Antony Kervazo-Canut

Azure DevOps for Teenagers

SUITABLE FOR ADULTS

Introduction

Azure DevOps is the result of the evolution of several generations of Microsoft's software development tools, now offering a robust platform that integrates DevOps practices with Microsoft's cloud ecosystem.

- Visual Studio Team System (VSTS): Launched in 2005, it was Microsoft's first offering for integrated software development, the direct precursor to TFS and Azure DevOps. Team Foundation
- Server (TFS): Stemming from VSTS, TFS provided a complete On Premise solution.
- Visual Studio Online (VSO): With the advent of the cloud, TFS evolved into VSO in 2013, offering the same features in an online version, facilitating access and flexibility.
- Azure DevOps: In 2018, VSO was renamed Azure DevOps, integrating more deeply with Azure services and reinforcing Microsoft's commitment to DevOps practices. This version marks a more coherent and integrated offering for modern software development.
- Azure DevOps Server: Concurrently, the latest version of TFS was renamed Azure DevOps Server, providing an on-site option for those who prefer it, while staying in sync with Azure DevOps cloud services.

Azure DevOps is divided into several main services:

- Azure Boards: Work tracking and project management.
- Azure Repos: Version control with Git.
- Azure Pipelines: CI/CD for all languages and platforms.
- Azure Test Plans: Manual and automated testing.
- Azure Artifacts: Sharing Maven, npm, and NuGet packages.

Get Azure DevOps

To start your journey with Azure DevOps, the first step is to create an account and set up your workspace.

dev.azure.com

Visit the Azure DevOps site and click on "Start free".

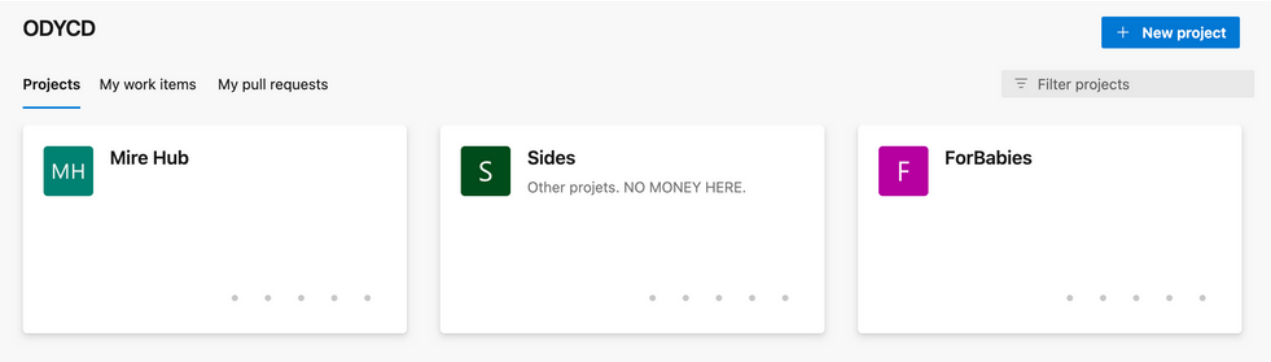
Sign in or create a Microsoft account: If you already have a Microsoft account (Outlook, Office 365, etc.), sign in with it. Otherwise, create a new account.

Create your Azure DevOps organization: Once signed in, you will be prompted to create a new organization. Give it a unique name that represents your team, project, or even your company.

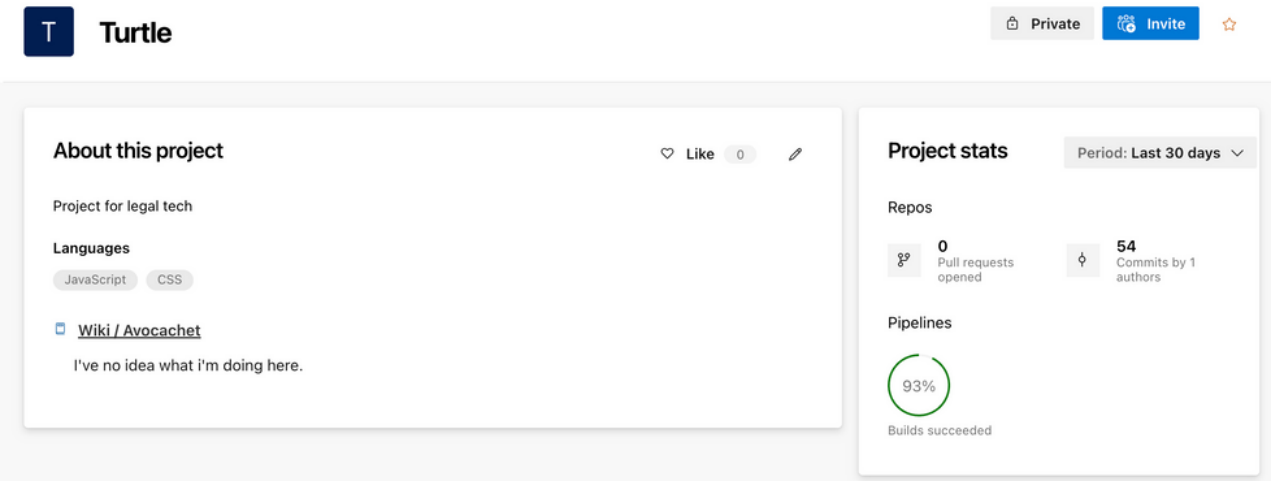
Set up your first project: After creating your organization, create your first project by giving it a name and choosing its visibility (private or public).

Organization & Projects

In Azure DevOps, an Organization is the primary level of structure for managing your projects, users, and software development resources in a shared environment. It serves as a container for your Azure DevOps projects and provides a framework for team collaboration, security management, and integration with other Azure services.

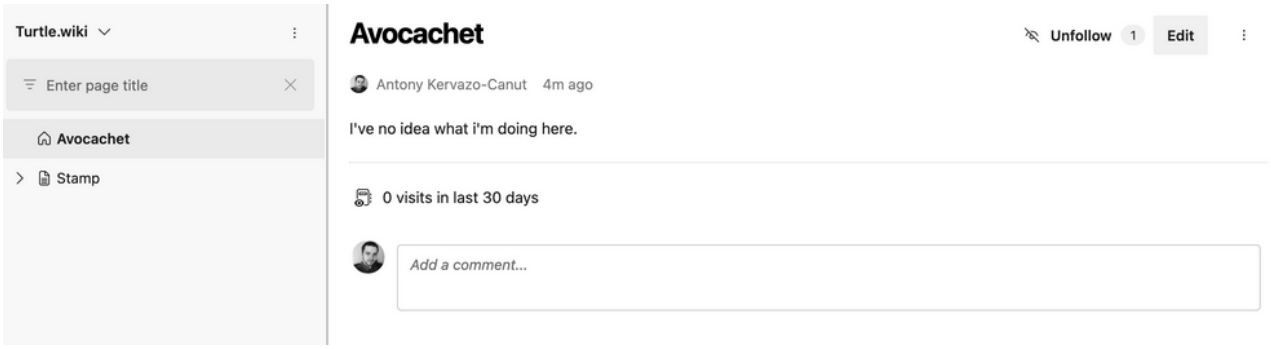


You can have multiple projects within an organization, each configured with its own settings, permissions, and team members.

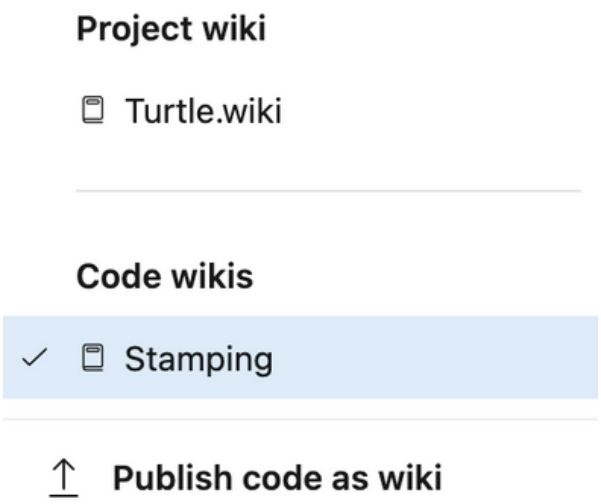


Wiki

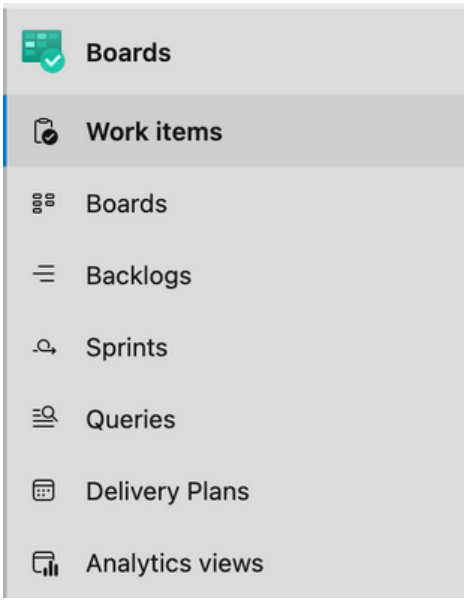
The Wiki in Azure DevOps is an integrated collaborative tool that allows teams to create, share, and manage project documentation directly within their Azure DevOps workspace. It is designed to capture and organize project knowledge, facilitate communication among team members, and serve as a central repository for important project information.



Azure DevOps has the advantage of having a global wiki for a project, but also of gathering all the documentation contained in your projects directly in the Wiki section.



Azure Boards



Azure Boards is an Azure DevOps service that offers comprehensive tools for agile planning, work tracking, and project management. It allows teams to track the progress of their projects through Kanban boards, backlogs, sprints, and reports.

Hierarchy: Azure Boards allows organizing work items into a hierarchy, helping to visualize the relationships between Epics, Features, User Stories, and Tasks. This structure facilitates planning and tracking at different levels of abstraction.

Statuses: Each work item goes through various statuses that represent its progress, such as "New", "Active", "Resolved", and "Closed". You can customize the status workflow to fit your development process.

Tags and Filters: Use tags to categorize work items and filter them to facilitate searching and organizing. Filters can be based on criteria such as work item type, status, assignee, and more.

Process Boards

Azure DevOps supports different types of work processes to adapt to the specific methodologies of a team or project. These processes define the framework for work items, boards, and workflows in Azure Boards, offering a set of templates suited to various project management approaches. The main types of processes available in Azure DevOps are: Agile, Scrum, CMMI, and Basic.

Agile

Designed for teams following the agile methodology, emphasizing iterative and incremental development cycles, and a high responsiveness to changes.
Main Work Items: Epics, Features, User Stories, Tasks, Bugs.

Scrum

Suited for teams using the Scrum framework, a subcategory of the agile methodology focused on fixed sprints, sprint reviews, and daily stand-ups.
Main Work Items: Epics, Features, Backlog Items, Tasks, Bugs.

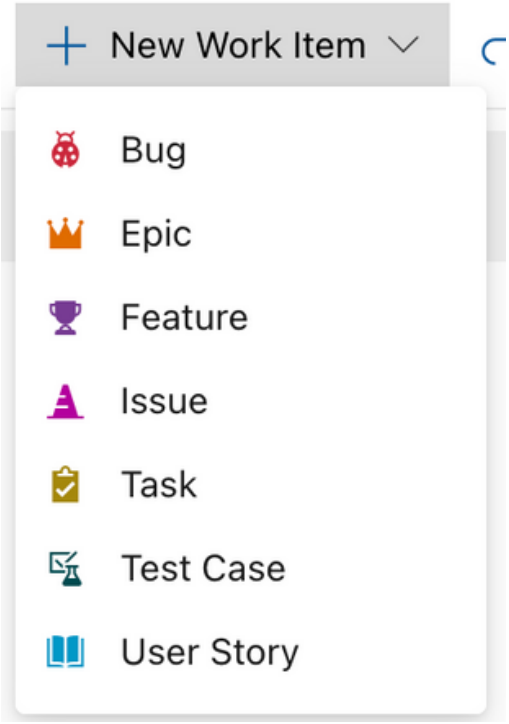
CMMI

Oriented towards teams that apply more formal and structured project management approaches, with an emphasis on continuous process improvement.
Main Work Items: Requirements, Change Requests, Defects, Tasks.

Basic

A simplified framework, with a minimal set of work item types. It's a good option for small teams or projects that don't need the formalities of the Agile, Scrum, or CMMI frameworks.
Main Work Items: Epics, Tasks, Bugs.

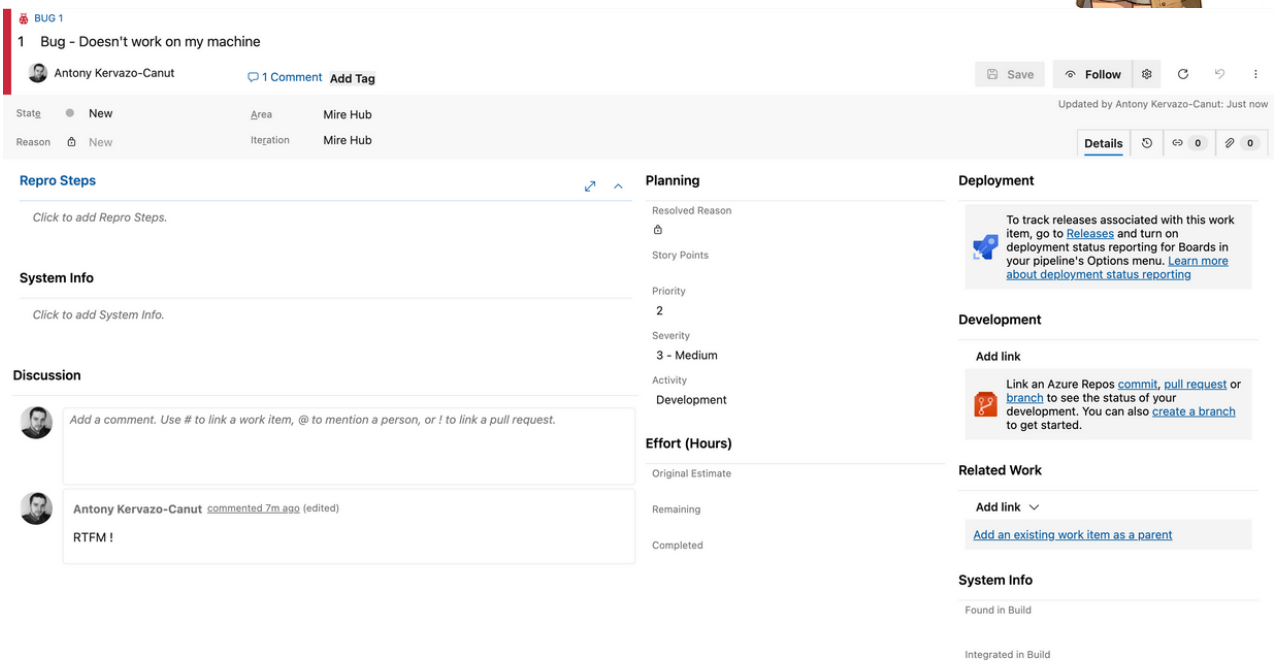
Work Items



Azure Boards offers different types of work items to suit various project needs and methodologies. The basic types include:

- Bug: Used to track issues, errors, or bugs in the software. Bugs can be linked to specific User Stories for easy tracking.
- Epic: Represents a large block of work that can be broken down into several features. Epics are often used to track work across multiple sprints or even product releases.
- Feature: Used to define a functionality or a set of functionalities under an Epic. A feature is usually delivered over several sprints.
- Task: Represents a specific unit of work needed to complete a User Story or a Feature. Tasks are often assigned to individuals for execution.
- User Story (or simply "Story"): Describes a functionality from the end-user's perspective. Stories are smaller work items than features and aim to deliver value to the user.

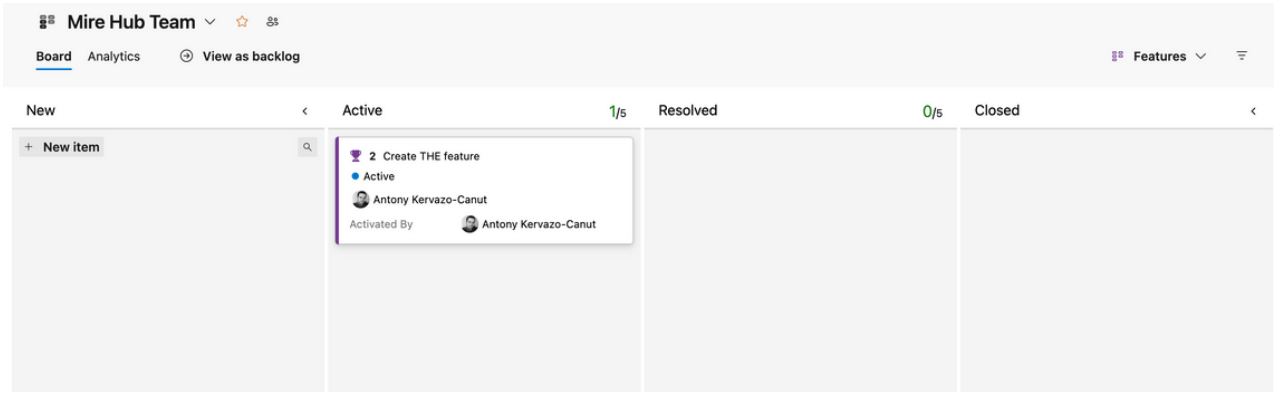
Work Items



- Comments and Attachments: Team members can leave comments on work items and attach files or images to provide more context or share resources.
- Links and Integrations: Azure Boards allows linking work items to each other (for example, linking a Task to a User Story) or to other elements like code commits, pull requests, and builds in Azure Repos and Azure Pipelines. This creates a complete and integrated audit trail of project activity.

Kanban

Kanban is a project and workflow management method that aims to improve the efficiency and flexibility of work processes. Originating from Japan and initially developed by Toyota in the 1940s to optimize its manufacturing processes, the Kanban method has since been adopted in many other fields, including software development and IT project management, thanks to its simplicity and effectiveness. Azure Boards integrates the Kanban system into its features, allowing software development teams to manage their projects in an agile and visual manner.

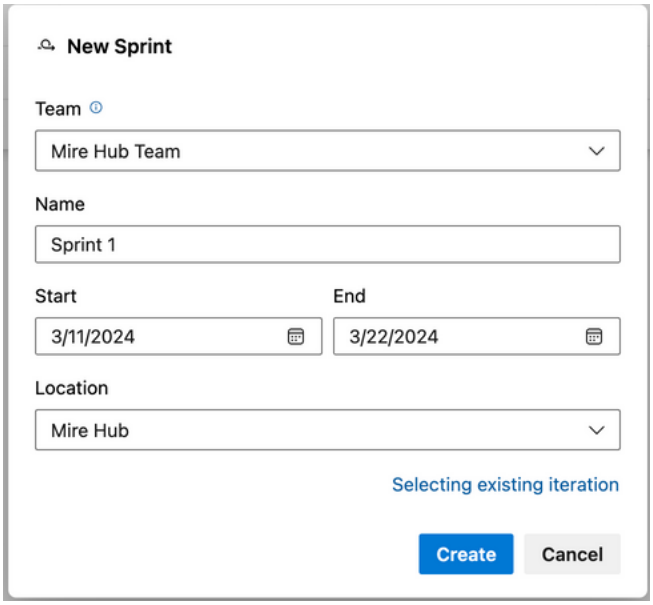


The implementation of Kanban in Azure Boards allows development teams to maximize their productivity, optimize their processes, and achieve faster and more reliable value delivery to their customers.

With this in mind, it's possible to link a commit in a repository to a ticket.

Sprints

Sprints in Azure Boards are an integral part of agile project management, allowing teams to plan, execute, and track work in defined time intervals, called sprints. This feature is essential for teams adopting agile methodologies such as Scrum, as it helps to organize work into manageable batches, keeps the team focused on short-term goals, and provides continuous and incremental product improvements.

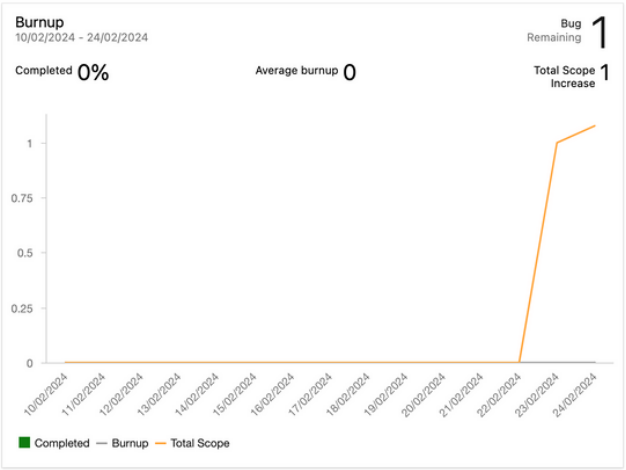


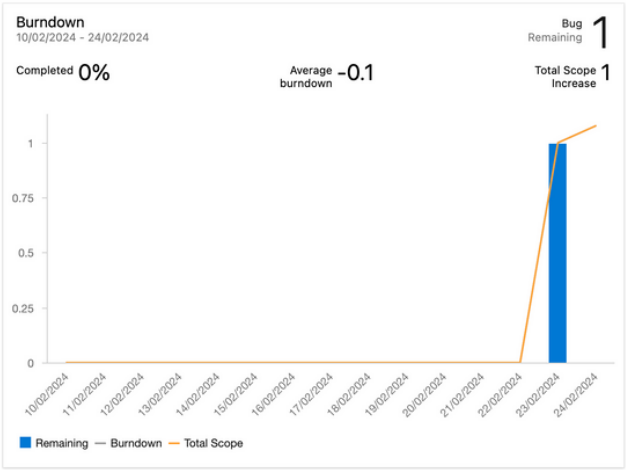
Set your iterations: In the settings of your Azure DevOps project, define the structure of your iterations. You can create a hierarchy of iterations (for example, by dividing them by year, quarter, etc.) and specify the duration of each sprint.

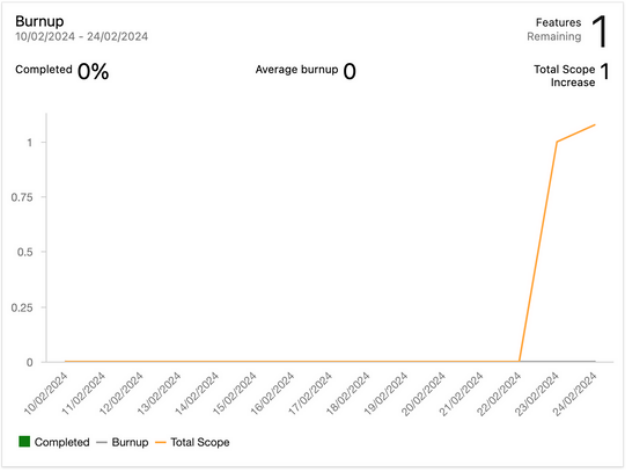
Plan the work: Assign work items to specific sprints by setting the iteration for each work item. You can do this by dragging and dropping items into the iteration backlog or by modifying the iteration in the work item details.

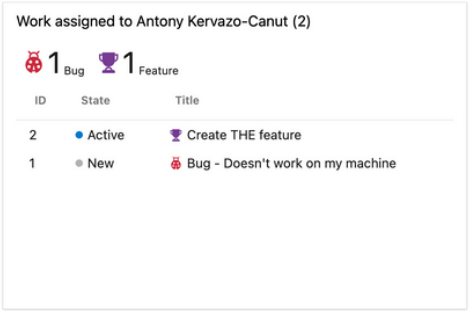
Dashboard & Widgets

Dashboards in Azure Boards offer a customizable and interactive view of various aspects of your project, thanks to the use of widgets. These widgets can display a wide range of information from work progress status, bug tracking, results of the latest builds or deployments, to custom metrics important to the team. Here's a more detailed overview of some key widgets available in Azure Boards and how they can be used to improve the visibility and tracking of your projects.



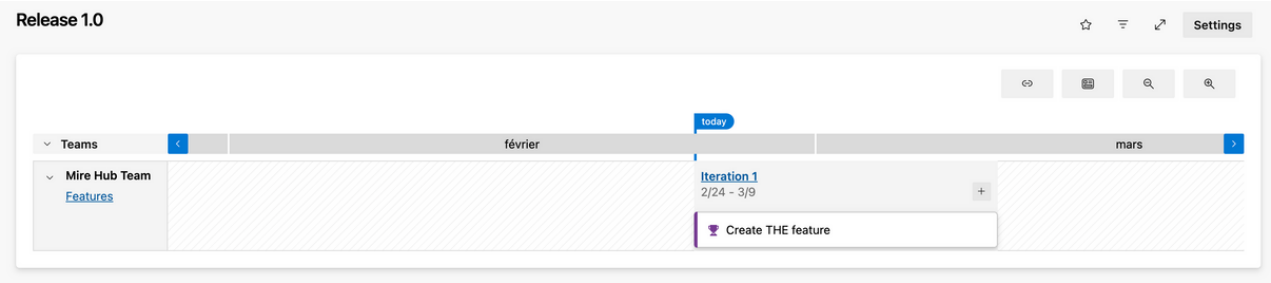






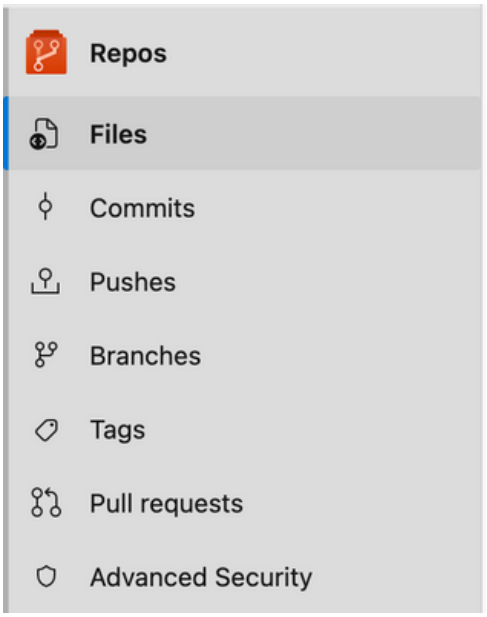
Delivery Plan

The Delivery Plan is a powerful extension for Azure Boards that provides a calendar view to visualize work across multiple teams and projects. It assists organizations in understanding how work aligns with delivery schedules, facilitating the planning and tracking of dependencies between teams. This view allows managers and teams to stay synchronized on objectives and delivery timelines, while identifying potential risks and bottlenecks at an early stage.

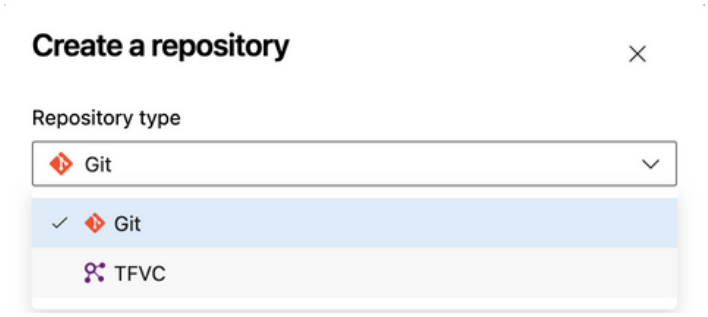


The Delivery Plan is an essential tool for project managers, team leaders, and stakeholders looking to have a comprehensive understanding of project delivery in an Agile environment. It provides the necessary insights to make informed decisions, manage resources effectively, and achieve delivery objectives within the planned timelines.

Azure Repos



Azure Repos is a set of version control hosting services that allows you to manage your source code in the cloud. It is part of Microsoft's Azure DevOps suite and offers two main types of version control: Git, a distributed version control system, and Team Foundation Version Control (TFVC), a centralized version control system. Here's a simplified overview of what you need to know about Azure Repos and how to use it to manage your code.



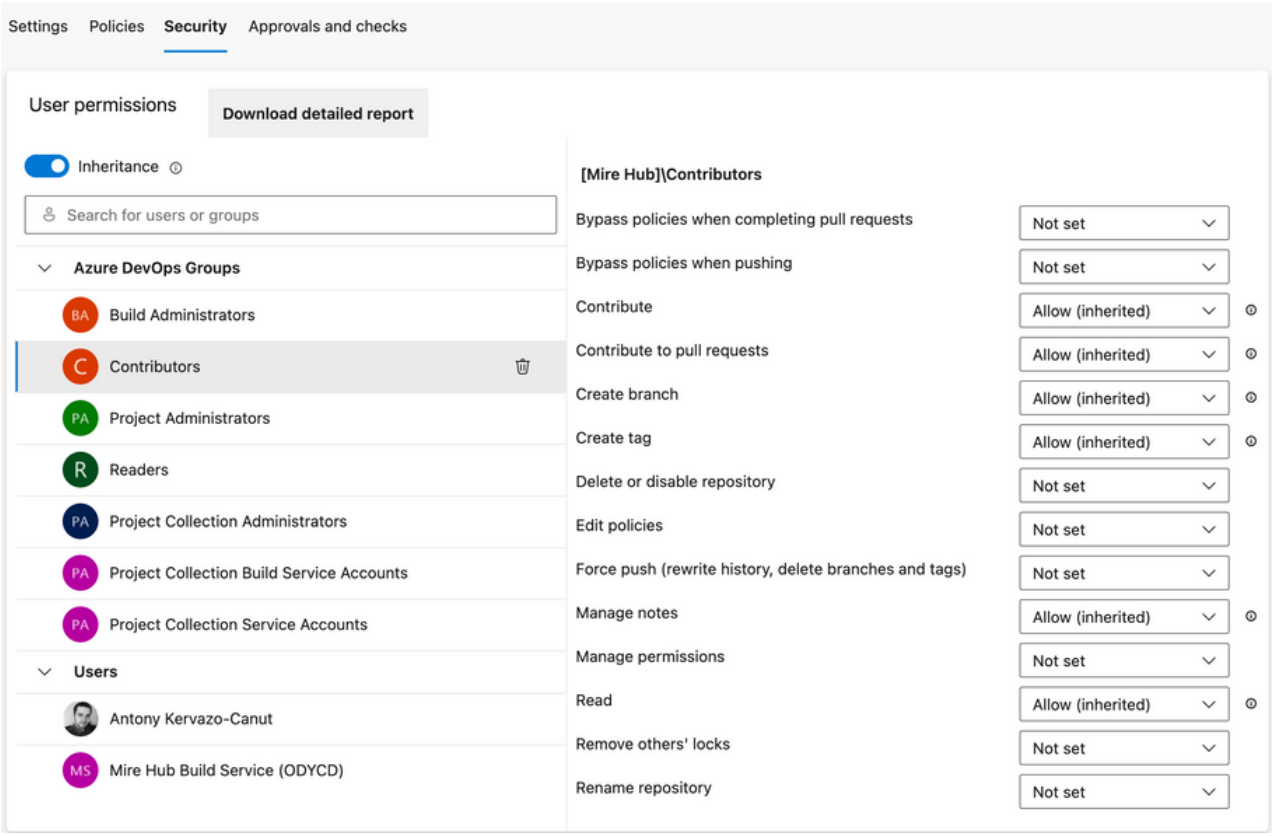
Git Repositories: Azure Repos provides unlimited Git hosting, supporting collaborative development workflows. This includes features such as code reviews, feature branches, and pull requests to facilitate collaboration and code management.

TFVC (Team Foundation Version Control): For those who prefer a centralized version control model, TFVC allows tracking changes in the code, maintaining version histories, and working on large projects with many dependencies.

Security and Permissions

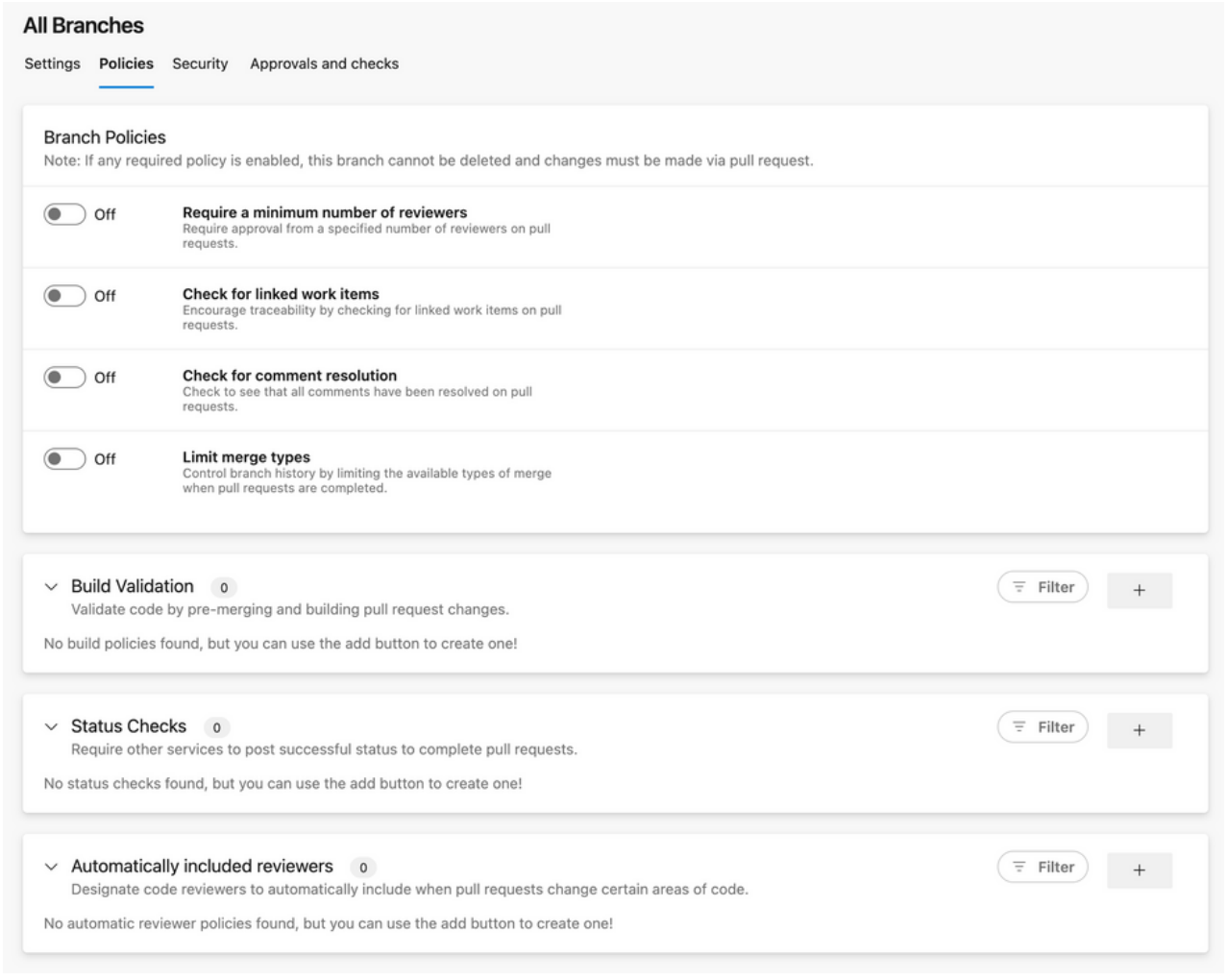
Azure Repos uses a role-based permission system that allows you to finely control access to your repositories. Permissions can be assigned at different levels, including:

- Project Level: You can set permissions that apply to all repos within an Azure DevOps project. This is useful for configuring project-wide roles, such as project administrators who need access to all repos.
- Repository Level: Permissions can also be specifically set for each repository, allowing you to customize access based on the needs of each team or project. This includes controlling who can read, write, or manage the repository.



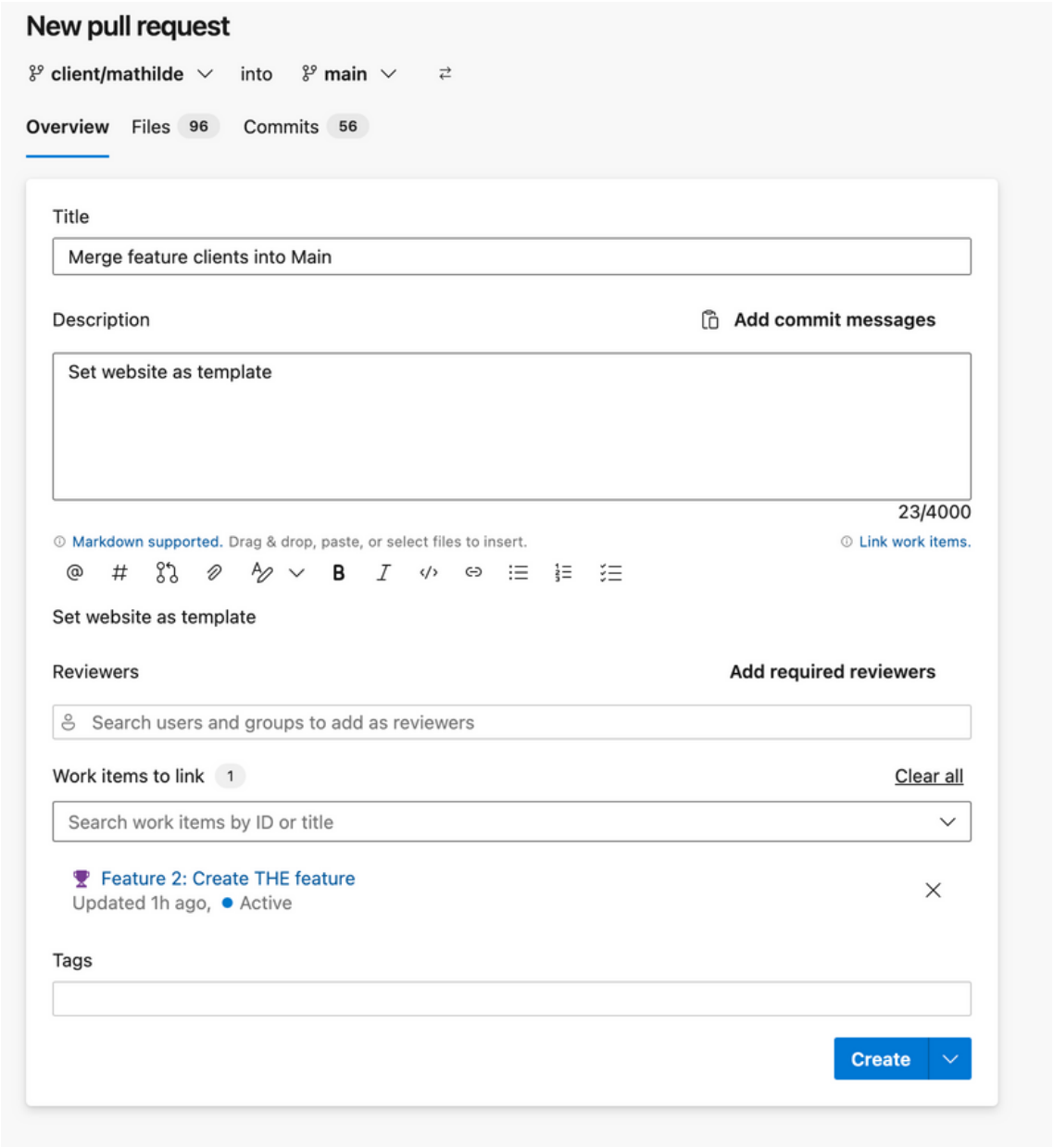
Security and Permissions

Branch Level: Azure Repos allows setting branch policies that can restrict changes to certain branches, require code reviews for pull requests, or enforce validation builds. This helps to protect important branches, like the main or release branches.



Pull Request

Pull requests in Azure Repos are a central mechanism to facilitate collaboration and code review within development teams using Git. They allow developers to notify team members that they have completed changes in a branch and request that these changes be integrated (or "merged") into the main branch or any other target branch. Here is a detailed overview of pull requests and their use in Azure Repos.



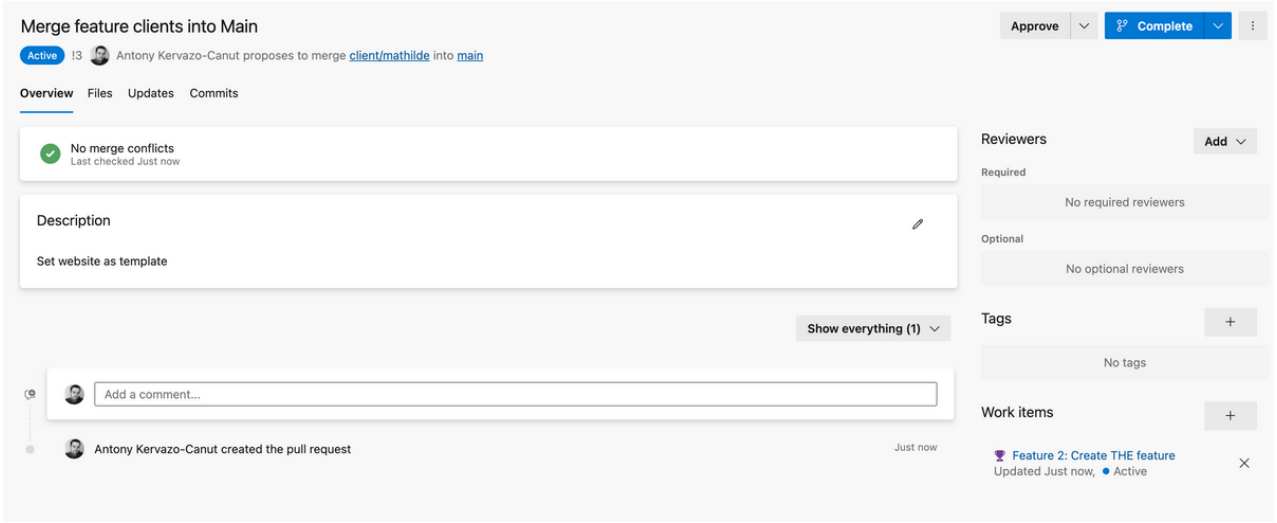
Pull Request

Code Reviews: Pull requests provide a framework for code reviews, where peers can comment and suggest changes before the code is integrated. This helps maintain code quality and share knowledge within the team.

Continuous Integration: With the integration of Azure Pipelines, you can set up automatic builds for pull requests, ensuring the code passes all tests and checks before merging.

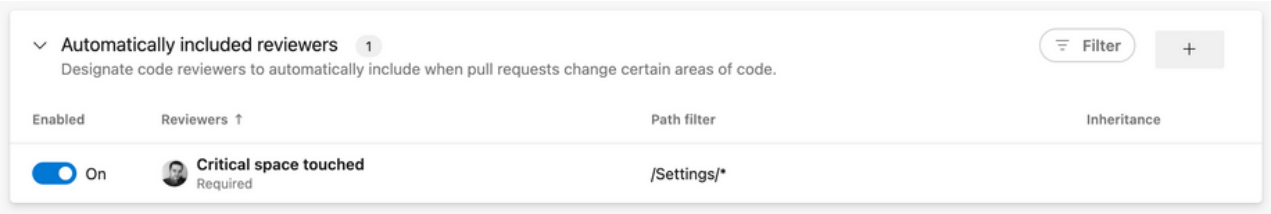
Branch Policies: You can apply branch policies to pull requests to require code reviews, successful builds, or other criteria before code can be merged, reinforcing quality standards.

Automation: Pull requests can trigger automated workflows, such as closing associated tasks in Azure Boards once the pull request is completed.



Pull Request

Tip :
If a specific part of the code has been touched, you can automatically include a reviewer or a specific team to validate the Pull Request.



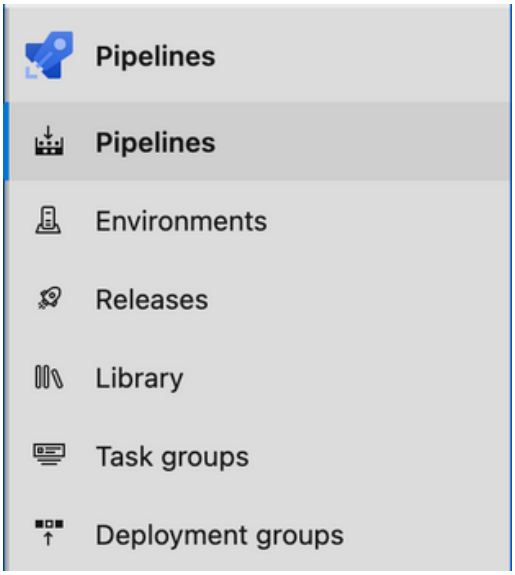
This technique can be used for triggering pipelines. Thus, certain tests, certain builds can be performed specifically only if certain parts of the source code are touched.
A greater separation that can save time on builds, or improve security and processes by only including individuals who are actually necessary for a review.

Azure Pipelines

Azure DevOps has undergone several evolutions in terms of pipelines, reflecting changes in software development practices and Continuous Integration (CI) / Continuous Delivery (CD). Here is a brief history of the different types of pipelines used over time in Azure DevOps (formerly known as Visual Studio Team Services (VSTS) and Team Foundation Server (TFS)):

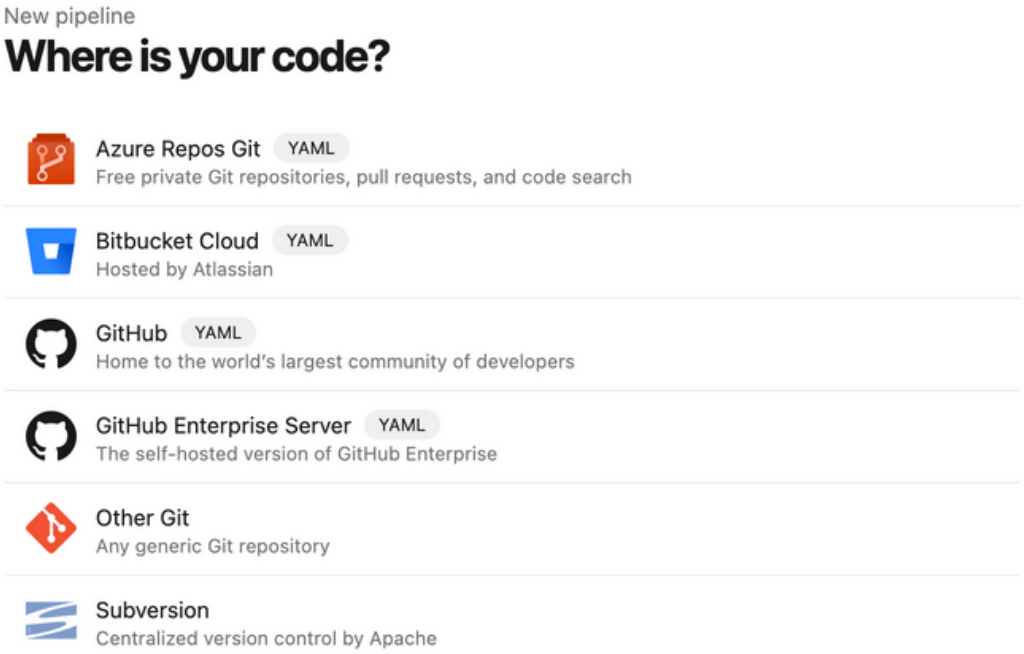
- XAML Pipelines, first generation, using XAML for configuration. Complex and difficult to manage.
- Classic Pipelines, configuration via a user interface without scripts. Accessible, easy to use.
- YAML Pipelines, configuration "as code" in YAML files. Flexible, versionable, and promote collaboration.

Azure DevOps still supports classic pipelines and XAML pipelines (although the latter is at the end of its life), but strongly encourages the use of YAML pipelines for most new projects.



External Sources

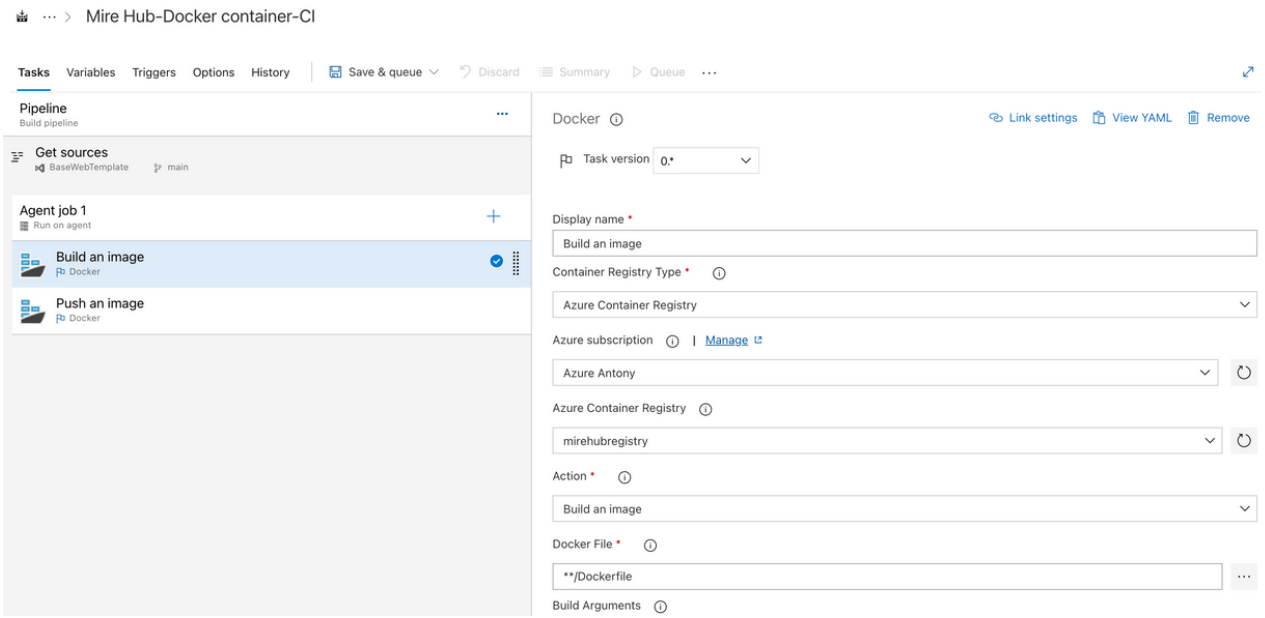
Creating a pipeline in Azure Pipelines is a flexible process that allows not only the use of code hosted on Azure Repos but also the connection to code repositories located on other platforms such as GitHub, GitLab, or Bitbucket.



If you choose an external code source, you will need to authorize Azure Pipelines to access your repository. This generally involves signing in to your account on the external platform and granting the necessary permissions for Azure Pipelines to read the code and configure webhooks for CI/CD triggers.

Pipeline Classic

Classic pipelines in Azure DevOps offer a graphical and user-friendly approach to configuring and managing Continuous Integration (CI) and Continuous Delivery (CD) without writing YAML code.



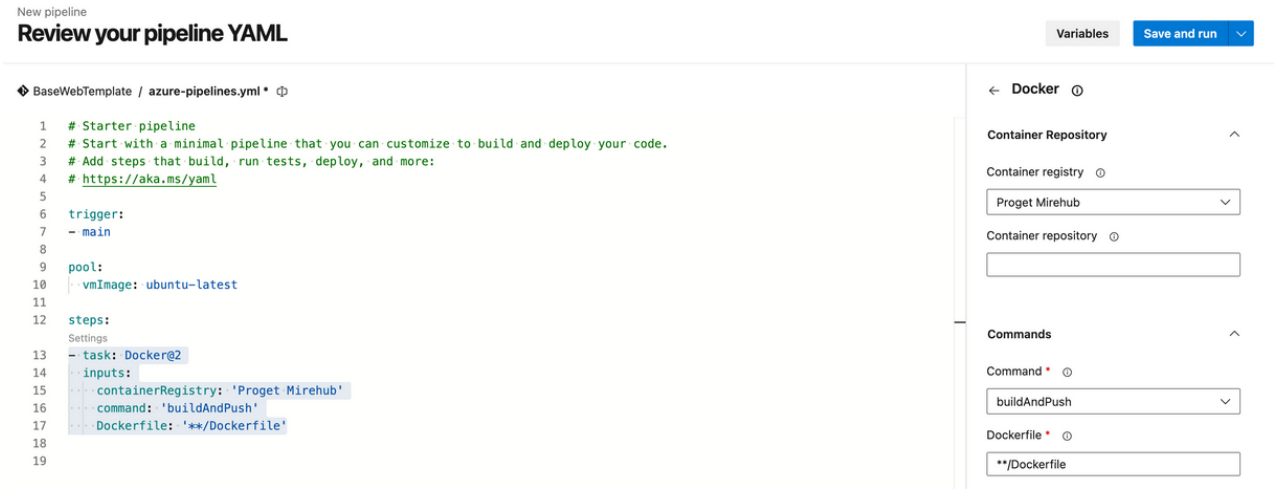
Classic pipelines have many screens for configuring variables, triggers, and managing sources. It's a historical model, easy to understand and very flexible.

Tip :
Azure DevOps offers the possibility to export an existing Classic pipeline to a YAML format to start migrating your old pipelines. However, you might encounter difficulties if you use features such as "Task Groups".

Pipeline YAML

YAML pipelines in Azure DevOps represent a modern, code-centric approach to defining Continuous Integration (CI) and Continuous Delivery (CD) processes. Using the YAML (YAML Ain't Markup Language), these pipelines allow for a declarative description of how to build, test, and deploy your code directly in a text file that can be versioned with your source code.

However, building pipelines within Azure DevOps also allows for having an assistant.



Changes made to the pipeline configuration are traceable through the version control system, offering full visibility into the evolution of the CI/CD process.

Tip :
By using branch policies, you can ensure to systematically include individuals who will validate the modification of the pipeline.

Agent Pool

Agent Pools in Azure DevOps are collections of agents that execute the tasks defined in your CI/CD pipelines. An agent is an application installed on a physical or virtual machine that runs builds, deployments, or other automated tasks defined in your pipelines. Agent pools allow you to manage and organize these agents into groups, facilitating the allocation of the necessary resources for pipeline execution.

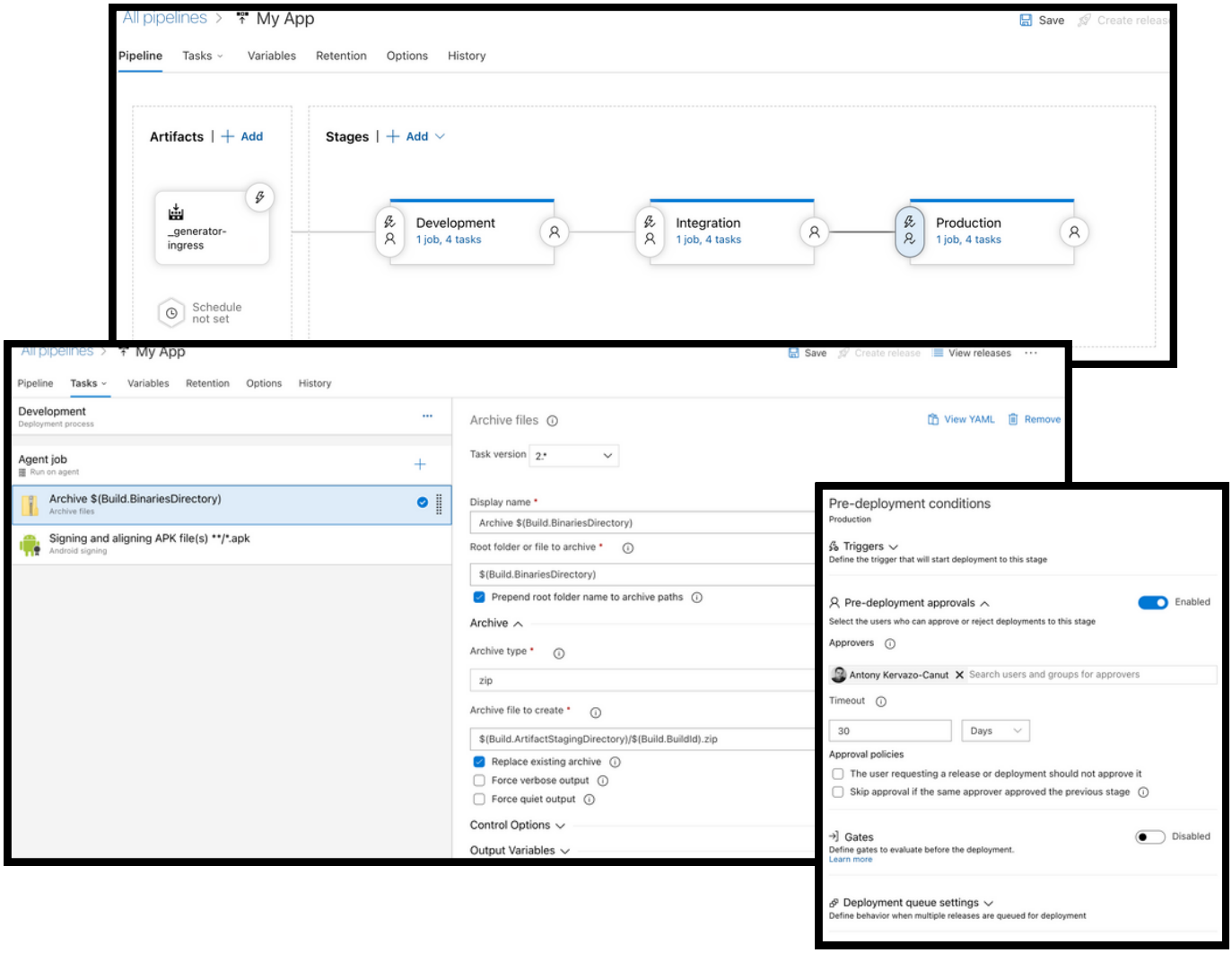
- Hosted Agents: Azure DevOps provides hosted agents, which are virtual machines managed by Microsoft with pre-installed development environments. These agents are ready to use and require no maintenance. There are several types of hosted agents to target different operating systems or configurations.
- Self-Hosted Agents: You can also set up your own agents on your machines. This is useful for scenarios where you need specific configurations, access to internal resources, or to use software that is not available on hosted agents.

The configuration of a self-hosted agent is done in the project settings.

Releases

Releases in Azure DevOps are a crucial part of the Continuous Delivery (CD) process, allowing for the management of application deployments across different environments, such as development, testing, and production. The management of releases in Azure DevOps is designed to automate, schedule, and track the deployment of your code, while ensuring the quality and stability of your applications.

Release pipelines are primarily configured through a graphical user interface, without a direct option to write the configuration in YAML.



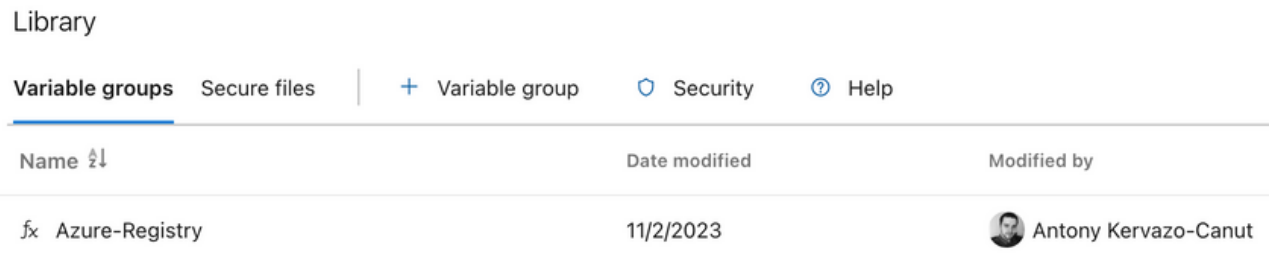
Multi-Stage Pipelines in YAML

With the introduction and evolution of multi-stage pipelines in YAML, Azure DevOps has begun to offer a unified experience for defining Continuous Integration (CI) and Continuous Delivery (CD) in a single YAML file. This approach extends the flexibility and power of YAML to deployment processes, allowing users to define deployment stages, jobs, steps, environments, deployment strategies, and other aspects of delivery in the same YAML file used for CI.

```yaml
trigger:
- main

stages:
- stage: Build
  jobs:
  - job: BuildJob
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - script: echo Building ...
    - script: dotnet build --configuration Release

- stage: Deploy
  dependsOn: Test
  condition: and(succeeded(),
eq(variables['Build.SourceBranch'], 'refs/heads/main'))
  jobs:
  - deployment: DeployJob
    environment: production
    pool:
      vmImage: 'ubuntu-latest'
    strategy:
      runOnce:
        deploy:
          steps:
          - script: echo Deploying to production ...
```

Library

The Library in Azure DevOps is a feature that allows storing, managing, and sharing variables and secrets across multiple pipelines, facilitating the reuse of configurations and centralizing the management of sensitive data.



Best Practices:

- Principle of Least Privilege: Limit access to variables and variable groups as needed. Only give access to team members who require this information for their work.
- Regular Review: Regularly review your variables and variable groups to ensure they are up-to-date and that only appropriate individuals have access.
- Use of Variable Groups by Environment: Organize your variables into groups specific to each environment to simplify management and reduce the risk of accidental deployment with the wrong configuration.

Variable Groups

Library > Registry

Variable group | Save | Clone | Security | Pipeline permissions | Approvals and checks

Properties

Variable group name

Registry

Description

Link secrets from an Azure key vault as variables

Variables

Name ↑	Value
Account	AccountName
Token	********

```yaml
trigger:
- main

variables:
- group: Registry

jobs:
- job: ExampleJob
  pool:
    vmImage: 'ubuntu-latest'
  steps:
  - script: echo Using account $(account)
    displayName: 'Display Account Variable'
```

Variable Groups

Library > Registry

Variable group | Save | Clone | Security | Pipeline permissions | Approvals and checks

Properties

Variable group name

Registry

Description

Link secrets from an Azure key vault as variables

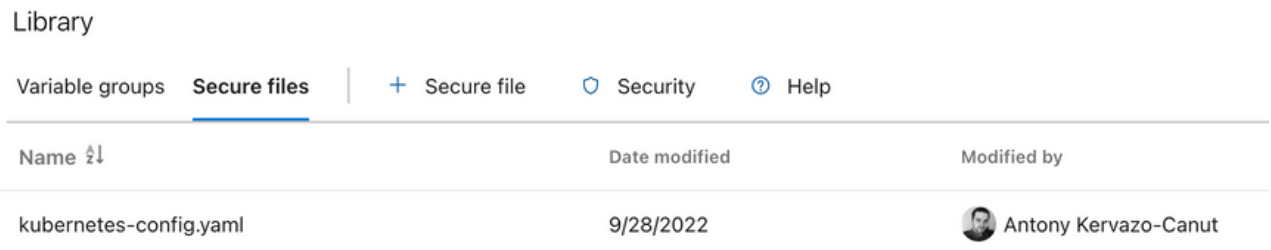
Variables

Name ↑	Value
Account	AccountName
Token	********

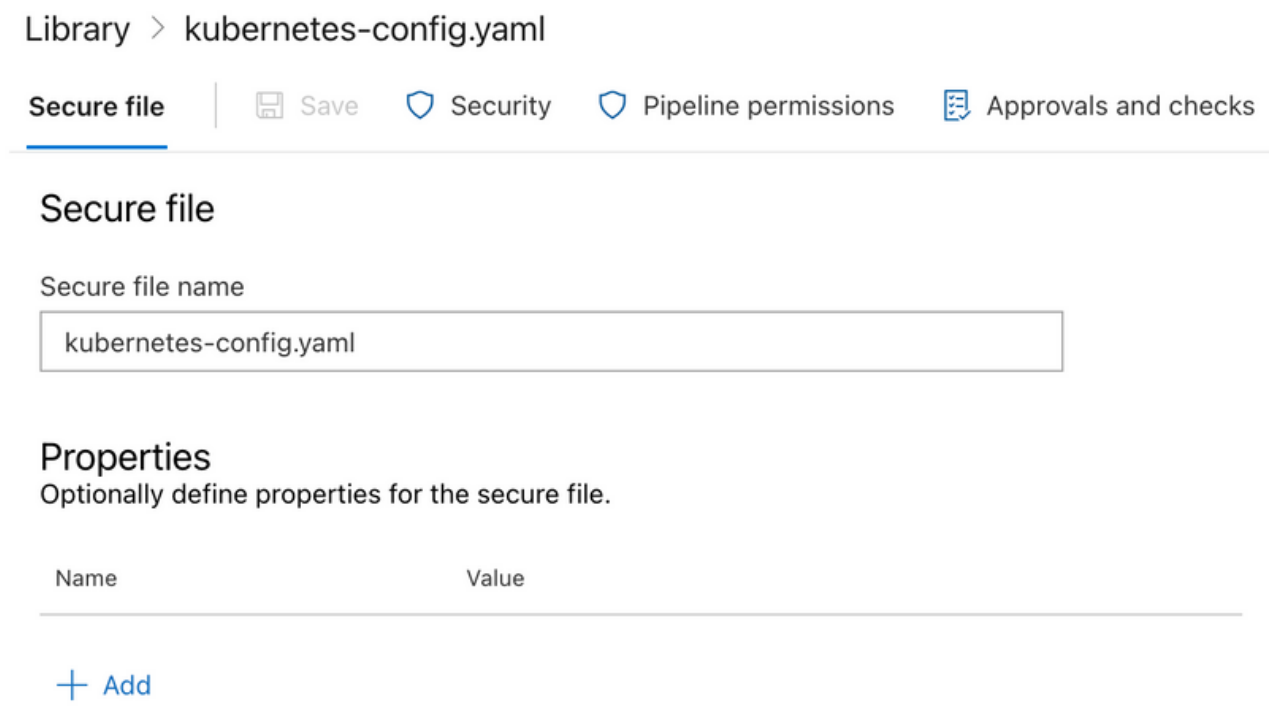
```yaml
trigger:
- main

variables:
- group: Registry

jobs:
- job: ExampleJob
  pool:
    vmImage: 'ubuntu-latest'
  steps:
  - script: echo Using account $(account)
    displayName: 'Display Account Variable'
```

Secure Files

The Secure Files section in Azure DevOps is a feature of the Library that allows storing and managing secure files, such as certificates, configuration files containing secrets, or any other type of sensitive file, which you can then use in your CI/CD pipelines. These files are stored securely and can be referenced by pipelines to automate deployments and builds while keeping sensitive information protected.



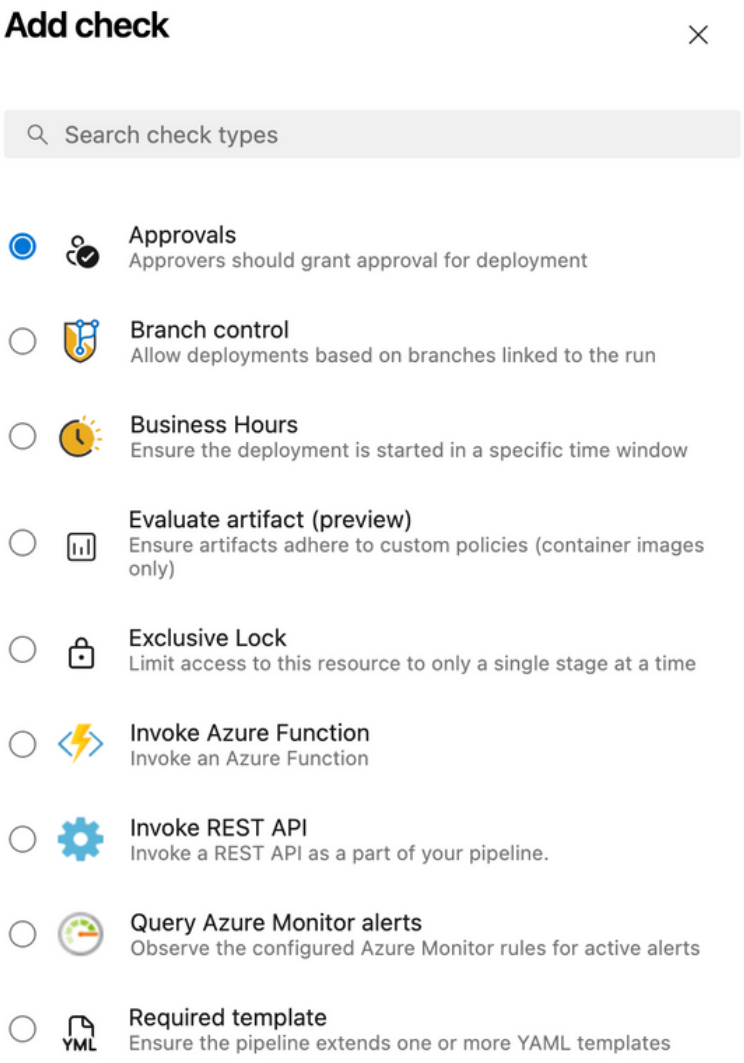
The file cannot be read from its interface but has permission management and access conditions.



Approuvals and Checks

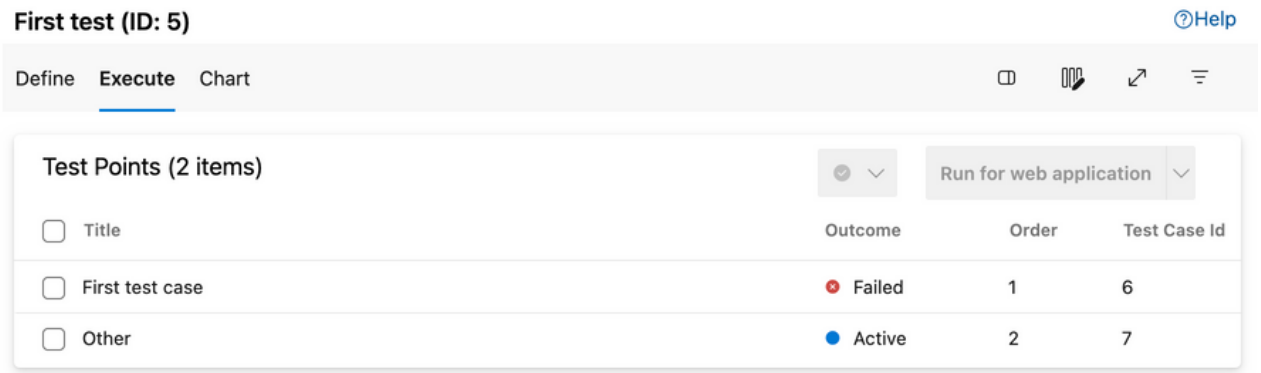
Variable groups or secure files can have usage conditions on pipelines. These can apply to specific branches, people who can initiate pipelines using secret elements, or even the schedule of use to avoid, for example, an awkward deployment during a specific time slot or day of the week.

Other conditions can even be written directly via Azure Functions or by calling REST APIs to obtain validation for the use of the secret.



Azure Test Plans

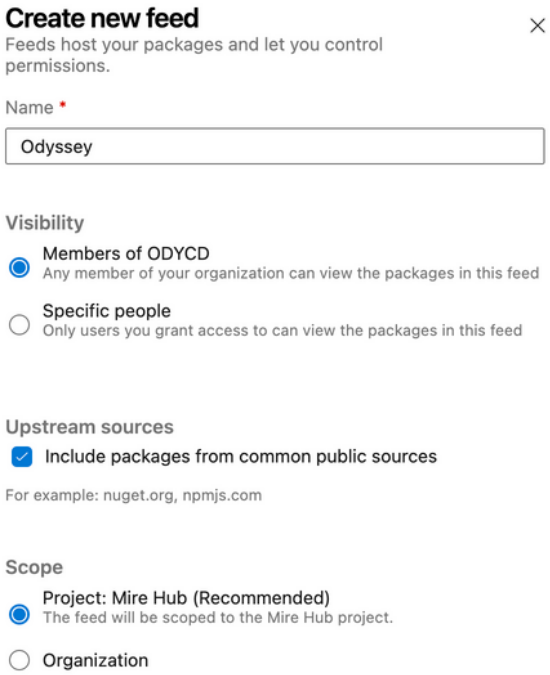
Azure Test Plans, a component of Azure DevOps, is a set of powerful tools designed for test planning, manual test execution, test automation, and tracking the overall quality of software within your development projects. Azure Test Plans supports an integrated approach to quality management, facilitating collaboration among developers, testers, and project managers to improve the reliability and performance of applications.



The free version of Azure DevOps includes access to Azure Boards, Azure Repos, Azure Pipelines, and limited capacity for Azure Artifacts. However, access to Azure Test Plans is more restricted.

Azure Artifacts

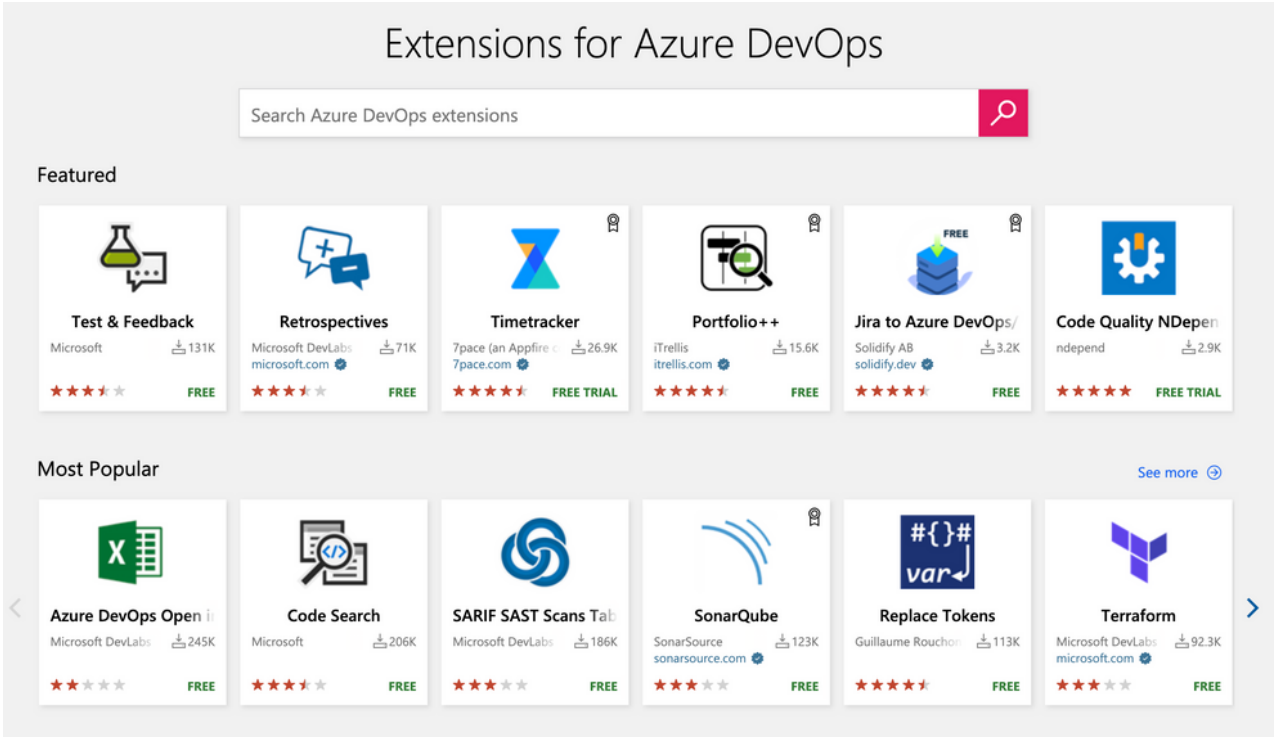
Azure Artifacts is a powerful feature for managing and sharing packages within your organization or with the community. A "feed" is a package repository where you can publish, consume, and share packages of different types. This facilitates collaboration among development teams and code reuse, while supporting DevOps practices by enabling smooth project dependency management.



Type	Source	Location
	npmjs	https://registry.npmjs.org/
	NuGet Gallery	https://api.nuget.org/v3/index.json
	PowerShell Gallery	https://www.powershellgallery.com/api/v2/
	PyPI	https://pypi.org/
	Maven Central	https://repo.maven.apache.org/maven2/
	Google Maven Repository	https://dl.google.com/android/maven2/
	JitPack	https://jitpack.io/
	Gradle Plugins	https://plugins.gradle.org/m2/
	crates.io	https://index.crates.io/

Visual Studio Marketplace

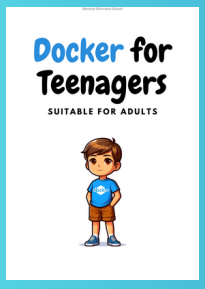
The Visual Studio Marketplace is an online platform where users can discover, acquire, and install extensions to enhance and customize their experience with Visual Studio, Azure DevOps, and GitHub products. It offers a wide range of extensions, from integrations with other services and tools, to productivity enhancements for developers, as well as dashboard widgets and custom tasks for Azure DevOps pipelines.

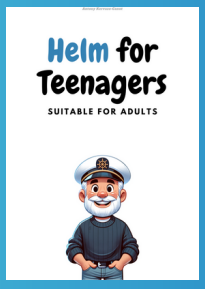


Visual Studio Marketplace offers extensions that can extend and customize the functionalities of Azure Boards, Azure Repos, Azure Pipelines, and more.

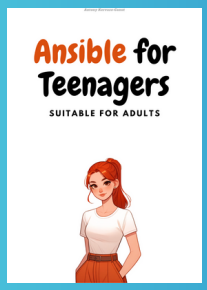
In the same collection

Container Orchestration and Management



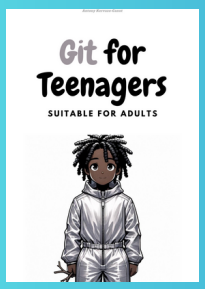


Infrastructure as Code



Security & Secrets Management

Development & CI/CD



↓ FOLLOW ME ↓

ANTONYCANUT

ANTONY KERVAZO-CANUT