server.

Category	Requirements
Access levels	Contributor for the project.
Permissions	To create subscriptions in a Slack channel for work item events: Member of the <a href="#">Project Administrators</a> or <a href="#">Team Administrators</a> group.
Organization policies	To receive notifications, the organization must have the <a href="#">Third party application access via OAuth policy</a> enabled.
Microsoft Entra account	If your organization is connected to a Microsoft Entra ID tenant, sign in with an account that's a native member of that tenant; guest or external accounts can experience authentication errors when signing in through Slack.

## Add the Azure Boards app to your Slack workspace

1. To install the Azure Boards app to your Slack workspace, open a web browser, sign into Slack, and open the [Azure Boards app](#).
2. Once added, a welcome message displays from the app as shown in the following image.



3. Use the `/azboards` Slack handle to interact with the app. A list of commands is provided later in this article, [Command reference](#).

## Link your Azure Boards project to the Azure Boards app

To use the app, you must first link your Azure Boards project to your Slack channel.

1. Once the app gets installed in your Slack workspace, connect and authenticate yourself to Azure Boards.

The screenshot shows a Slack message from the 'Azure Boards' app at 1:28 PM. It displays a note: 'Looks like you are not signed in to Azure Boards. Please sign in before making any request.' Below this is a 'Sign in' button and a placeholder for entering a verification code.

```
/azboards link [project url]
```

- After signing in, use the following slash command inside a Slack channel to link to the Azure Boards project that you specify with the URL:

```
/azboards link [project url]
```

For example:

```
/azboards link https://dev.azure.com/myorg/myproject
```

Once the project is linked, you can create work items using `/azboards create` command or use message actions.

## Set up subscriptions to monitor work items

You can create subscriptions to monitor work items at any time using the `/azboards subscriptions` command. You have an option of setting up subscriptions just after linking a project.

- Select area path you want, the event that you're interested in, and use the associated filters to customize your Slack channel. To easily set up subscriptions, your recently accessed area paths are shown in the area path dropdown.

Azure Boards APP 11:33 PM  
Add a new subscription to this channel. Can't find an area path? Add a new one with: /azboards addareapath [area path]

Area path: Smart 360 Vision

Event: Work item updated

Work item type: Bug

Field: Assigned To

Links are added or removed: Yes

**Cancel** **Add filters** **Save**

In case your team's area path doesn't appear in the Area path dropdown menu, follow the instructions mentioned in the next section, [Add area paths](#). Area paths added using the `/azboards addAreopath` command and area paths for which subscriptions are created in the Slack channel always appear in the Area path dropdown along with recently accessed area paths.

## Add area paths

You can add areas that your team works on to the channel so that they're always available for creating work items and subscriptions. This action is important mainly for the teams with more than 100 area paths.

- Use the following command to add area paths from your project to the Slack channel.

/azboards addAreopath [area path]

For example:

/azboards addAreopath myproject\fabrikam

Only visible to you

 **Azure Boards** APP 11:46 PM

Successfully added area path 'Smart 360 Vision\Mobile' to this channel.

- If you choose project name as your area path, then you receive notifications for all the area paths in the project. It's logically equivalent to choosing 'Any' area path.

## Create a work item with a command

1. With the Azure Boards app, you can create work items from your channel. The app supports custom work items as well.
- To create a work item, use `/azboards create`.

The image consists of three vertically stacked screenshots of the Azure Boards app interface:

- Screenshot 1: Choose work item type**  
Shows a message: "Which type of work item would you like to create?" with options: Bug, User Story, Task, and Choose an option... A note says "Only visible to you".
- Screenshot 2: Select area path**  
Shows a message: "Please choose an area path. Can't find an area path? Add a new one using `/azboards addareapath [area path]`". A dropdown menu shows "Smart 360 Vision".
- Screenshot 3: Fill necessary details**  
Shows a modal dialog titled "New Feature" with fields for "Title" and "Description (optional)". Buttons include "Cancel" and "Submit". A note says "Only visible to you".

2. You can create work items directly from a command by passing work item type and title as parameters. Work items get created only if they don't have any fields to be mandatorily filled.

```
/azboards create [work item type] [work item title]
```

For example:

/azboards create 'user story' Push cloud monitoring alerts to mobile devices

# Create a work item from message actions

Often, discussions in a channel call for creation of work items. You can use message actions to create a work item. The selected message is prefilled in the description section of the work item. A link back to the conversation in the channel is stored in the Discussion section of the newly created work item, giving users access to the discussion that led to the creation of the work item.

- To create work items using message actions

The screenshot illustrates the workflow for creating a work item from a message action:

- 1) Use message actions to create work item from a conversation in the channel:** A message from Kyle in a Slack channel is shown. The message content is: "Jarvis, service call to network monitors are failing. Please create a task to investigate this." A context menu is open over the message, with the "Create a work item Azure Boards" option highlighted by a red box.
- 2) Conversations are prepopulated in the work item:** The "New Task" dialog box is displayed. The "Description (optional)" field contains the message text: "Jarvis, service call to network monitors are failing. Please create a task to investigate this.", which is also highlighted by a red box.
- 3) Work item will have a link back to the conversation in the Slack channel. This helps to provide complete context for the work item:** The newly created work item, titled "29 Investigate network monitors", is shown. The "Discussion" section includes a comment from Karthik: "Karthik commented just now This work item has been created from Slack: Link". This comment is also highlighted by a red box.

# Manage Azure Boards subscriptions

1. To view, add and remove subscriptions for a channel, use the `/azboards subscriptions` command:

```
/azboards subscriptions
```

This command lists all the current subscriptions for the channel and allows you to add new subscriptions and remove existing ones. As part of adding subscriptions, you can also customize what you get notified on by using various filters.

## ⓘ Note

Team administrators can't remove or modify subscriptions created by Project administrators.

⌚ Only visible to you

 **Azure Boards** APP 1:03 AM

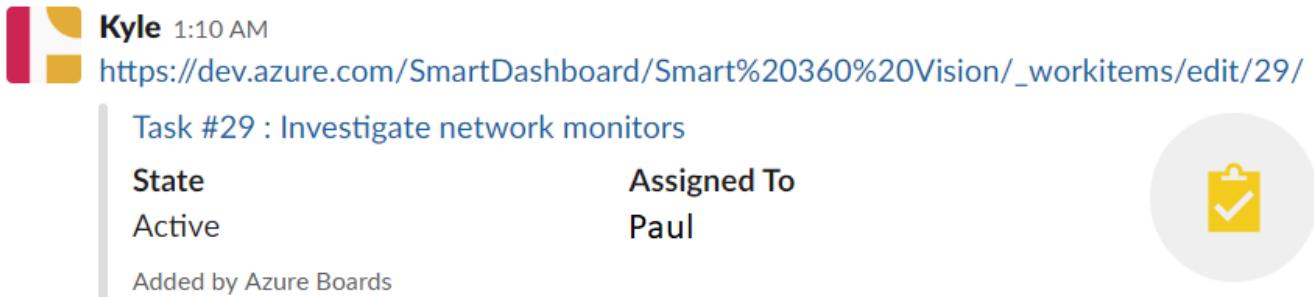
This channel has 3 subscriptions from 2 area paths in project **Smart 360 Vision**:

- Smart 360 Vision**  
Work item commented on  
For Work item type: *Feature*  
[Remove](#)
- Smart 360 Vision**  
Work item created  
For Work item type: Any  
[Remove](#)
- Smart 360 Vision\Mobile**  
Work item updated  
For Work item type: Bug, Tag: *Prod Issues*, Links are added or removed: Yes  
[Remove](#)

[Add subscription...](#)

## Previews of work item URLs

To support collaboration around work items discussed within a channel, a preview of work items referenced in the channel is displayed. When a user pastes the work item URL, a preview is shown similar to the following image. This preview helps to keep work-item-related conversations relevant and correct.



A screenshot of a Slack message from a channel. The message is from a user named Kyle at 1:10 AM. It contains a link to an Azure Boards work item: [https://dev.azure.com/SmartDashboard/Smart%20360%20Vision/\\_workitems/edit/29/](https://dev.azure.com/SmartDashboard/Smart%20360%20Vision/_workitems/edit/29/). The work item preview shows the title "Task #29 : Investigate network monitors", the state "Active", the assignee "Paul", and a yellow checkmark icon indicating it's completed. The message also includes the text "Added by Azure Boards".

For this feature to work, users have to be signed-in. Once they're signed in, this feature works for all channels in a workspace.

## Unlink a project from a channel

A Slack channel can only link to one Azure Boards project at a time. To link to a different project, you must first unlink the current project using `/azboards unlink` command.

Unlinking a project deletes all the subscriptions along with added area paths from the channel. If the channel has no subscriptions, any user can unlink a project. However if a channel has subscriptions, only project admins can unlink a project from a channel.

## Command reference

The following table lists all the `/azboards` commands you can use in your Slack channel.

[ ] Expand table

Slash command	Functionality
<code>/azboards link [project url]</code>	Link a project to this channel to create work items and receive notifications
<code>/azboards subscriptions</code>	Add or remove subscriptions for this channel
<code>/azboards create or /azboards create [work item type] [title]</code>	Create a work item
<code>/azboards addAreapath [area path]</code>	Add an area path from your project to this channel
<code>/azboards signin</code>	Sign in to your Azure Boards organization

Slash command	Functionality
/azboards signout	Sign out from your Azure Boards organization
/azboards unlink	Unlink a project from this channel
/azboards feedback	Report a problem or suggest a feature
/azboards help	Get help on the commands

## Manage work in private channels

The Azure Boards app for Slack can help you create work items and monitor the work item activity in your private channels as well. To invite the bot to your private channel, enter `/invite @azboards`. After you post that, you can create work items and manage your notifications in the same way as you would for a public channel.

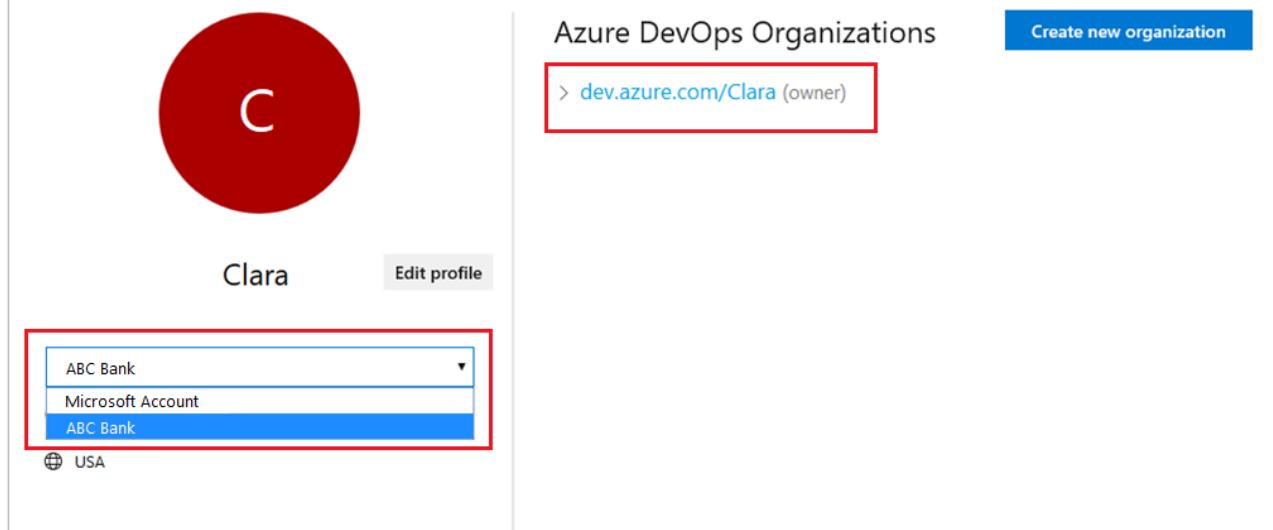
## Troubleshoot errors

If you're experiencing the following errors when using the [Azure Boards App for Slack](#), follow the procedures in this section.

### Configuration failed. Please make sure that the organization '{organization name}' exists and that you have sufficient permissions.

To resolve this authentication issue, complete the following steps:

1. Sign out of Azure DevOps by going to `https://aka.ms/VsSignout` in your browser.
2. Open an **InPrivate** (Microsoft Edge) or **Incognito** (Chrome) browser window.
3. Go to `https://aex.dev.azure.com/me` and sign in with your credentials.
4. Select the correct directory from the dropdown menu under your profile icon. Choose the directory that contains the organization with the project you want to link.



Azure DevOps Organizations

> dev.azure.com/Clara (owner)

Clara Edit profile

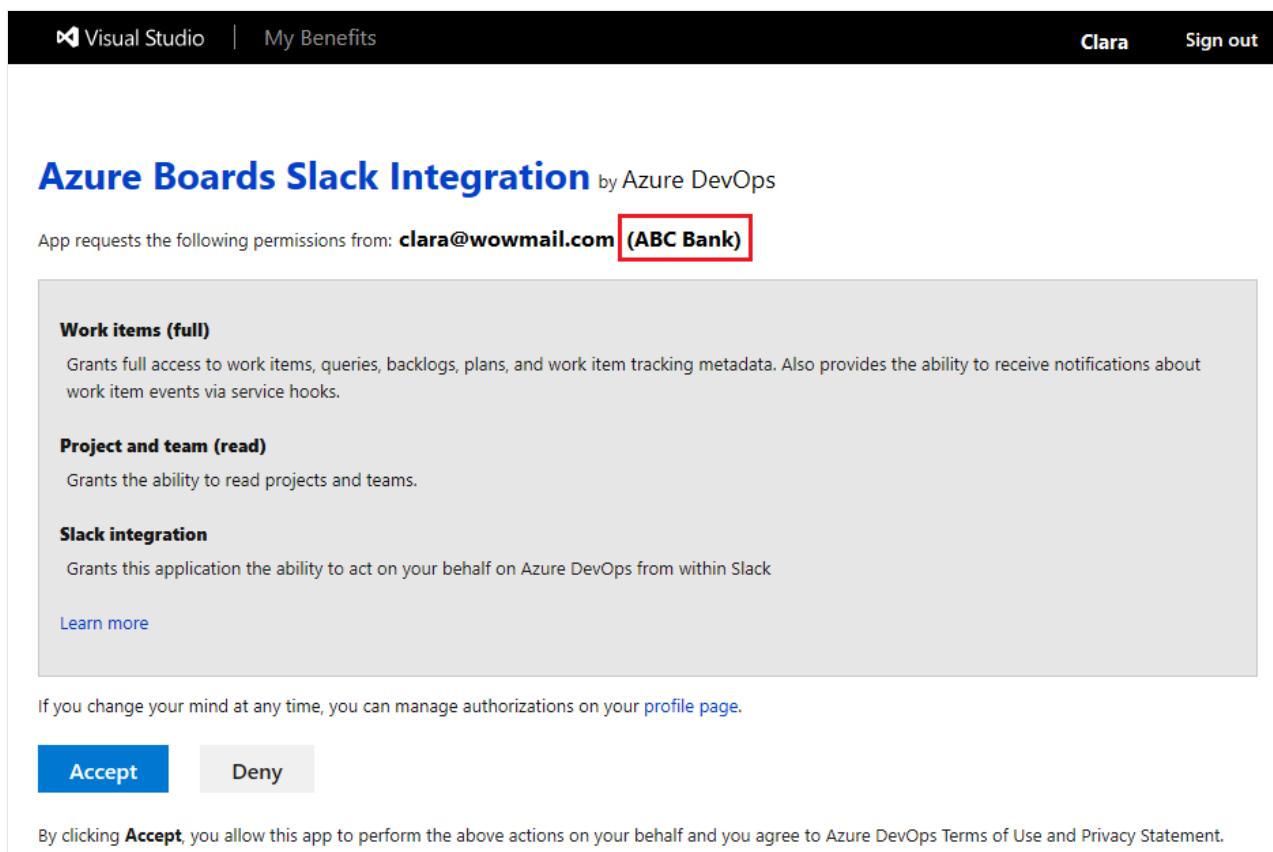
ABC Bank

Microsoft Account

ABC Bank

USA

5. In the **same browser**, start a new tab, go to <https://slack.com>, and sign in to your work space (**use web client**). Run the `/azboards signout` command followed by the `/azboards signin` command.
6. Select the **Sign in** button and you get redirected to a consent page like the one in the following example. Ensure that the directory shown beside the email is same as what was chosen in the previous step. Accept and complete the sign-in process.



Visual Studio | My Benefits Clara Sign out

## Azure Boards Slack Integration by Azure DevOps

App requests the following permissions from: clara@wowmail.com (ABC Bank)

**Work items (full)**  
Grants full access to work items, queries, backlogs, plans, and work item tracking metadata. Also provides the ability to receive notifications about work item events via service hooks.

**Project and team (read)**  
Grants the ability to read projects and teams.

**Slack integration**  
Grants this application the ability to act on your behalf on Azure DevOps from within Slack

Learn more

If you change your mind at any time, you can manage authorizations on your [profile page](#).

**Accept** **Deny**

By clicking **Accept**, you allow this app to perform the above actions on your behalf and you agree to Azure DevOps Terms of Use and Privacy Statement.

If these steps don't resolve your authentication issue, reach out to us at [Developer Community](#).

# Conditions and limitations

- The Azure Boards app for Slack works only with Azure DevOps Services (cloud); it isn't supported on Azure DevOps Server.
- Notifications go to channels only (they don't appear in direct messages).
- To create channel subscriptions, be a Project or Team Administrator (see the [Prerequisites](#) section).

## Related content

- [Define area paths and assign to a team](#)
- [Use Azure Pipelines with Slack](#)
- [Use Azure Repos with Slack](#)
- [Create a service hook for Azure DevOps with Slack](#)

# Use Azure Boards in Microsoft Teams

Article • 01/30/2025

## Azure DevOps Services

This article describes how to use the Azure Boards and Azure DevOps apps for Microsoft Teams to create and monitor Azure Boards work items from your Teams channels.

### Note

This feature is only available on Azure DevOps Services. Typically, new features are introduced in the cloud service first, and then made available on-premises in the next major version or update of Azure DevOps Server. For more information, see [Azure DevOps Feature Timeline](#).

By using the Azure Boards app for Microsoft Teams, you can:

- Set up subscriptions to create and manage work items and work item events in your Teams channels.
- Create work items directly from channel conversations.
- Search for and share work items across channels using the messaging extension.
- View previews of work items from URLs.

### Note

- Azure Boards notifications aren't supported inside Teams chat or direct messages.
- The Azure Boards app for Microsoft Teams isn't supported for O365 Government Community Cloud (GCC) customers that use an Azure Commercial subscription in conjunction with a GCC tenant.

## Prerequisites

 Expand table

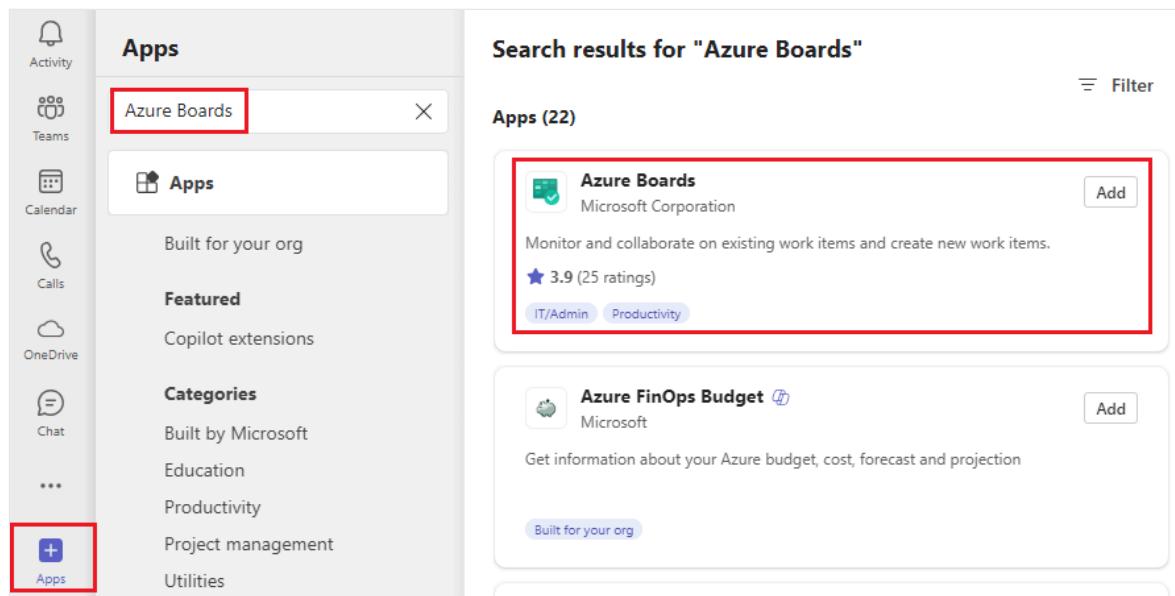
Category	Requirements
Access levels	<ul style="list-style-type: none"><li>- Contributor for the project.</li><li>- Access to a team and channel in Microsoft Teams where you can install an app.</li></ul>

Category	Requirements
Permissions	To create subscriptions in a Slack channel for work item events: Member of the <a href="#">Project Administrators</a> or <a href="#">Team Administrators</a> group.
Policies	To receive notifications: <a href="#">Third party application access via OAuth policy enabled for the organization</a> .

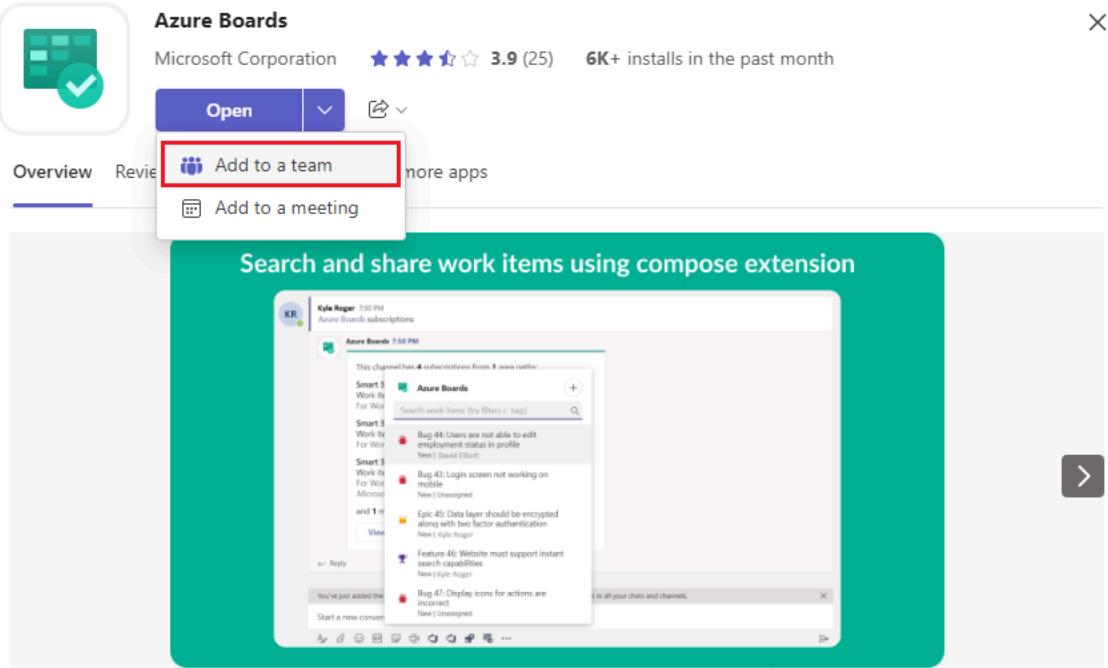
## Add the Azure Boards app to Teams

To add the Azure Boards app to your Teams channels, do the following steps.

1. In Microsoft Teams, select **Apps**, search for *Azure Boards*, and then select **Azure Boards**.



2. Select the dropdown arrow next to **Open**, and select **Add to a team**.



**Azure Boards**

Microsoft Corporation 3.9 (25) 6K+ installs in the past month

**Open** Add to a team more apps Add to a meeting

Search and share work items using compose extension

Kyle Roger 7:50 PM Azure Boards subscriptions

Azure Boards 7:50 PM

This channel has 4 unread messages from 3 areas active.

Smart 3 Work Items For Work Smart 3 Work Items For Work Smart 3 Work Items For Work Microsoft and 1 more View

+ Search work items by filter or tag

Bug 44: Users are not able to edit employment status in profile New | Create | Edit

Bug 43: Login screen not working on mobile devices New | Create | Edit

Epic 45: Data layer should be encrypted along with two factor authentication New | Create | Edit

Feature 46: Website must support instant search capabilities New | Create | Edit

Bug 47: Display icons for actions are incorrect New | Create | Edit

You've just added the Azure Boards app to your team!

Start a new conversation

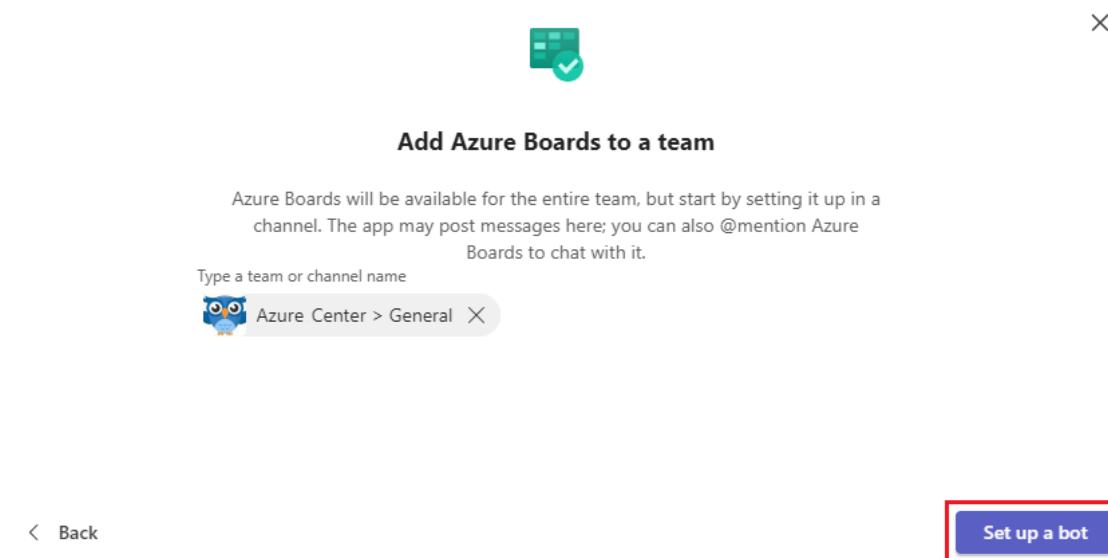
In all your chats and channels.

Monitor and collaborate on existing work items and create new work items.

Azure Boards enables you to plan, track and discuss work across your teams. Azure Boards app for Microsoft Teams lets you monitor work item activity in your projects, create new work items, search and share work items with your colleagues and get previews of work items using URLs.

By using Azure Boards, you agree to the [privacy policy](#), [terms of use](#), and [permissions](#).

3. Select or enter your team name, and then select **Set up a bot**.



### Add Azure Boards to a team

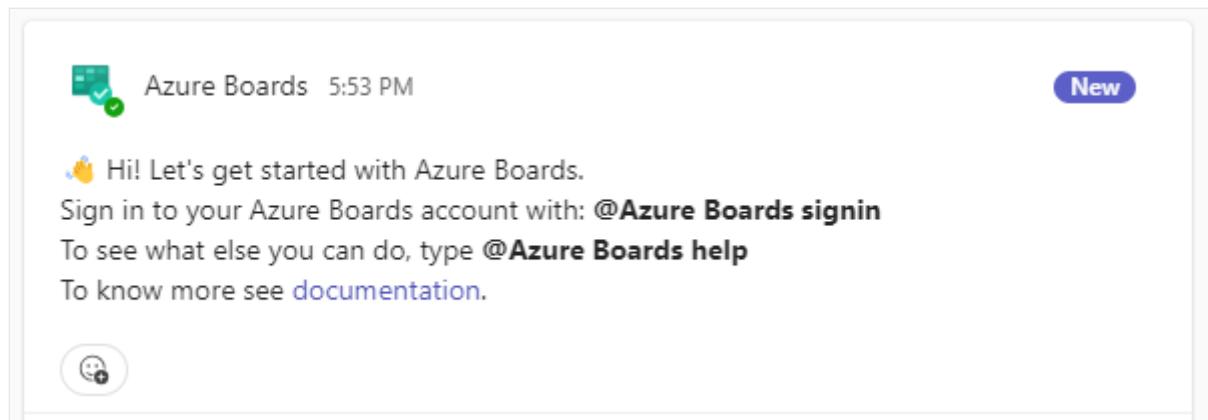
Azure Boards will be available for the entire team, but start by setting it up in a channel. The app may post messages here; you can also @mention Azure Boards to chat with it.

Type a team or channel name

 Azure Center > General X

Back Set up a bot

After the app installs, you see a welcome message in the Teams conversation pane.



## Azure Pipelines app commands

You can use the following `@azure boards` commands to interact with the Azure Boards app in your Teams channel.

[ Expand table

Command	Functionality
<code>@azure boards sign in</code>	Sign in to your Azure Boards organization.
<code>@azure boards sign out</code>	Sign out from your Azure Boards organization.
<code>@azure boards link &lt;project url&gt;</code>	Link a project to this channel to create work items and receive notifications.
<code>@azure boards unlink</code>	Unlink a project from this channel.
<code>@azure boards subscriptions</code>	Add or remove subscriptions for this channel.
<code>@azure boards addAreapath &lt;area path&gt;</code>	Add an area path from your project to this channel.
<code>@azure boards feedback</code>	Report a problem or suggest a feature.
<code>@azure boards help</code>	Get help on the commands.

## Link your Azure Boards project to the app

To use the app, sign in to Azure Boards and link your Azure Boards project to your Teams channel.

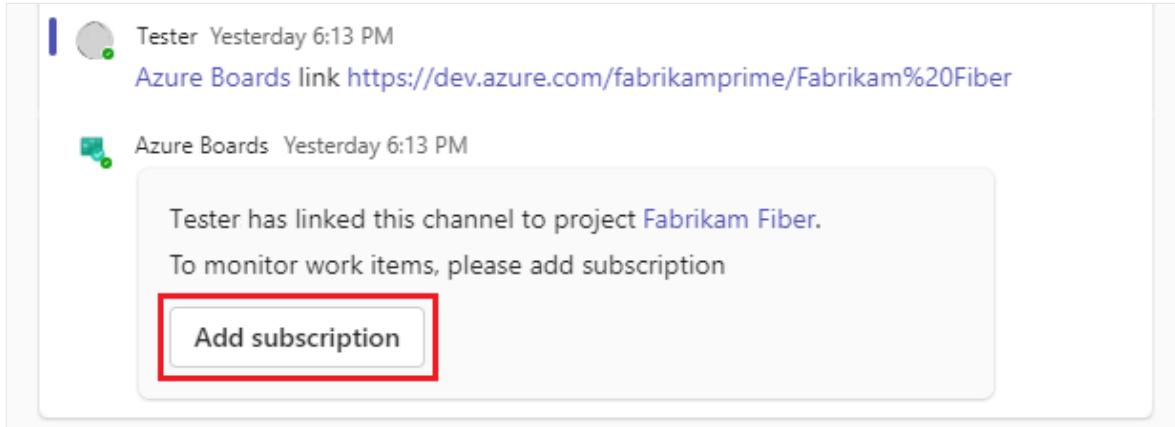
1. In the Teams conversation pane, enter `@azure boards signin`.
2. Select **Sign in** and complete authentication to Azure Boards. Select **Sign in with different email** if your Microsoft Teams and Azure Boards are in [different tenants](#).

3. Use the `@azure boards link` command to link to your Azure DevOps project URL.

For example:

```
@azure boards link https://dev.azure.com/myorg/myproject/
```

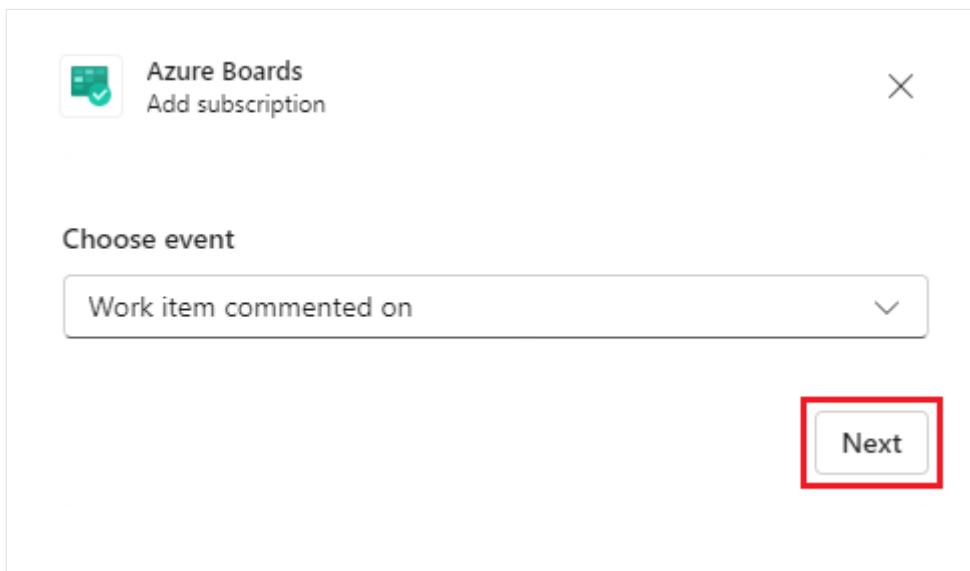
4. Select the **Add subscription** button in the linked project notification to start monitoring your project.



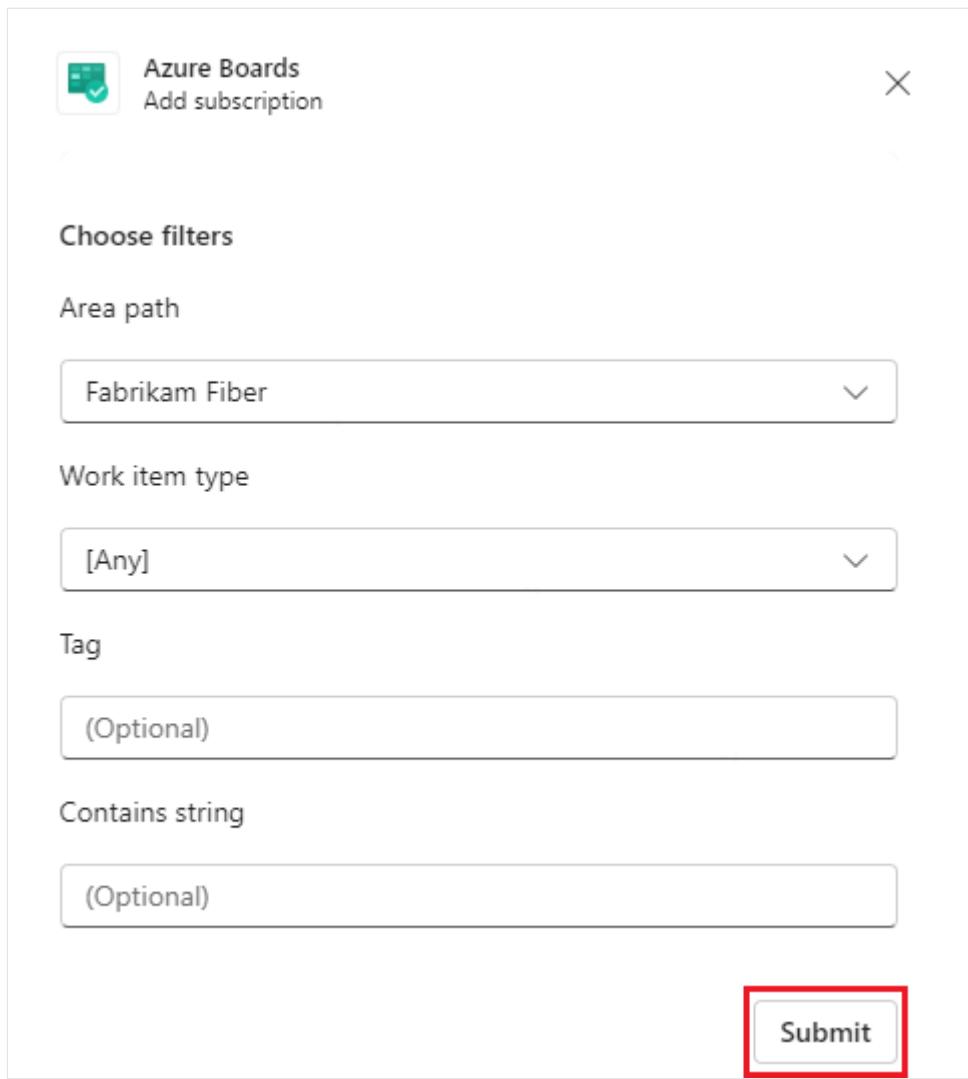
## Set up subscriptions

After your project is linked, begin monitoring project work items by selecting the **Add subscription** button. You can add more subscriptions anytime by using the `@azure boards subscriptions` command.

1. Under **Choose event**, select the event you want to subscribe to and select **Next**.



2. Under **Choose filters**, select the **Area path**, **Work item type**, and optionally specify **Tags** or a specific string to filter on, and then select **Submit**.



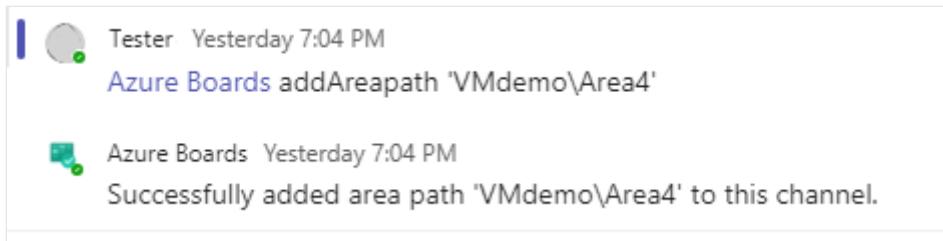
## Add area paths

Area paths that have subscriptions in the channel, recently accessed area paths, and area paths that you add by using the `@azure boards addAreapath` command appear in the **Area path** dropdown menu when you create a subscription. If the area path that your team works on doesn't appear in the dropdown menu, you can add it so that it's always available for creating work items and subscriptions. This feature is especially useful for teams with more than 100 area paths in their project.

Use the `@azure boards addAreapath` command to add area paths from your project to the Teams channel. For example:

```
@azure boards addAreapath 'VMdemo\Area4'
```

You get a success message.

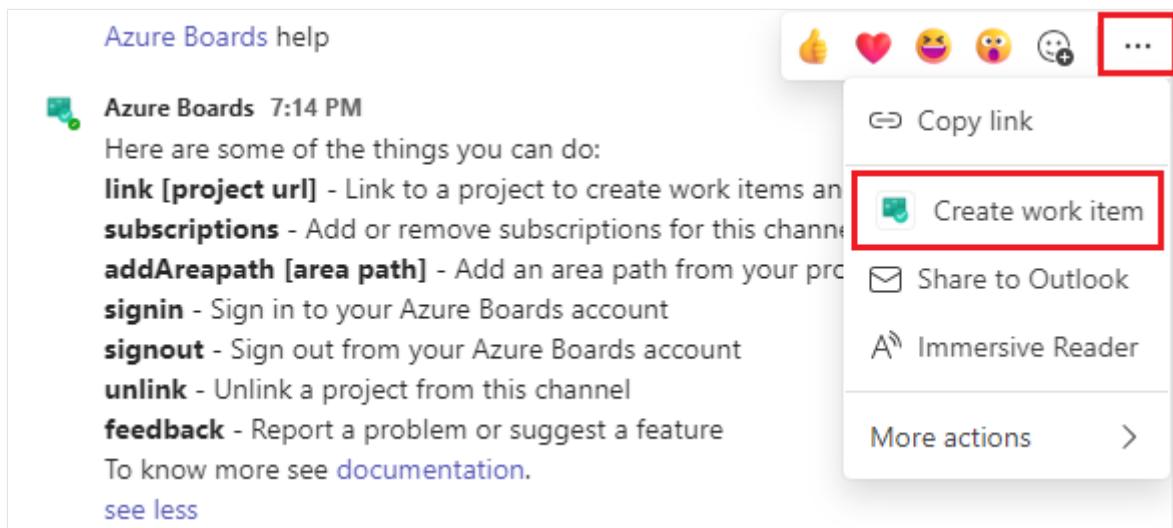


If you choose the project name as your area path, you get notifications for all the area paths in the project.

## Create work items

You can use the Azure Boards app to create work items from your channel by using a message action.

1. In any message in the channel, select the **More actions** ellipse in the actions panel, and then select **Create work item**.



2. Select the type of work item you want to create, and select **Next**.
3. Enter a **Title** and select an **Area path** for the work item.
4. The text of the message becomes the work item **Description** or **Repro Steps**, depending on work item type, or you can edit this text. Select **Create**.

Azure Boards  
New Bug

**Title**  
Bug from Teams chat

**Area path**  
VMdemo\Team3

**Repro Steps**

```
<p>Here are some of the things you can do:<br>
<b>link [project url]</b> - Link to a project to create work items and receive notifications<br>
<b>subscriptions</b> - Add or remove subscriptions for this
```

**Create**

The new work item appears in Azure Boards and contains a link back to the Teams item that generated the work item.

BUG 253

253 Bug from Teams chat

Unassigned    1 comment    Add tag    Save    Follow    ...

State	Proposed	Area	VMdemo\Team3	Updated: Just now
Reason	New	Iteration	VMdemo	

**Symptom**  
Click to add Symptom

**Effort (Hours)**  
Original Estimate

**System Info**

**Fix**  
Remaining work

**Discussion**

Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Tester commented just now  
Created from Microsoft Teams: [Link](#)

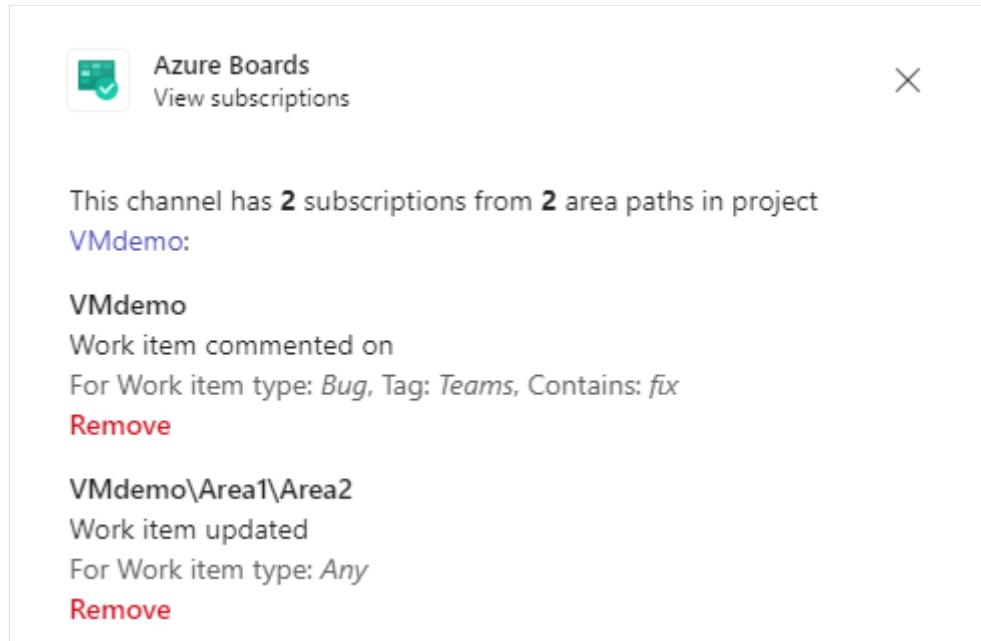
**Classification**  
Discipline

Root Cause  
Unknown

**Checklist**

## Manage Azure Boards subscriptions

To view, add, and remove subscriptions for a channel, use the `@azure boards subscriptions` command. This command lists all the current subscriptions for the channel. You can add new subscriptions and remove existing ones. When you add a subscription, you can customize notifications by using filters.



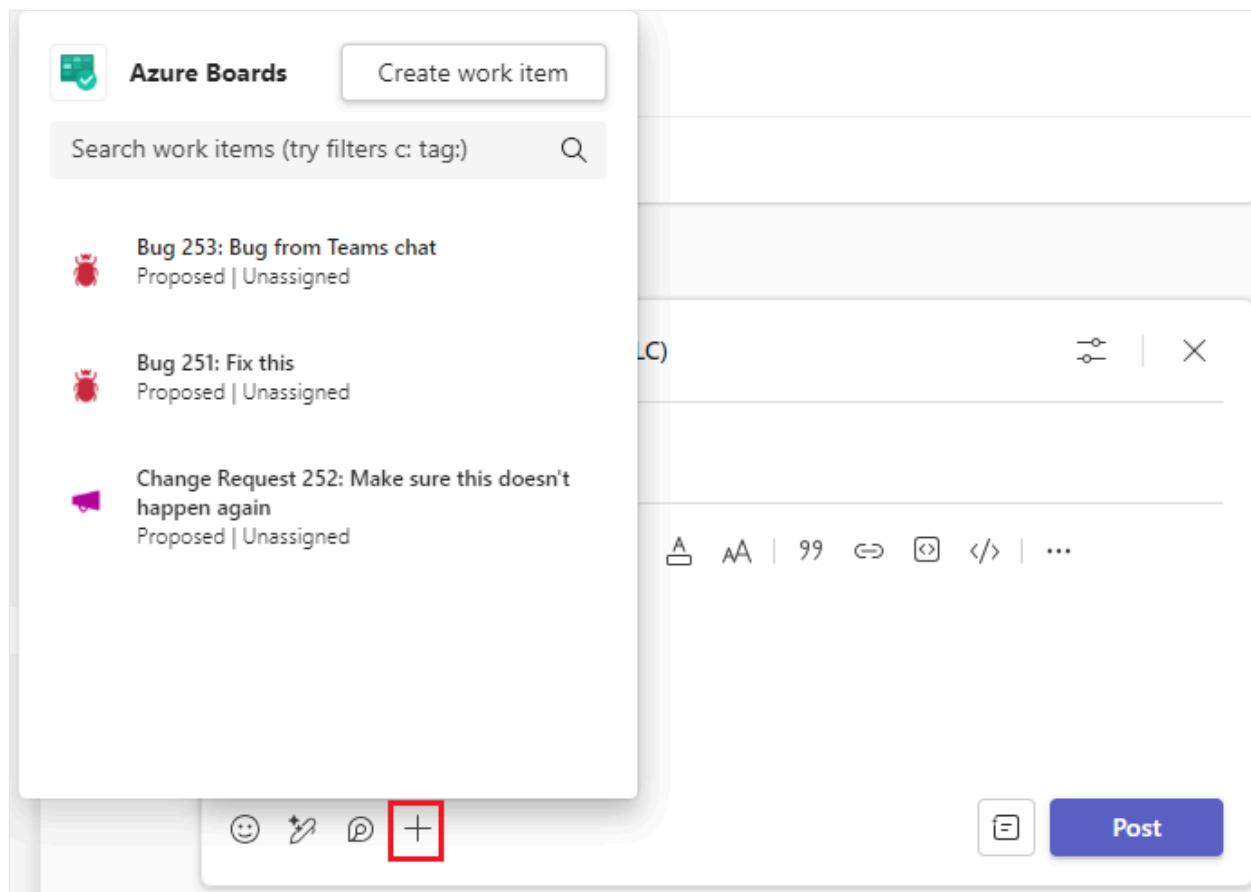
! Note

Team Administrators can't remove or modify subscriptions that Project Administrators created.

## Use the compose extension

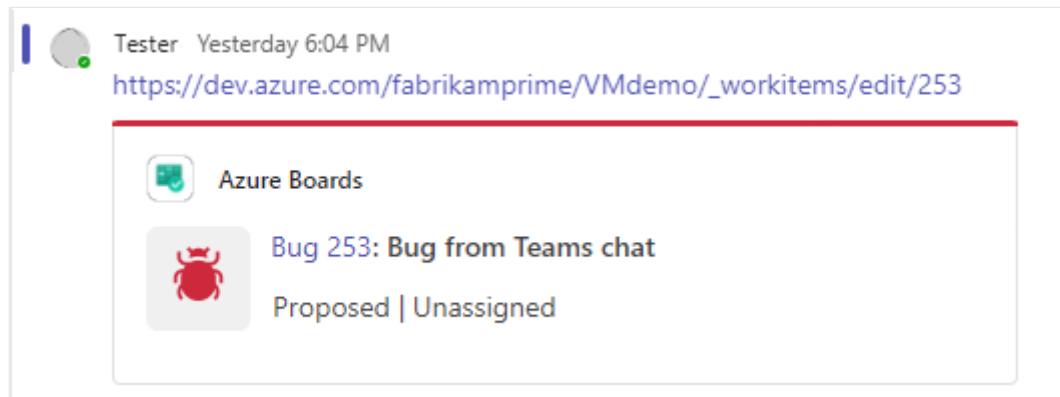
To help you search for and share work items, the Azure Boards app for Microsoft Teams supports the compose extension. You can search for work items by work item ID, title, or supported functional command. For a list of supported commands, see [Functional work item search](#).

To use the compose extension, be signed in to the Azure Boards project in the Teams channel. Select the + symbol in the message field, select **Azure Boards**, and then search for a work item. You can also select **Create work item** to create a new work item.



## Preview work items

To support collaboration around work items you discuss in a channel, the Azure Boards app displays a preview of work items you reference. When you paste in a work item URL or select a work item from the compose extension, the app shows a preview similar to the following image. This URL unfurling feature works for all channels in the team.



## Unlink a project from a channel

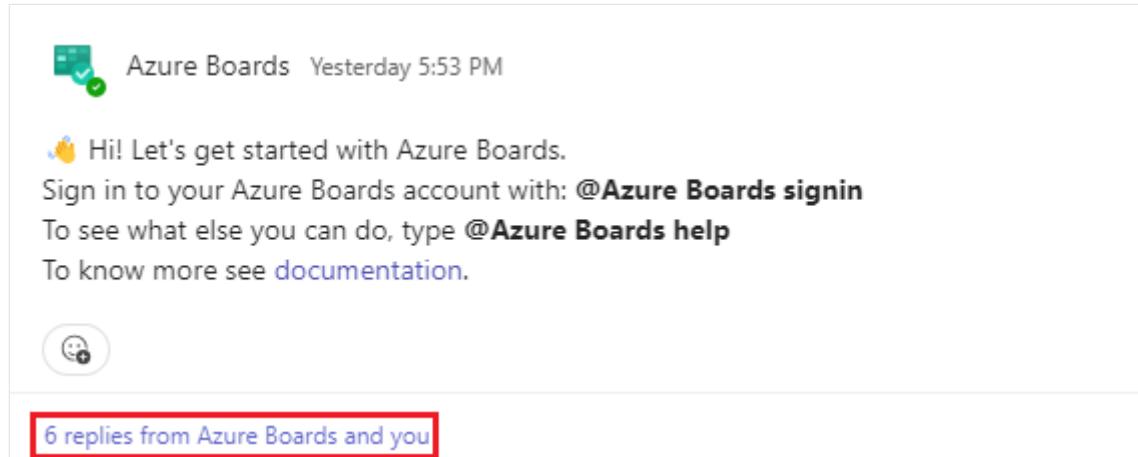
A Teams channel can link to only one Azure Boards project at a time. To link to a different project, you must first unlink the current project by using `@azure boards unlink`.

Unlinking a project deletes all the project subscriptions and added area paths from the channel. If the channel has no subscriptions, any user can unlink a project. If the channel has subscriptions, only Project Administrators can unlink the project.

## Expand and collapse threads

A Teams channel collapses posts in threads to logically link and reduce related posts in the channel. All replies to a particular post are linked together.

To expand the thread, select the compacted thread link.



The screenshot shows a Microsoft Teams channel interface. A message from 'Azure Boards' is displayed, which reads: 'Hi! Let's get started with Azure Boards. Sign in to your Azure Boards account with: [@Azure Boards signin](#). To see what else you can do, type [@Azure Boards help](#). To know more see [documentation](#).'. Below the message is a small circular icon with a smiley face. At the bottom of the message card, there is a red-bordered box containing the text '6 replies from Azure Boards and you'.

To return to the channel and collapse the thread, select **Go to channel**.

Post by Azure Boards

General

Go to channel

Azure Boards Yesterday 5:53 PM

Hi! Let's get started with Azure Boards.  
Sign in to your Azure Boards account with: **@Azure Boards signin**  
To see what else you can do, type **@Azure Boards help**  
To know more see documentation.

Tester Yesterday 6:04 PM

Azure Boards signin

Azure Boards Yesterday 6:05 PM

Please sign in before making any request.

Sign in

Tester Yesterday 6:13 PM

Azure Boards link <https://dev.azure.com/fabrikamprime/Fabrikam%20Fiber>

Azure Boards Yesterday 6:13 PM

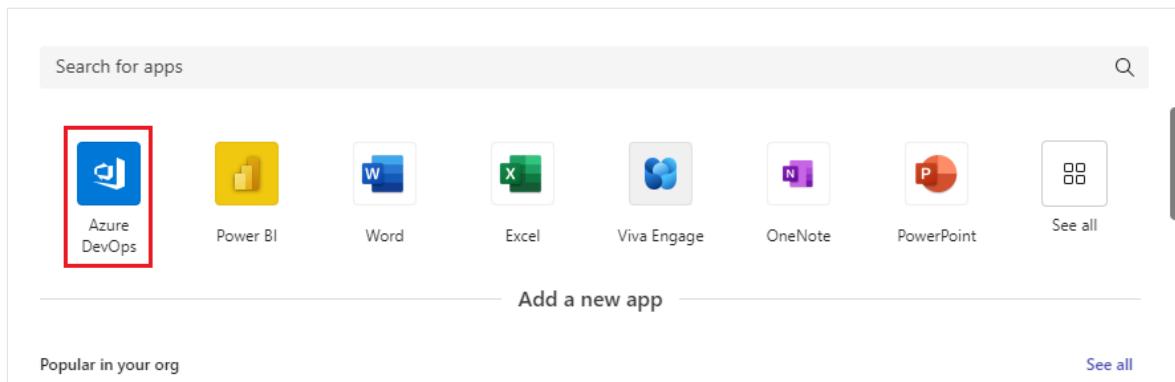
Tester has linked this channel to project **Fabrikam Fiber**.  
To monitor work items, please add subscription

Add subscription

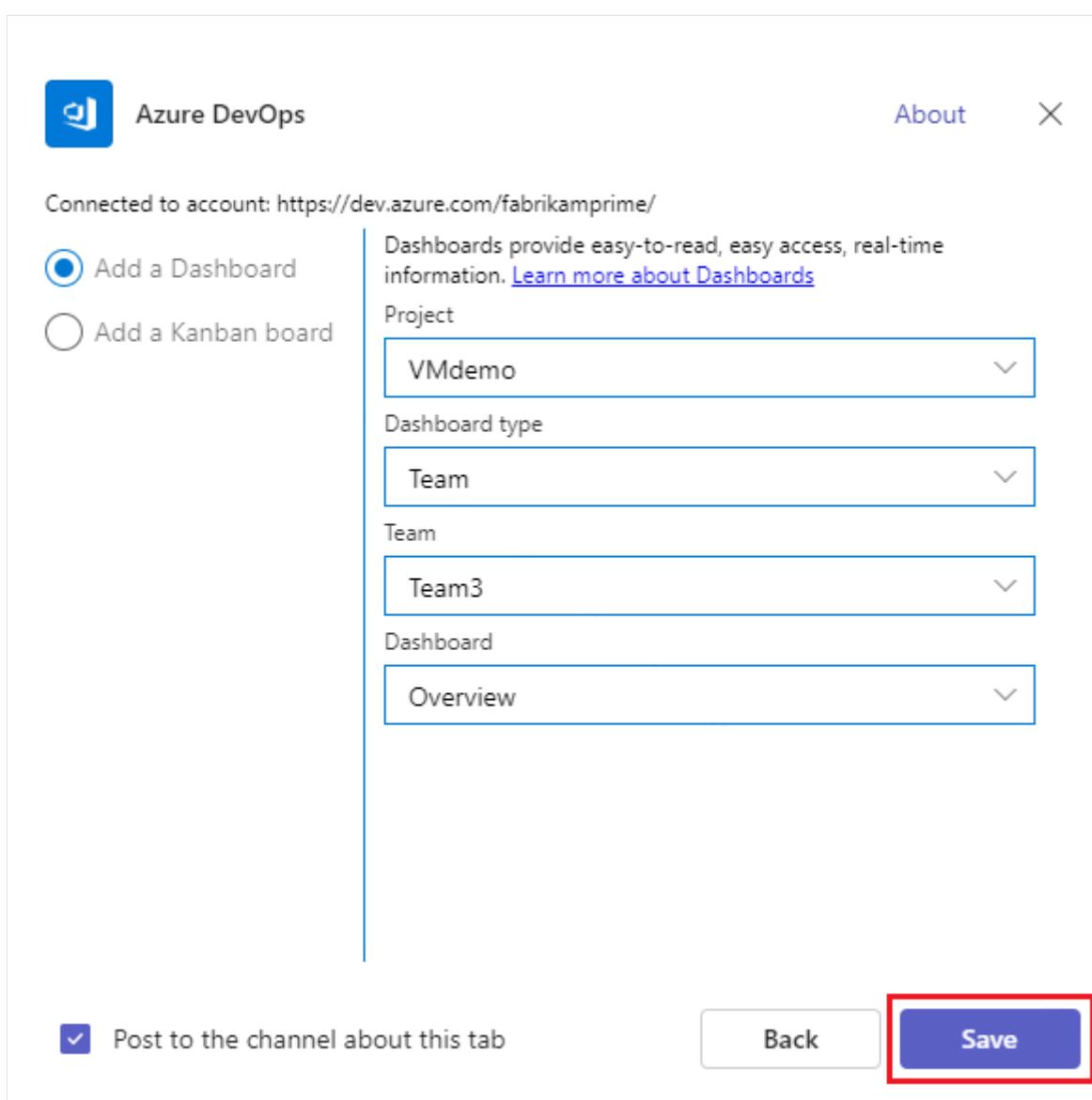
## Configure Azure DevOps tabs

To bring your project dashboard or Kanban board into Teams, you can install the Azure DevOps app in a tab in your Teams channel. The Azure DevOps app lets you insert content from the app in messages, and get notifications from the app in your channels.

1. In Teams, select the + symbol on the top navigation of your channel or select Apps from the left menu.
2. Search for if necessary, and then select **Azure DevOps**.



3. Select and sign in to your Azure DevOps organization.
4. On the **Azure DevOps** screen, select a **Project**, and whether to add a **Dashboard** or a **Kanban board** to the tab. Select other configurations depending on your choice, and select whether you want to post to the channel about adding the tab.



5. Select **Save**. The new tab and board appear in your channel.

## Connect different tenants

If you use different emails or tenants for Microsoft Teams and Azure DevOps, follow these steps to sign in and connect based on your settings.

[+] Expand table

Microsoft Teams	Azure DevOps	Sign in action
email1@abc.com (tenant1)	email1@abc.com (tenant1)	Select Sign in
email1@abc.com (tenant1)	email2@abc.com (tenant2)	1. Sign in to Azure DevOps. 2. In the same browser, start a new tab and go to <a href="https://teams.microsoft.com/">https://teams.microsoft.com/</a> . 3. Run the <code>sign in</code> command and select Sign in.
email1@abc.com (tenant1)	email2@pqr.com (tenant2)	1. Select Sign in with different email address. 2. In the email ID picker, use the email2 to sign in.
email1@abc.com (tenant1)	email2@pqr.com (nondefault tenant3)	Not supported.

## Troubleshoot authentication issues

If you receive the error **Configuration failed. Please make sure that the organization '{organization name}' exists and that you have sufficient permissions**, try the following steps to resolve the error.

1. In the same browser, start a new tab and sign in to <https://teams.microsoft.com/>.
2. In this tab, go to the channel where the Azure Boards app for Microsoft Teams is installed and run the `@azure boards signout` command and then the `@azure boards signin` command.
3. Select the **Sign in** button, and complete the sign-in process. Ensure that the directory shown is the same one you chose in the previous step.

If these steps don't resolve your authentication issue, reach out to the [Developer Community](#).

## Related articles

- [Define area paths and assign to a team](#)
- [Use Azure Pipelines with Microsoft Teams](#)

- Use Azure Repos with Microsoft Teams
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#)

# Use Azure Repos with Slack

09/22/2025

## Azure DevOps Services

If you use [Slack](#), you can use the [Azure Repos app for Slack](#) to easily monitor your Azure Repos repositories. Set up and manage subscriptions to receive notifications in your channel whenever code is pushed or checked in and whenever a pull request (PR) gets created, updated, or merged. This app supports both Git and Team Foundation Version Control (TFVC) events.

## Prerequisites

[ ] [Expand table](#)

Category	Requirements
Platform	Azure Repos Slack app works only with Azure DevOps Services (cloud); it isn't supported on Azure DevOps Server.
Permissions	To create subscriptions in a Slack channel for repository-related events: Member of the <b>Project Administrators</b> group or team administrator. For more information, see <a href="#">Change project-level permissions</a> or <a href="#">Add a team administrator</a> . To receive notifications: <b>Third-party application access via OAuth</b> setting enabled for the organization. For more information, see <a href="#">Change application access policies for your organization</a> .
Organization policies	Your organization must allow third-party apps to sign in using OAuth. Check <b>Organization settings &gt; Security &gt; Policies</b> and enable <b>Third party application access via OAuth</b> if it's not already on.
Microsoft Entra account	If your organization is connected to a Microsoft Entra ID tenant, sign in with an account that is a native member of that tenant; guest or external accounts can experience authentication errors when signing in through Slack.

### ! Note

- Notifications are sent to channels—they don't appear in direct messages.
- To create channel subscriptions be a Project or Team Administrator (see the [Prerequisites](#) section).
- Your organization must allow third-party apps to sign in using OAuth. Check **Organization settings > Security > Policies** and enable **Third party application access via OAuth** if it's not already on.

- If your organization is connected to a Microsoft Entra ID tenant, sign in with an account that is a native member of that tenant; guest or external accounts can run into authentication errors when signing in through Slack.

## Add the Azure Repos app to your Slack workspace

1. Go to the [Azure Repos Slack app](#) and select Add to Slack.

The screenshot shows the Azure Repos app card in the Slack App Directory. The card includes the following details:

- Icon:** A red icon with a white gear and branch symbol.
- Name:** Azure Repos
- Description:** Monitor pull request activity
- Details:**
  - Alice Taranova created a pull request.
  - 13098: Allow data import if alm extension is installed and data loader is used in Web layout.
  - Source branch: master
  - Target branch: master
  - Reviewers: @Alice Okorop, @Zola Irem, [Fabrikam] PR Approvers and 2 more
  - This PR has merge conflicts | Repository: Fabrikam
- Supported Languages:** English
- Pricing:** Free
- Learn more & Support:**
  - Get app support
  - Visit developer website
  - ✉️ AzDevInMlChatOps@microsoft.com
  - 🔗 Privacy policy
- Categories:**
  - Developer Tools
  - New & Noteworthy

Once added, you receive something like the following welcome message.

The screenshot shows a welcome message from the Azure Repos app in Slack:

**Azure Repos APP 4:49 PM**

Hi! Let's get started with Azure Repos.

Subscribe to one or more repositories with: `/azrepos subscribe [repository url]`

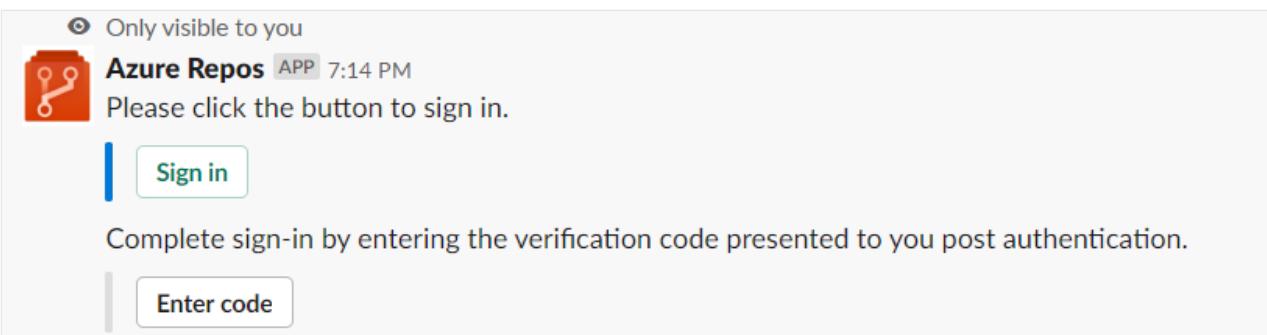
To see what else you can do, type `/azrepos help`

To know more see [documentation](#).

2. Use the `/azrepos` Slack handle to interact with the app. A full list of commands is provided in the [Command reference](#) section of this article.

## Connect the Azure Repos app to your repositories

1. Connect and authenticate yourself to Azure Repos using `/azrepos signin` command.



2. Select **Sign in**.

3. Accept the Azure Repos Slack Integration.

The page title is "Azure Repos Slack Integration by Azure DevOps". It says "App requests the following permissions from:"

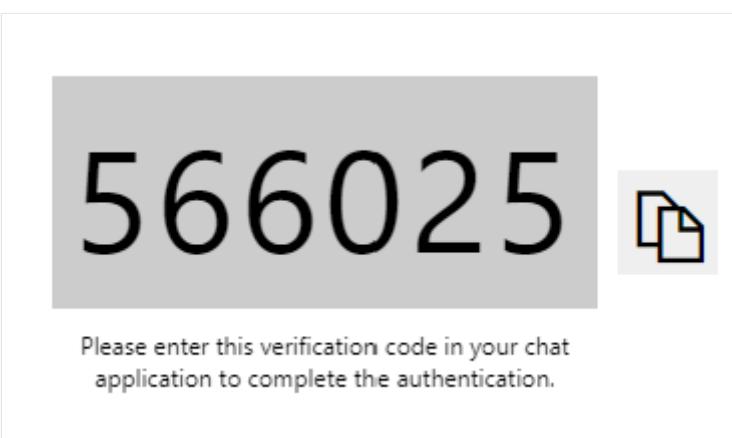
- Code (full)**: Grants full access to source code, access metadata about commits, changesets, branches, and other version control artifacts. Also grants the ability to create and manage code repositories, create and manage pull requests and code reviews, and to receive notifications about version control events via service hooks.
- Project and team (read)**: Grants the ability to read projects and teams.
- Slack integration**: Grants this application the ability to act on your behalf on Azure DevOps from within Slack

Below the permissions is a link "Learn more". At the bottom of the box, it says "If you change your mind at any time, you can manage authorizations on your profile page." There are two buttons: "Accept" (blue) and "Deny".

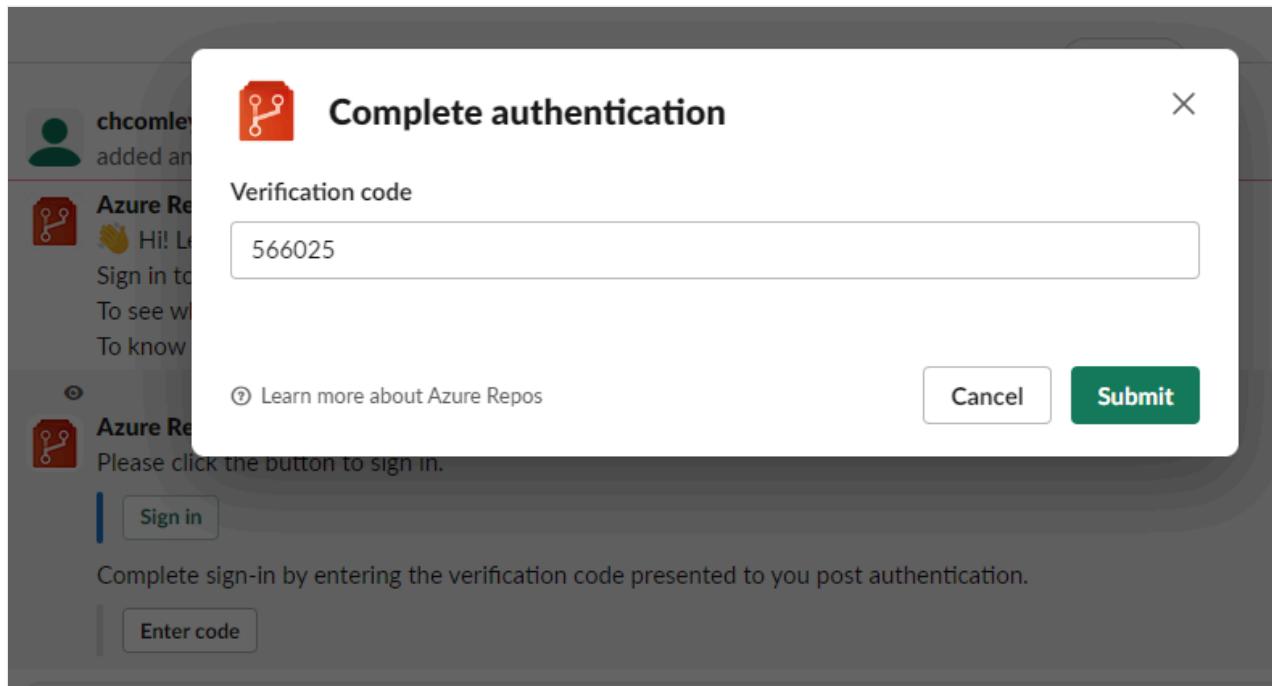
By clicking **Accept**, you allow this app to perform the above actions on your behalf and you agree to Azure DevOps Terms of Use and Privacy Statement.

A verification code displays for use in your chat app to complete authentication.

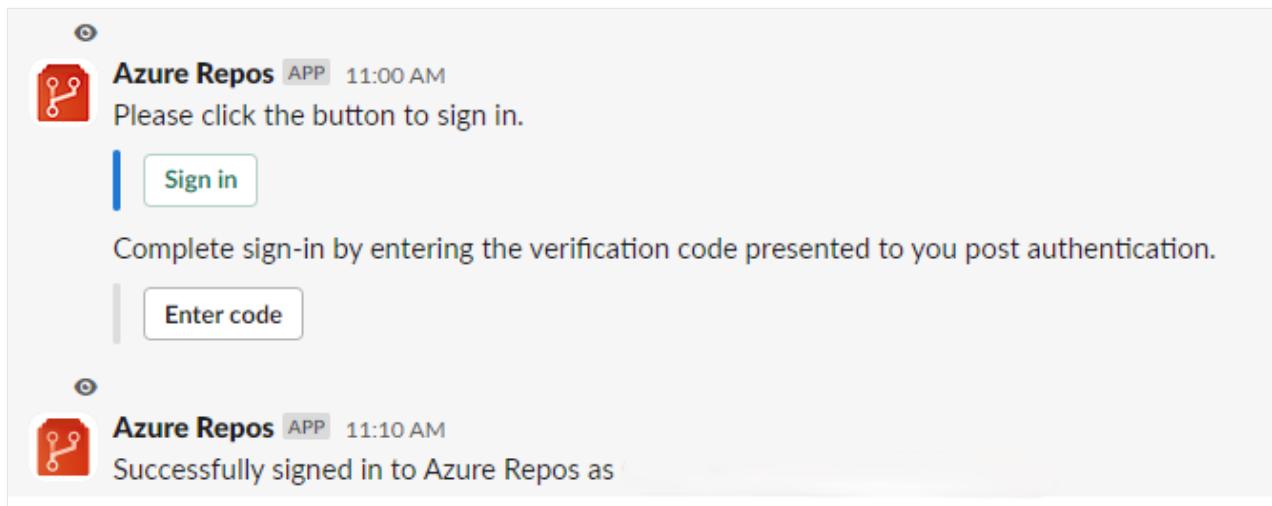
4. Copy the authentication code.



5. Select **Enter code**, paste the code, and then select **Submit**.



Confirmation of sign-in displays within the chat.



To start monitoring all Git repositories in a project, use the following slash command inside a channel:

```
slash  
  
/azrepos subscribe [project url]
```

The project URL can be to any page within your project (except URLs to repositories). For example:

```
slash  
  
/azrepos subscribe https://dev.azure.com/myorg/myproject/
```

You can also monitor a specific repository using the following command:

```
slash
```

```
/azrepos subscribe [repository url]
```

The repository URL can be to any page within your repository that has your repository name. For example, for Git repositories, use:

```
slash
```

```
/azrepos subscribe https://dev.azure.com/myorg/myproject/_git/myrepository
```

For TFVC repositories, use:

```
slash
```

```
/azrepos subscribe https://dev.azure.com/myorg/myproject/_versionControl
```

### ➊ Note

You can only subscribe to public repositories.

The subscribe command gets you started with a default subscription. For Git repositories, the channel is subscribed to the **Pull request created** event (with target branch = main), and for TFVC repositories, the channel is subscribed to the **Code checked in** event.



Azure Repos 2:55 PM

Kylie Rogers has added subscription to **Pull request created** notification for repository [SmartHotel360](#). Add or remove subscriptions for this channel with : [@Azure Repos subscriptions](#)

↪ Reply

## Manage subscriptions

To view, add, or remove subscriptions for a channel, use the following `subscriptions` command:

```
slash
```

```
/azrepos subscriptions
```

This command lists all the current subscriptions for the channel and allows you to add new subscriptions or remove existing ones. When adding subscriptions, you can customize the notifications you get by using various filters, as described in the following section.

## (!) Note

Team administrators can't remove or modify subscriptions created by Project administrators.

The screenshot shows a Microsoft Teams channel interface for the 'Azure Repos' app. At the top, there's a note that says 'Only visible to you'. Below that, it shows 'Azure Repos APP 9:52 PM' and a message stating 'This channel has 5 subscriptions from 2 repositories:'. The list of subscriptions is as follows:

- SmartHotel360**  
Code pushed  
For Branch: master  
[Remove](#)
- SmartHotel360**  
Pull request created  
For Target branch: Any  
[Remove](#)
- SmartHotel360**  
Pull request merge attempted  
For Target branch: Any, Merge result: Merge Unsuccessful - Rejected By Policy  
[Remove](#)
- SmartHotel360**  
Pull request updated  
For Target branch: WebHookRepo, Change: Reviewers changed  
[Remove](#)
- \$/TFVC Smart Repo**  
Code checked in  
Under path : \$/TFVC Smart Repo  
[Remove](#)

At the bottom of the list, there's a button labeled 'Add subscription...'

## Use filters to customize subscriptions

When a user subscribes to a repository using the `/azrepos subscribe` command, a default subscription gets created. Often, users need to customize these subscriptions. For example, users might want to get notified only when PRs have a specific reviewer.

The following steps demonstrate how to customize subscriptions.

1. Run the `/azrepos subscriptions` command.
2. In the list of subscriptions, if there's a subscription that's unwanted or must be modified (Example: creating noise in the channel), select the **Remove** button.

3. Select the Add subscription button.
4. Select the required repository and the desired event.
5. Select the appropriate filters.

## Example: Get notifications only when my team is in the reviewer list for a PR

Azure Repos APP 12:47 AM

Add a new subscription to this channel. Can't find a repository? Subscribe to a new one with: [/azrepos subscribe \[repository url\]](#)

Repository: SmartHotel360

Event: Pull request created

Target branch: [Any]

Requested by a member of group: [Any]

**Reviewer includes group:** [Smart city project]\Web Design Dev Leads

**Buttons:** Cancel, Save

## Example: Tell me when merge attempts fail due to a policy violation

Azure Repos APP 12:47 AM

Add a new subscription to this channel. Can't find a repository? Subscribe to a new one with: [/azrepos subscribe \[repository url\]](#)

Repository: SmartHotel360

Event: Pull request merge attempted

Target branch: [Any]

**Merge result:** Merge Unsuccessful - Rejected By Policy

Requested by a member of group: [Any]

Reviewer includes group: [Any]

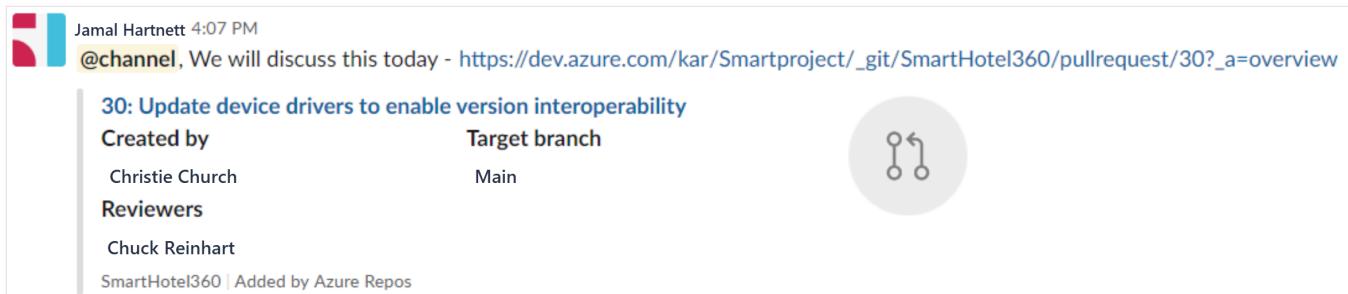
**Buttons:** Cancel, Save

### (!) Note

- All the filters are typically drop-downs. But if the drop-down has greater than 100 items, then users must enter the values manually.
- For the TFVC Code Checked in event, the filter **Under path** must be of the format `$/myproject/path`.

## Preview pull request URLs

When a user pastes the URL of a PR, a preview display like the following image, which helps to keep PR-related conversations contextual and accurate.



Once users sign in, this feature works for all channels in a workspace.

## Remove subscriptions and repositories from a channel

Use the following command to clean up your channel by removing repositories and subscriptions.

```
slash
```

```
/azrepos unsubscribe all [project url]
```

For example, the following command deletes all the subscriptions related to any repository in the project and removes the repositories from the channel. Only project admins can run this command.

```
slash
```

```
/azrepos unsubscribe all https://dev.azure.com/myorg/myproject
```

# Command reference

The following table lists all the `/azrepos commands` you can use in your Slack channel.

 Expand table

Slash command	Functionality
<code>/azrepos subscribe [repository url/ project url]</code>	Subscribe to a repository or all repositories in a project for notifications
<code>/azrepos subscriptions</code>	Add or remove subscriptions for this channel
<code>/azrepos signin</code>	Sign in to your Azure Repos organization
<code>/azrepos signout</code>	Sign out from your Azure Repos organization
<code>/azrepos feedback</code>	Report a problem or suggest a feature
<code>/azrepos unsubscribe all [project url]</code>	Remove all repositories (belonging to a project) and their associated subscriptions from a channel
<code>/azrepos help</code>	Get help on the commands

## Notifications in private channels

The Azure Repos app can help you monitor the repository events in your private channels, too. Invite the bot to your private channel by using `/invite @azrepos`. Then, you can manage your notifications the same way you would for a public channel.

## Troubleshoot

If you're experiencing the following errors when using the [Azure Repos App for Slack](#), follow the procedures in this section.

- Sorry, something went wrong. Please try again.
- Configuration failed. Please make sure that the organization '{organization name}' exists and that you have sufficient permissions.

## Sorry, something went wrong. Please try again.

The Azure Repos app uses the OAuth authentication protocol, and requires [Third-party application access via OAuth for the organization](#) to be enabled. To enable this setting,

navigate to **Organization Settings > Security > Policies**, and set the **Third-party application access via OAuth for the organization** setting to **On**.

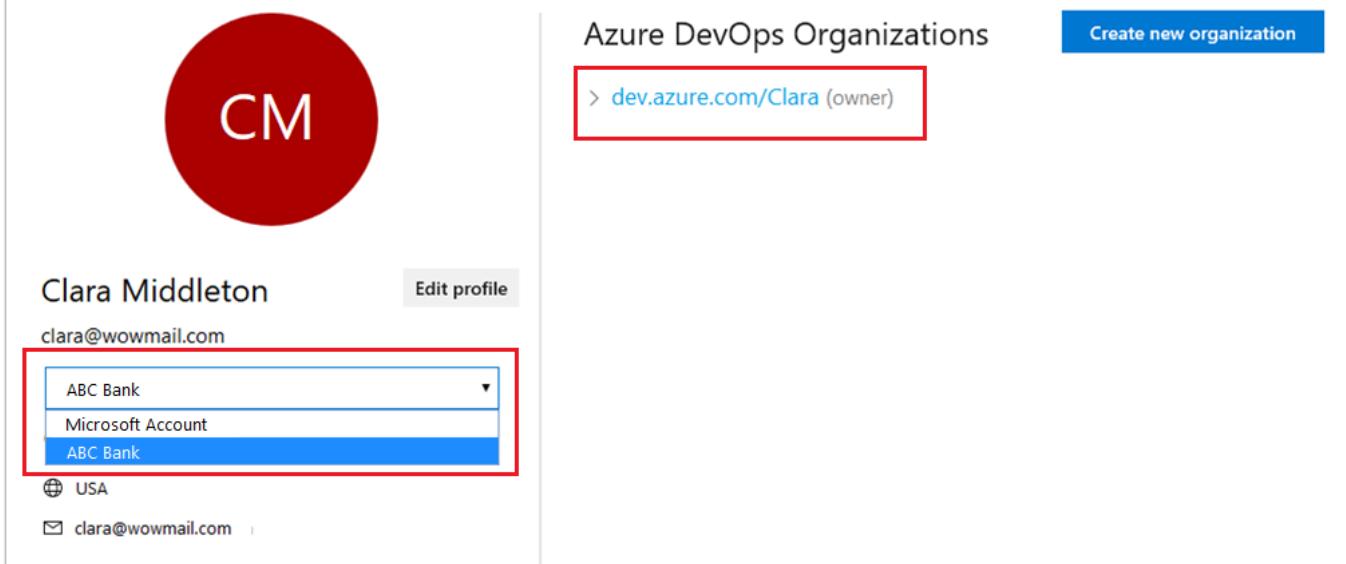
The screenshot shows the Azure DevOps Organization Settings Policies page. The left sidebar has sections for General, Security, and Boards. The Security section is expanded, showing Policies selected. The main area shows Application connection policies and Security policies. The 'Third-party application access via OAuth' policy is set to Off.

Policy Type	Status	Description
Application connection policies	Off	Third-party application access via OAuth
Security policies	On	Log Audit Events
	Off	Allow public projects
	On	Enterprise access to projects
	On	Additional protections when using public package registries
	Off	Enable IP Conditional Access policy validation

**Configuration failed. Please make sure that the organization '{organization name}' exists and that you have sufficient permissions.**

Sign out of Azure DevOps by navigating to <https://aka.ms/VsSignout> using your browser.

Open an **In private or incognito** browser window and navigate to <https://aex.dev.azure.com/me> and sign in. In the dropdown under the profile icon to the left, select the directory that contains the organization containing the repository to which you wish to subscribe.



Azure DevOps Organizations

> dev.azure.com/Clara (owner)

Create new organization

Clara Middleton

Edit profile

clara@wowmail.com

ABC Bank

Microsoft Account

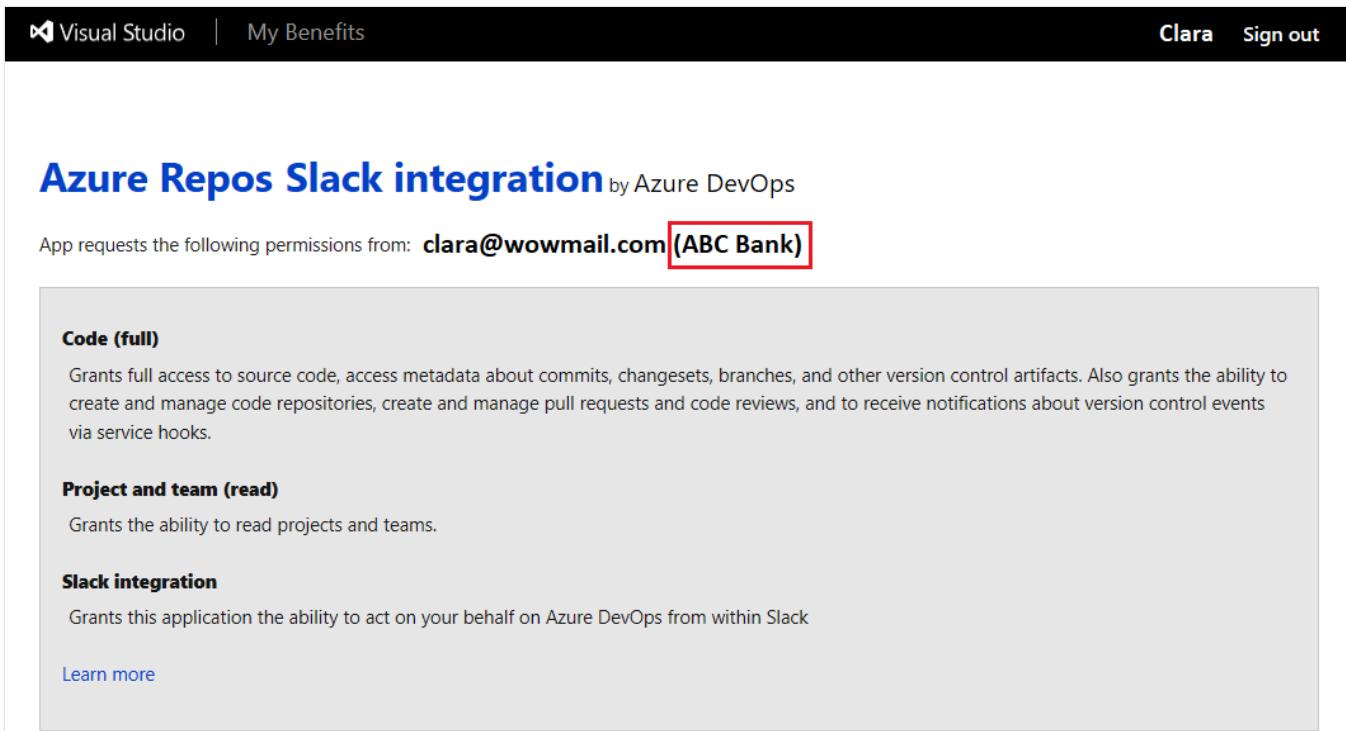
ABC Bank

USA

clara@wowmail.com

In the **same browser**, start a new tab, navigate to <https://slack.com>, and sign in to your work space (**use web client**). Run the `/azrepos signout` command followed by the `/azrepos signin` command.

Select the **Sign in** button and you're redirected to a consent page like the one in the following example. Ensure that the directory shown beside the email is same as what was chosen in the previous step. Accept and complete the sign in process.



Visual Studio | My Benefits

Clara Sign out

## Azure Repos Slack integration by Azure DevOps

App requests the following permissions from: clara@wowmail.com (ABC Bank)

**Code (full)**  
Grants full access to source code, access metadata about commits, changesets, branches, and other version control artifacts. Also grants the ability to create and manage code repositories, create and manage pull requests and code reviews, and to receive notifications about version control events via service hooks.

**Project and team (read)**  
Grants the ability to read projects and teams.

**Slack integration**  
Grants this application the ability to act on your behalf on Azure DevOps from within Slack

[Learn more](#)

If you change your mind at any time, you can manage authorizations on your [profile page](#).

**Accept**

**Deny**

By clicking **Accept**, you allow this app to perform the above actions on your behalf and you agree to Azure DevOps Terms of Use and Privacy Statement.

If these steps don't resolve your authentication issue, reach out to us at [Developer Community](#).

## Conditions and limitations

- The Azure Repos app for Slack works with Azure DevOps Services (cloud) only—it isn't available for Azure DevOps Server.
- Notifications are sent to channels (not to direct messages).
- To create channel subscriptions be a Project or Team Administrator.
- Your organization must allow third-party apps to sign in via OAuth. Check **Organization settings > Security > Policies** and enable **Third party application access via OAuth** if needed.
- If your organization uses Microsoft Entra ID, sign in with an account that is a native member of that tenant—guest or external accounts can experience authentication issues when signing in through Slack.

## Related articles

- [Integrate Azure Boards with Slack](#)
- [Integrate Azure Pipelines with Slack](#)
- [Create a service hook with Slack](#)

# Use Azure Repos with Microsoft Teams

Article • 05/22/2023

## Azure DevOps Services

If you use [Microsoft Teams](#) and Azure Repos, you can use the [Azure Repos app for Teams](#) to monitor your repos. The app supports monitoring both Git and Team Foundation Version Control (TFVC) repos, but it doesn't support integration with GitHub repos.

In this article, learn how to do the following tasks:

- ✓ Add the Azure Repos app to your team in Microsoft Teams
- ✓ Connect the Azure Repos app to your repos
- ✓ Manage subscriptions to repo related events in your channel
- ✓ Search and share PR info using compose extension
- ✓ Preview PR URLs
- ✓ Remove subscriptions and repos from a channel

## Prerequisites

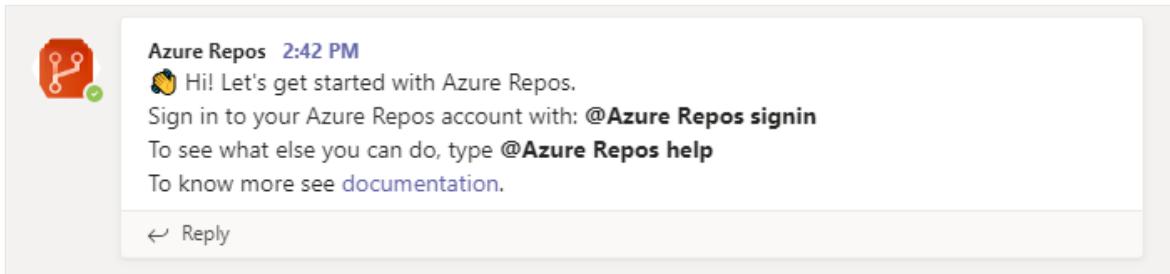
- Manage your subscription, so you receive notifications in your channel whenever code gets pushed or checked in, or when a pull request (PR) gets created, updated, or merged. To create subscriptions for repo-related events, you must be a member of the **Project Administrators** group, or a team administrator. To get added, see [Change project-level permissions](#) or [Add a team administrator](#).
- To receive notifications, enable the **Third-party application access via OAuth** setting for the Azure DevOps organization. See [Change application access policies for your organization](#).

### Note

- Notifications are currently not supported inside direct messages.
- You can only link the Azure Repos app for Microsoft Teams to a project hosted on Azure DevOps Services.

## Add the Azure Repos app to a team in Microsoft Teams

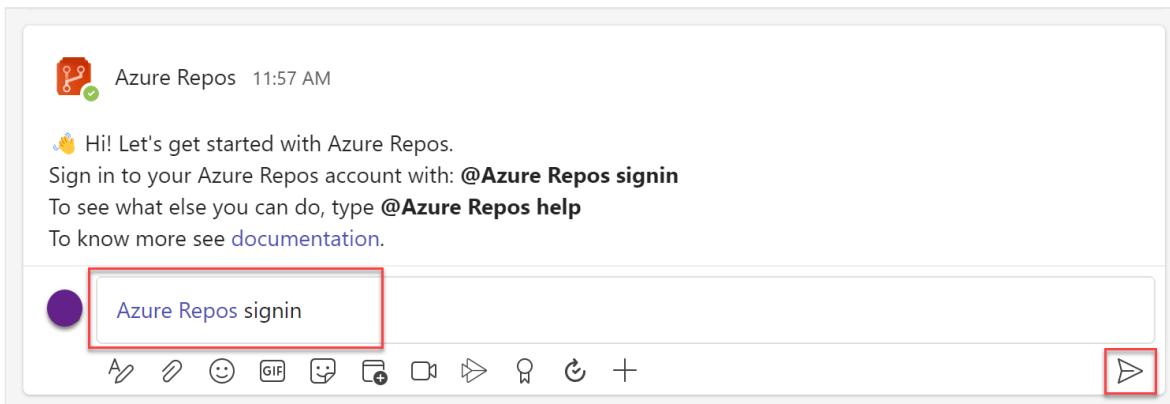
1. Go to the Azure Repos app in Microsoft Teams, [Microsoft Teams > Azure Repos](#).
2. Select **Add** or if you already downloaded the app, select the dropdown menu next to **Open**, and then select **Add to a team**.
3. Enter a team or channel name, and then select **Set up a bot**.

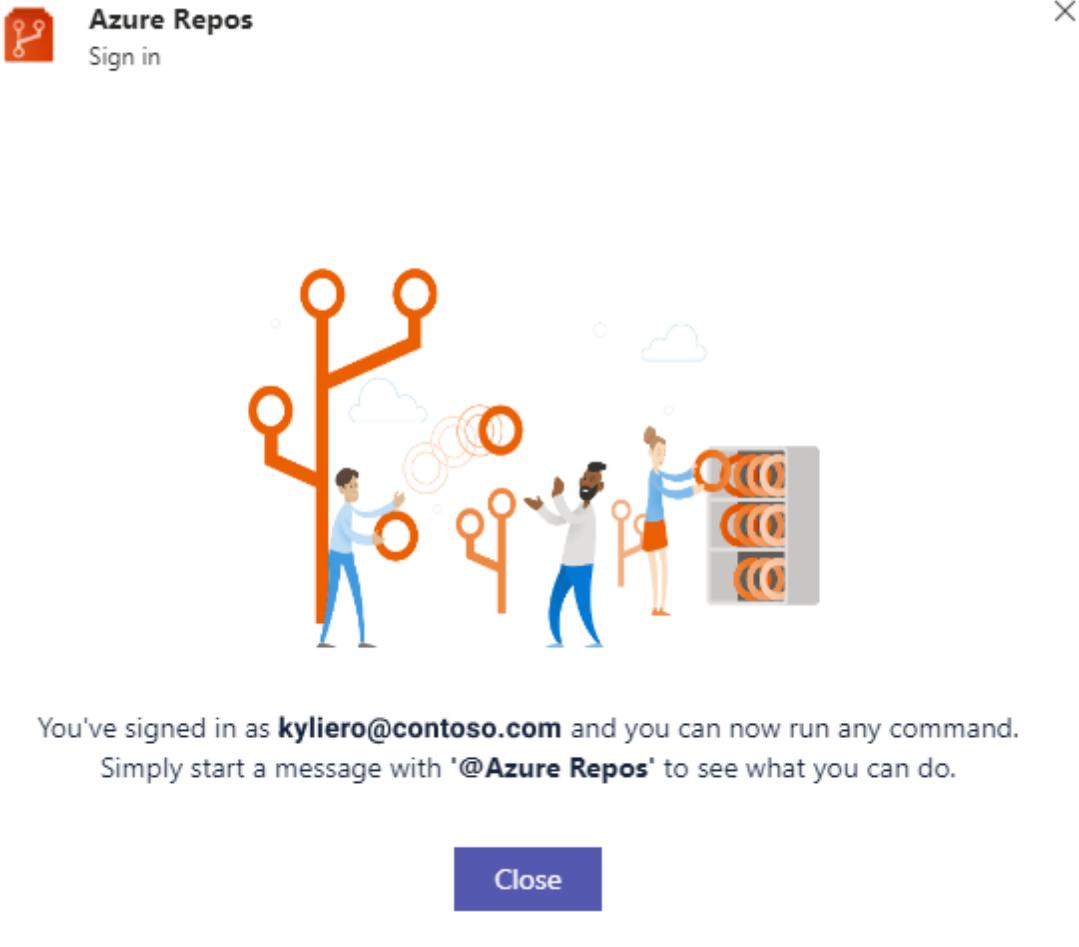


## Connect the Azure Repos app to your repos

1. Once the app is installed in your team, enter the following text into the reply field:  
`@azure repos signin`

If your Microsoft Teams and Azure Boards are in different tenants, select **Sign in with different email**.





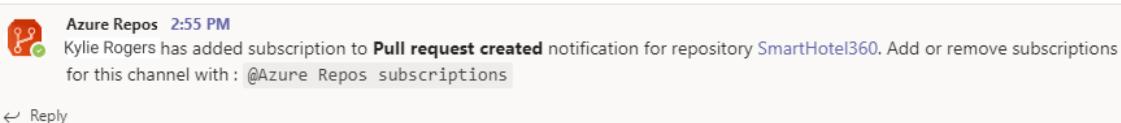
2. To monitor all Git repos in a project, enter `@azure repos subscribe [project url]` into the channel. Be sure to add your project URL. The project URL can be to any page within your project (except URLs to repos).

You can also monitor a specific repo using: `@azure repos subscribe [repo url]`.

The repo URL can be to any page within your repo that has your repo name, for example, `@azure repos subscribe`

`https://dev.azure.com/myorg/myproject/_git/myrepo`, or for TFVC repos: `@azure repos subscribe https://dev.azure.com/myorg/myproject/_versionControl`.

3. The `subscribe` command gets you started with a default subscription. For Git repos, the channel gets subscribed to the **Pull request created** event (with target branch = main). For TFVC repos, the channel is subscribed to the **Code checked in** event.



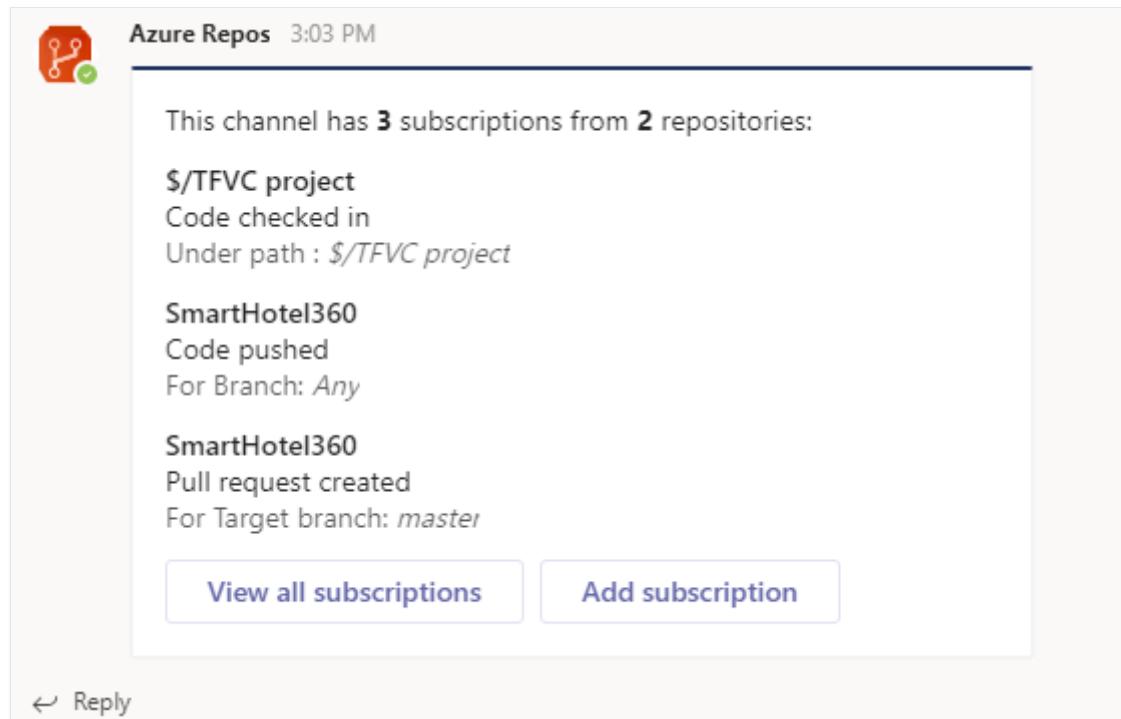
# Manage subscriptions

To view, add, and remove subscriptions for a channel, enter the following text: `@azure repos subscriptions`.

You see a list of all the current subscriptions for the channel and you can add new subscriptions or remove existing ones. Customize your notifications with various filters, as described in the following section.

## ⓘ Note

Team administrators can't remove or modify subscriptions created by Project administrators.



The screenshot shows a channel named "Azure Repos" at 3:03 PM. It displays a list of three subscriptions from two repositories:

- \$/TFVC project**  
Code checked in  
Under path : *\$/TFVC project*
- SmartHotel360**  
Code pushed  
For Branch: *Any*
- SmartHotel360**  
Pull request created  
For Target branch: *master*

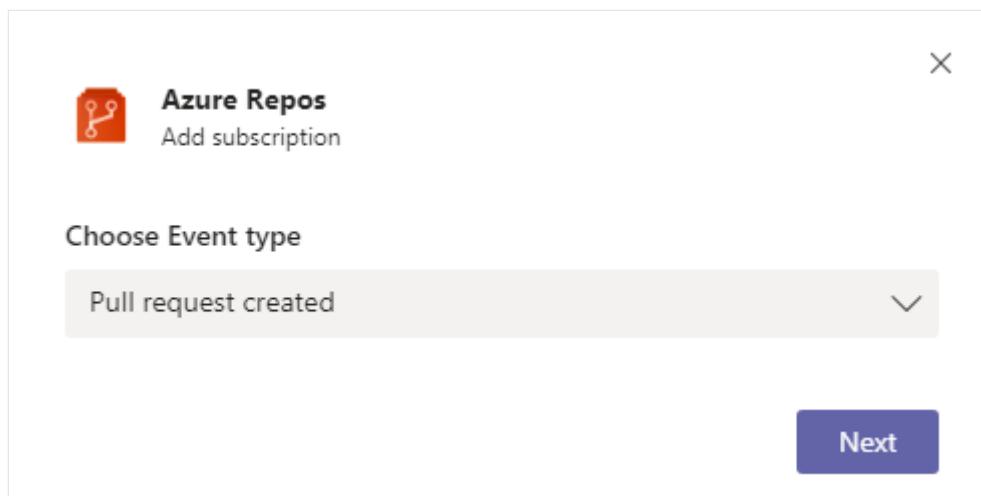
At the bottom, there are "View all subscriptions" and "Add subscription" buttons. A "Reply" button is located below the channel header.

## Use filters to get only notifications that you want

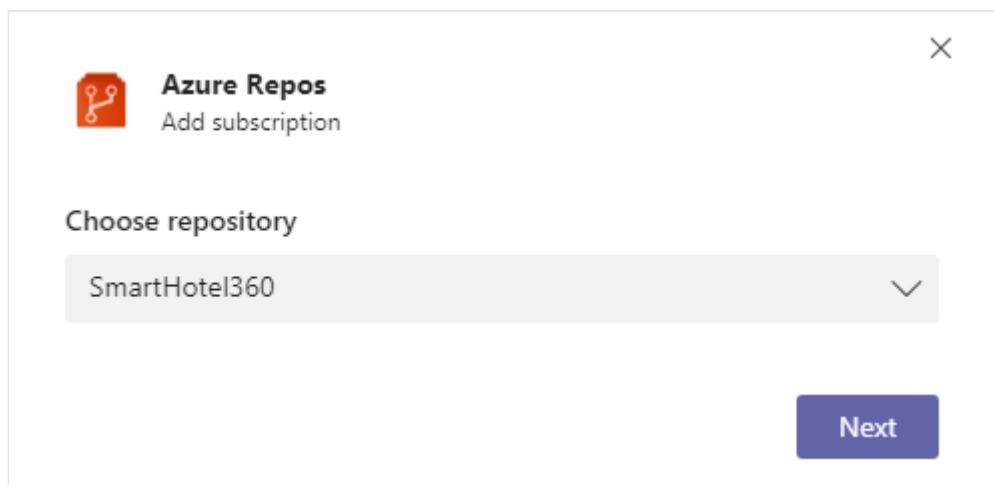
When you subscribe to a repo with `@azure repos subscribe`, a default subscription gets created with no filters applied. Often, users need to customize these subscriptions to be notified only when certain conditions are met. The following screenshots show an example of setting up notifications only when our team is in the reviewer list for a PR.

1. Enter the following text into your channel: `@azure repos subscriptions`.
2. In the list of subscriptions, if there's a subscription that you don't want or must be modified, select **Remove** to delete it.

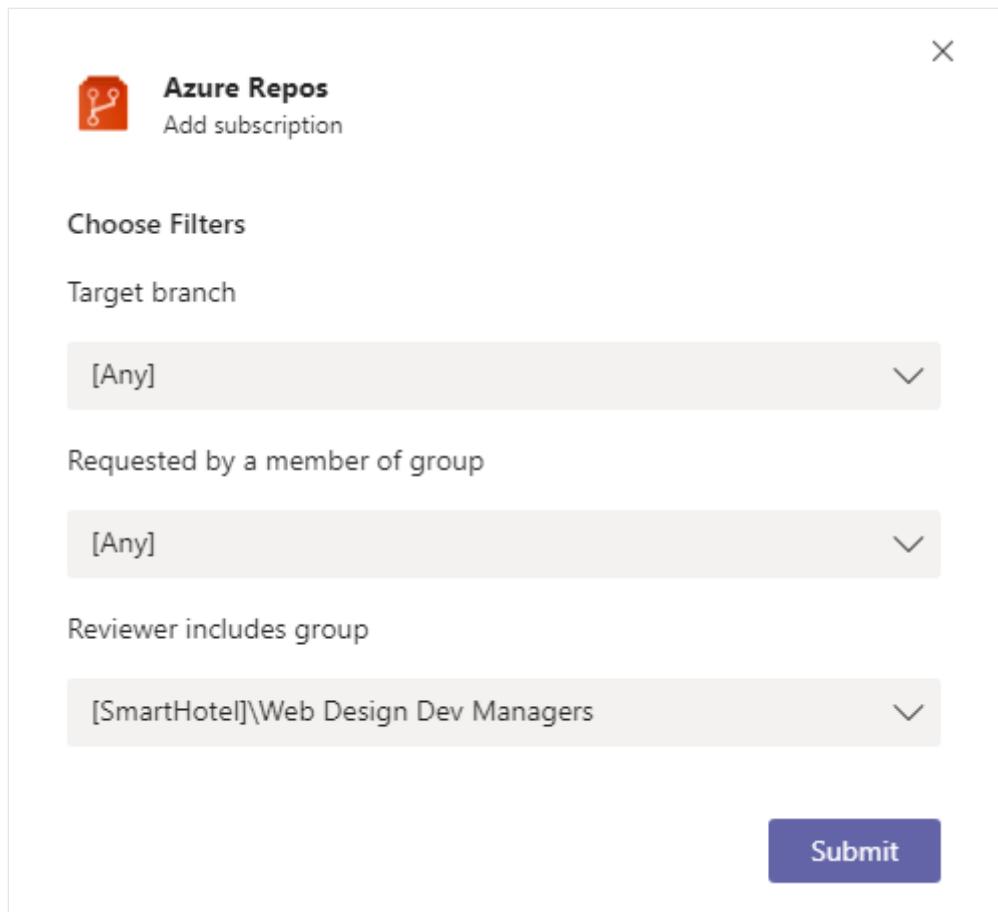
3. Select the **Add subscription** button.
4. Choose an event type, and then select **Next**.



5. Choose a repo, and then select **Next**.



6. Select the appropriate filters to customize your subscription, and then select **Submit**.

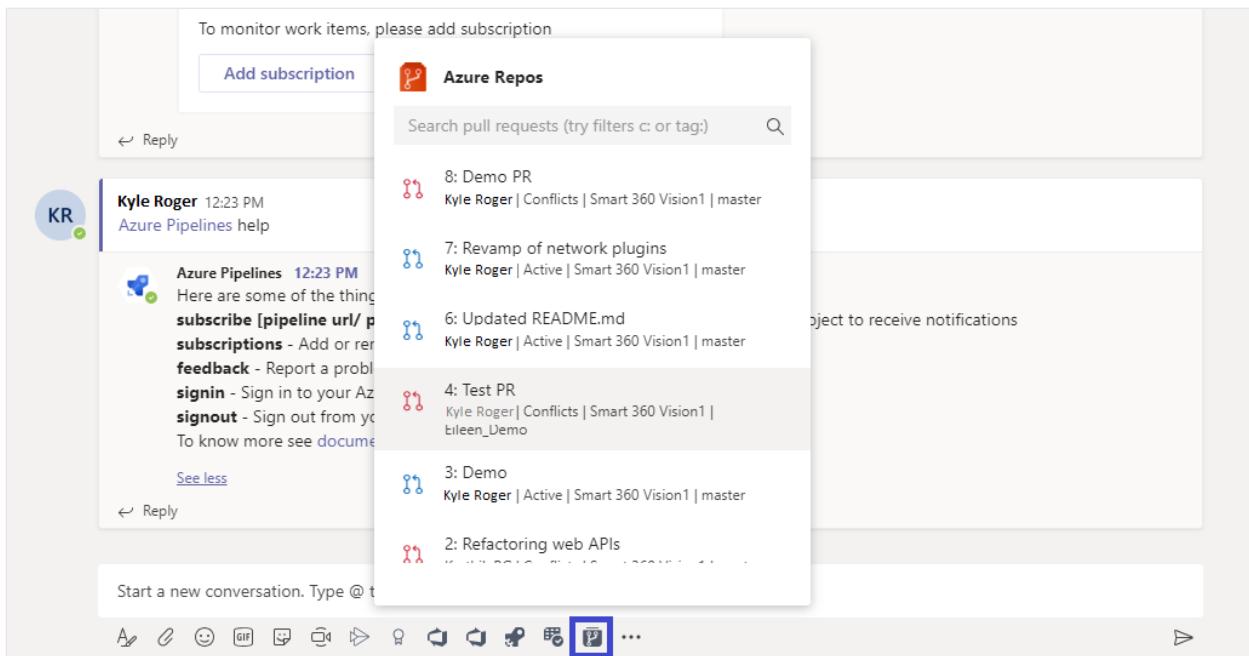


### ⓘ Note

- All the filters are typically drop-downs. However, if the drop-down has greater than 100 items, you must manually enter the values.
- For the **TFVC Code Checked in** event, the filter **Under path** must be of the format `$/myproject/path`.

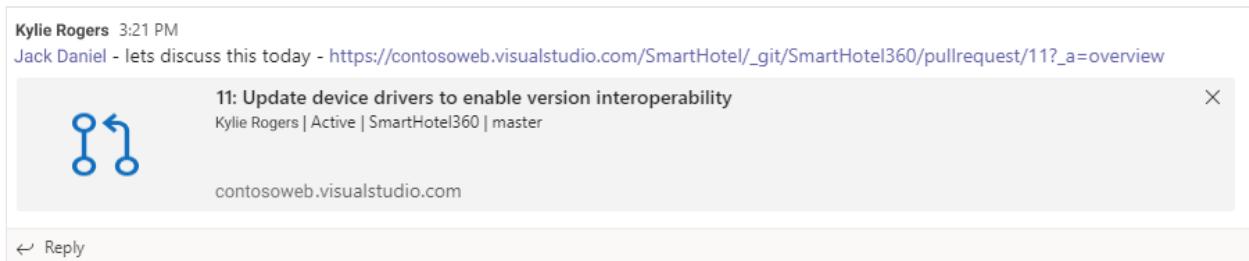
## Search and share pull request information using compose extension

To help users search and share information about pull requests, Azure Repos app for Microsoft Teams supports a compose extension. You can search for pull requests by ID or name. For the extension to work, sign in to the Azure Repos project by entering `@azure repos signin` or by signing into the compose extension directly.



## Preview pull request URLs

When you paste the URL of a PR, a preview shows like the one in the following image, which helps to keep PR-related conversations contextual and accurate. You must be signed in, and then you can preview PRs for URLs in all channels in a Team.



## Remove subscriptions and repos from a channel

To delete all the subscriptions related to any repo in the project and remove the repos from the channel, enter the following text into Teams: `@azure repos unsubscribe all [project url]`. Make sure to enter the project URL. Only project admins can do this task.

## Threaded notifications

To link a set of related notifications and also to reduce the space occupied by notifications in a channel, notifications get threaded. All notifications linked to a particular pull request are linked together.

## Compact view of threaded notifications

Azure Repos 6:46 PM

Kyle Roger created a pull request

 39: Updated network drivers to be compliant with latest firewall policies  
Active | chatopstest

Target branch: master  
Source branch: ThreadingDemo  
Reviewers: Gunther, [chatopstest]\chatopstest Team

2 replies from Azure Repos

Azure Repos 6:47 PM

Kyle Roger completed pull request

 39: Updated network drivers to be compliant with latest firewall policies  
Completed | chatopstest

Created by: Kyle Roger  
Target branch: master

← Reply

Compact  
Threaded  
Notifications

## Expanded view of threaded notifications

The screenshot shows a Microsoft Teams channel with a message from 'Azure Repos' at 6:46 PM. The message is: 'Kyle Roger created a pull request'. Below this, another message from 'Azure Repos' says: 'Merge attempt succeeded for pull request 39: Updated network drivers to be compliant with latest firewall policies in repository chatopstest'. A 'Collapse all' button is visible above the second message. The third message is from 'Gunther' and says: 'Gunther has approved and left suggestions in pull request'. It includes details about the pull request: '39: Updated network drivers to be compliant with latest firewall policies' (Active | chatopstest), 'Created by: Kyle Roger', and 'Target branch: master'. The fourth message is from 'Kyle Roger' and says: 'Completed pull request'. It also includes the same pull request details. At the bottom left of the message area, there is a 'Reply' button.

**Expanded view  
of Threaded  
Notifications**

## Command reference

The following table lists all the `azure repos` commands you can use in your Teams channel.

Command	Functionality
<code>@azure repos subscribe [repo url/ project url]</code>	Subscribe to a repo or all repos in a project to receive notifications
<code>@azure repos subscriptions</code>	Add or remove subscriptions for this channel
<code>@azure repos signin</code>	Sign in to your Azure Repos organization
<code>@azure repos signout</code>	Sign out from your Azure Repos organization

Command	Functionality
<code>@azure repos feedback</code>	Report a problem or suggest a feature
<code>@azure repos unsubscribe all [project url]</code>	Remove all repos (belonging to a project) and their associated subscriptions from a channel

## Multi-tenant support

If you're using a different email or tenant for Microsoft Teams and Azure DevOps, do the following steps to sign in, based on your use case.

Use case	Email ID + Teams tenant	Email ID + Azure DevOps tenant	Steps
1	email1@abc.com (tenant 1)	email1@abc.com (tenant 1)	Select <b>Sign in</b> .
2	email1@abc.com (tenant 1)	email1@abc.com (tenant 2)	Sign in to Azure DevOps. In the same browser, start a new tab and go to <a href="https://teams.microsoft.com/">https://teams.microsoft.com/</a> . Run the signin command and select <b>Sign in</b> .
3	email1@abc.com (tenant 1)	email2@pqr.com (tenant 2)	Select <b>Sign in with different email address</b> , and then in the email ID picker use the email2 to sign in to Azure DevOps.
4	email1@abc.com (tenant 1)	email2@pqr.com (non default tenant 3)	This scenario isn't supported.

## Troubleshoot

If you're experiencing the following errors when using the Azure Repos App, follow the procedures in this section.

- [Sorry, something went wrong. Please try again.](#)
- [Configuration failed. Please make sure that the organization '{organization name}' exists and that you have sufficient permissions.](#)

**Sorry, something went wrong. Please try again.**

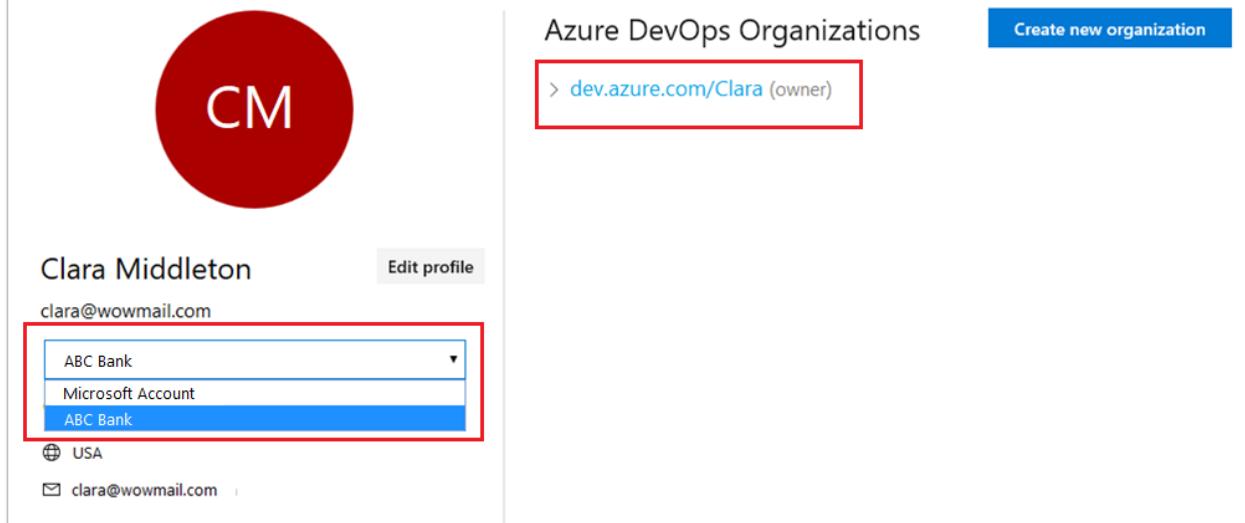
The Azure Repos app uses the OAuth authentication protocol, and requires [Third-party application access via OAuth for the organization](#) to be enabled. To enable this setting, navigate to **Organization Settings > Security > Policies**, and set the **Third-party application access via OAuth for the organization** setting to **On**.

The screenshot shows the Azure DevOps Organization Settings page for 'ABC Domestic Banking'. The left sidebar has 'Organization Settings' (1) selected. Under 'Security' (2), 'Policies' (3) is selected. The main content area is titled 'Policy' and contains sections for 'Application connection policies' and 'Security policies'. In 'Application connection policies', 'Alternate authentication credentials' is set to 'Off' and 'Third-party application access via OAuth' is highlighted with a red box and set to 'Off'. In 'Security policies', 'Allow public projects' is set to 'On' and 'Enable Azure Active Directory Conditional Access Policy Validation' is set to 'Off'. In 'User policies', 'External guest access' is set to 'On'.

**Configuration failed. Please make sure that the organization '{organization name}' exists and that you have sufficient permissions.**

Sign out of Azure DevOps by navigating to <https://aka.ms/VsSignout> using your browser.

Open an **In private or incognito** browser window and navigate to <https://aex.dev.azure.com/me> and sign in. In the dropdown under the profile icon to the left, select the directory that contains the organization containing the repository to which you wish to subscribe.



Azure DevOps Organizations [Create new organization](#)

> dev.azure.com/Clara (owner)

Clara Middleton [Edit profile](#)

clara@wowmail.com

ABC Bank

Microsoft Account

ABC Bank

USA

clara@wowmail.com

In the **same browser**, start a new tab and sign in to <https://teams.microsoft.com/>. Run the `@Azure Repos signout` command and then run the `@Azure Repos signin` command in the channel where the Azure Repos app for Microsoft Teams is installed.

Select the `Sign in` button and you're redirected to a consent page like the one in the following example. Ensure that the directory shown beside the email is same as what was chosen in the previous step. Accept and complete the sign-in process.

## Azure Repos Microsoft Teams Integration by Azure DevOps

App requests the following permissions from **kyliero@contoso.com (Contoso Corp)**

### Code (full)

Grants full access to source code, access metadata about commits, changesets, branches, and other version control artifacts. Also grants the ability to create and manage code repositories, create and manage pull requests and code reviews, and to receive notifications about version control events via service hooks.

### Project and team (read)

Grants the ability to read projects and teams.

### Identity Picker (read)

Grants the ability to interact with federated identities.

### Teams Integration

Grants this application the ability to act on your behalf on Azure DevOps from within Microsoft Teams

[Learn more](#)

If you change your mind at any time, you can manage authorizations on your [profile page](#).

**Accept**

**Deny**

By clicking **Accept**, you allow this app to perform the above actions on your behalf and you agree to Azure DevOps Terms of Use and Privacy Statement.

If these steps don't resolve your authentication issue, reach out to us at [Developer Community](#).

## Related articles

- [Azure Boards with Teams](#)
- [Azure Pipelines with Teams](#)

# Use Azure Pipelines with Slack

08/18/2025

## Azure DevOps Services

This article shows you how to use the [Azure Pipelines app for Slack](#) to monitor your pipeline events. You can establish and manage subscriptions for pipeline events like builds, releases, and pending approvals. Notifications for these events are delivered directly to your Slack channels.

### ⓘ Note

This feature is only available on Azure DevOps Services. Typically, new features are introduced in the cloud service first, and then made available on-premises in the next major version or update of Azure DevOps Server. For more information, see [Azure DevOps Feature Timeline](#).

## Prerequisites

- A Slack account with permission to install an app to your Slack workspace.
- An Azure DevOps project with **Project Collection Administrators** or **Project Administrators** permissions.
- Must not be an external user in Entra.

## Install the Azure Pipelines app

Install the [Azure Pipelines Slack app](#) to your Slack workspace. Once the app installs, you see the following welcome message. Enter `/azpipelines` to start interacting with the app.



Azure Pipelines APP 9:22 PM

👋 Hi! Let's start monitoring your pipelines

Subscribe to one or more pipelines with: `/azpipelines subscribe [pipeline url]`

To see what else you can do, type `/azpipelines help`

## Connect to your pipeline

Once the app is installed in your Slack workspace, you can connect the app to any pipeline you want to monitor. You must authenticate to Azure Pipelines before running any commands.

Only visible to you

 Azure Pipelines APP 12:04 PM

Looks like you are not signed in to Azure Pipelines. Please authenticate first.

[Sign in](#)

Complete sign-in by entering the verification code presented to you post authentication.

[Enter Code](#)

 Note

If your Azure DevOps organization is connected to an Entra ID tenant, you must sign in with a native member of that tenant. External users will see the following error if they try to sign in to the Azure Pipelines app: Configuration failed. Please make sure that the organization exists and that you have sufficient permissions.

## Subscribe to pipelines

To start monitoring all pipelines in a project, enter `/azpipelines subscribe <project url>` in a channel, replacing `<project url>` with your Azure DevOps project URL. The project URL can link to any page within your project except pipeline pages, for example `/azpipelines subscribe https://dev.azure.com/myorg/myproject/`.

You can monitor a specific pipeline by using `/azpipelines subscribe <pipeline url>`. The pipeline URL can link to any page within your pipeline that has a `definitionId` or a `buildId/releaseId` in the URL. For example:

- `/azpipelines subscribe https://dev.azure.com/myorg/myproject/_build?definitionId=123`
- `/azpipelines subscribe https://dev.azure.com/myorg/myproject/_release?definitionId=123&view=mine&_a=releases`

The `subscribe` command subscribes you to the following notifications by default:

- For YAML pipelines:
  - **Run stage state changed**
  - **Run stage waiting for approval**
- For Classic build pipelines, **Builds completed**
- For Classic release pipelines:
  - **Release deployment started**
  - **Release deployment completed**
  - **Release deployment approval pending**

Azure Pipelines APP 3:08 PM  
Add or remove subscriptions for this channel with: /azpipelines subscriptions  
@publisher has added subscriptions to Run stage state changed and Run stage waiting for approval notifications for pipeline publish to nuget.org

# Manage subscriptions

To manage the subscriptions for a channel, enter `/azpipelines subscriptions`. This command lists all the current subscriptions for the channel and lets you add or remove subscriptions.

Only visible to you

3:14 This channel has 6 subscriptions from 3 pipelines:

 **Anaconda (1): Anaconda**

Run stage state changed

For stage: *any*, state: *any*

[Remove](#)

 **Anaconda (1): Anaconda**

Run stage waiting for approval

For stage: *any*, environment: *any*

[Remove](#)

 **AzureContainerRegistryDemo: AzureContainerRegistryDemo**

Run stage state changed

For stage: *any*, state: *any*

[Remove](#)

 **AzureContainerRegistryDemo: AzureContainerRegistryDemo**

Run stage waiting for approval

For stage: *any*, environment: *any*

[Remove](#)

 **publish to nuget.org: Caching-NuGet**

Run stage state changed

For stage: *any*, state: *any*

[Remove](#)

 **publish to nuget.org: Caching-NuGet**

Run stage waiting for approval

For stage: *any*, environment: *any*

[Remove](#)

 Note

Team Administrators can't remove or modify subscriptions created by Project Administrators.

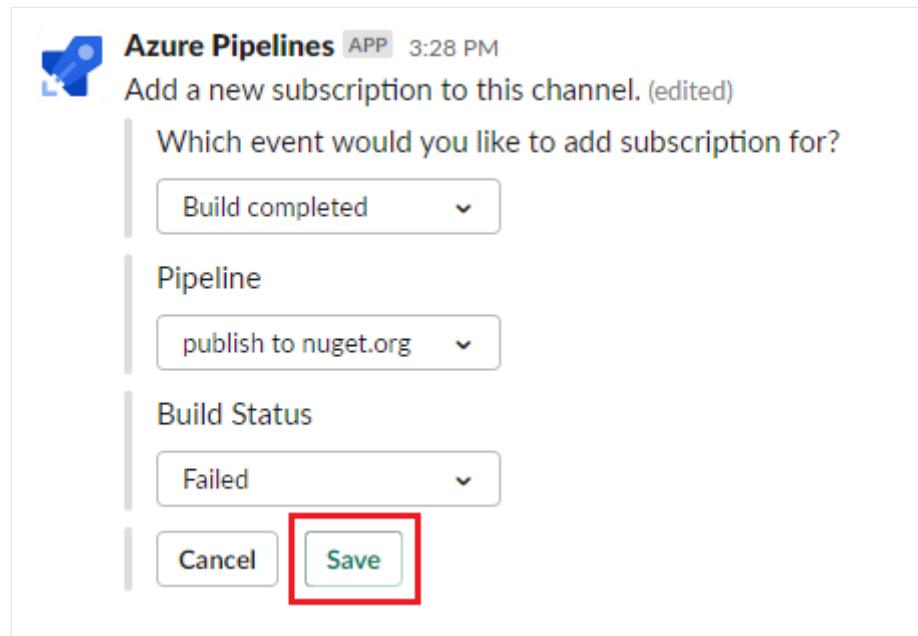
## Customize subscriptions

The default subscriptions don't have any filters applied, but you can customize these subscriptions according to your preferences. For instance, you might want to receive notifications only for failed builds or deployments to production. You can apply filters to customize which messages you receive in your channel.

To customize a subscription:

1. Run the `/azpipelines subscriptions` command to list all your subscriptions.
2. Select **Add subscription**.
3. Select the event you want to subscribe to, and then select your desired configuration.
4. Select **Save**.

For example, to get notifications only for failed builds, select **Failed** under **Build status**.



## Approve deployments

You can approve deployments from within your Slack channel without going to Azure Pipelines. Subscribe to the **Run stage waiting for approval** notifications for YAML pipelines or the **Release deployment approval pending** notifications for Classic releases. Both of these subscriptions are created by default when you subscribe to a pipeline.

 Artist 6:55 PM  
@Releaser I have created a release to deploy hotfix for bug #1441

 Azure Pipelines APP 6:56 PM  
Pre-deployment approval pending for release Prod Release-6 on stage Linux\_Deployment

Hotfix deployment to address bug #1441 - global scorecard

Release pipeline	Artifacts
Integrated_Reportng	879767c0b
Branch	
main	
Approvers	
Releaser	

Approve Reject

The Azure Pipelines app for Slack lets you handle all the checks and approval scenarios that are available in the Azure Pipelines portal. These scenarios include single approver, multiple approvers, and team-based approval. You can approve requests either individually or on behalf of a team.

 Azure Pipelines APP 6:56 PM  
Pre-deployment approval pending for release Prod Release-6 on stage Linux\_Deployment

Hotfix deployment to address bug #1441 - global scorecard

Release pipeline	Artifacts
Integrated_Reportng	879767c0b
Branch	
main	
Approvers	
<span style="color: green;">✓ Releaser</span>	

## Remove all subscriptions

To declutter your channel, you can use the `/azpipelines unsubscribe all <project url>` command to unsubscribe from all pipelines in a project. For example, `/azpipelines unsubscribe all https://dev.azure.com/myorg/myproject.`

### Important

Only Project Administrators can run this command.

## Command reference

The Azure Pipelines app for Slack supports the following commands:

[Expand table](#)

Command	Description
/azpipelines subscribe <pipeline url or project url>	Subscribe to a pipeline or all pipelines in a project and receive notifications.
/azpipelines subscriptions	Add or remove subscriptions for this channel.
/azpipelines feedback	Report a problem or suggest a feature.
/azpipelines help	Get help on the commands.
/azpipelines signin	Sign in to your Azure Pipelines account.
/azpipelines signout	Sign out of your Azure Pipelines account.
/azpipelines unsubscribe all <project url>	Remove all project pipelines and their associated subscriptions from a channel.

## Notifications in private channels

The Azure Pipelines app can also help you monitor pipelines activity in your private channels. You need to invite the bot to your private channel by using `/invite @azpipelines`. Once you add the bot, you can configure and control your notifications the same way as for a public channel.

## Conditions and limitations

- You can use the Azure Pipelines app for Slack only with Azure DevOps Services.
- To set up the subscriptions, you must be an admin of the project containing the pipeline.
- Notifications aren't supported inside direct messages.
- Deployment approvals that have the **Revalidate identity of approver before completing the approval** policy applied aren't supported.
- To use the app, **Third party application access via OAuth** must be enabled in Azure DevOps Organization settings > Security > Policies.

## Troubleshooting

If you get the following errors when using the Azure Pipelines App for Slack, try the procedures in this section.

# Sorry, something went wrong. Please try again.

The Azure Pipelines app uses the OAuth authentication protocol, and requires [Third-party application access via OAuth](#) to be enabled. To enable this setting, navigate to [Organization Settings > Security > Policies](#), and enable [Third-party application access via OAuth](#).

Azure DevOps fabrikamprime / Settings / Policies

**Organization Settings**

Search Settings

**General**

- Overview
- Projects
- Users
- Billing
- Auditing
- Global notifications
- Usage
- Extensions
- Microsoft Entra

**Security**

Policies

Permissions

Boards

Process

**Policies**

**Application connection policies**

- Off Third-party application access via OAuth ↗
- Off SSH authentication ↗

**Security policies**

- On Log Audit Events ↗
- Off Allow public projects ↗
- On Enterprise access to projects
- On Additional protections when using public package registries ↗
- Off Enable IP Conditional Access policy validation ↗

**User policies**

## Configuration failed. Please make sure that the organization exists and that you have sufficient permissions.

If you're seeing this error, you're most likely a Guest user in the Entra tenant connected to your Azure DevOps organization. You'll need to have an Entra Member go through the configuration steps.

## Related articles

- [Azure Boards with Slack](#)
- [Azure Repos with Slack](#)

- Service hook for Azure DevOps with Slack

# Manage Microsoft 365 connectors and custom connectors

10/18/2024 Applies to: Microsoft Teams

Connectors in Microsoft Teams deliver content and service updates directly from third-party services into a Teams channel. By using connectors, users can receive updates from popular services such as Azure DevOps Services, Trello, Wunderlist, GitHub, and more. Connectors post these updates directly into the chat stream. This functionality makes it easy for all the team members to stay in sync and quickly receive the relevant information.

Teams and Microsoft 365 groups use connectors. You can use the same connectors in Teams and Microsoft Exchange.

Any team member can add a connector to a channel, if the team permissions allow it. The updates from the service, that the connector fetches information from, notifies all the team members.

## Important

Developers can't register new connectors on the [Connector developer portal](#).

## Update connectors URL

The [Teams connectors are transitioning](#) to a new URL to enhance security. During this transition, you may receive notifications to update your configured connector to use the new URL. We strongly recommend that you update your connector immediately to prevent any disruption to connector services.

This change is needed only for webhook-based Connectors such as Incoming Webhook and third-party connectors. The change isn't required for polling connectors such as RSS. You must update the URL for the connector to continue posting notifications into Teams after December 31, 2024.

## Tip

We recommend that you note the URL that you update. After it is updated, you can't view the previous URL anymore. An old URL can be useful to find and identify the systems still using it that you must update with the new URL.

To update the URL, follow these steps:

1. Go to the **Posts** tab in a Teams channel, select **Open channel details**, select **Manage channel**, select **Edit** under the **Connectors** option, and select **Configured** section. Check the existing connector connections on this page.

The screenshot shows the Microsoft Teams Channel Details page for a channel named "Test".

At the top, there is a header bar with icons for video, search, and more. Below it, the channel name "In this channel" and an "X" button are visible.

The main content area includes:

- A "People" section with a placeholder icon and a "See all" link.
- A "Description" section containing "Test team" and an "Edit description" link.
- An "Options" section with a "Find in channel" search bar and a "Manage channel" button, which is highlighted with a red box.
- A "Channel notifications" section with a bell icon.
- A navigation bar with tabs: TT (highlighted), Test (selected), Settings, and Analytics.
- A "Channel details" section with an "Edit" button.
- A "Moderation" section with a "General Channel:" dropdown. The option "Anyone can post messages" is selected (radio button is checked).
- A "Connectors" section with an "Edit" button, which is highlighted with a red box.

Search  All Sort by Popularity ▾

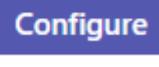
MANAGE  
Configured

My Accounts

CATEGORY  
All

Connectors for your team

 RSS Get RSS feeds for your group. 

 Incoming Webhook Send data from a service to your Office 365 group in real time. 

2. Update only those connections that display **Attention required** under the **Manage** option.

 Incoming Webhook Send data from a service to your Office 365 group in real time.   
**3 Configured ^**  
**Attention required**

alejandroRepro  
Added by: Pradeep Anantharaman 

test99  
Added by: Pradeep Anantharaman   
**Attention required**

3. Do one of the following:

- For connectors that contain a webhook URL, select **Manage** and **Update URL**.

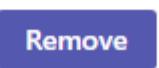
Copy the URL below to save it to the clipboard, then select Save. You'll need this URL when you go to the service that you want to send data to your group.

<https://microsoft.webhook.office.com/we>

By clicking the **Update URL** button, a new webhook URL will be generated and the existing URL will be deleted.

 Save

 Remove

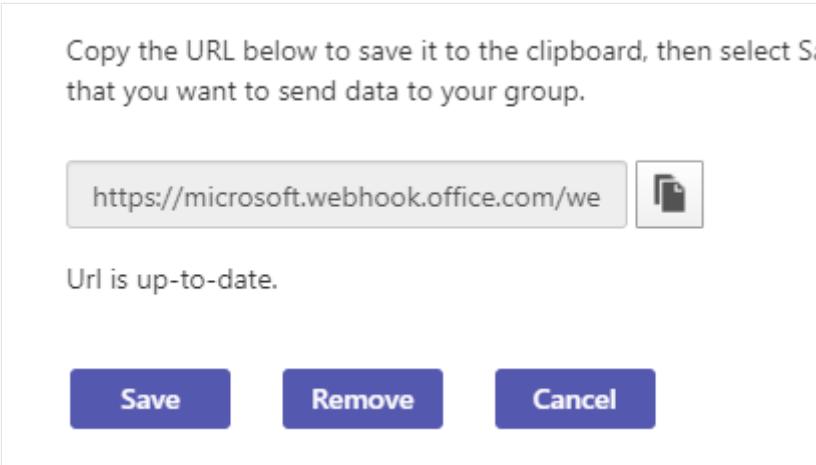
 Cancel

 Note

If the **Update URL** button is greyed out and you're unable to update the URL, create a new connector configuration.

- For other types of connectors, remove the connector and recreate the connector configuration.

4. Use the updated URL or the new connection in the systems that were posting to the old URL. The Configure page displays that the URL is updated.



To know more or to share more information with your app developers, see [Connectors deprecation information](#).

## Enable or disable connectors in Teams

The Exchange Online PowerShell v2 module uses modern authentication and works with multifactor authentication (MFA) to connect to all Exchange related PowerShell environments in Microsoft 365. Admins can use Exchange Online PowerShell to disable connectors for an entire organization or a specific group mailbox. If a connector is disabled, it affects all users in that org or mailbox. You can't disable a connector for a few specific users.

The organization setting overrides the group setting. For example, if an admin enables connectors for the group and disables the same connectors for the organization, then the connectors are disabled for the group. To enable a connector in Teams, [connect to Exchange Online PowerShell](#) using modern authentication with or without MFA.

To enable or disable a connector, execute the following commands in Microsoft Exchange Online PowerShell:

1. Open PowerShell as an administrator.
2. Use the command `Import-Module ExchangeOnlineManagement` to import the Microsoft Exchange module.

3. To disable connectors in your organization, use the command `Set-OrganizationConfig -ConnectorsEnabled:$false`.

4. To connect the admin account, use the command `Connect-ExchangeOnline -UserPrincipalName UPN -ExchangeEnvironmentName 0365USGovGCCHigh`. Replace `UPN` with your User Principal Name.

5. To enable connectors for Teams, use the following commands. To disable connectors or actionable messages, set the value to `false` instead of `true` in the following commands.

- `Set-OrganizationConfig -ConnectorsEnabled:$true`
- `Set-OrganizationConfig -ConnectorsEnabledForTeams:$true`
- `Set-OrganizationConfig -ConnectorsActionableMessagesEnabled:$true`

For more information about PowerShell module exchange, see [Set-OrganizationConfig](#). To enable or disable Outlook connectors, [connect apps to your groups in Microsoft Outlook](#). To know more about User Principal Name (UPN), see [what is UPN in Microsoft 365](#).

 **Note**

It takes up to 24 hours for these changes to propagate.

## Publish connectors for your organization

To make a custom connector available to your organization's users, upload a custom connector app to your org's app catalog. Users within the org can install, configure, and use the connector in a team.

 **Important**

Custom connectors are not available in Government Community Cloud (GCC), Government Community Cloud-High (GCCH), and Department of Defense (DOD) environments.

To use connectors in a team or a channel, open the More Options menu from the upper right corner of a channel. From the menu, select **Connectors** and then locate or search for the required connector. Configure the selected connector if necessary.

Keep your group current with content and updates from other services.

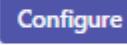
Search  All Sort by: Popularity ▾

MANAGE Connectors for your team

Configured Azure DevOps Collaborate on and manage software projects online. 

My Accounts RSS Get RSS feeds for your group. 

CATEGORY All

Analytics Incoming Webhook Send data from a service to your Office 365 group in real time. 

CRM GitHub Manage and collaborate on code projects. 

Customer Support BitBucket Manage and collaborate on your code projects. 

Developer Tools Azure DevOps Server \* New Share code. Track work. Ship software. 

HR Marketing

News & Social

Project Management Flipgrid Receive notifications when videos and Topics are created. 

Others

## Use connectors in GCC or GCCH

Connectors are disabled by default in the Government Cloud Community (GCC) and Government Community Cloud-High (GCCH) environments. To let your users use connectors in GCC or GCCH environments, follow these steps:

1. You must [enable connectors in Teams](#).
2. To set the parameters, connect to the [Exchange Online PowerShell](#).
3. To use an incoming webhook in Teams, create a custom app using the following [manifest.json](#). To use icons in the custom app, follow the [guidelines to create app icons](#).

```
JSON

{
  "$schema": "https://developer.microsoft.com/en-us/json-
schemas/teams/v1.5/MicrosoftTeams.schema.json",
  "manifestVersion": "1.5",
  "id": "203a1e2c-26cc-47ca-83ae-be98f960b6b2",
  "version": "1.0.0",
  "packageName": "com.incomingwebhook.microsoft",
  "developer": {
    "name": "Microsoft Corporation",
    "website": "https://github.com/microsoft/IncomingWebhook"
  }
}
```

```

    "websiteUrl": "https://go.microsoft.com/fwlink/?linkid=837668",
    "privacyUrl": "https://privacy.microsoft.com/privacystatement",
    "termsOfUseUrl": "https://www.microsoft.com/servicesagreement"
},
"description": {
    "full": "The Incoming Webhook connector enables external services to notify you about activities that you want to track.",
    "short": "Send data from a service to your Microsoft 365 group in real time."
},
"icons": {
    "outline": "outline.png",
    "color": "color.png"
},
"connectors": [
{
    "connectorId": "203a1e2c-26cc-47ca-83ae-be98f960b6b2",
    "scopes": ["team"]
}
],
"name": {
    "full": "Incoming Webhook",
    "short": "Incoming Webhook"
},
"accentColor": "#FFFFFF",
"permissions": ["identity", "messageTeamMembers"]
}

```

4. [Upload the custom app](#) in your Teams admin center.

## Considerations when using Connectors in Teams

- Connectors are disabled by default in the Government Cloud Community (GCC) environments and Government Community Cloud-High (GCCH). To enable connectors, set the `ConnectorsEnabled` or `ConnectorsEnabledForTeams` parameters to `$true` with the `SetOrganizationConfig` cmdlet. To set the parameters, connect to the [Exchange Online PowerShell](#).
- If the user who added a connector to a team leaves the team, the connector continues to work.
- You can't configure new connections for the following connectors:
  - Aha!
  - Airbrake
  - Aircall
  - App Links
  - AppSignal

- Beanstalk
- Bitbucket
- Buddy
- Buildkite
- CATS
- Chatra
- CircleCI
- CodeShip
- Constant Contact
- GetResponse
- Ghost Inspector
- Groove
- Heroku
- Honeybadger
- Intercom
- Logentries
- Mailchimp
- Microsoft Forms
- Opsgenie
- PagerDuty
- Papertrail
- Pivotal Tracker
- Raygun
- Runscope
- SatisMeter
- Semaphore
- Sentry
- Simple In/Out
- Stack Exchange
- SUBVERSION
- TestFairy
- Travis CI
- Trello
- Uptodown
- Userlike
- Wrike
- XP-Dev
- Zendesk

## Related articles

- Understand connectors and webhooks
- Create Microsoft 365 connectors

# Build secure applications with Azure DevOps

07/16/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Build secure, scalable applications and services that integrate with Azure DevOps to access user resources and automate processes programmatically. Whether you're creating internal automation tools or building commercial products, Azure DevOps provides robust APIs and modern authentication options to support your integration needs.

## Why integrate with Azure DevOps?

Azure DevOps integration enables you to:

### Automate workflows

- [Create and track bugs](#) automatically from customer reports
- [Monitor work items](#) and display status on custom dashboards
- Sync data between Azure DevOps and external systems
- Generate reports and analytics from Azure DevOps data

### Build commercial solutions

- Develop marketplace extensions for Azure DevOps customers
- Create SaaS products that integrate with Azure DevOps
- Build mobile apps that connect to Azure DevOps services
- Integrate Azure DevOps with enterprise systems

## Getting started: Choose your path

### Quick start options

 Expand table

Need	Recommended approach	Best for
Simple automation	<a href="#">REST API</a> with personal access tokens (PATs)	Scripts, personal tools

Need	Recommended approach	Best for
Production applications	.NET client libraries with managed identity	Enterprise apps, Azure-hosted services
Interactive applications	Microsoft Entra authentication	User-facing apps, desktop tools
Custom UI components	Azure DevOps extensions	Team customizations, marketplace products

## 🔒 Authentication: Security first

Choose the right authentication method:

 Recommended for production:

- [Managed Identity](#) - For Azure-hosted applications (most secure)
- [Service Principal](#) - For CI/CD pipelines and automated services
- [Microsoft Entra ID](#) - For user-facing applications requiring OAuth flows

 Quick development:

- [Personal Access Tokens \(PATs\)](#) - For testing and personal automation only

 Avoid for production:

- Username/password authentication (deprecated)
- Hardcoded credentials in source code
- Overly broad permission scopes

## Development approaches

### 🔌 REST API integration

**Best for:** Direct HTTP calls, platform-agnostic development, simple automation

markdown

**\*\*Key benefits:\*\***

- Works with any programming language
- Full control over HTTP requests and responses
- Lightweight integration for simple scenarios
- Easy to debug and test

**\*\*Get started:\*\***

- [Learn REST API basics](./how-to/call-rest-api.md)
- [Browse API reference](/rest/api/azure/devops/)
- [Try APIs in the browser](https://docs.microsoft.com/rest/api/azure/devops/)

## .NET client libraries

**Best for:** C# applications, enterprise development, complex integrations

markdown

### **\*\*Key benefits:\*\***

- Strongly typed APIs with IntelliSense support
- Built-in retry logic and error handling
- Async/await patterns for better performance
- Production-ready authentication options

### **\*\*Get started:\*\***

- [.NET client library samples](./get-started/client-libraries/samples.md)
- [Authentication guidance](./get-started/authentication/authentication-guidance.md)
- [Client library concepts](./concepts/dotnet-client-libraries.md)

## Event-driven integration

**Best for:** Real-time responses, webhook-based automation, external system synchronization

markdown

### **\*\*Key benefits:\*\***

- Real-time event notifications
- Reduced polling and improved efficiency
- Support for multiple event types
- Easy integration with external services

### **\*\*Get started:\*\***

- [Service hooks overview](../service-hooks/overview.md)
- [Webhook configuration guide](../service-hooks/services/webhooks.md)
- [Event reference documentation](../service-hooks/events.md)

## Platform extensions

**Best for:** Custom UI components, team-specific features, marketplace products

markdown

**\*\*Key benefits:\*\***

- Native integration with Azure DevOps UI
- Access to platform APIs and services
- Distribution through Visual Studio Marketplace
- Rich customization capabilities

**\*\*Get started:\*\***

- [Extension development overview]([../extend/overview.md](#))
- [Extension samples and tutorials]([../extend/develop/samples-overview.md](#))
- [Marketplace publishing guide]([../extend/publish/overview.md](#))

# Architecture patterns

## Recommended architectures

### Microservices integration:

markdown

Azure Function/App Service → Managed Identity → Azure DevOps APIs

- Secure, serverless integration
- Automatic credential management
- Scalable and cost-effective

### Enterprise application:

markdown

On-premises App → Service Principal → Azure DevOps REST APIs

- Certificate-based authentication
- Centralized credential management
- Audit logging and compliance

### User-facing application:

markdown

Web/Mobile App → Microsoft Entra OAuth → Azure DevOps on behalf of user

- User consent flows
- Secure token management
- Granular permission control

# Security and compliance

# Security best practices

## Authentication security:

- Use managed identities when possible
- Implement proper token refresh logic
- Apply principle of least privilege
- Enable audit logging for all API calls
- Never commit credentials to source control
- Don't use overly broad PAT scopes

## Application security:

- Implement proper error handling and logging
- Use HTTPS for all communications
- Validate all input data
- Handle rate limiting gracefully
- Store sensitive data in Azure Key Vault

## Compliance considerations:

- Review [Azure DevOps security overview](#)
- Understand data residency requirements
- Implement proper access controls and auditing
- Follow industry-specific compliance guidelines

# Resources and next steps

## Essential documentation

### Core concepts:

- [Authentication guidance](#) - Choose the right auth method
- [Microsoft Entra integration](#) - OAuth and modern auth patterns
- [Integration best practices](#) - Production-ready development patterns

### API references:

- [Azure DevOps REST API](#) - Complete API documentation
- [.NET client libraries](#) - Managed client library information
- [Service hooks reference](#) - Event-driven integration

### Code samples:

- [.NET client samples](#) - Production-ready C# examples
- [Azure DevOps auth samples](#) ↗ - Authentication examples
- [Extension samples](#) - Platform extension examples

## Quick actions

Start building today:

1. [Set up authentication](#) - Choose your auth method
2. [Try the REST API](#) - Make your first API call
3. [Run client library samples](#) - See working code examples
4. [Review security practices](#) - Build securely from the start

Need help?

- [Azure DevOps Developer Community](#) ↗ - Ask questions and get help
- [Stack Overflow](#) ↗ - Community support and examples
- [GitHub samples repository](#) ↗ - Working code examples

### Tip

New to Azure DevOps integration? Start with the [authentication guidance](#) to understand your options, then try the [REST API quickstart](#) to make your first successful API call.

 **Note:** The author created this article with assistance from AI. [Learn more](#)

# Rate and usage limits

09/15/2025

## Azure DevOps Services

Azure DevOps Services uses multi-tenancy to reduce costs and improve performance. This design can cause performance issues or outages when other users of shared resources have spikes in consumption. To help prevent this, Azure DevOps limits the resources each user can consume, and the number of requests they can make to certain commands. If you exceed these limits, future requests can be delayed or blocked.

Learn more in [Git limits](#) and [Best practices to avoid hitting rate limits](#).

## Global consumption limit

Azure DevOps has a global consumption limit that delays requests from individual users when shared resources are at risk of being overwhelmed. This limit helps avoid outages when shared resources are close to being overwhelmed. Individual users typically experience delayed requests only when one of the following incidents occurs:

- One of their shared resources is at risk of being overwhelmed.
- Their personal usage exceeds 200 times the consumption of a typical user within a sliding five-minute window.

The delay depends on the user's sustained level of consumption. Delays range from a few milliseconds per request up to 30 seconds. When consumption drops to zero or the resource isn't overwhelmed, the delays stop within five minutes. If consumption stays high, delays can continue indefinitely to protect the resource.

When a user request is delayed by a significant amount, the user receives an email and a warning banner in the web. For the build service account and others without an email address, members of the Project Collection Administrators group receive the email. For more information, see [Usage monitoring](#).

When an individual user's requests are blocked, the user receives responses with HTTP code 429 (too many requests) and a message similar to the following:

text

TF400733: The request has been canceled: Request was blocked due to exceeding usage of resource <resource name> in namespace <namespace ID>.

# Azure DevOps throughput units

Azure DevOps users consume many shared resources, and the level of consumption depends on factors like:

- Uploading a large number of files to version control, which puts load on databases and storage accounts.
- Running complex work item queries, which increases database load based on the number of work items being searched.
- Running builds, which download files from version control and produce log output.
- General operations, which consume CPU and memory across different parts of the service.

To measure this activity, Azure DevOps expresses resource consumption in **Azure DevOps throughput units (TSTUs)**. A TSTU is an abstract unit of load that represents a blend of different resources, including:

- Database usage—measured primarily through Azure SQL Database DTUs.
- Compute usage—CPU, memory, and I/O from application tiers and job agents.
- Storage usage—Azure Storage bandwidth.

## (!) Note

TSTUs are intentionally abstract. They aggregate resource consumption across compute, storage, and database layers within a distributed infrastructure. The underlying metrics (CPU, memory, I/O, DTUs) aren't directly exposed or meaningful on their own. TSTUs provide a unified way to represent load, making it easier to manage and monitor usage without exposing the full complexity of individual resource components. You can't calculate usage in TSTUs for an action with a formula, but you can see how many TSTUs an operation consumes on the [usage monitoring](#) page. Some operations, like work item queries, vary in consumption as your organization grows and changes, so you might need to benchmark periodically to stay accurate.

Currently, TSTUs focus primarily on Azure SQL Database DTUs because databases are the shared resource most likely to be overwhelmed by excessive consumption.

- One TSTU represents the average load generated by a typical Azure DevOps user over five minutes.
- Normal user activity can generate spikes of 10 TSTUs or fewer per five minutes.
- Larger but less frequent spikes can reach up to 100 TSTUs.
- The global limit is 200 TSTUs within any sliding five-minute window.

## Best practices

- Honor the Retry-After header: If you receive it in a response, wait the specified time before sending another request. The response still returns HTTP 200, so retry logic isn't required.
- Monitor X-RateLimit headers: If available, track `X-RateLimit-Remaining` and `X-RateLimit-Limit` to approximate how quickly you're approaching the threshold. This lets your client smooth out request bursts and avoid enforced delays.

#### ⓘ Note

Identities used by tools and applications to integrate with Azure DevOps can occasionally need higher rate and usage limits beyond the allowed consumption limit. Increase these limits by assigning the [Basic + Test Plans](#) access level to the identities your application uses. After you no longer need higher rate limits, revert to the previous access level. You're charged for the [Basic + Test Plans](#) access level only for the duration assigned to the identity. Identities already assigned a Visual Studio Enterprise subscription can't be assigned the [Basic + Test Plans](#) access level until you remove the subscription.

## Pipelines

Rate limiting works the same way for Azure Pipelines. Each pipeline is an individual entity, and its resource consumption is tracked separately. Even if build agents are self-hosted, they generate load by cloning and sending logs.

There's a 200 TSTU limit for each pipeline in a sliding 5-minute window. This limit matches the global consumption limit for users. If rate limiting delays or blocks a pipeline, you see a message in the attached logs.

## API client experience

When requests are delayed or blocked, Azure DevOps returns response headers to help API clients react. While not fully standardized, these headers are [broadly in line with other popular services](#).

The following table lists the available headers and what they mean. Except for `X-RateLimit-Delay`, all these headers are sent before requests start getting delayed. This design lets clients proactively slow down their rate of requests.

### Header name

### Description

---

`Retry- A custom header that shows the service and type of threshold reached. Threshold types and service names can vary over time and without warning. Display this string to a human, but don't rely on it for computation.

---

#### X-RateLimit-Delay

How long the request is delayed. Units: seconds with up to three decimal places (milliseconds).

---

#### X-RateLimit-Limit

Total number of TSTUs allowed before delays are imposed.

---

#### X-RateLimit-Remaining

Number of TSTUs remaining before delays start. If requests are already delayed or blocked, it's 0.

---

#### X-RateLimit-Reset

Time when, if all resource consumption stops immediately, tracked usage returns to 0 TSTUs. Expressed in Unix epoch time.

---

## Work tracking, process, & project limits

Azure DevOps limits the number of projects you can have in an organization and the number of teams you can have in each project. There are also limits for work items, queries, backlogs, boards, dashboards, and more. For more information, see [Work tracking, process, and project limits](#).

## Wiki

In addition to the usual [repository limits](#), a wiki file in a project can be up to 25 MB.

# Service connections

There aren't any per-project limits on creating service connections. However, limits might be imposed through Microsoft Entra ID. For more information, see the following articles:

- [Microsoft Entra service limits and restrictions](#)
- [Azure subscription and service limits, quotas, and constraints](#)

## Related content

- [Track work, manage processes, and set project limits](#)
- [Configure and customize Azure Boards](#)
- [Monitor usage](#)
- [Understand Git limits](#)
- [Follow best practices for avoiding hitting rate limits](#)

# Deprecation of the Work Item Tracking (WIT) and Test Client OM

Article • 02/24/2023

## Azure DevOps Services

WIT and Test Client OM are part of the broader TFS SDK. They are a set of SOAP-based APIs that can be used to create tools to run operations against the Azure DevOps. These SOAP-based APIs have been replaced by the set of modern REST-based endpoints. Therefore we are in the process of deprecating the SOAP-based APIs (Client OM) for WIT and Test.

### Note

WIT and Test Client OM for Azure DevOps has been deprecated in latest version Visual Studio 2019 and the [Microsoft.TeamFoundationServer.ExtendedClient](#) package. This means, there's no new work item tracking or test functionality included into the Client OM.

## Affected object models

Only the WIT and Test Client OM are being deprecated. Other areas like build and source control aren't on a deprecation schedule at this time.

## Impact

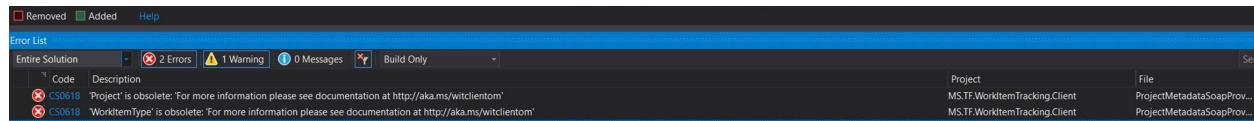
If your organization is using the WIT Client OM in tools that run against Azure DevOps, we recommend you start migrating those tools to stop using the WIT Client OM and start using the new [REST](#) based endpoints.

Updating your code to use REST is vital to ensure your tools work as we release new functionality into Azure Boards and Azure Test.

## Visual Studio warnings

Any references to the latest versions of WIT Client OM result in an obsolete warning. This includes the dlls that come with Visual Studio 2019 and Azure DevOps 2019. Here is an example warning you may receive...

CS0618: 'WorkItemType' is obsolete: 'for more information, see documentation at <https://aka.ms/witclientom>'



## Q&A

### How do I fix my tools?

First thing you should investigate and determine if the tool is still needed. We have found that many organizations create a tool for a specific reason for a short period of time. Then for one reason or another, that tool continues to run when it's no longer needed. For this scenario, you can shut down the tool.

If the tool is still of use, you need to re-refactor that tool using the REST based [.NET client libraries](#). Refactoring requires some reaction work on your part to remove references to the old WIT Client OM and replace them with the appropriate .NET Client Libraries.

We have also put together this handy [migration guide](#) with [sample code](#) to aid your transition from WIT Client OM to the REST-based .NET Client libraries.

## Resources

- [Migration Guide](#)
- [Azure DevOps Services REST API Reference](#)
- [.NET client libraries for Azure DevOps](#)
- [Understanding Rate limits](#)
- [Integration Best Practices](#)
- [Sample Code on GitHub](#)

# Migrate from WIT Client OM to REST APIs

07/16/2025

Azure DevOps Services

## Important

### Legacy technology replacement required

The [WIT Client OM](#) (Work Item Tracking Client Object Model) is legacy technology that should be replaced with modern [REST-based .NET client libraries](#). Migration provides better performance, security, and cross-platform support.

This guide helps you migrate your .NET code from the deprecated WIT Client OM to modern REST APIs. The migration offers significant benefits:

#### Modern advantages:

- Asynchronous operations for better performance
- Modern authentication with managed identities and service principals
- Cross-platform support (.NET Core, .NET 5+, and .NET Framework)
- Active development and ongoing support

#### Legacy limitations:

- Limited to .NET Framework and Windows only
- Synchronous, blocking operations
- Outdated authentication methods

## Migration overview

Step 1: Update NuGet packages - Replace WIT Client OM with modern REST client packages

Step 2: Update authentication - Migrate to secure, modern authentication methods

Step 3: Convert operations - Replace synchronous calls with asynchronous REST operations

For detailed code examples and step-by-step migration samples, see the [GitHub Azure DevOps WIT Client OM Migration Guide](#).

## Common scenarios migration table

The following table shows how to migrate common work item operations from legacy WIT Client OM to modern REST APIs:

[ ] Expand table

Scenario	Legacy WIT Client OM	Modern REST API
Get list of work items	<a href="#">WorkItemStore.Query</a>	<a href="#">Work Items - List</a>
Get single work item	<a href="#">WorkItemStore.GetWorkItem</a>	<a href="#">Work Items - Get Work Item</a>
Create new work item	<a href="#">WorkItem</a>	<a href="#">Work Items - Create</a>
Update existing work item	<a href="#">WorkItem.Fields</a>	<a href="#">Work Items - Update</a>
Validate a work item	<a href="#">WorkItem.IsValid()</a> , <a href="#">WorkItem.Validate()</a>	<a href="#">Work Items - Update (validate only)</a>
Create a link to an existing work item	<a href="#">WorkItem.WorkItemLinks.Add</a>	<a href="#">Work Items - Update (add link)</a>
Add a comment	<a href="#">WorkItem.History</a>	<a href="#">Work Items - Update (add comment)</a>
Create a hyperlink	<a href="#">WorkItem.Links.Add()</a>	<a href="#">Work Items - Update (add hyperlink)</a>
Add an attachment	<a href="#">WorkItem.Attachments.Add()</a>	<a href="#">Work Items - Update (add attachment)</a>
Query work items using WIQL	<a href="#">WorkItemStore.Query()</a>	<a href="#">Wiql - Query by Wiql</a>
Run an existing query to get work items	<a href="#">WorkItemStore.Query()</a>	<a href="#">Wiql - Query by ID</a>
Get list of work item types for project	<a href="#">Category.WorkItemTypes</a>	<a href="#">Work Item Types - List</a>
Get work item type details	<a href="#">Category.WorkItemTypes</a>	<a href="#">Work Item Types - Get</a>
Get list of fields for a work item type	<a href="#">WorkItemType.FieldDefinitions</a>	<a href="#">Work Item Types Field - List</a>
Get field details	<a href="#">WorkItemType.FieldDefinitions</a>	<a href="#">Work Item Types Field - Get</a>

## Authentication migration

Legacy authentication (X Replace):

C#

```
// WIT Client OM with basic authentication
using (var tpc = new TfsTeamProjectCollection(new Uri(collectionUri)))
{
    tpc.Authenticate();
    var workItemStore = tpc.GetService<WorkItemStore>();
}
```

Modern authentication (✓ Recommended):

C#

```
// REST client with managed identity (for Azure-hosted apps)
var credentials = new VssAzureIdentityCredential();
using var connection = new VssConnection(new Uri(collectionUri), credentials);
var witClient = connection.GetClient<WorkItemTrackingHttpClient>();

// Alternative: Service principal for CI/CD
// var credentials = new VssServicePrincipalCredential(clientId, clientSecret,
// tenantId);

// Alternative: PAT for development/testing
// var credentials = new VssBasicCredential(string.Empty, personalAccessToken);
```

## Next steps and resources

### Essential migration resources

- [Modern .NET client library samples](#) - Production-ready code examples with modern authentication
- [Authentication guidance](#) - Choose the right authentication method for your scenario
- [.NET client libraries concepts](#) - Understanding the modern client architecture

### Code examples and tools

- [Migration guide with code samples](#) - GitHub repository with detailed migration examples
- [Work Item Tracking REST API documentation](#) - Complete API reference with examples

### Support and community

- [Azure DevOps Developer Community](#) - Ask questions and get help
- [Migration guide issues](#) - Report missing scenarios or get specific help

## Related migrations

- [Migrate data from Azure DevOps Server to Azure DevOps Services](#) - Service migration guidance
- [Legacy SOAP client samples](#) - Reference for other legacy client patterns

### Tip

Start your migration: Begin with [authentication guidance](#) to choose the right approach, then see [.NET client library samples](#) for working code examples.

 **Note:** The author created this article with assistance from AI. [Learn more](#)

# Azure DevOps Services REST API Reference

Article • 02/26/2025

Welcome to the Azure DevOps Services/Azure DevOps Server REST API Reference.

Representational State Transfer (REST) APIs are service endpoints that support sets of HTTP operations (methods), which provide create, retrieve, update, or delete access to the service's resources. This article walks you through:

- The basic components of a REST API request/response pair.
- Overviews of creating and sending a REST request, and handling the response.

Most REST APIs are accessible through our [client libraries](#), which can be used to greatly simplify your client code.

## Components of a REST API request/response pair

A REST API request/response pair can be separated into five components:

1. The **request URI**, in the following form: `VERB https://{{instance}}[/{{team-project}}]/_apis[/{{area}}]/{{resource}}?api-version={{version}}`

- *instance*: The Azure DevOps Services organization or TFS server you're sending the request to. They are structured as follows:
  - Azure DevOps Services: `dev.azure.com/{{organization}}`
  - TFS: `{{server:port}}/tfss/{{collection}}` (the default port is 8080, and the value for collection should be `DefaultCollection` but can be any collection)
- *resource path*: The resource path is as follows: `_apis/{{area}}/{{resource}}`. For example `_apis/wit/workitems`.
- *api-version*: Every API request should include an api-version to avoid having your app or service break as APIs evolve. api-versions are in the following format: `{{major}}.{{minor}}[-{{stage}}[.{{resource-version}}]]`, for example:
  - `api-version=1.0`
  - `api-version=1.2-preview`
  - `api-version=2.0-preview.1`

Note: *area* and *team-project* are optional, depending on the API request. Check out the [TFS to REST API version mapping matrix](#) below to find which REST API versions apply to your version of TFS.

2. HTTP request message header fields:

- A required [HTTP method](#) (also known as an operation or verb), which tells the service what type of operation you are requesting. Azure REST APIs support GET, HEAD, PUT, POST, and PATCH methods.
  - Optional additional header fields, as required by the specified URI and HTTP method. For example, an Authorization header that provides a bearer token containing client authorization information for the request.
3. Optional HTTP **request message body** fields, to support the URI and HTTP operation. For example, POST operations contain MIME-encoded objects that are passed as complex parameters.
- For POST or PUT operations, the MIME-encoding type for the body should be specified in the Content-type request header as well. Some services require you to use a specific MIME type, such as `application/json`.

#### 4. HTTP **response message header** fields:

- An [HTTP status code](#), ranging from 2xx success codes to 4xx or 5xx error codes. Alternatively, a service-defined status code may be returned, as indicated in the API documentation.
- Optional additional header fields, as required to support the request's response, such as a `Content-type` response header.

#### 5. Optional HTTP **response message body** fields:

- MIME-encoded response objects may be returned in the HTTP response body, such as a response from a GET method that is returning data. Typically, these objects are returned in a structured format such as JSON or XML, as indicated by the `Content-type` response header. For example, when you request an access token from Azure AD, it will be returned in the response body as the `access_token` element, one of several name/value paired objects in a data collection. In this example, a response header of `Content-Type: application/json` is also included.

## Create the request

### Authenticate

There are many ways to authenticate your application or service with Azure DevOps Services or TFS. The following table is an excellent way to decide which method is the best for you:

[ ] Expand table

Type of application	Description	example	Authentication mechanism	Code samples
Interactive client-side	Client application, that allows user interaction, calling <a href="#">Azure DevOps Services REST APIs</a>	Console application enumerating projects in an organization	<a href="#">Microsoft Authentication Library (MSAL)</a>	<a href="#">sample ↗</a>
Interactive JavaScript	GUI based JavaScript application	AngularJS single page app displaying project information for a user	<a href="#">MSAL</a>	<a href="#">sample ↗</a>
Non-interactive client-side	Headless text only client side application	Console app displaying all bugs assigned to a user	<a href="#">Device Profile</a>	<a href="#">sample ↗</a>
Interactive web	GUI based web application	Custom Web dashboard displaying build summaries	<a href="#">OAuth</a>	<a href="#">sample ↗</a>
TFS application	TFS app using the Client OM library	TFS extension displaying team bug dashboards	<a href="#">Client Libraries</a>	<a href="#">sample ↗</a>
<a href="#">Azure DevOps Services Extension ↗</a>	Azure DevOps Services extension	<a href="#">Azure DevOps extension samples ↗</a>	<a href="#">VSS Web Extension SDK ↗</a>	<a href="#">sample walkthrough ↗</a>

**Note:** You can find more information on authentication on our [authentication guidance page](#).

## Assemble the request

### Azure DevOps Services

For Azure DevOps Services, *instance* is `dev.azure.com/{organization}`, so the pattern looks like this:

```
VERB https://dev.azure.com/{organization}/_apis[/{area}]/ {resource}?api-version={version}
```

For example, here's how to get a list of team projects in a Azure DevOps Services organization.

```
dos
```

```
curl -u {username}[:{personalaccesstoken}]
https://dev.azure.com/{organization}/_apis/projects?api-version=2.0
```

If you wish to provide the personal access token through an HTTP header, you must first convert it to a Base64 string (the following example shows how to convert to Base64 using C#). (Some tools apply a Base64 encoding by default. If you are trying the API via such tools, Base64 encoding of the PAT is not required) The resulting string can then be provided as an HTTP header in the format:

```
Authorization: Basic BASE64PATSTRING
```

Here it is in C# using the [HttpClient class](/previous-versions/visualstudio/hh193681(v=vs.118)).

```
cs
```

```
public static async void GetProjects()
{
    try
    {
        var personalaccesstoken = "PAT_FROM_WEBSITE";

        using (HttpClient client = new HttpClient())
        {
            client.DefaultRequestHeaders.Accept.Add(
                new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));

            client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Basic",
Convert.ToBase64String(
    System.Text.Encoding.ASCII.GetBytes(
        string.Format("{0}:{1}", "", personalaccesstoken))));

            using (HttpResponseMessage response = await client.GetAsync(
                "https://dev.azure.com/{organization}/_apis/projects"))
            {
                response.EnsureSuccessStatusCode();
                string responseBody = await response.Content.ReadAsStringAsync();
                Console.WriteLine(responseBody);
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

```
}
```

Most samples on this site use Personal Access Tokens as they're a compact example for authenticating with the service. However, there are a variety of authentication mechanisms available for Azure DevOps Services including MSAL, OAuth and Session Tokens. Refer to the [Authentication](#) section for guidance on which one is best suited for your scenario.

## TFS

For TFS, `instance` is `{server:port}/tfss/{collection}` and by default the port is 8080. The default collection is `DefaultCollection`, but can be any collection.

Here's how to get a list of team projects from TFS using the default port and collection.

```
dos
```

```
curl -u {username}[:{personalaccesstoken}]
https://:{server}:8080/tfs/DefaultCollection/_apis/projects?api-version=2.0
```

The examples above use personal access tokens, which requires that you [create a personal access token](#).

## Process the response

You should get a response like this.

```
JSON
```

```
{
  "value": [
    {
      "id": "eb6e4656-77fc-42a1-9181-4c6d8e9da5d1",
      "name": "Fabrikam-Fiber-TFVC",
      "url": "https://dev.azure.com/fabrikam-fiber-
inc/_apis/projects/eb6e4656-77fc-42a1-9181-4c6d8e9da5d1",
      "description": "TeamFoundationVersionControlprojects",
      "collection": {
        "id": "d81542e4-cdfa-4333-b082-1ae2d6c3ad16",
        "name": "DefaultCollection",
        "url": "https://dev.azure.com/fabrikam-fiber-
inc/_apis/projectCollections/d81542e4-cdfa-4333-b082-1ae2d6c3ad16",
        "collectionUrl": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection"
      },
      "defaultTeam": {
        "id": "66df9be7-3586-467b-9c5f-425b29afedfd",
        "name": "Default Team"
      }
    }
  ]
}
```

```

        "name": "Fabrikam-Fiber-TFVCTeam",
        "url": "https://dev.azure.com/fabrikam-fiber-
inc/_apis/projects/eb6e4656-77fc-42a1-9181-4c6d8e9da5d1/teams/66df9be7-3586-467b-
9c5f-425b29afedfd"
    }
},
{
    "id": "6ce954b1-ce1f-45d1-b94d-e6bf2464ba2c",
    "name": "Fabrikam-Fiber-Git",
    "url": "https://dev.azure.com/fabrikam-fiber-
inc/_apis/projects/6ce954b1-ce1f-45d1-b94d-e6bf2464ba2c",
    "description": "Gitprojects",
    "collection": {
        "id": "d81542e4-cdfa-4333-b082-1ae2d6c3ad16",
        "name": "DefaultCollection",
        "url": "https://dev.azure.com/fabrikam-fiber-
inc/_apis/projectCollections/d81542e4-cdfa-4333-b082-1ae2d6c3ad16",
        "collectionUrl": "https://dev.azure.com/fabrikam-fiber-
inc/DefaultCollection"
    },
    "defaultTeam": {
        "id": "8bd35c5e-30bb-4834-a0c4-d576ce1b8df7",
        "name": "Fabrikam-Fiber-GitTeam",
        "url": "https://dev.azure.com/fabrikam-fiber-
inc/_apis/projects/6ce954b1-ce1f-45d1-b94d-e6bf2464ba2c/teams/8bd35c5e-30bb-4834-
a0c4-d576ce1b8df7"
    }
}
],
"count": 2
}

```

The response is [JSON](#). That's generally what you'll get back from the REST APIs although there are a few exceptions, like [Git blobs](#).

Now you should be able to look around the specific API areas like [work item tracking](#) or [Git](#) and get to the resources that you need. Keep reading to learn more about the general patterns that are used in these APIs.

## API and TFS version mapping

All supported Rest APIs are included in this section of the Azure DevOps documentation. Azure DevOps Service is compatible with the most recent Rest API version (including preview versions), as well as previous versions. We highly recommend that you use the latest release version of the REST API with Azure DevOps Service. If you are using Azure DevOps Server or Team Foundation Server (TFS), you will find the REST API versions and their corresponding Server version below. REST API versions are compatible with the Server version listed, as well as Server versions that are newer than the Server version listed. Please ensure the Documentation

being viewed is for a Rest API version that is compatible with the Server version you are using; it can be changed via the menu at the top of the table of contents.

[ ] Expand table

Server Version	REST API Version	Build Version
Azure DevOps Server vNext	7.2	
Azure DevOps Server 2022.1	7.1	versions >= 19.225.34309.2
Azure DevOps Server 2022	7.0	versions >= 19.205.33122.1
Azure DevOps Server 2020	6.0	versions >= 18.170.30525.1
Azure DevOps Server 2019	5.0	versions >= 17.143.28621.4
TFS 2018 Update 3	4.1	versions >= 16.131.28106.2
TFS 2018 Update 2	4.1	versions >= 16.131.27701.1
TFS 2018 Update 1	4.0	versions >= 16.122.27409.2
TFS 2018 RTW	4.0	versions >= 16.122.27102.1
TFS 2017 Update 2	3.2	versions >= 15.117.26714.0
TFS 2017 Update 1	3.1	versions >= 15.112.26301.0
TFS 2017 RTW	3.0	versions >= 15.105.25910.0
TFS 2015 Update 4	2.3	versions >= 14.114.26403.0
TFS 2015 Update 3	2.3	versions >= 14.102.25423.0
TFS 2015 Update 2	2.2	versions >= 14.95.25122.0
TFS 2015 Update 1	2.1	versions >= 14.0.24712.0
TFS 2015 RTW	2.0	versions >= 14.0.23128.0

## Related Content

Check out the [Integrate documentation](#) for REST API samples and use cases.

- [Authentication guidance](#)
- [Samples](#)

# Client Libraries

Discover the client libraries for these REST APIs.

- [.NET conceptual documentation](#) and [.NET reference documentation](#)
- [Go ↗](#)
- [Node.js ↗](#)
- [Python ↗](#)
- [Swagger 2.0 ↗](#)
- [Web Extensions SDK ↗](#)

## Where are the earlier versions of REST APIs? (Before 4.1)

We recently made a change to our engineering system and documentation generation process; we made this change to provide clearer, more in-depth, and more accurate documentation for everyone trying to use these REST APIs. Due to technical constraints, we are only able to document **API Version 4.1 and newer** using this method. We believe the documentation for API Version 4.1 and newer will be easier to use due to this change.

If you are working in TFS or are looking for the older versions of REST APIs, you can take a look at the [REST API Overview for TFS 2015, 2017, and 2018](#).

# Billing overview

03/05/2025

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

If you need more than the [free tier](#) of resources in your organization, [set up billing](#). When you set up billing you can buy other features offered by Microsoft or other companies.

## Services billed via Azure

During your first purchase for your organization, you're prompted to select the Azure subscription to use for billing. The subscription can be part of your Enterprise Agreement, pay-as-you-go, Cloud Solution Provider (CSP), or other types of Azure subscriptions. All services get billed via Azure. You aren't required to pay for or use any other Azure services.

### 💡 Tip

To estimate costs for Azure DevOps, see the [Pricing calculator](#) or the [Azure DevOps pricing page](#).

For more information, see the following articles about the paid services offered by Microsoft:

- [Basic access for users](#)
- [Basic + Test Plans](#)
- [Azure Pipelines](#)
- [Azure Artifacts](#)

### ❗ Note

The Azure DevOps cloud-based load testing service is deprecated, but [Azure Load Testing](#) remains available. This fully managed load testing service allows you to generate high-scale load using your existing Apache JMeter scripts. For more information, see [What is Azure Load Testing?](#) and [Changes to load test functionality in Visual Studio and cloud load testing in Azure DevOps](#).

## Free tier

The *free tier* includes the following aspects:

- First five users free (Basic license)

- **Azure Pipelines:**
  - One [Microsoft-hosted CI/CD](#) (one concurrent job, up to 30 hours per month)
  - One self-hosted CI/CD concurrent job
- **Azure Boards:** Work item tracking and boards
- **Azure Repos:** Unlimited private Git repos
- **Azure Artifacts:** Two GiB free per organization

All charges appear on your monthly Azure bill. Azure supports payment by credit card and invoiced billing through the Enterprise Agreement (EA), Cloud Solution Providers (CSP), and more.

## Next steps

[Set up billing](#)

## Related content

- [Change the Azure subscription for billing](#)
- [Marketplace & Extensibility](#)
- [Buy Visual Studio cloud subscriptions](#)

# Shipping Visual Studio Extensions

06/13/2025

After you've finished developing your extension, you can install it on other machines, share it with your friends and coworkers, or publish it on the Visual Studio Marketplace. In this section we explain all the things you need to do in order to publish and maintain your extension: working with .vsix files, publishing, localizing, and updating.

## Working with VSIX Extensions

You can create a VSIX extensions by creating a blank VSIX Project, and then adding different item templates to it. For more information, see [VSIX Project Template](#).

You can use the VSIX format to package project templates, item templates, VSPackages, Managed Extensibility Framework (MEF) components, **Toolbox** controls, assemblies, and custom types (this includes custom Start Pages for Visual Studio 2017). The VSIX format uses file-based deployment. For more information about VSIX packages, see [Anatomy of a VSIX Package](#).

The VSIX format does not support the installation of code snippets. It also does not support certain other scenarios such as writing to the Global Assembly Cache (GAC) or to the system registry. If you need to write to the GAC or the registry in the installation, you must use the Windows Installer. For more information, see [Preparing Extensions for Windows Installer Deployment](#).

## Publishing your Extension to the Visual Studio Marketplace

You can distribute your extension to other people simply by mailing them the .vsix file or putting it on a server. But the best way to get your code in the hands of a lot of people is to put it on the [Visual Studio Marketplace](#). Visual Studio Marketplace extensions are available to Visual Studio users through **Extensions and Updates**. For more information, see [Finding and Using Visual Studio Extensions](#).

For a full example that shows how to upload an extension to the Visual Studio Marketplace, see [Walkthrough: Publishing a Visual Studio Extension](#).

## Private Galleries

As you develop controls, templates, and tools, you can share them with your organization by posting them to a private gallery on your intranet. For more information, see [Private Galleries](#).

## Localizing your Extension

If you're planning to release your extension in different locales, you should consider localizing it. For an explanation of what's involved, see [Localizing VSIX Packages](#).

## Updating and Versioning your Extension

After you've published your extension, there will come a time when you need to update it. To find out how to update an extension that has been published on the Visual Studio Marketplace, see [How to: Update an Extension](#).

You can set your extension to support multiple versions of Visual Studio. For more information, see [Supporting Multiple Versions of Visual Studio](#).

## Related Topics

 [Expand table](#)

Title	Description
<a href="#">Getting Started with the VSIX Project Template</a>	Explains how to use the VSIX project template to install a custom project template.
<a href="#">Anatomy of a VSIX Package</a>	Describes the components of a VSIX package.
<a href="#">VSIX Project Template</a>	Provides step-by-step instructions about how to package and publish an extension.
<a href="#">Localizing VSIX Packages</a>	Explains how to provide localized text for the installation process by using extension.vsixlangpack files.
<a href="#">How to: Update an Extension</a>	Describes how to update an extension on your system and how to deploy an update to an existing Visual Studio extension.
<a href="#">How to: Add a Dependency to a VSIX Package</a>	Describes how to add references to VSIX deployment packages.
<a href="#">Preparing Extensions for Windows Installer Deployment</a>	Explains how to deploy your extension with Windows Installer.
<a href="#">Signing VSIX Packages</a>	Explains how to sign VSIX packages.

Title	Description
Private Galleries	Explains how to create private galleries for extensions.
Supporting Multiple Versions of Visual Studio	Shows how to have your extension support multiple versions of Visual Studio.
Locating Visual Studio	Describes how to locate Visual Studio instances for custom extension deployment.

# Windows app development documentation

Learn how to design, develop, and deploy apps and solutions for Windows PCs and other devices.



**QUICKSTART**  
[Start developing Windows apps](#)



**OVERVIEW**  
[App framework options](#)



**OVERVIEW**  
[Enhance your Windows apps with AI](#)



**OVERVIEW**  
[Set up your environment with Dev Home, WinGet, Terminal, WSL, and more](#)



**REFERENCE**  
[Windows APIs](#)



**DOWNLOAD**  
[Download Visual Studio](#)



**DOWNLOAD**  
[Windows Code Samples](#)



**TRAINING**  
[Online training for Windows app developers](#)

## Dive into developing apps for Windows

### Get started

[Overview of framework options](#)

[Get started with WinUI](#)

[Windows App SDK and supported Windows releases](#)



### Design

[Design and code your app UI](#)

[Design basics for Windows apps](#)

[Windows 11 design principles](#)

[Install tools for the Windows App SDK](#)

[Sample apps for Windows](#)

[Design toolkits](#)



## Develop

[Windows development features](#)

[Windows App SDK features](#)

[WinUI](#)

[Windows Copilot Runtime](#)



## Deploy

[Deployment architecture](#)

[Windows App SDK deployment guide](#)

[Package apps using MSIX](#)

[Manage your MSIX deployment](#)

# More development options



[Windows Application SDK](#)



[.NET](#)



[UWP](#)



[Games](#)

# Contribute to Windows Open Source projects

## [Windows App SDK](#)

A set of developer tools and APIs that represent the next evolution in the Windows app platform.

## [WinUI](#)

The native user interface layer for building Windows apps including modern controls and styles.

## [MSIX SDK](#)

## [.NET](#)

Enables developers to pack and unpack app packages on a variety of platforms.

Engage with .NET open source projects including .NET, WPF, Windows Forms, and much more.

[@WindowsDocs](#) - [OneDevMinute on YouTube](#) - [Windows developer support](#) - [Stack Overflow](#)

# About migrating and integrating work tracking data

10/08/2025

Azure DevOps Services | Azure DevOps Server | Azure DevOps Server 2022 | Azure DevOps Server 2020

You can migrate work tracking data into Azure Boards and integrate Azure Boards with many non-Microsoft tools. This article gives an overview of migration options, common scenarios, and extensions that help with migration and integration.

## Tip

Browse Azure Boards extensions in the Visual Studio Marketplace to customize and extend your boards experience. See the "Extensions for Azure Boards" section later in this article.

## Migrate from Azure DevOps Server

Use the Data Migration Tool for Azure DevOps to migrate collection databases from Azure DevOps Server to Azure DevOps Services with high fidelity. For details and guidance, see [Migrate data from Azure DevOps Server to Azure DevOps Services](#).

## Migrate data between projects

Search the Visual Studio Marketplace for extensions that help you bulk edit, migrate, or synchronize work items between projects. These tools typically support these tasks:

- Migrate work items (including custom fields and history) from one project to another and synchronize changes after migration.
- Merge multiple projects into a single project, or split a project into multiple projects.
- Assist with process template changes and mapping fields between processes.
- Bulk edit work items.
- Migrate test plans, test suites, test cases, and test results.

## Migrate data between projects without downtime

Some non-Microsoft tools enable migration with minimal disruption, even when the target environment contains data or uses a different process template. These tools commonly preserve:

- Version control history, including original dates and authors for commits and comments.
- Work items and their history (standard and custom), preserving original dates and authors.
- Test plans, test suites, test cases, and test results.
- Area and iteration paths, teams, and user mappings.
- Dashboards, queries, widgets, and pipeline references (where supported).
- Compatibility with Azure DevOps Server (2010+) and Azure DevOps Services (cloud), depending on the tool.

Before you migrate, test the tool in a staging environment, confirm support for your Server or Services versions, and back up your data.

## Migrate process models between Azure DevOps organizations

When you import process templates, you might hit validation constraints. For troubleshooting process import errors, see [Resolve validation errors for process import](#).

## Export and import work tracking data

Use Microsoft Excel to export and import work item lists or hierarchical work item trees. Excel supports publishing flat lists or parent-child hierarchies. For step-by-step instructions, see [Bulk add or modify work items with Excel](#).

## Integrate with GitHub

You can connect Azure Boards with GitHub to link commits, pull requests, and issues to Azure Boards work items. This integration lets you use GitHub for development while tracking work in Azure Boards. See [Connect Azure Boards to GitHub](#) for setup steps.

## Integrate with non-Microsoft tools using service hooks

Use service hooks to trigger actions in external services (for example, Jenkins or Trello) when Azure DevOps events occur. For examples and service-specific guidance, see:

- [Create a service hook for Azure DevOps with Jenkins](#)
- [Create a service hook for Azure DevOps with Trello](#)
- [Integrate with service hooks overview](#)

# Extensions for Azure Boards

The table below lists representative Microsoft extensions and useful categories. Search the Visual Studio Marketplace for more Microsoft and non-Microsoft extensions that match your scenario.

 Expand table

Category	Extensions
Automation	<a href="#">Power Automate, Azure DevOps</a> ↗
Command-line interface	<a href="#">Azure DevOps CLI</a>
Customizing work item types	<ul style="list-style-type: none"><li>- <a href="#">Cascading Lists</a> ↗</li><li>- <a href="#">Color picklist control</a> ↗</li><li>- <a href="#">Multi-value control</a> ↗</li><li>- <a href="#">Work Item Visualization</a> ↗</li><li>- <a href="#">WSJF (Weighted Shortest Job First)</a> ↗</li></ul>
Dashboard widgets	<ul style="list-style-type: none"><li>- <a href="#">Azure Application Insights widget</a> ↗</li><li>- <a href="#">Work Item Details widget</a> ↗</li><li>- <a href="#">Roll-up Board widget</a> ↗</li></ul>
Product planning	<ul style="list-style-type: none"><li>- <a href="#">Azure DevOps Open in Excel</a> ↗</li><li>- <a href="#">Epic Roadmap extension</a> ↗</li></ul>
Querying and reporting	<a href="#">WIQL to OData</a> ↗

## 💡 Tip

When you evaluate extensions for migration or integration, verify compatibility with your Azure DevOps Server or Services version and test in a nonproduction environment.

## Next step

- [Bulk add or modify work items with Microsoft Excel](#)

## Related content

- [Use the Azure DevOps REST API](#)
- [Migrate data from Azure DevOps Server to Azure DevOps Services](#)
- [Integrate with service hooks](#)

 **Note:** The author created this article with assistance from AI. [Learn more](#)

# Find, install, and manage extensions for Visual Studio

08/18/2025

This article describes how to find, install, and manage extension packages in Visual Studio.

Extensions are code packages that run inside Visual Studio and provide new or improved features. Extensions can be controls, samples, templates, tools, or other components that add functionality to Visual Studio, for example, [Live Share](#) or [GitHub Copilot](#).

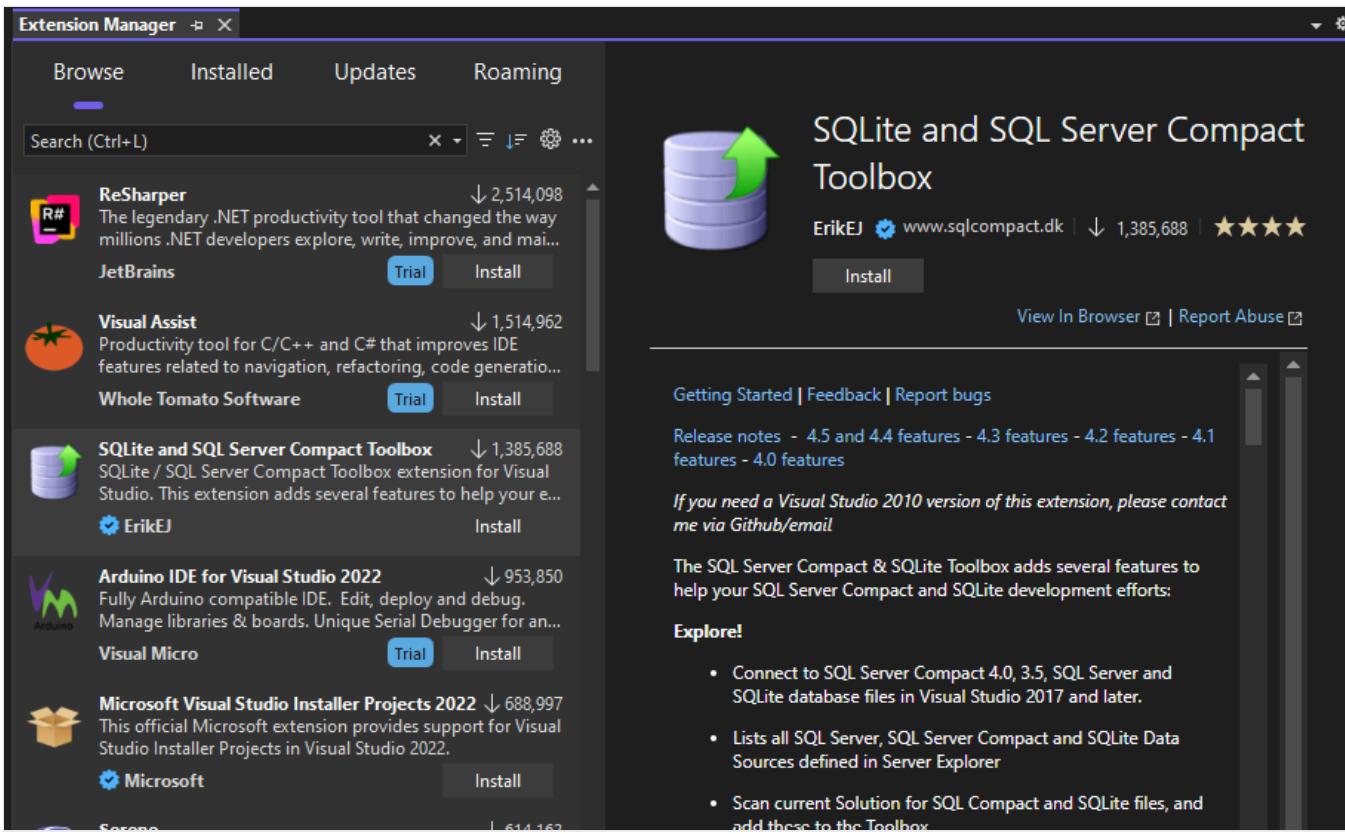
For information about creating Visual Studio extensions, see [Visual Studio SDK](#). For information about using extensions, see the individual extension page on [Visual Studio Marketplace](#). For information about finding extensions, see the [Where Are My Favorite Extensions in Visual Studio 2022?](#) blog post.

## ⓘ Note

To learn more about an extensibility model that's currently in preview, see the [Build Better Extensions with VisualStudio.Extensibility\\_\(Preview 3\)](#) blog post.

## Use Extension Manager

In the Visual Studio IDE, Extension Manager is the tool that you use to find, install, and manage Visual Studio extensions. To open Extension Manager, select **Extensions > Manage Extensions**. Or enter **extensions** in the search box and select **Manage Extensions**.



The left pane categorizes extensions by those that are available on Visual Studio Marketplace (**Browse**), those that are installed, and those that have updates available. The **Roaming** tab lists all the Visual Studio extensions that you have installed on any machine or instance of Visual Studio. It's designed to let you find your favorite extensions more easily.

Tabs have buttons to filter and sort extensions, and a gear button that takes you to the **Tools > Options** screen.

You can use the filter button to filter extensions by category. Categories for **Installed** extensions include **Enabled**, **Disabled**, and **Incompatible**. In 17.14 and later, you can also filter by **Pending**, which filters by extensions with updates that will be applied the next time you restart Visual Studio.

Use the sort button to sort by extension name, extension author, or date.

## Find and install extensions

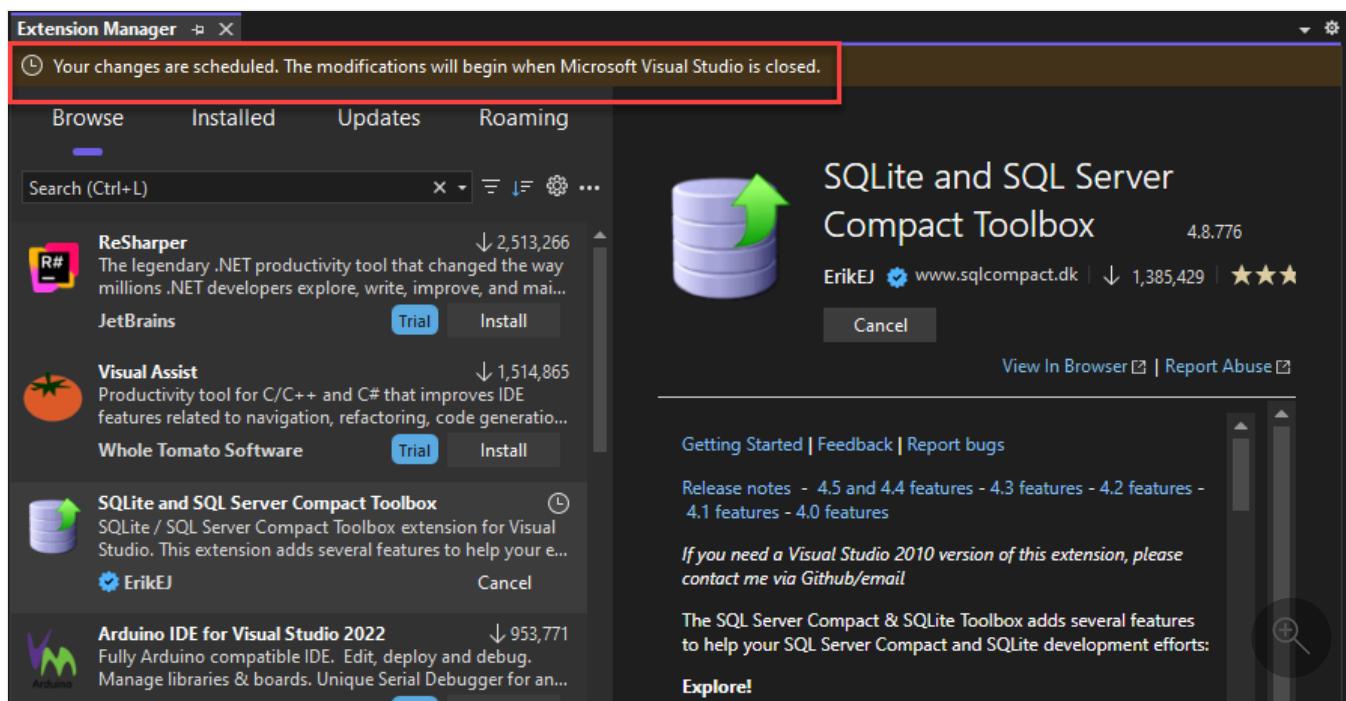
You can install extensions from [Visual Studio Marketplace](#) or from Extension Manager in Visual Studio.

To install extensions from Visual Studio:

1. Select **Extensions > Manage Extensions**. Find the extension you want to install. (If you know the name or part of the name of the extension, you can search in the search box.)

## 2. Select Install.

After the download completes, you see a notification at the top of Extension Manager: "Your changes are scheduled. The modifications will begin when Microsoft Visual Studio is closed."



If you try to install an extension that has dependencies, the installer determines whether they're already installed. If they aren't installed, Extension Manager lists the dependencies that must be installed before you can install the extension.

## Install extensions without using Extension Manager

Extensions that are packaged in .vsix files might be available in locations other than Visual Studio Marketplace. The **Extensions > Extension Manager** dialog can't detect these files, but you can install a .vsix file by double-clicking the file or selecting the file and then selecting **Enter**. If you encounter permission issues, ensure that you're running Visual Studio as an administrator. After that, just follow the instructions. When the extension is installed, you can use Extension Manager to enable it, disable it, or uninstall it.

### ! Note

- Visual Studio Marketplace contains both VSIX-based and MSI-based extensions. Extension Manager can't enable or disable MSI-based extensions.
- If an MSI-based extension includes an *extension.vsixmanifest* file, the extension appears in Extension Manager.

# Uninstall or disable an extension

If you want to stop using an extension, you can either disable it or uninstall it. Disabling an extension keeps it installed but unloaded. Find the extension and select **Uninstall or Disable**. Restart Visual Studio to unload a disabled extension.

## ⓘ Note

You can disable VSIX-based extensions but not extensions that are installed via MSI. MSI-installed extensions can only be uninstalled.

## Manage extensions

This section provides information about managing extensions for an organization, automatic extension updates, and unresponsiveness notifications.

### Per-user and administrative extensions

Most extensions are per-user extensions and are installed in the `%LocalAppData%\Microsoft\VisualStudio\<Visual Studio version>\Extensions\` folder. A few extensions are administrative extensions and are installed in the `<Visual Studio installation folder>\Common7\IDE\Extensions\` folder.

To protect your system against extensions that might contain errors or malicious code, you can restrict per-user extensions to load only when Visual Studio is run with normal user permissions. This causes per-user extensions to be disabled when Visual Studio is run with elevated permissions.

To restrict when per-user extensions load:

1. Open the extensions options page (**Tools > Options > Environment > Extensions**).
2. Clear the **Load per user extensions when running as administrator** checkbox.
3. Restart Visual Studio.

### Automatic extension updates

Extensions are updated automatically when a new version is available on Visual Studio Marketplace. The new version of the extension is detected and installed in the background. The next time you open Visual Studio, the new version of the extension will be running.

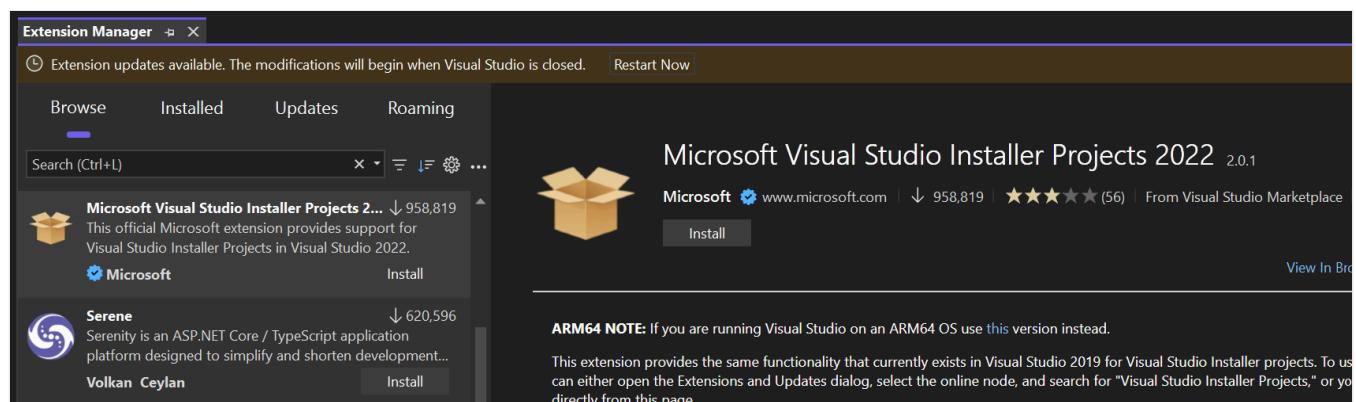
If you want to disable automatic updates, you can disable the feature for all extensions or for only specific extensions.

- To disable automatic updates for all extensions, select the **View Settings** button in the **Extensions > Manage Extensions** dialog. In the **Options** dialog, under **Environment > Extensions**, clear **Install updates automatically**.
- To disable automatic updates for a specific extension, clear the **Automatically update this extension** option in the extension's details pane on the right side of Extension Manager.

The rest of this section describes changes available in Visual Studio 2022 17.14 and later. The changes make it easier to see what's going on with extension updates, and manage automatic update settings from Extension Manager or in **Tools > Options**.

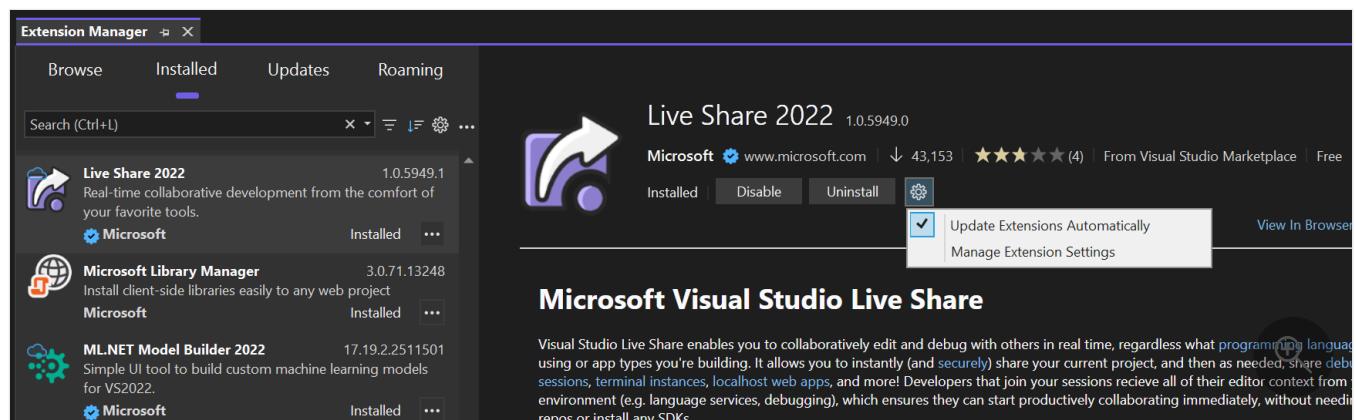
Whenever you open the Extension Manager window, you automatically trigger updates. This ensures that you're consistently working with the latest features and don't need to manually initiate updates.

When an update is available for an extension, you see a yellow notification infobar:

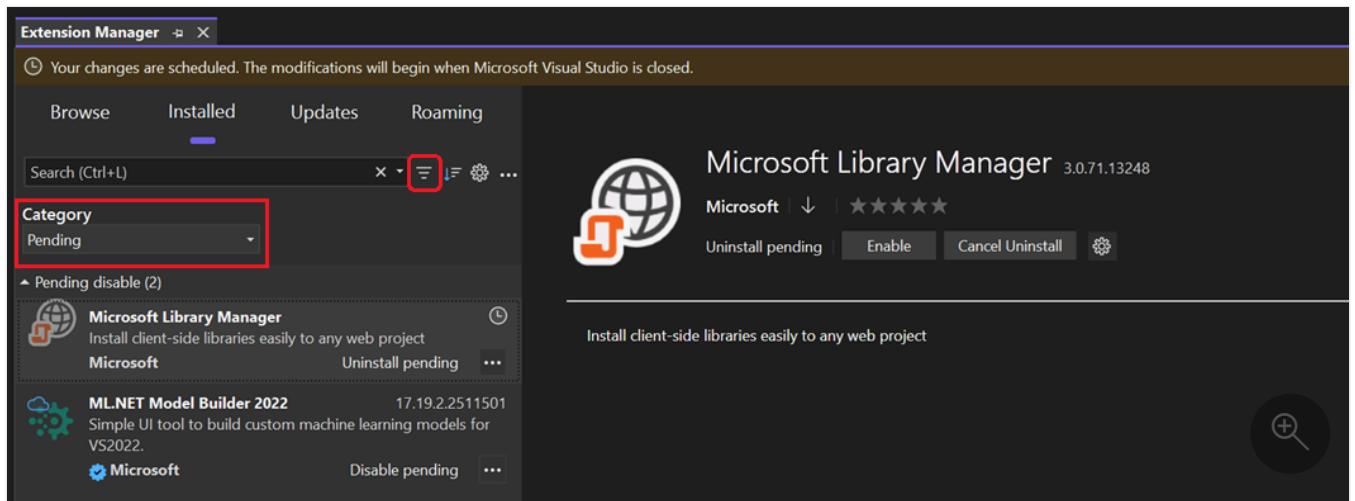


Also, a notification appears when updates are applied, reminding you to restart Visual Studio so that the extensions take effect.

You can change settings for automatic updates on the extension's page in Extension Manager.

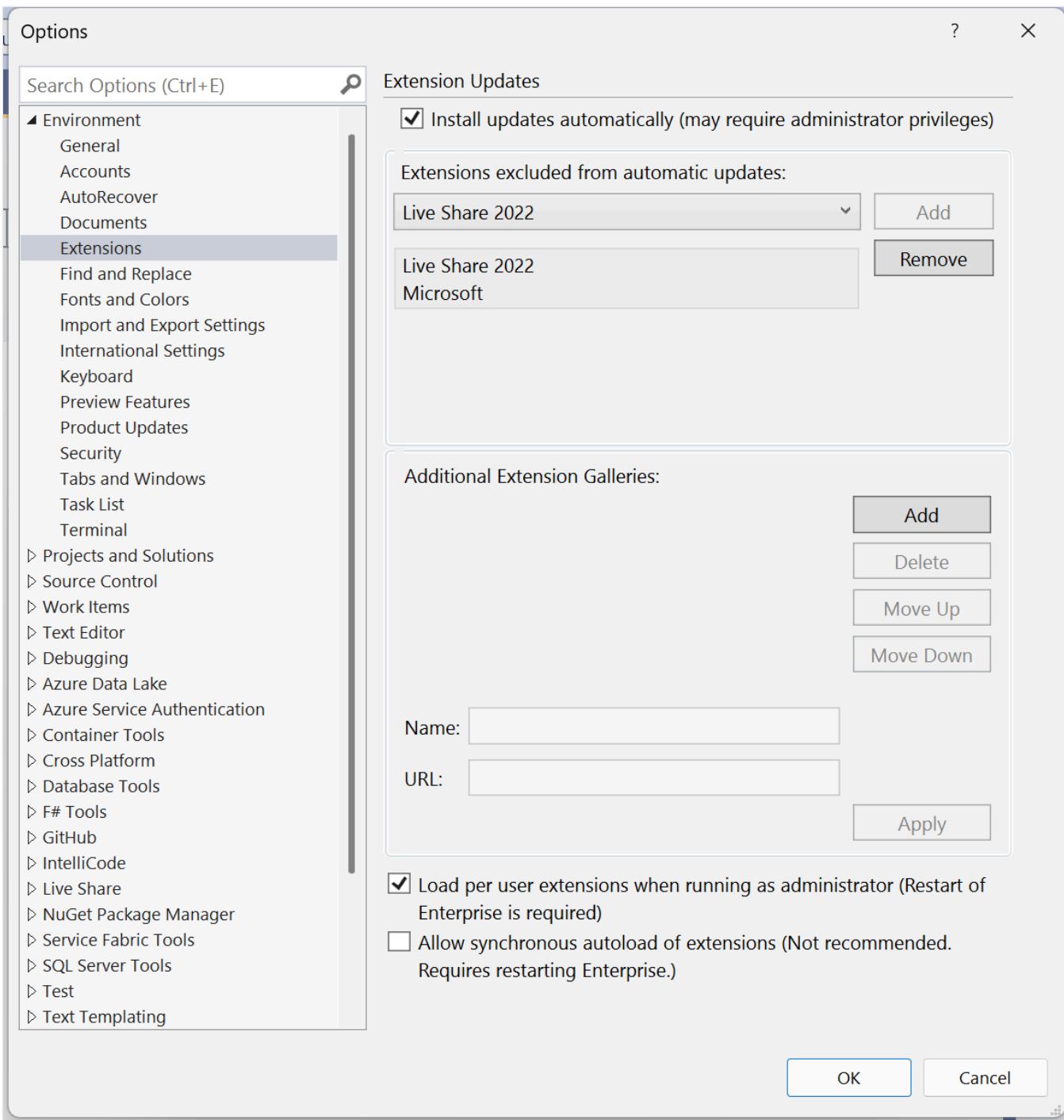


The **Pending** category enables you to view only extensions that have pending updates.



## Automatic update settings

You can also configure automatic update settings in **Tools > Options**.



The list under **Extensions excluded from automatic updates** shows the extensions that won't be updated automatically. You might use this to ensure the stability and consistency of the extensions you're using during a critical phase of your development lifecycle.

## Crash and unresponsiveness notifications

Visual Studio notifies you if it suspects that an extension was involved in a crash during a previous session. When Visual Studio crashes, it stores the exception stack. The next time Visual Studio starts, it examines the stack, starting with the leaf and working towards the base. If Visual Studio determines that a frame belongs to a module that's part of an installed and enabled extension, it shows a notification.

Visual Studio also notifies you if it suspects that an extension is causing the UI to be unresponsive.

When you see one of these notifications, you can ignore it or take one of the following actions:

- Select **Disable this extension**. Visual Studio disables the extension and lets you know whether you need to restart your system for the disable operation to take effect. You can re-enable the extension in the **Extensions > Manage Extensions** dialog.
- Select **Never show this message again**.
  - If the notification concerns a crash in a previous session, Visual Studio no longer shows a notification when a crash associated with the extension occurs. Visual Studio still shows notifications when unresponsiveness can be associated with the extension or for crashes or unresponsiveness that can be associated with other extensions.
  - If the notification concerns unresponsiveness, the IDE no longer shows a notification when the extension is associated with unresponsiveness. Visual Studio still shows crash-related notifications for the extension and crash-related and unresponsiveness-related notifications for other extensions.
- Select **Learn more**.
- Select the **X** at the end of the notification to dismiss the notification. A new notification appears if the extension is associated with a crash or with UI unresponsiveness in the future.

#### Note

A UI unresponsiveness notification or crash notification means that one of the extension's modules was on the stack when the UI was unresponsive or when the crash occurred. It doesn't necessarily mean that the extension caused the problem. It's possible that the extension called code that's part of Visual Studio, which in turn resulted in unresponsive UI or a crash. However, the notification might still be useful if the extension that led to the UI unresponsiveness or crash isn't important to you. In this case, disabling the extension avoids the UI unresponsiveness or crash in the future.

## Marketplace protections

The Visual Studio Marketplace for extensions employs several mechanisms to protect you from malicious extensions:

- **Malware scanning**: The Marketplace runs a malware scan on each extension package that's published to ensure its safety. The scan, which uses several antivirus engines, is run

for each new extension and for each extension update. Until the scan is all clear, the extension won't be published in the Marketplace for public usage.

- **Verified publishers:** Publishers can verify (blue check mark) their identity by proving domain ownership. It shows that the publisher has proven domain-name ownership to the Marketplace. It also shows that the Marketplace has verified both the existence of the domain and the good standing of the publisher on the Marketplace for at least six months.
- **Unusual usage monitoring:** The Marketplace monitors the downloads and usage patterns of extensions to detect unusual behavior.
- **Name squatting:** The Marketplace stops extension authors from stealing the names of official publishers, such as Microsoft or RedHat, and popular extensions, like GitHub Copilot.
- **Block List:** If a malicious extension is reported and verified, or a vulnerability is found in an extension dependency, the extension is removed from the Marketplace.
- **Extension Signature Verification:** The Visual Studio Marketplace signs all extensions when they're published. Visual Studio checks this signature when you install an extension to verify the integrity and the source of the extension package.
- **Secret Scanning:** The Marketplace automatically scans every newly published extension for secrets such as API keys or credentials (e.g., Azure DevOps PAT tokens). If any secrets are detected, publishing is blocked to prevent potential security risks.

Learn about these measures in the [Security and Trust in Visual Studio Marketplace blog post](#).

## Samples

When you install an online sample, the solution is stored in two locations:

- A working copy is stored in the location that you specified when you created the project.
- A separate master copy is stored on your computer.

You can use the **Extensions > Manage Extensions** dialog to perform these samples-related tasks:

- List the master copies of samples that you have installed.
- Disable or uninstall the master copy of a sample.

- Install Sample Packs, which are collections of samples that relate to a technology or feature.
- Install individual online samples.
- View update notifications when source code changes are published for installed samples.
- Update the master copy of an installed sample when there's an update notification.

## Related content

- [Visual Studio SDK](#)
- [Visual Studio Extensibility](#)
- [Visual Studio Marketplace ↗](#)