Azure DevOps for Web Developers

Streamlined Application Development Using Azure DevOps Features

Ambily K K

Apress®

Azure DevOps for Web Developers

Streamlined Application Development Using Azure DevOps Features

Ambily K K

Apress®

Azure DevOps for Web Developers: Streamlined Application Development Using Azure DevOps Features

Ambily K K
Hyderabad, India

ISBN-13 (pbk): 978-1-4842-6411-9 ISBN-13 (electronic): 978-1-4842-6412-6
https://doi.org/10.1007/978-1-4842-6412-6

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Matthew Moodie
Coordinating Editor: Shrikant Vishwakarma

Cover designed by eStudioCalamar

Cover image designed by Pexels

Distributed to the book trade worldwide by Springer Science+Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6411-9. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

To Pranav & Pavitra,
my little mentors

Table of Contents

About the Author

For more than a dozen years **Ambily** has worked on cloud adoption and accelerating software delivery through DevOps. As the head of Azure, DevOps, and UI practices at TCS HiTech Industry, she supports major public- and private-sector companies across the globe in their cloud journeys and DevOps implementations. Ambily blogs about her experiences and speaks at conferences to share what she has learned. You can find her blog at https://ambilykk.com/.

About the Technical Reviewer

Swapneelkumar Deshpande is a software engineer. He is a Microsoft Certified Professional and Microsoft Certified Trainer. Swapneel has been working and leading teams in various aspects of the software development life cycle for more than 20 years.

Acknowledgments

Writing a book is harder than I thought and more rewarding than I could have ever imagined. I'm grateful to my teachers, and I've been lucky enough to have a lot of great teachers, but one in particular shaped the person I am today: Minu K K. My heartfelt thanks to him for being an inspiration throughout my life.

I am grateful to my loving husband, Rajeev, for all his support and caring. To my little children, Pranav and Pavitra: thank you for letting me spend time on writing.

A very special thanks to my mentor, Bala Peddigari, who inspired me to write articles and books.

Finally, to all those who have been part of my getting there: Smriti Srivastava, Matthew Moodie, Shrikant Vishwakarma, and Swapneelkumar Deshpande.

Introduction

DevOps is one of the mandatory elements of the entire IT spectrum, which integrates people, process, and technology with an objective of delivering value faster. At the same time, DevOps comes in many varieties with varying levels of understanding, which can be an obstacle for many implementations. In this book, we will explore the various concepts of DevOps and take you on a DevOps journey using Azure DevOps. We'll look at the end-to-end process of DevOps implementation using Azure DevOps.

Specifically, the following are the topics covered:

- Project management including user and permission management
- Requirement management
- Version control management
- Test management
- Build automation and release management
- Test management
- Continuous feedback

Also, we will discuss the current IT landscape and how we can leverage the benefits of DevOps in various areas such as support projects. Every concept will be covered from different perspectives, including a novice view, a DevOps developer view, a DevOps architect view, and a business view.

This book is intended for novices in DevOps who want to learn the concepts of DevOps across a modern application spectrum. Also, it provides enough information for DevOps engineers to fine-tune their skills in various aspects of DevOps.

CHAPTER 1

DevOps Basics and Variations

Software development processes and the release requirements can change over time in order to continue bringing value to customers and gaining market adoption. To scale up and get some early feedback, many organizations have adopted a frequent release methodology for new features or defect fixes. These releases are deployed to the production system as often as once a week, and sometimes they are even once a day, once an hour, or even every ten seconds. To maintain this edge in the market or gain this quick feedback cycle, companies are adopting DevOps practices such as continuous delivery and continuous deployment. In this chapter, you will get an introduction to DevOps concepts and practices.

Software Development

The software development industry gradually adopted the agile model after many years of following the waterfall model. The agile development model provides the ability to adapt to changing requirements, and it offers better collaboration between teams and improved productivity. In a waterfall model, application development is done as a sequential operation, and the application is released at the end of the project. Sometimes, though, the development process takes a long time to complete, and the relevance of lot of features envisioned at the start may not be useful anymore. Moreover, this approach lacks the end-user feedback loop, the expectations of the end user and any feedback will be received only after the entire development/deployment is complete.

The agile process demands the release of finished features (production-ready) more frequently to receive early feedback from customers and respond accordingly. Agile without DevOps takes more cycle time and manual involvement. DevOps addresses the

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_1

shortfalls in agile development and deployment practices using automated releases, frequently with automated validations and gates in between. Figure 1-1 shows all three models.

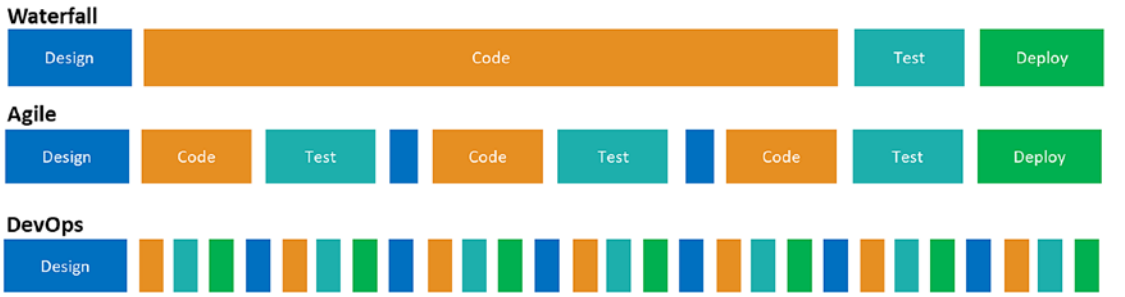


Figure 1-1. *Software development models*

DevOps

DevOps is one of the hottest topics in the IT field, but it has many different variations. If ten people started discussing DevOps, they would have ten different explanations for it. Here is a popular definition of DevOps from https://en.wikipedia.org/wiki/DevOps shows the process angle of DevOps, in other words, how we can use the single process across the development and operations teams:

> *DevOps is a software development methodology that combines software development (Dev) with information technology operations (Ops).*

The following definition from https://searchitoperations.techtarget.com/definition/DevOps. DevOps is defined from a person's perspective, specifying how the team culture changes and enables more collaborative working environments.

> *DevOps is the blending of the terms development and operations, meant to represent a collaborative or shared approach to the tasks performed by a company's application development and IT operations teams.*

In the following definition, defines DevOps from a technical perspective, where DevOps is for automating end-to-end development processes (www.atlassian.com/devops):

> *DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably.*

Considering the importance of DevOps in the IT industry and the fact that DevOps is an integral part of software development, let's discuss these different possibilities of what it encompasses.

DevOps is a combination of cultural elements, practices, and tools for delivering value at a higher velocity. DevOps consists of people, process, and technology. Technology enables the automation of an end-to-end pipeline for delivery. Moreover, using the proper tools increases a team's collaboration. The agile process provides agility in execution, collaboration between teams, management of end-to-end activities, and an innovative team culture. DevOps culture means people are trained on multiple skills to improve the breadth of understanding on the overall process, a proper handshake between the teams, a single team mindset, and multiskilled team members.

As part of DevOps adoption, most companies start with or focus on the technology and process implementation. Agile is the de facto development methodology for many companies, and automation is the driving force for DevOps adoption. DevOps Culture adoption is complex, and it requires that changes in organizational structure and that people's mindsets change.

DevOps Team

If DevOps is a mixture of development and operations, then what about the test department and other stakeholders?

All the stakeholders have their own importance and roles in DevOps. The *Dev* in DevOps includes the scrum masters, developers, testers, project managers, subject-matter experts (SMEs), business analysts, domain experts, security architects, solution architects, performance engineers, and so on. The roles and responsibilities differ from project to project based on the size, complexity, and business impact of the solution. The *Ops* in DevOps includes the operational members such as the build engineers, production support, monitoring, and so on.

Remember that the goal of DevOps is to deliver value to the customer, not to reduce the cost or automate the process; these are the by-products of enabling proper DevOps practices. Team structure plays a major role in driving the DevOps practice in an organization. Based on various parameters, we can identify a matching team topology for our project. Identify whether you are following an anti-type or a correct DevOps team type by reading the article by Matthew Skelton https://web.devopstopologies.com/.

DevOps Practices

DevOps involves different components to establish an end-to-end DevOps workflow. These are some of the main practices:

- *Requirement management*: Establishing an end-to-end tractability, quick view, validity, and usability priority on the status of each of the requirements.
- *Configuration management*: Managing the configuration based on the target environments, users, company policies, etc.
- *Release management*: Planning, scheduling and controlling the releases, approvals, and other policies.
- *Continuous integration*: Merging the latest updates to remote repositories and ensuring that the quality builds are passed.
- *Continuous deployment*: Deploying the code changes to higher environments after ensuring the quality of the build through automated quality gates.
- *Infrastructure as a code*: Provisioning the infrastructure as part of the code deployment and controlling the configurations in script format.
- *Test automation*: Automating tests to ensure faster execution and more coverage.
- *Continuous monitoring*: Systems to automatically monitor and act on the result. This can include logging the bugs, doing corrective actions, or sending notifications.

There are many more areas such as collaboration, knowledge management, identity/access management, portfolio management, artifacts repositories, performance testing, security test, backlog management, and so on.

DevOps Variations

DevOps is an integral part of software development. Most projects implement at least the technical and process parts of DevOps, even though this may not result in the actual benefits of DevOps. Culture changes really help you to realize the real benefit of DevOps.

But, most of the time, we end up with limiting our DevOps scope to automation and cost reduction instead of releasing value.

Currently, many variations of DevOps exist based on the implementation. Some of the main variants are as follows:

- *BizDevOps*: DevOps with the involvement of the business view throughout the process. Even though regular DevOps defines the participation of the business stakeholders in DevOps, this variation emphasizes the role of the business people in the overall execution.
- *DevSecOps/SecDevOps*: This integrates security into DevOps. This is the current trend, and most customers are adopting this model now. Why security? We have security implemented in our code, so why do we need security as part of our DevOps implementation? The following are some scenarios in traditional DevOps where security threats can occur:

  a) In a DevOps implementation using multiple tools, there will be requirements for integration between the tools through passwords, secrets, tokens, etc., which can be stored in secure vaults.

  b) If we have enabled the CI/CD pipeline, which performs the code commits and deploys to production, then how do we ensure the code getting deployed to production is secure enough? We do this through automated security verification tests.

  c) DevOps talks about shifting left, so why do we need to wait for a build to explore our security vulnerabilities? Can we have an IDE integrated with security vulnerability features at the time of coding? If so, we can use security IntelliSense.

  d) How do we secure a cloud subscription? This is called *subscription security*.

  e) We monitor the security issues in a production environment and take the necessary actions. Security monitoring for the applications deployment in the containers adds another level of security requirement. This is called the *hosting environment/container security*.

SecDevOps tries to address the end-to-end security needs of our DevOps implementation. Based on the solution technology, hosting environment, and DevOps tools, we can integrate the proper security tools throughout the DevOps life cycle to implement SecDevOps.

- *CloudOps*: A variation of DevOps that targets cloud-based systems. The main difference is the implementation of infrastructure as code and handling the SaaS and PaaS deployments.
- *MLOPs*: DevOps is an essential component in machine learning (ML) model development. There is a slight change in the MLOps implementations compared to the other cloud-based application developments. In MLOps, the stages will be to import the data from various sources; do data processing including data transformation, data cleaning, etc.; train the model using a sample data set; deploy the model; and pass the actual data for processing. Also, it may require retraining of the model as part of the end-to-end orchestration. MLOps is handled by specific DevOps tools such as MLFlow and Azure ML Pipeline.
- *Intelligent DevOps*: This is the future of DevOps and is a collaboration between DevOps and artificial intelligence. In DevOps, the main three components are the people, process, and technology. A lot of data will be generated as part of the DevOps life cycle, but not consumed properly. Intelligent DevOps use the data collected in DevOps setup likes the logs, code, test cases, bugs, build reports, etc to predict or derive more meaningful insights. Deployment failure prediction, quality risk prediction, identification of new test scenarios, and defect prediction based on log analytics are some of the insights we can derive from this DevOps data.

The DevOps process has evolved over time from DevOps 1.0 to intelligent DevOps.

DevOps 1.0

In DevOps 1.0, the main focus is to resolve the conflict and increase the collaboration between various stakeholders such as the development team, QA, and operations, as shown in Figure 1-2. Bridging the gap in understanding the priority of the other team played a major role here.

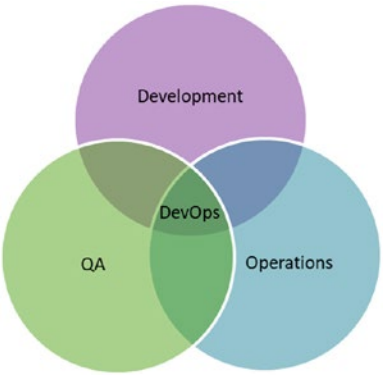


Figure 1-2. *DevOps 1.0*

In this scenario, mainly the development team focuses on new features to improve the business, whereas the Ops team looks for stability in the production environment. Figure 1-3 shows some of the conflict or deviations between the dev team and the ops team.

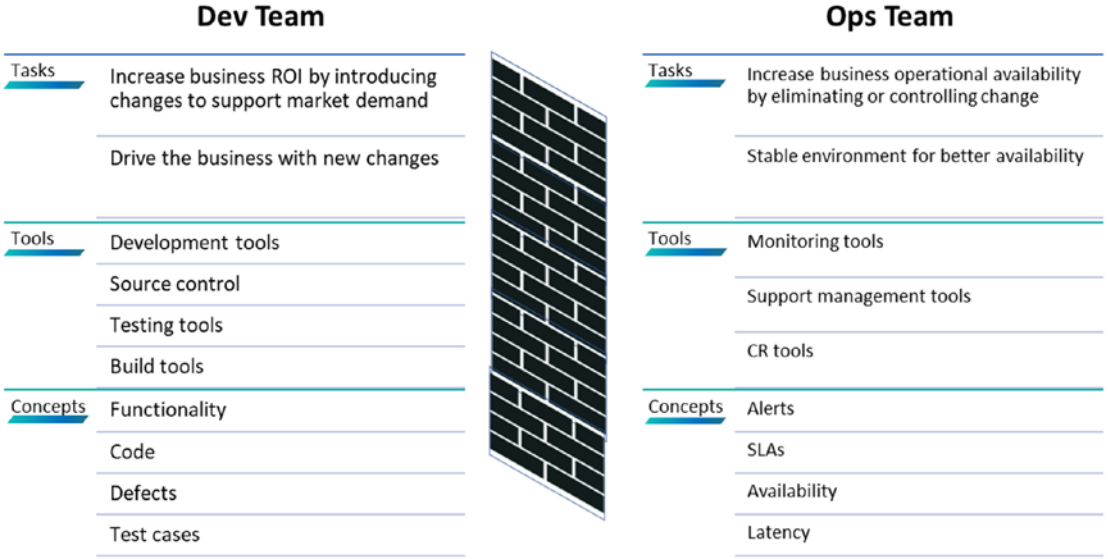


Figure 1-3. *Dev team versus the ops team*

DevOps 2.0

DevOps 2.0 focused on the user-centric deployments, as shown in Figure 1-4. This is especially true for SaaS or products where the preview features are released to a set of users who give feedback. Based on the feedback, the team incorporates the changes and releases a final version to all the users.

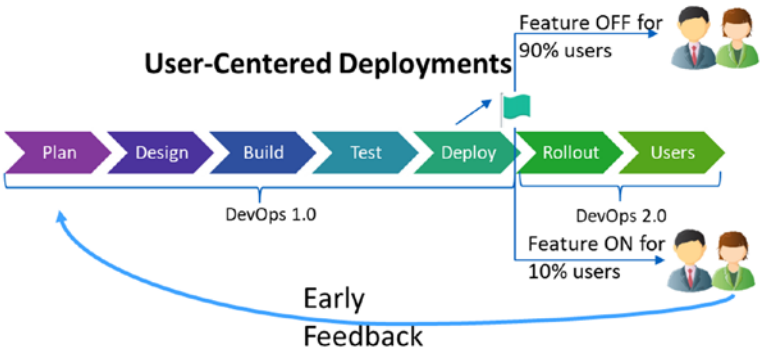


Figure 1-4. *DevOps 2.0*

This kind of DevOps implementation is handled using feature flags. Feature flags are the implementation of labeling the features and releasing them to selected users only.

SecDevOps

Security has become one of the main focuses in a DevOps implementation because of the recent remote work culture and cloud adoption. More and more DevOps tools are moving to a cloud-based subscription model to address the growing demand of cloud adoption. Another reason for security by design is the adoption of open source libraries in application development. For example, most modern web UI development is based on Angular or React, JavaScript frameworks are based on npm packages. Lots of npm packages are used along with the base package to design the rich UI. Deploying all these dependent libraries without proper screening will lead to security threats.

Intelligent DevOps

In intelligent DevOps, the team will look into implementing different intelligent models to derive proper insights from the data collected throughout the DevOps life cycle. Here, data will be added as a fourth element to the DevOps ecosystem. Figure 1-5 shows an example of intelligent DevOps.

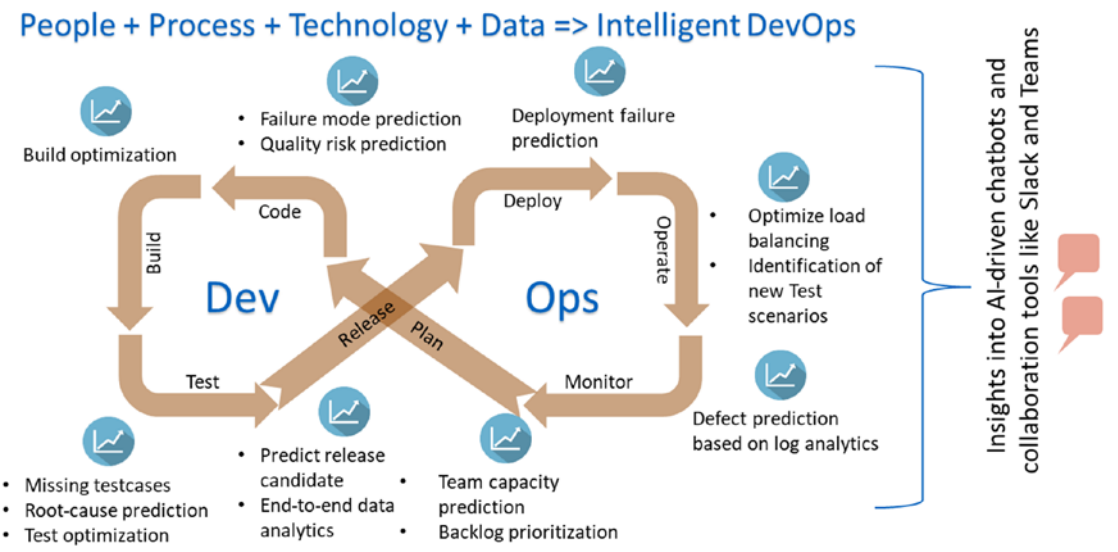


Figure 1-5. *Intelligent DevOps*

The DevOps model has become an essential part of modern application development and supports the automation of the agile process. A traditional DevOps model talks about people, process, and technology. But, the entire software/product/application development process involves a lot of data, specifically related to requirements fulfillment, team capabilities and sprint execution, build failures and quality of builds, untested areas, test failure patterns, candidate features for release, A/B release plans, operations aspects related to load balancers, exceptions, defects, and so on. Moreover, end customer tickets/feedbacks and monitoring trigger new data related to expected or broken features, expected defects, or anomalies. New DevOps models driven by AI focus on various insights from all this data. Intelligent DevOps tries to provide more proactive information to the stakeholders, at the same time tries to address some of the defects using self-healing mechanisms. Collaboration tools integrated with chatbots will add faster actions based on insights.

Benefits

The benefits of DevOps include building and shipping software changes faster to the market, from idea to release and through automation, orchestration of software delivery processes, and continuous customer feedback. DevOps continuously delivers value to end customers with lots of added benefits in terms of fewer test cycles, increased

ticket resolutions, reduced end-to-end effort, and so on. The transition from traditional DevOps to intelligent DevOps will bring exponential value to end users and businesses by combining the power of DevOps and AI.

The following are some of the DevOps benefits in the business area:

- More responsiveness to business needs
- Improved visibility into IT process and requirements
- Increased customer satisfaction
- Increased sales
- Lower risk

The following are the technical benefits:

- Improved Quality with code
- More Agile development
- Improved quality of deployments
- Faster release cycles
- Reduced complexity

The following are the team benefits:

- Collaborative culture
- Productive teams
- Higher team engagements
- Increased Growth opportunities
- Self driven team

Azure DevOps

Azure DevOps provides two offerings: Azure DevOps Server and Azure DevOps Services. Azure DevOps Server, previously known as Team Foundation Server (TFS), is the on-premise offering from Microsoft. Azure DevOps Services, formerly known as Visual Studio Team Services (VSTS), provides a SaaS-based offering to manage the end-to-end DevOps life cycle. This book will focus on Azure DevOps Services only.

Azure DevOps Services provides a platform for implementing the DevOps process across different IT segments. This tool supports the various practices under DevOps such as continuous planning, continuous development, continuous integration, continuous testing, continuous deployment/delivery, and continuous monitoring/feedback. Moreover, this tool supports integration with various tools such as code analysis tools to verify the quality of code, security tools to scan the vulnerabilities in code, infrastructure provisioning tools to automatically provision infrastructure components, and so on.

Summary

DevOps is an integral part of modern application development irrespective of commercial off-the-shelf (COTS) products or bespoke applications, machine learning models or web applications, and database or serverless models. This book will take you through the DevOps implementation journey using Azure DevOps. It will also address some of the high-level DevOps blueprints at the end.

CHAPTER 2

Project Management Using Azure DevOps

The field of project management has evolved over the years from the management of resources to include areas such as optimization and productivity boosters. Project managers have always managed resources and maintained schedules, but today's project managers add multiple dimensions such as agility and technology to these regular duties. Technical managers are the new norm to handle the growing adoption of agile and DevOps practices and automation. The collaborative and self-driven culture of the modern workforce provides more room for project managers to optimize their activities.

Azure DevOps complements project management activities by facilitating more automated features for the managers to track the work, optimize areas, and tightly control and maintain schedule. Project execution means creating a project, managing the end-to-end development cycle of the project, and onboarding teams with the proper permissions. In Azure DevOps, projects related to the same domain or business unit are grouped under an *organization*. Therefore, an organization is a collection of projects. In earlier versions, this was called a *project collection*.

As discussed in Chapter 1, an enterprise can have multiple Azure DevOps subscriptions to keep the billing related to each business unit separate, as shown in Figure 2-1. Multiple organizations can be configured using one subscription to capture the billing on a granular level or sub-business unit level. Organizations host related projects under that, which supports the execution of a project from requirements gathering until the deployment.

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_2

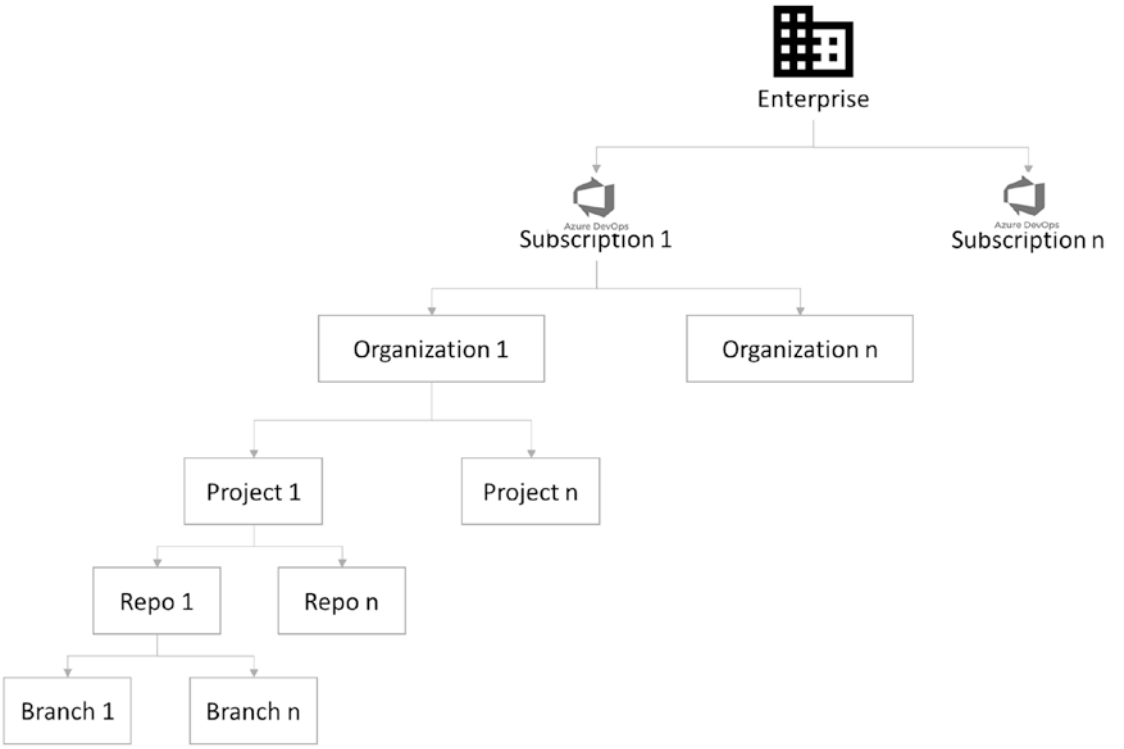


Figure 2-1. *Organization structure*

Each project can have multiple repositories to hold the codebase in separate areas. For example, a project handling the microservices will have one repo for each microservice. The repos will be further divided into branches to manage the code versioning and collaborative work. This chapter covers the high-level features of organizations and projects and touches upon a few configurations related to repos and branches. Repos and branches will be covered in more detail in Chapter 4. Also, most of the configurations discussed in this chapter will be revisited in later chapters based on their importance when setting up the end-to-date DevOps configuration.

Organizations

Organizations are the highest level of aggregation in Azure DevOps. From an enterprise perspective, organizations can be used to group the user licenses, the billings for a set of related projects, connections to an enterprise Azure AD, and global notifications and policies. More features of organizations will be discussed later in this book.

Creating an Organization

You can create a free organization by navigating to the Azure DevOps site: https://azure.microsoft.com/en-us/services/devops/.

Select the option "Start free" to activate a free instance of Azure DevOps. Azure DevOps supports using a Microsoft ID for login, where you can provide an existing live email address (`@hotmail.com`, `@outlook.com`, `@live.com`, etc.) or create one to proceed.

Once the login completes, Azure DevOps prompts you to continue with organization creation.

Provide an organization name to continue the journey. Once the organization is created, the system will prompt you to create a project, as shown in Figure 2-2. Create a new project with the minimum details.

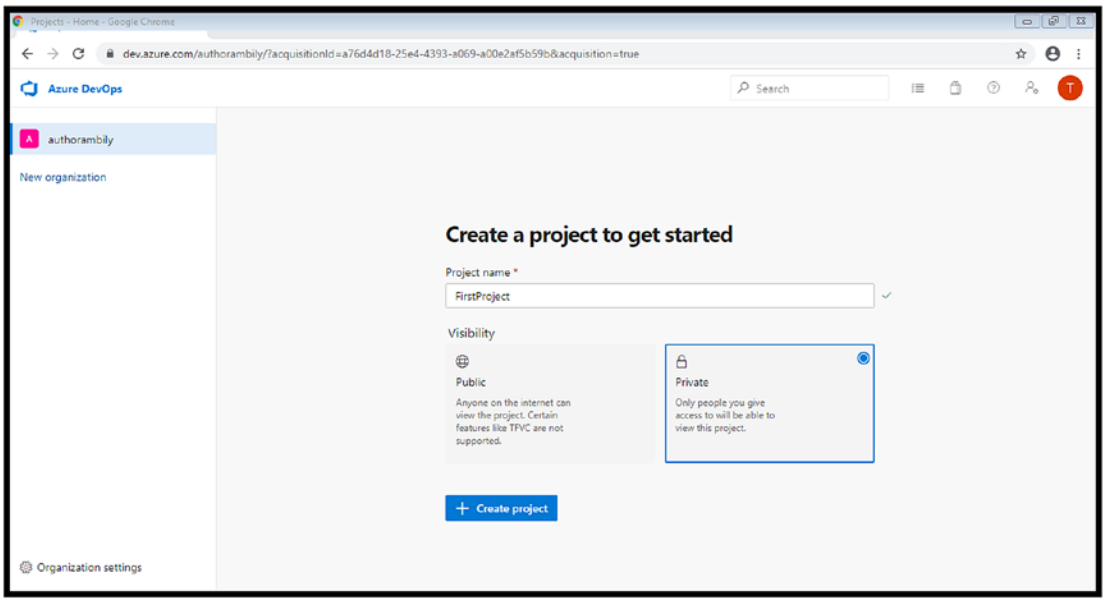


Figure 2-2. *Creating a new project*

Organization Settings

The organization home page shows the list of existing projects in the organization, "My work items" tab, and "My pull requests" tab as shown in Figure 2-3. Hovering the cursor over each of the project tiles allows users to navigate to the selected project's components such as the repo, board, pipeline, etc. The "My work items" tab displays all

the work items assigned to that user irrespective of the project. If a user contributes to multiple projects, this view provides a unified look at all the work items assigned to the user. Also, this view lists the activities related to the user. The "My pull requests" tab lists the pull requests (PRs) created by the user. More about the work items and PRs will be covered in subsequent chapters.

The organization settings allow you to configure a set of features required for the organization. Click the "Organization settings" option in the lower-left corner of the organization's home page to view the settings, as shown in Figure 2-3.

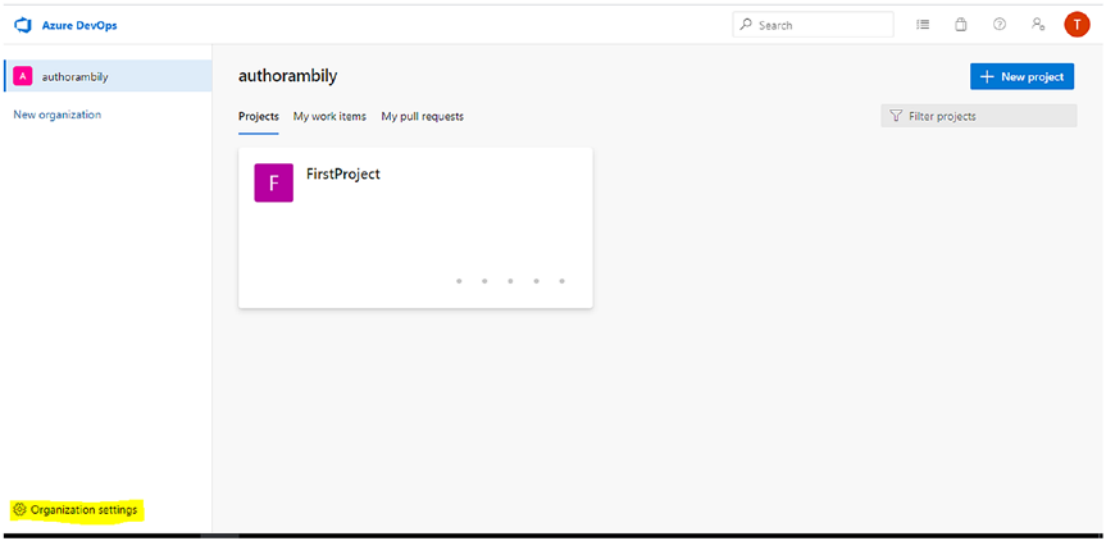


Figure 2-3. *Organization settings*

Briefly take a look at the different settings available in the organization settings. These settings are used as part of project execution. For example, the settings under Boards ➤ Process will be used to customize the project template when setting up the execution process. Similarly, the settings under Security help us to define organizational policies as part of the build and deployment operations.

General ➤ Overview

The Overview section allows you to manage the general properties associated with an organization such as the name of the organization, privacy URL, time zone, and description.

Scroll down in the Overview section to view the two main activities you can perform: changing the owner and deleting the organization, as shown in Figure 2-4.

- *Change owner*: This option is used to transfer the owner permission to another member of the organization.
- *Delete organization*: Remove the organization and associated projects. Only the owner can delete the organization along with all artifacts associated with it.

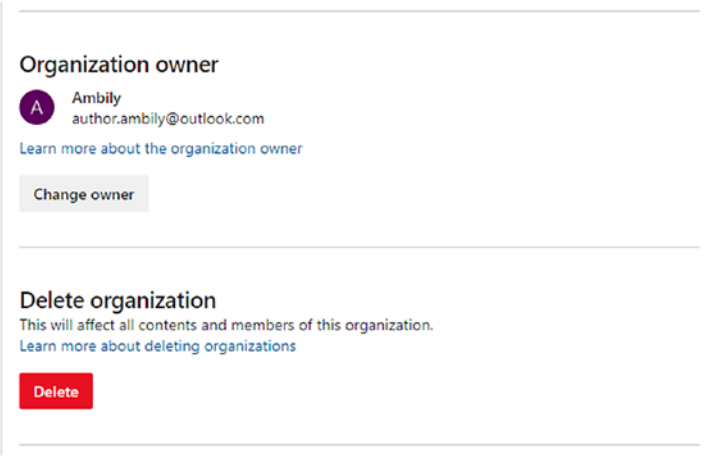


Figure 2-4. *Changing the owner*

General ➤ Projects

This section lists the projects associated with the organization. In this section, we can create a new project, delete a project, or rename an existing project, as shown in Figure 2-5.

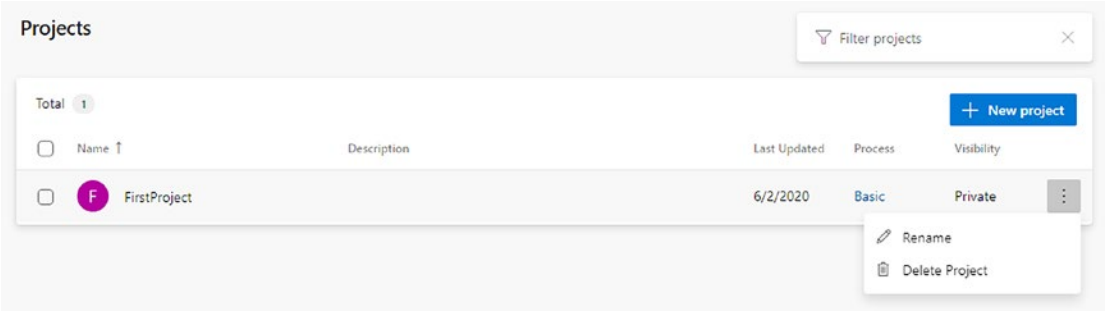


Figure 2-5. *Organization Settings ➤ General ➤ Projects*

General ➤ Users

This section lists the users and their access levels. Azure DevOps defines three access levels for users.

- Basic
- Stakeholders
- Visual Studio Subscriber

Note Only five Basic users are allowed for the free organization setup; but you can add unlimited stakeholders to your project.

Basic users have permission to access all the DevOps components, all the way from boards until the pipeline. This doesn't mean that access to all projects are allowed. The permission is there, but access is provided based on the access provided in each project at a granular level. Stakeholders have access to boards to view the backlog items, progress, sprint execution, and dashboards. They won't have access to the code repositories or the build and release pipeline. Visual Studio Subscriber has the same access as a Basic user but with an attached Visual Studio license. You can find details about the permissions and access levels at https://docs.microsoft.com/en-us/azure/devops/organizations/security/access-levels?view=azure-devops.

Add a few users as Basic users and a few users as Stakeholders using the Add Users option.

The Add Users option allows us to onboard new team members to the organization with specific permissions, as shown in Figure 2-6. If there is an existing project available, at the time of onboarding, provide the permission to different projects using Azure DevOps groups. Azure DevOps groups are used to group team members and assign specific permission. You can learn more about groups in the section "Security ➤ Permissions."

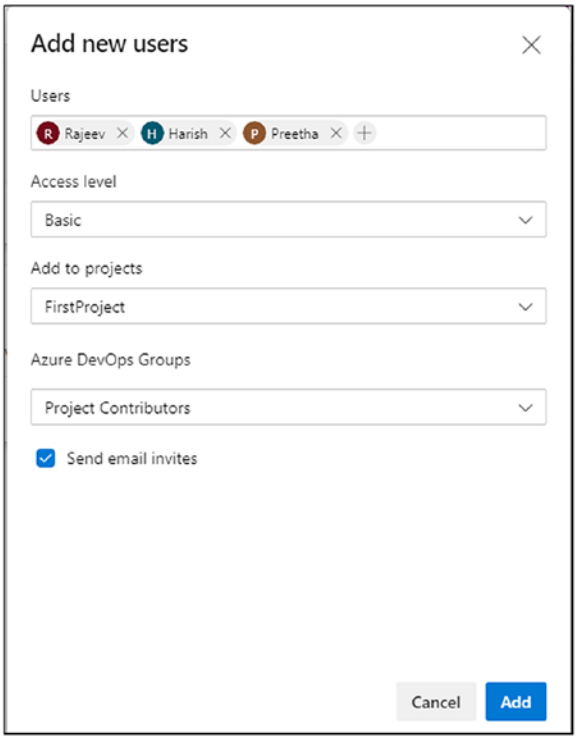


Figure 2-6. *Adding new users*

In the Users section, you can change the user access levels. This section allows you to reassign a user license to a new user, remove an existing user from the organization, or resend the invite to join the organization. Moreover, the Manage User option allows you to track the overall user permissions.

General ➤ Billing

The Billing section shows the billing of this subscription. To explore the various features of Azure DevOps, a free trial account is configured here. The billing shown in Figure 2-7 corresponds to the free trial account.

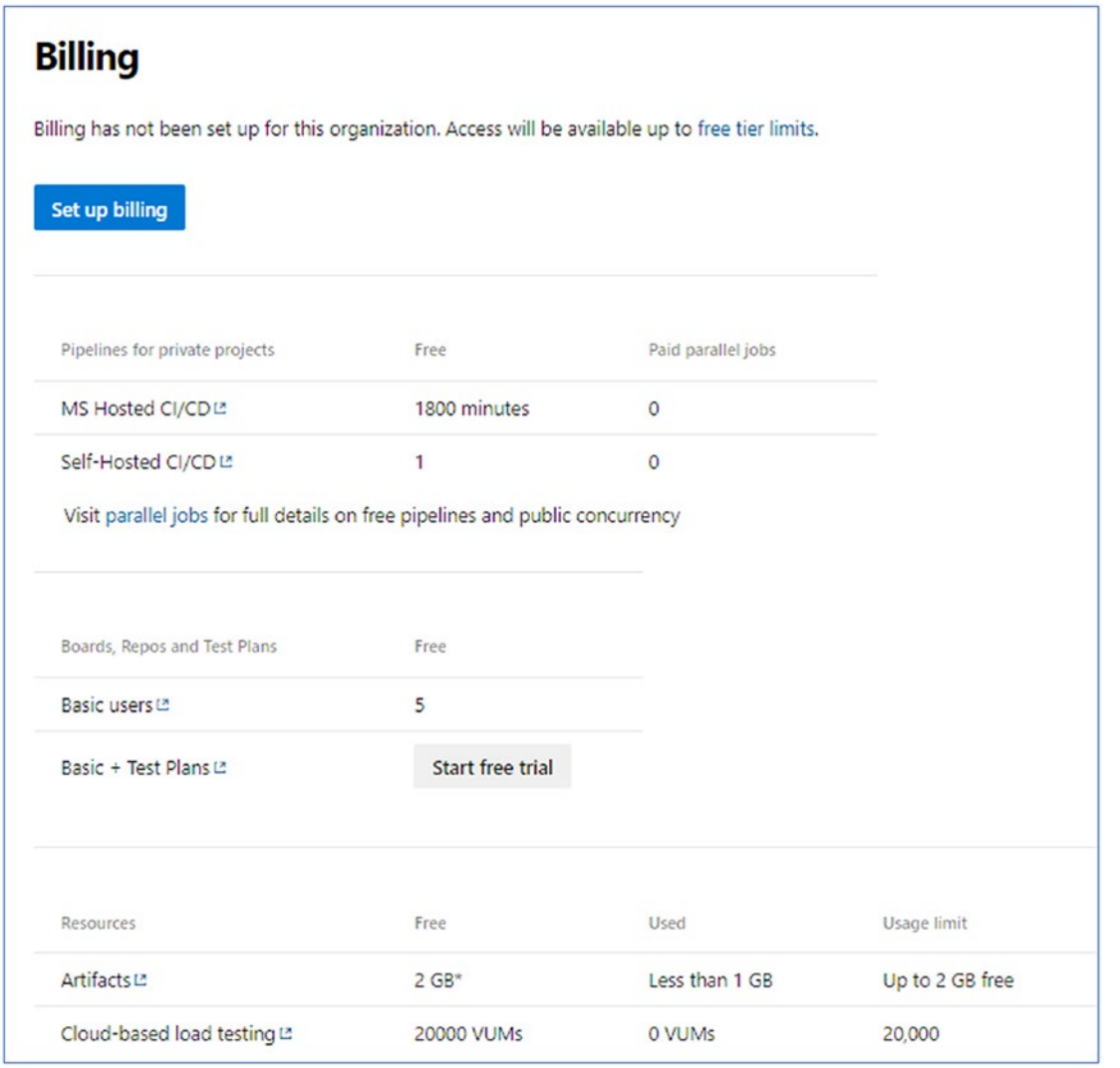


Figure 2-7. *Billing*

General ➤ Auditing

The Auditing section lists the activities performed across different projects in the organization. It also provides an option to export the logs in CSV and JSON formats, as shown in Figure 2-8.

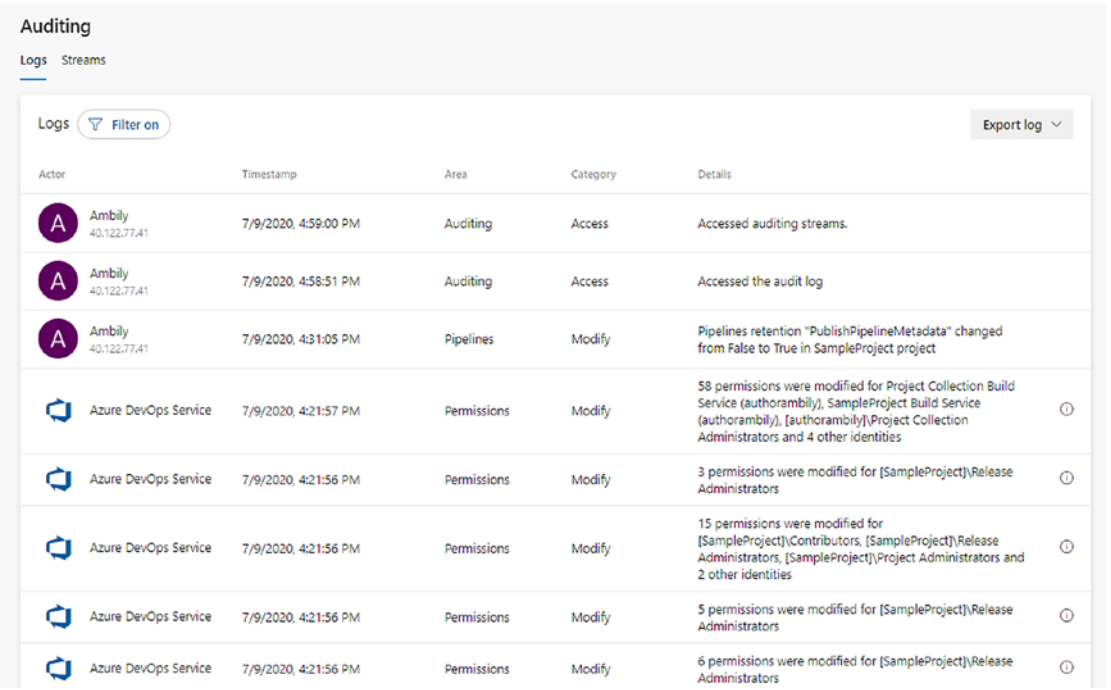


Figure 2-8. *Auditing section*

Moreover, the Streams tab allows you to configure new log streams from Azure Event Grid, Splunk, and Azure Monitor Logs. Mainly, streams are used to analyze the production or staging environment logs to identify any potential threats.

General ➤ Global Notifications

The Notifications section lists the default notifications, which can be disabled or enabled for all projects in this organization, as shown in Figure 2-9.

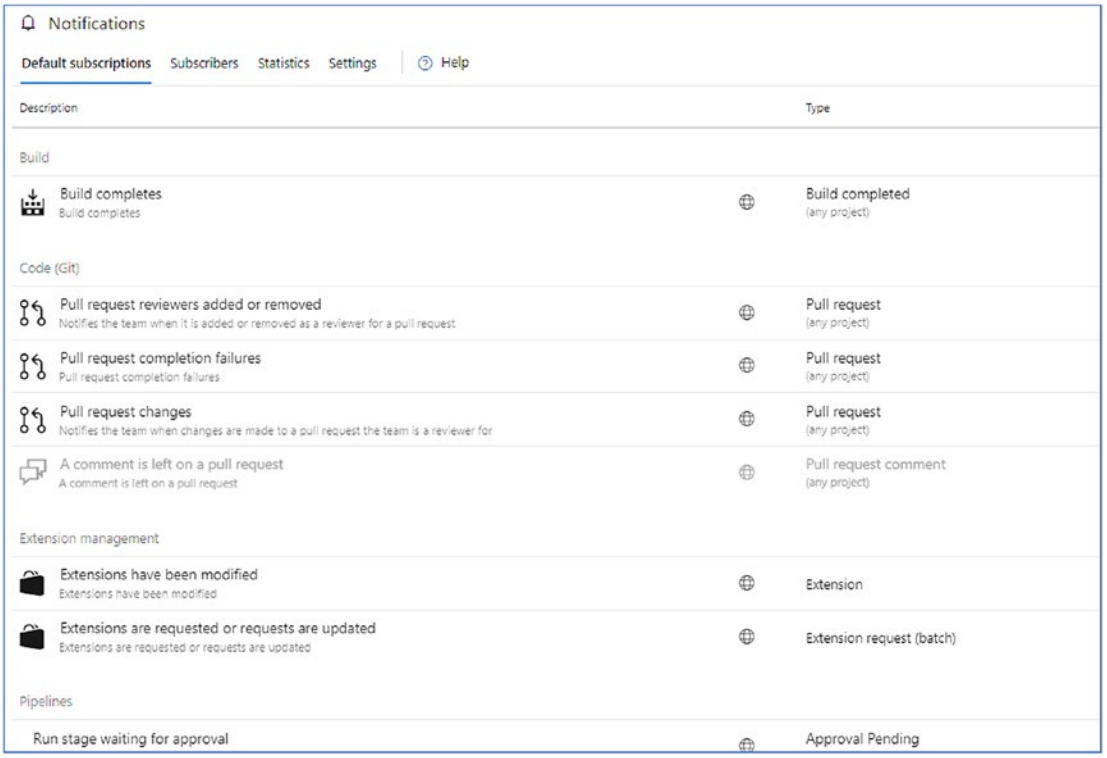


Figure 2-9. *Default notifications*

The Subscribers tab shows the notifications based on the users, in other words, what notifications are set for a selected user. The Statistics tab shows the active subscribers and top event initiators and deals with the delivery configuration, as shown in Figure 2-10.

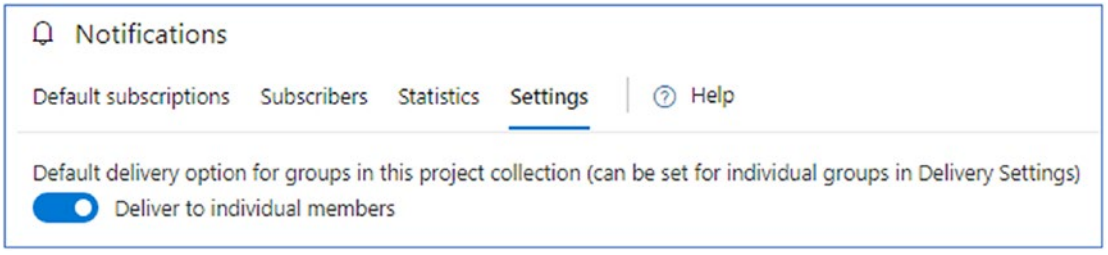


Figure 2-10. *Notifications: Settings*

General ➤ Usage

The Usage section shows the dynamic view of the usage of the system in terms of Team Services Through Units (TSTUs).

There are many filters and column selection options available to filter the data in lower levels. Moreover, this report provides an option to save the content in CSV format.

General ➤ Extensions

The Extensions section provides an option to add customized services from the community or other companies. The Extensions section provides an interface to customize Azure DevOps based on your requirements. Also, this option helps you explore the useful community-built services integration. Azure DevOps extensions are available in the Visual Studio Marketplace: https://marketplace.visualstudio.com/azuredevops?utm_source=vstsproduct&utm_medium=L1BrowseMarketplace&targetId=bc12d2e1-b56e-461e-a3eb-1852f2030883.

For example, the SonarQube extension (https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarqube) available in the Marketplace supports the build tasks to integrate the SonarQube code analysis as part of the build steps. We will explore different extensions in later chapters.

The Security option available in this section is important. The organization owner can provide permission for another member to install an extension using this feature. Otherwise, only the organization owner is allowed to install an external extension.

General ➤ Azure Active Directory

Azure AD integration helps map the organization members to the enterprise Azure AD identity. This helps track the activities of internal users.

Security ➤ Policies

Organization-level policies are configured in the Policies section, as shown in Figure 2-11. Project-level policies are explained under the "Project Settings ➤ Repo policies" section.

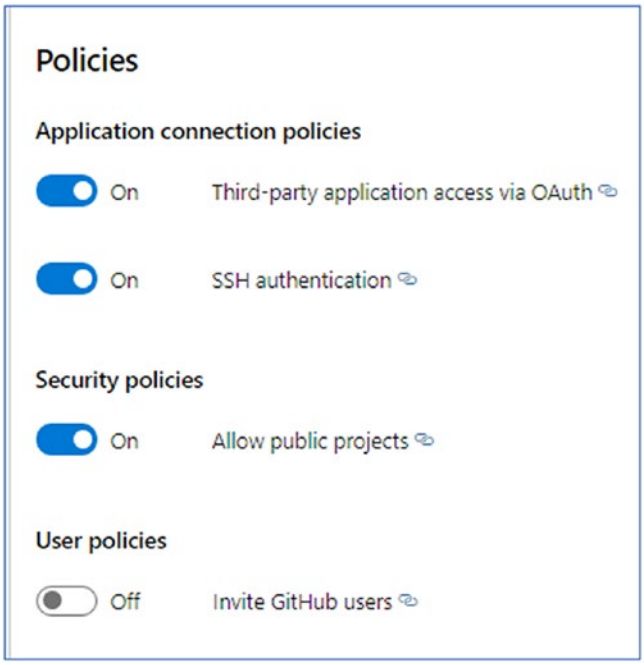


Figure 2-11. *Organization: Policies*

Security ➤ Permissions

The Permissions section has settings related to organization-level permissions for each group and user. From here, new groups can be configured with specific permissions as well.

You can learn more about permissions and their values in the "Project Settings ➤ General ➤ Permissions" section.

Boards ➤ Process

By default, Azure DevOps supports four process types to execute a project.

- *Basic*: This is suitable for those who are getting started with project management in Azure DevOps.
- *Agile*: This is the template defined for agile teams.
- *Scrum*: This is for scrum teams.
- *CMMI*: This is a standard template used for projects with lots of formalities and auditable requirements.

Understanding the process template is important to defining proper project management. Select a process template to explore this further.

Basic Process Template

The Basic process template provides the basic structure required to map the requirements to the system, as shown in Figure 2-12. Epics capture the business and functional requirements at a high level. Issues track the defects associated with the deliverables, and tasks define the implementation tasks related to the project. The Test Case, Test Plan, and Test Suites types provide facilities to manage the test artifacts properly in the system.

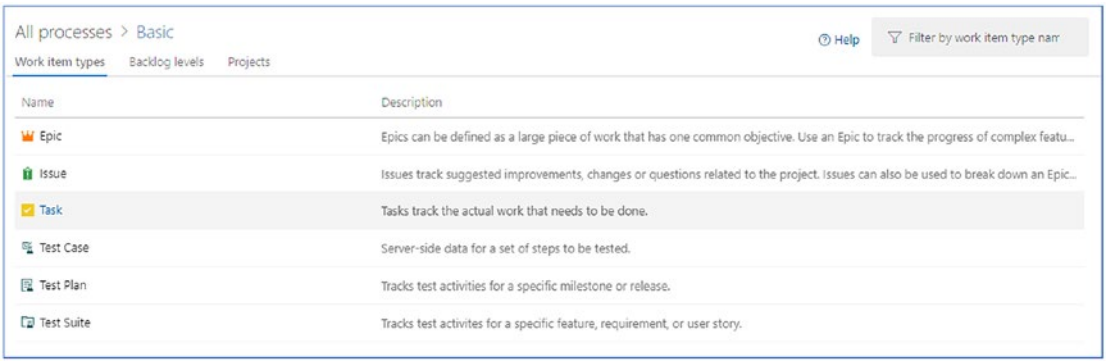


Figure 2-12. *Process template: Basic*

Agile Process Template

In the Agile process template, epics handle the high-level business requirements, whereas a feature defines the functional-level features of the application or product or software. User stories elaborate on the user activities and the expected behavior of the application. Bugs track the defects, whereas issues are challenges in executing the project. Tasks are the implementation tasks associated with a user story. In this template also, the items called Test Plan, Test Suite, and Test Case deal with test management, as shown in Figure 2-13.

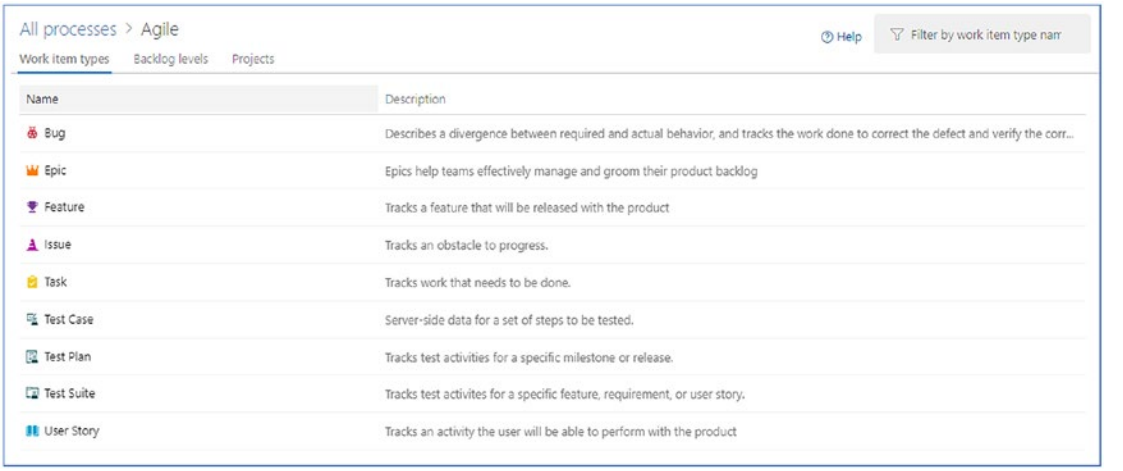


Figure 2-13. *Process template: Agile*

Scrum Process Template

In the Scrum process template, epics handle the high-level business requirements, whereas a feature defines the functional-level features of the application or software. Product Backlog Item elaborates on the user activities and the expected behavior of the application. Tasks define the implementation work. Bugs track the defects, whereas impediments are challenges in the execution of the project. In this template also, Test Plan, Test Suite, and Test Case all deal with test management, as shown in Figure 2-14.

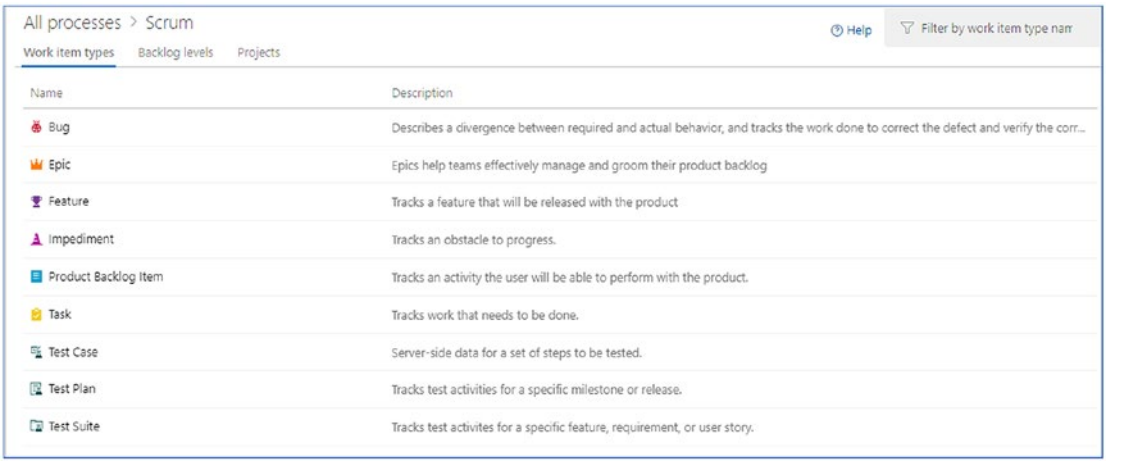


Figure 2-14. *Process template: Scrum*

CMMI Process Template

In the CMMI process template, epics handle the high-level business requirements, whereas a feature defines the functional-level features of the application or software. Bugs track the defects, whereas issues are challenges in the execution of the project. Change requests are used to track changes in the life cycle. Requirements track the requirements life cycle. Reviews track the review process and results. Risks track the risk mitigation. Tasks define the implementation work. In this template also, Test Plan, Test Suite, and Test Case deals with test management. Figure 2-15 shows the work items associated with the CMMI template.

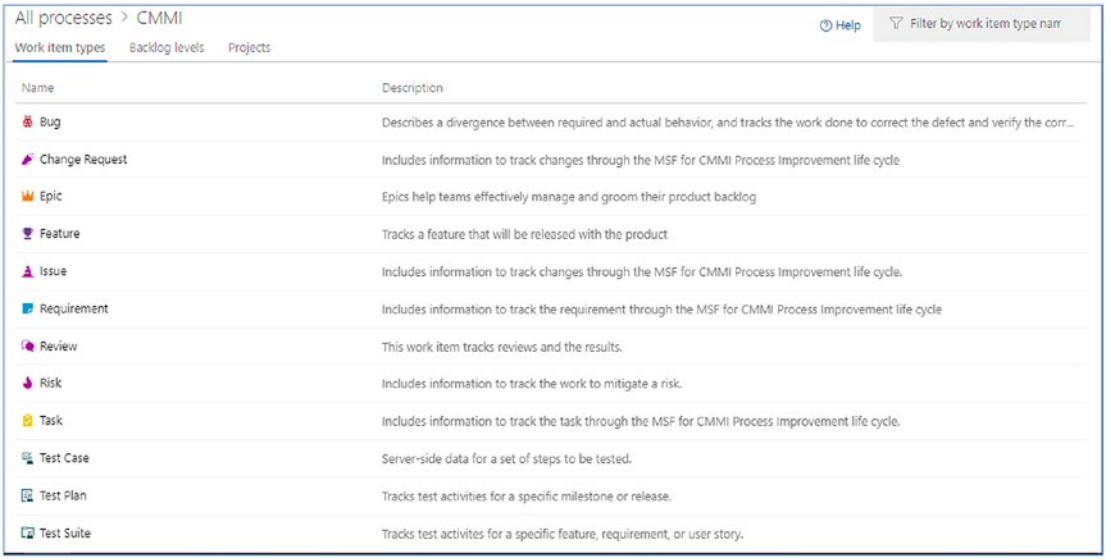


Figure 2-15. *Process template: CMMI*

Customization of the process templates will be covered in the "Process Template Customization" section of this chapter.

Pipelines ➤ Agent Pools

Pipeline jobs are executed by agents. Agent pools are collections of agents grouped together based on the location or features. By default, Microsoft provides a hosted agent pool to run the jobs. The following are the different agent configurations:

- *Microsoft-hosted agents*: Agents are hosted by Microsoft and available based on your subscription mode.
- *Self-hosted agents*: Users can host or configure agents on their own systems or in cloud subscriptions.
- *Azure Virtual Machine Scale Set (VMSS) agent*: These are self-hosted agents, autoscaled to support the demand.

Each agent pool shows the jobs running on the pool, the agents configured against the pool, the owner, and the permissions. We will look at agent pools in Chapter 6.

Pipelines ➤ Settings

The Settings section provides control over configuring global decisions, as shown in Figure 2-16. For example, if the enterprise policy defines the usage of Marketplace tasks are not secure, disable it on the organization level.

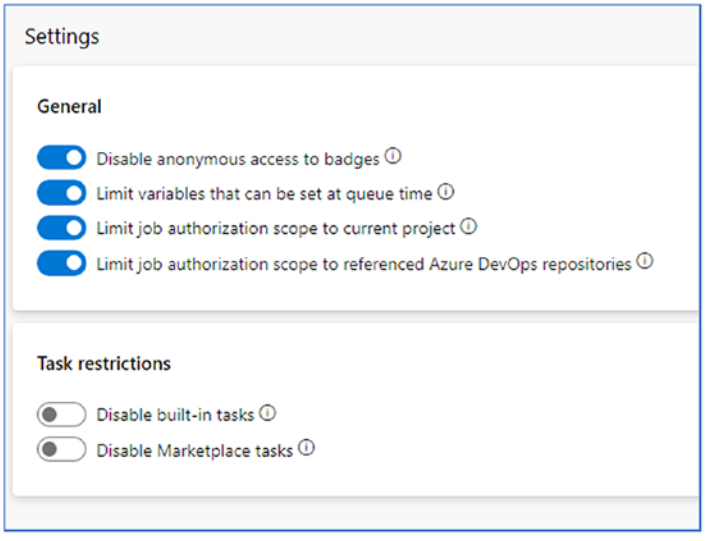


Figure 2-16. *Organization Settings ➤ Pipeline ➤ Settings*

Pipelines ➤ Deployment Pools

Deployment pools define the set of target systems that will be used for deployment by a few or all of the projects under this organization. Defining a new deployment pool requires you to configure the target systems using PowerShell commands, which will be explained in detail as part of Chapter 6.

Pipelines ➤ Parallel Jobs

Parallel job execution depends on the agent pool configuration and subscription tier. For a free trial account, the Microsoft hosted agent pool provides ten parallel jobs, which means the build can be configured to execute multiple tasks in parallel. You'll learn more about the job execution in Chapter 6.

Pipelines ➤ OAuth Configurations

The OAuth Configurations settings allow you to define connections with enterprise subscriptions such as GitHub, GitHub Enterprise, and Bitbucket over OAuth 2.0.

Artifacts ➤ Storage

Artifacts are the executable resulting from a build or the package used for deployment. The "Artifact storage" setting shows the storage used by different artifacts such as the pipeline artifacts, symbols used for debugging, NuGet package feeds, and so on.

This data helps the organization administrators analyze the storage usage and control the costs associated with the organization by configuring the proper retention policies.

Projects

Projects are used to group the activities related to the development of a solution or product. One project consists of the teams working on the project, different policies followed, sprint cadence followed, release pipelines, and daily activities. If the organization exists for the business group, new business project initiation starts by creating a new team project in Azure DevOps. The setup of the team project in Azure DevOPs determines how the project will execute.

Creating a Project

Under Create Organization, we have created a project with basic details such as the project name and the type of the project, whether public or private. In this section, click the "New project" option available on the organization's home page to explore more options corresponding to creating a project. Start with configuring a new project to understand the different elements of a project.

Click "New project," as shown in Figure 2-17, to start creating a new project.

Figure 2-17. *Starting a new project*

Provide a project name, description of the project, whether it is a public project or private project, version control, and work item processing in the "Create new project" window, as shown in Figure 2-18.

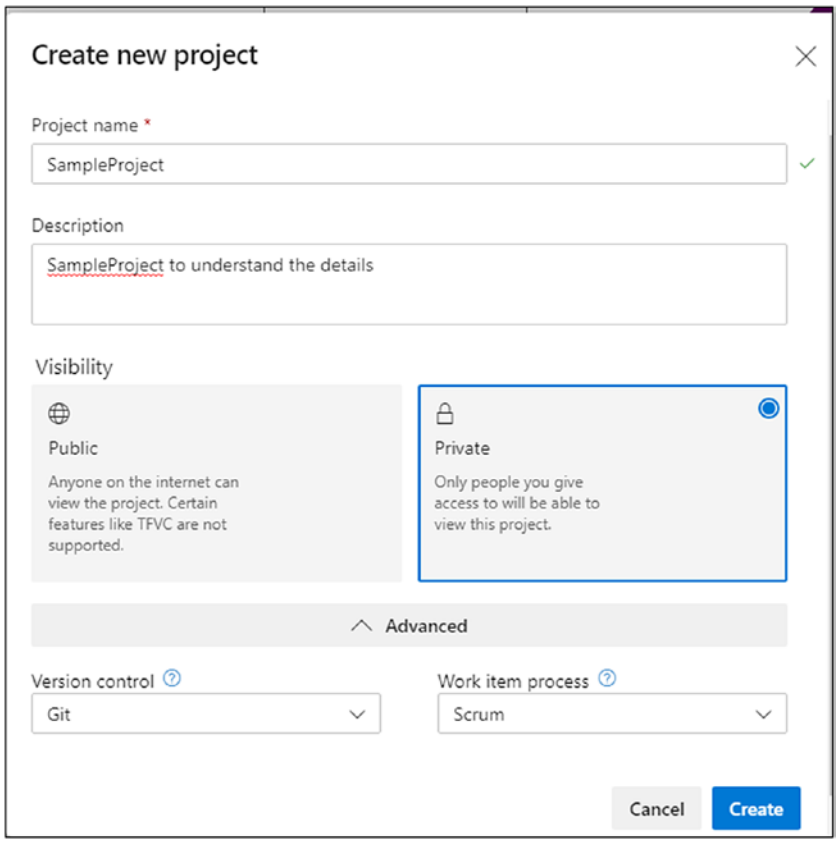


Figure 2-18. *New project settings*

There are two kinds of version control supported by Azure DevOps.

- *Git*: This is a distributed version control system, where the user branches the code locally. Users can create different branches locally and commit the changes in a logical completion of task or work. All the changes can be managed and versioned locally until merged back into the centralized repository.
- *Team Foundation Version Control*: In this kind of version control, versioning is done only in a centralized server. Users should merge the changes to the centralized repository often to avoid losing any work. Versioning is not available locally.

The "Work item process" setting defines the execution approach followed by the project. Azure DevOps supports different execution approaches and also allows the user to onboard custom process templates to align with an enterprise-defined procedure. The built-in process templates are as follows:

- *Agile*: Agile is one of the most popular execution approaches, where the requirements are realized in defined iterations. This iterative approach allows the team to deliver value to end customers in chunks, which enable the team to sensitize the real user expectations and feedback in an early stage.
- *Basic*: The Basic template provides a simple template for first-time users of Azure DevOps.
- *CMMI*: This approach was followed earlier in the waterflow model of execution, where the activities are executed in a sequential manner. All the design and development complete as one entire work item to deliver the end product all at once to the customer. This approach takes more time to get to market and gets real feedback from customers. But this approach is suitable for specific products where the partial release doesn't really deliver any kind of value to the end customer.
- *Scrum*: This approach follows agile principles where features are released before the complete project. In scrum, teams execute the activities in sprints and deliver different product backlogs in sprints. Most of the sprint deliverables don't release to the end customer; instead, scrum team define a separate release cadence to handle the end-user releases.

Select an appropriate version control system and process template to proceed with the project creation.

Project Settings

Before getting into the details of the project components, explore the project settings, especially the ones configured by the project administrator. Project settings provide a centralized place to handle all the project-related configurations.

General ➤ Overview

The Overview section allows you to reconfigure the project name, description, process template, and visibility. Along with these basic details, it also allows you to add a new project administrator and delete the project.

The "Azure DevOps services" section under Overview allows you to control the usage of the project. By default, the project will be configured to handle the following:

- *Boards*: Agile planning and execution
- *Repos*: Code versioning and management
- *Pipelines*: Build and release pipelines
- *Test Plans*: Manual test plans
- *Artifacts*: Management of packages ready for delivery

Based on the project, disable the components that will not be used. For example, some enterprise projects with high security concerns manage the code in local Git repositories instead of Azure DevOps and use the Azure DevOps boards for agile execution and Azure DevOps pipelines for build and release. In the case of a containerized application deployment, the built-in Artifacts component is not used; instead, an external container registry like Azure Container Registry or Docker Image Registry will be used. In this case, project administrator can disable the Artifacts option.

General ➤ Teams

By default, the system provisions a team with the name of the project to manage access to the project, as shown in Figure 2-19. New teams can be provisioned to control the access on the team level, such as the test team having access to the test plans and the stakeholders requiring only read-only access to certain artifacts.

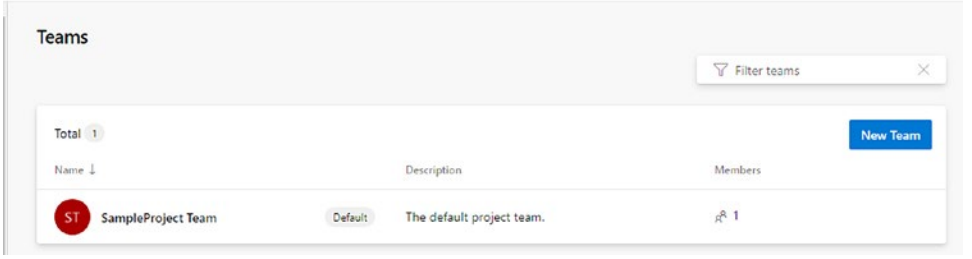


Figure 2-19. *Teams*

Select an existing team to add new members to the team, as shown in Figure 2-20. Team details such as name, avatar, and description can be modified by navigating to the Settings tab. Also, you can remove any team that is not the default team of the project using the Settings tab.

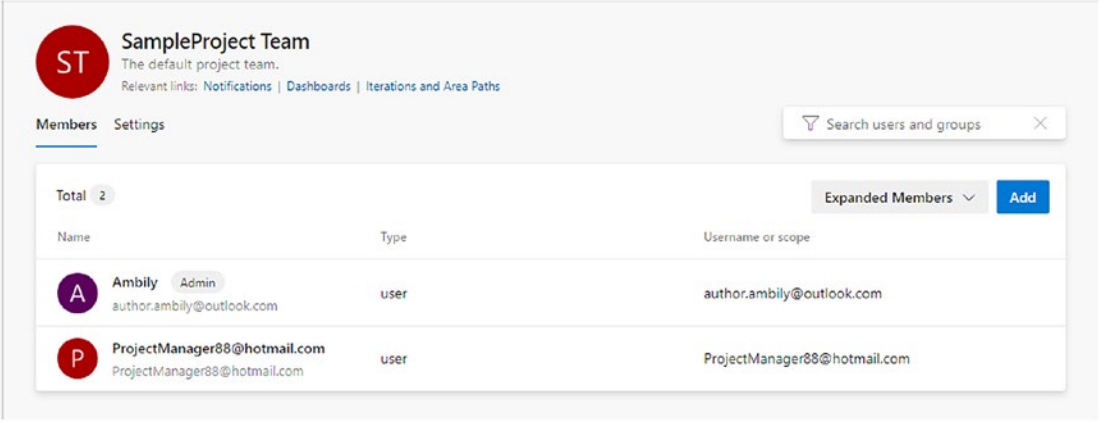


Figure 2-20. *Adding team members*

Creating a New Team

Like a project that has an avatar indicating the project logo or the real spirit of the project, we can have an avatar for each team. Specify the name of the team, a brief description, and add few a members as part of this team, as given in Figure 2-21.

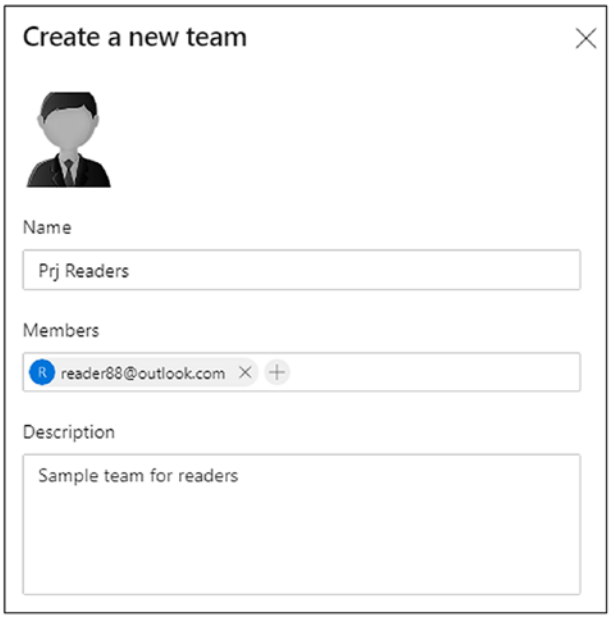


Figure 2-21. *Creating a new team*

This section will allow you to add the administrator of the project as a member of the team and provide permissions based on the built-in roles. You'll learn more about the built-in roles in later chapters. The area path is another aspect, as shown in Figure 2-22, that will be covered in subsequent sections.

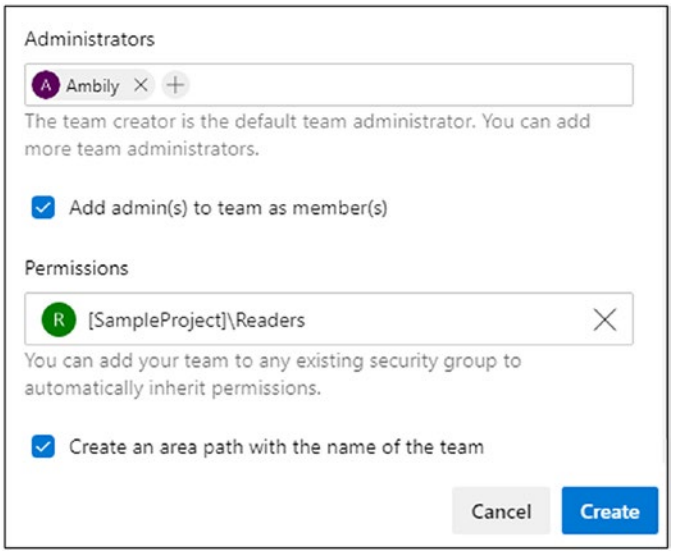


Figure 2-22. *Creating another new team*

General ➤ Permissions

End-to-end project execution involves many people with different roles and requirements. A project manager or a business stakeholder will be interested in task completion and how many pending activities there are. At the same time, a developer would concentrate on the source code and associated artifacts, a tester would focus on the test plan and overall functionality, and so on. Group the different team members under different groups to define the required permissions.

As observed in Figure 2-23, the teams are also added as groups. A new group can be provisioned from this section to assign permissions. Select a group to understand the permissions assigned to it.

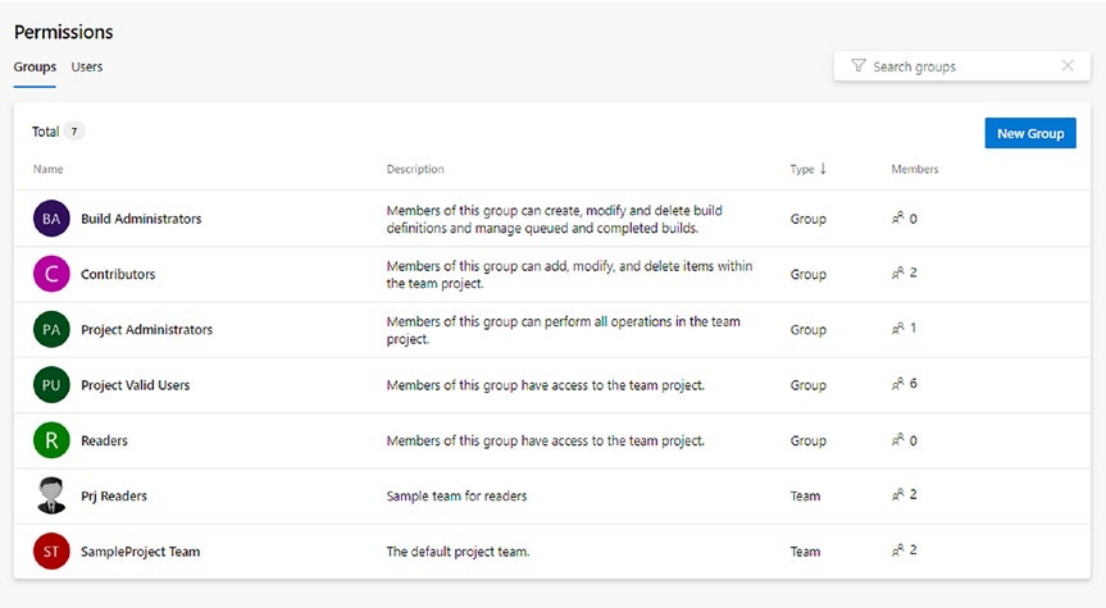


Figure 2-23. *Project permissions*

Permissions are grouped under General, Boards, Analytics, and Test Plan, as shown in Figure 2-24. Based on the need, change the permission value to Allow, Deny, or "Not set." Changes will be saved immediately and go into effect for the corresponding team members.

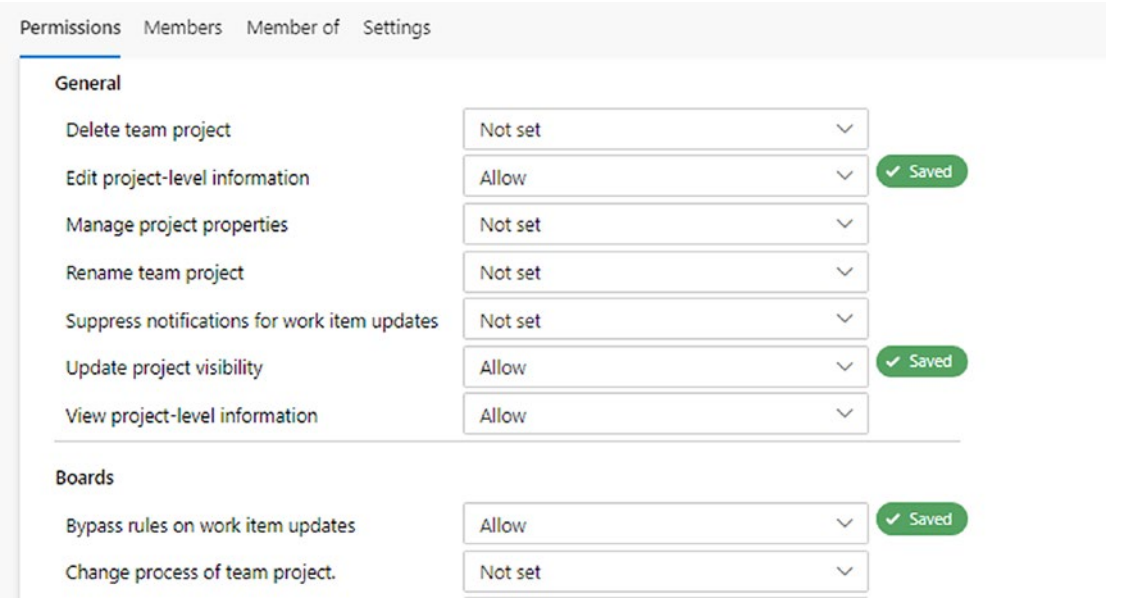


Figure 2-24. *Editing permissions*

The Members tab displays the members belonging to this group. Similarly, the "Member of" tab displays the groups that this group belongs to, as shown in Figure 2-25. By default, every group getting added to the project will be part of the Project Valid Users group. If required, add new groups.

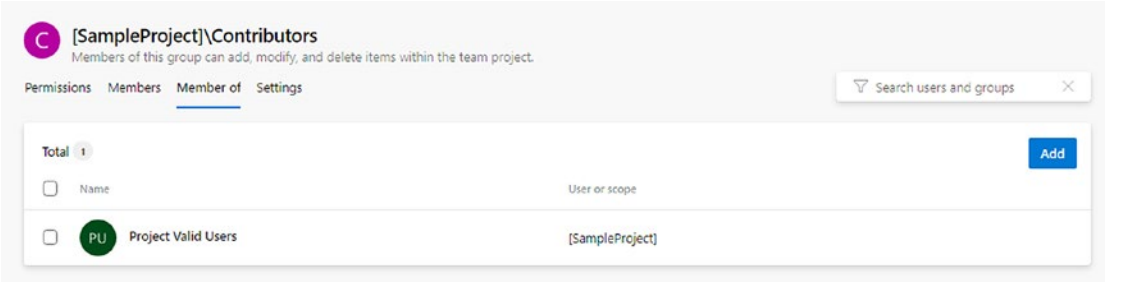


Figure 2-25. *Permissions: Member of tab*

The Settings tab allows you to modify the group image or avatar, name, and description. It also allows you to delete the group.

General ➤ Notifications

The Notifications section provides different built-in notification capabilities. Some notifications are configured by default. For example, when a build completes, members get a notification to their configured emails.

To understand the conditions associated with a notification, select the View option available next to the globe icon, as shown in Figure 2-26.

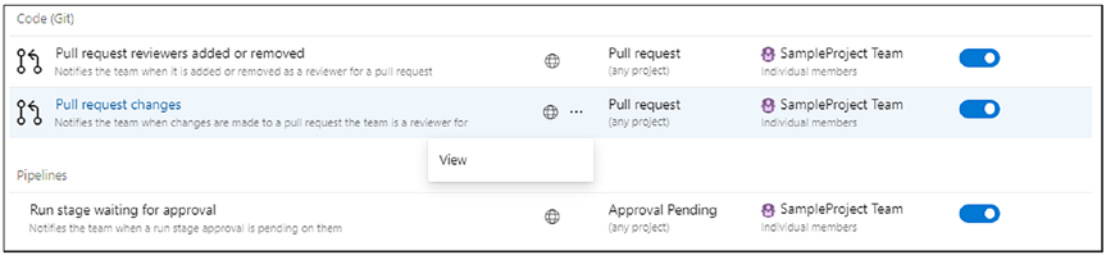


Figure 2-26. *Viewing the notification details*

Selecting the View option opens the configuration of the selected notification; you can define the description of the notification, who will receive the notification, and the filter criteria.

At any time, users can opt out of a notification using the switch key at the end of each notification, as shown in Figure 2-27.

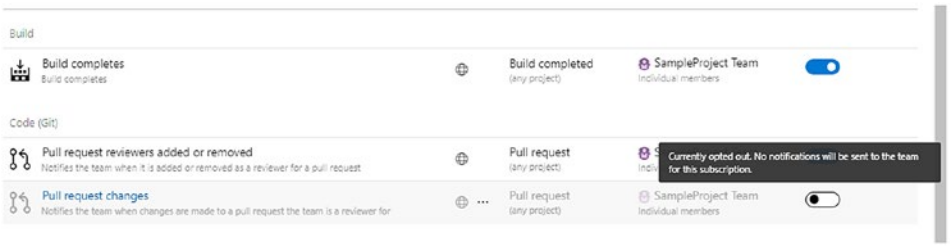


Figure 2-27. *Opting out of a notification*

Click “New subscription” to add a new notification to the list. For example, add a new notification for the build failed criteria by selecting Build for Category and “A Build fails” for Template, as shown in Figure 2-28.

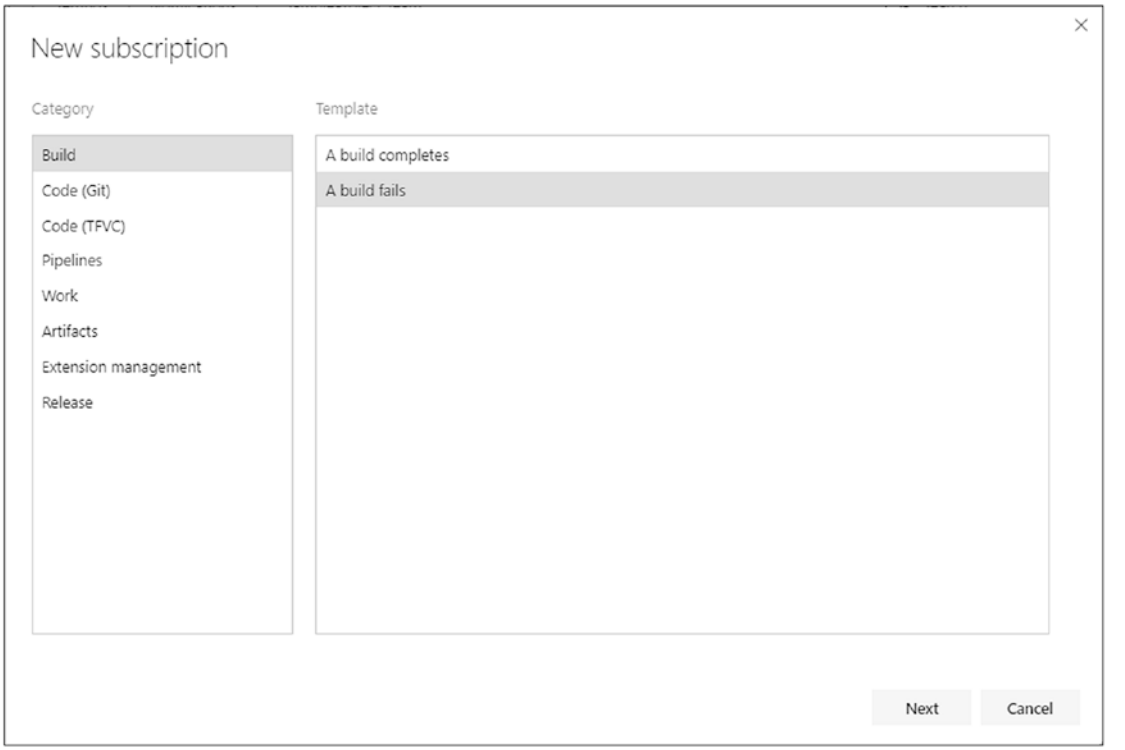


Figure 2-28. *New notification*

Click Next to define the notification criteria. For build fails, we can get the notification to a custom email address like the service group's ID, as shown in Figure 2-29. Using the Filter option, you can apply the notification configuration to all projects under this organization or to a specific project.

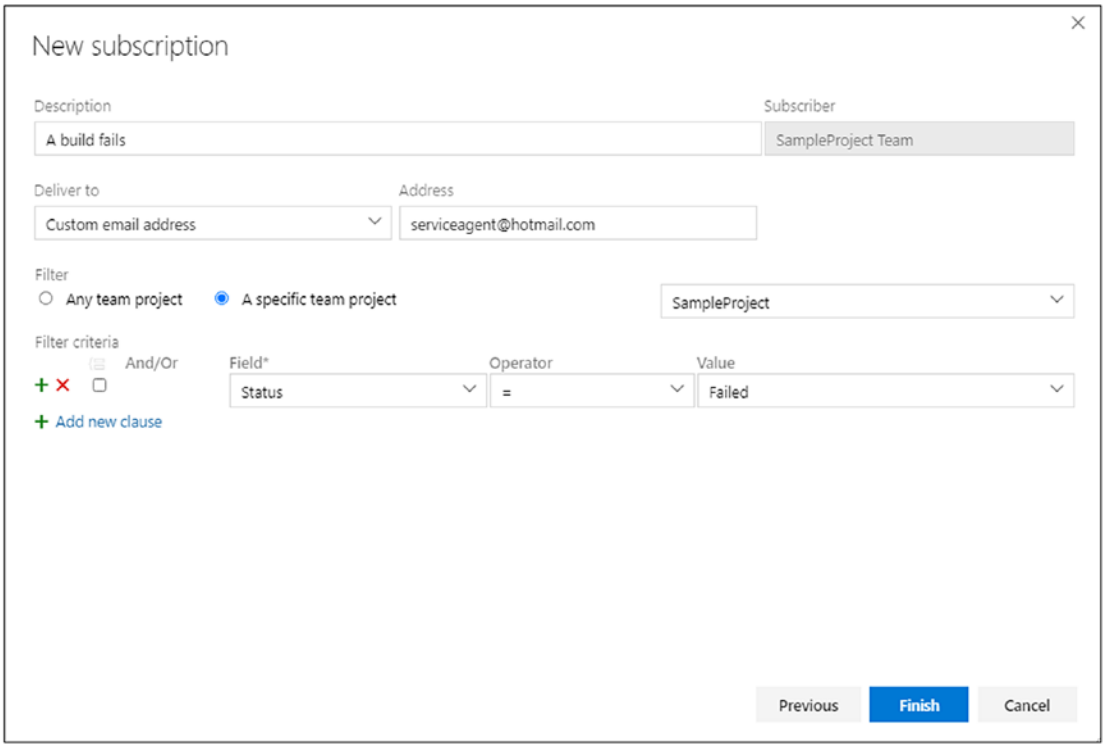


Figure 2-29. *New notification criteria*

Click Finish to complete the configuration of a new notification, which gets triggered on build fails.

The delivery of each notification can be controlled using the delivery settings, as shown in Figure 2-30. Select one of the notifications and select the Delivery Settings option available on top to reconfigure the delivery options.

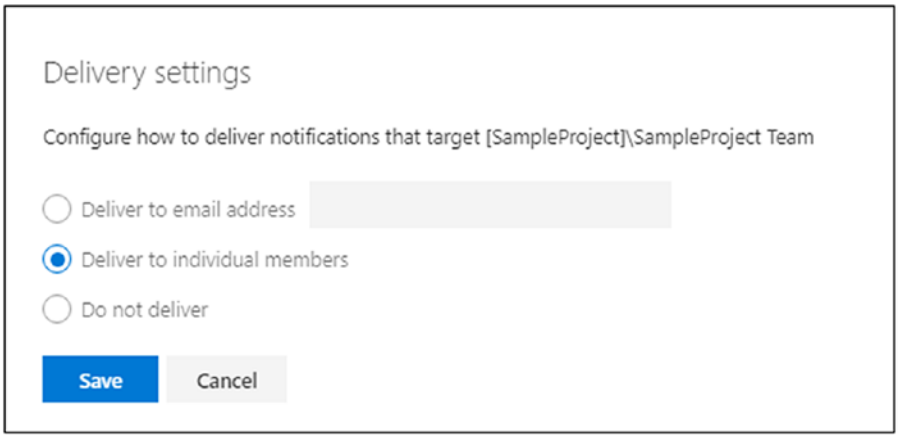


Figure 2-30. *Delivery settings*

Note Click [⤢] to open the section in full-screen mode.

General ➤ Service Hooks

Service hooks help to trigger or notify an external system about an internal event. For example, if you configured your build in an external Jenkins system and want to notify the code commits to Jenkins, utilize the service hooks with code push event defined in Azure DevOps.

Clicking "Create subscription" opens the New Service Hooks Subscription dialog with the available integration options, as shown in Figure 2-31. We will look into these integrations later in this book.

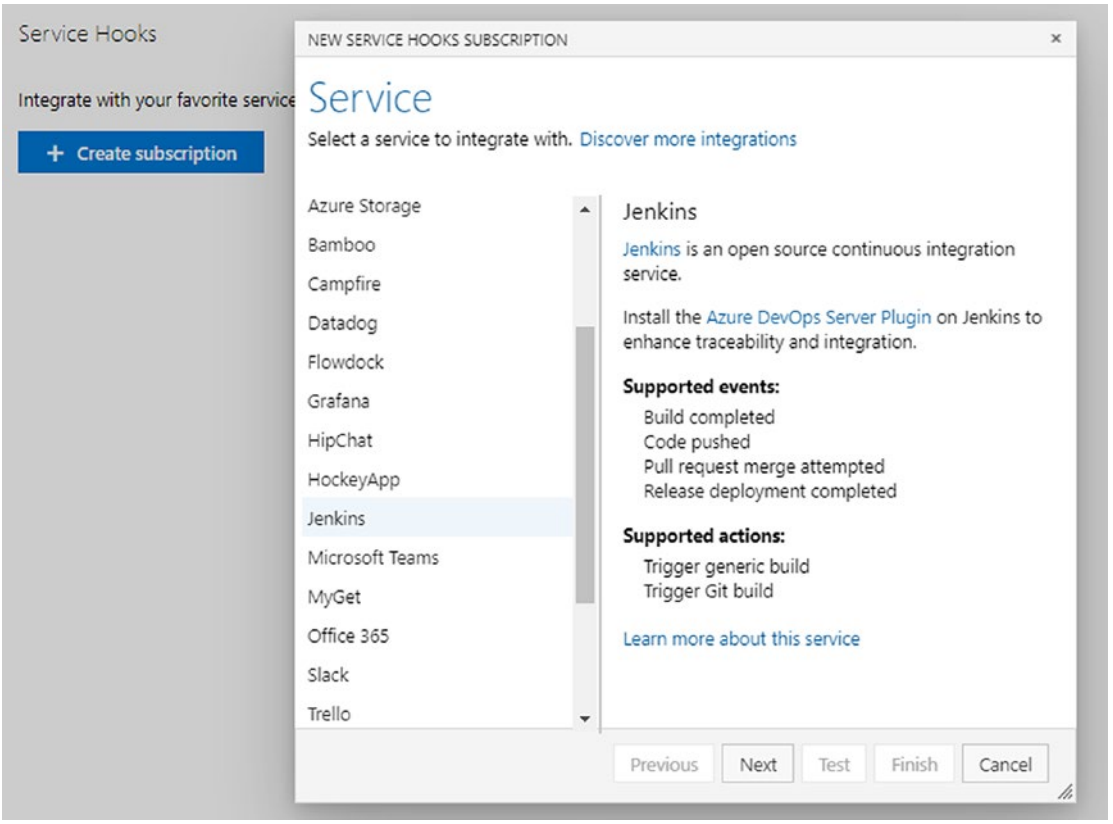


Figure 2-31. *Creating a subscription*

General ➤ Dashboards

The Dashboard section allows the project administrators to provide permission to team members to configure dashboards. By default, team members will not have permission to configure a new dashboard or edit or delete a dashboard, as shown in Figure 2-32.

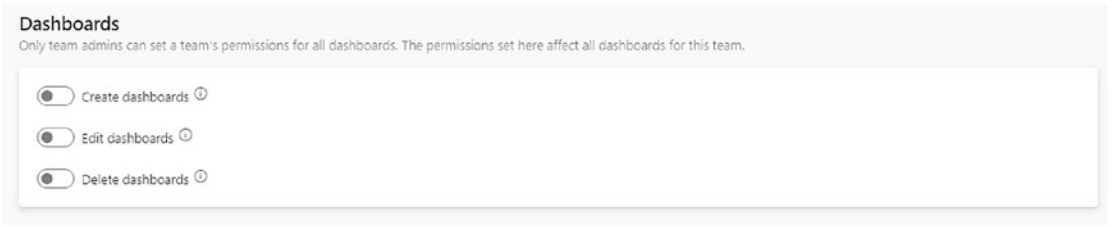


Figure 2-32. *Dashboard permissions*

Boards ➤ Project Configuration

In the Project Configuration section, configure the iterations or sprints and areas. Sprints define the duration of each iteration, which can range from one week to four weeks. Based on the agile planning and complexity of the project, a team decides on the sprint duration, called *sprint cadence*. In scrum, there will be a different release cycle, which may cover one or more sprints. Release cycles are called *release cadences*. By default, sprints are labeled Sprint 1, Sprint 2, and so on, and this can be changed based on the project requirements, as shown in Figure 2-33.

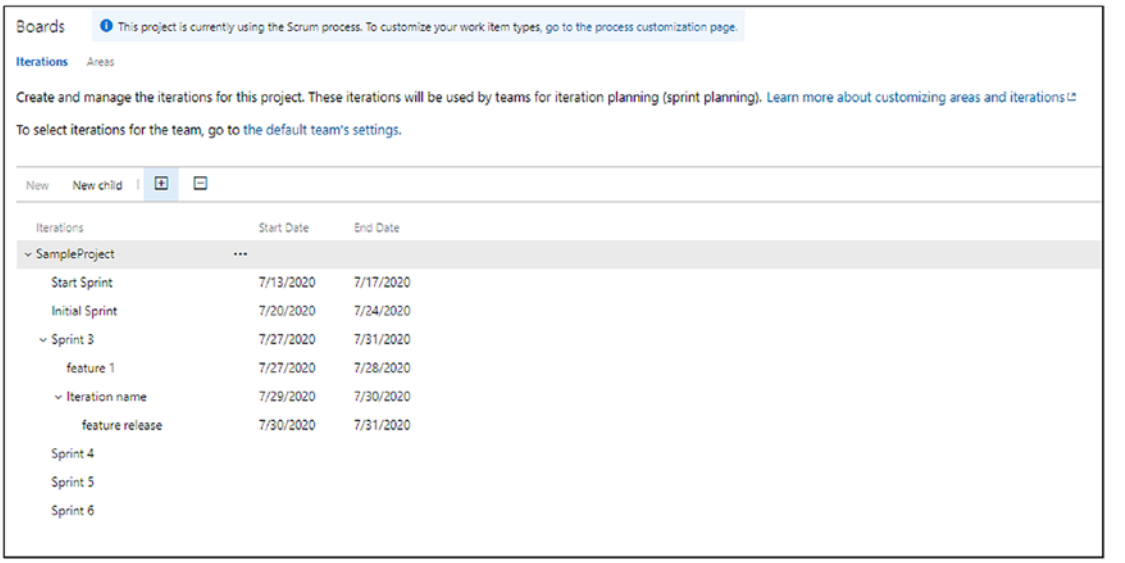


Figure 2-33. *Sprint iterations: multilevel*

Iterations can be defined in different levels as well. For example, a sprint with a cadence of four weeks can be further divided into one-week iterations to focus on specific tasks. Also, this can be configured without multilevel iterations, as shown in Figure 2-34.

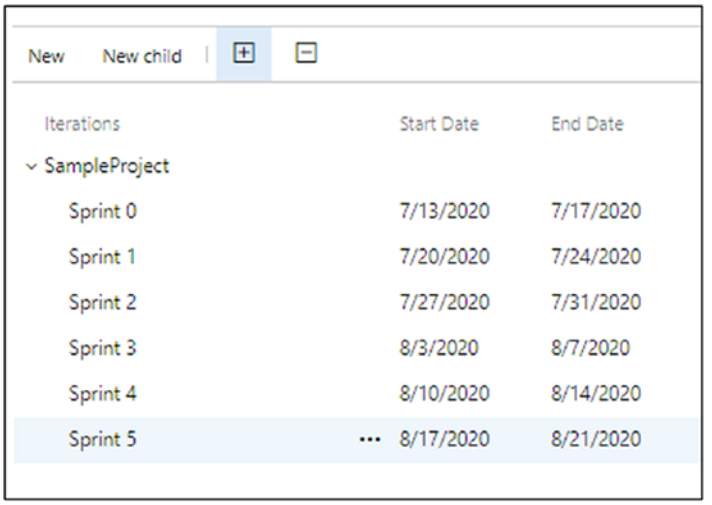


Figure 2-34. *Sprint iterations*

Areas can be defined in many ways, as shown in Figure 2-35. This can be used for separating the work items related to different teams; it can denote different features, can be different components, can be different layers, and so on.

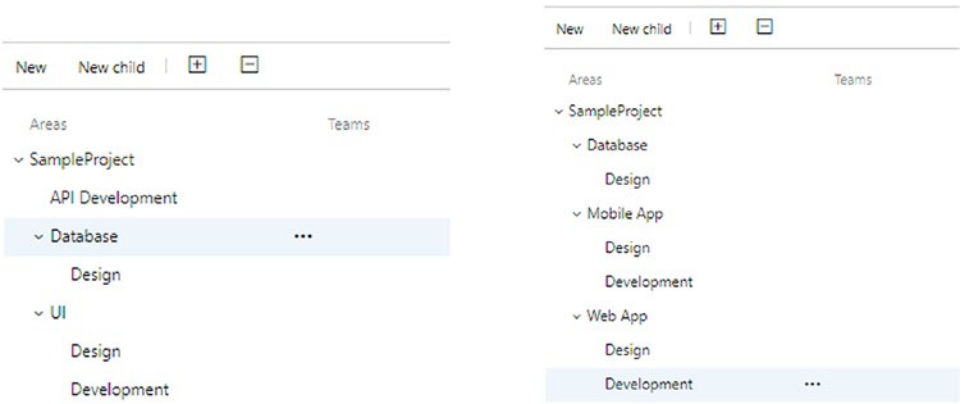


Figure 2-35. *Areas: based on different modules and components*

Areas are used to separate the work, especially the work items in boards and as part of queries. This will be discussed as part of Chapter 3.

Boards ➤ Team Configuration

The team configuration settings related to boards, iterations, areas, and templates for teams. Under the General settings, configure the navigation levels and working days. Also, this allows you to configure how the bugs or defects are treated as part of sprint boards.

The Iterations tab allows you to decide on what iterations this team is going to contribute. Teams such as the design team may be onboarded as part of the initial iterations. Sometimes, performance testers will be onboarded after a few iterations only.

Similarly, the Areas tab supports the team-level configuration of areas, as shown in Figure 2-36. For example, the mobile app development team may require access to the work items related to mobile app area, whereas a UI developer requires access to tasks related to the UI area only.

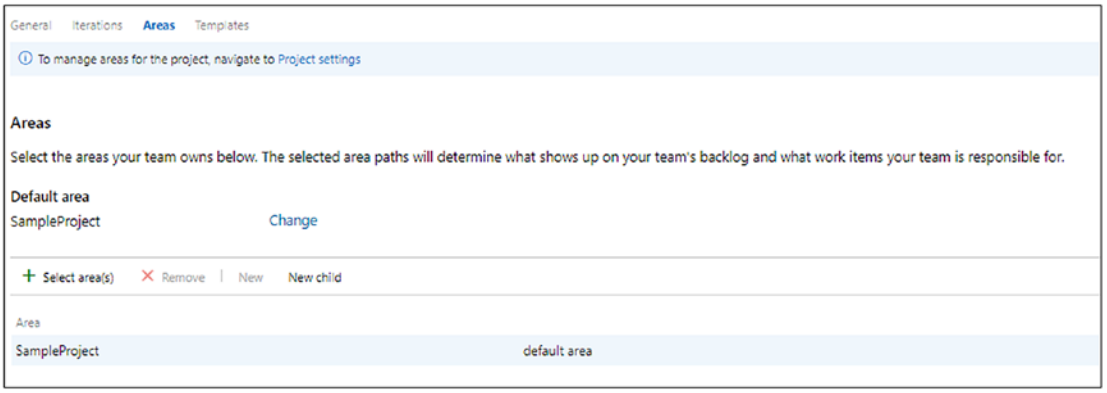


Figure 2-36. *Team configuration: Areas*

Boards ➤ GitHub Connections

Even though Azure DevOps provides a full-fledged end-to-end solution for DevOps implementation, enterprises may look to manage the overall DevOps process using multiple tools such as GitHub, GitLab, Jenkins, and so on. GitHub is a popular code repository, which can be integrated with Azure Boards to manage the end-to-end activities.

Connect GitHub with Azure DevOps using one of these three methods:

- *Using a GitHub account*: This option allows the user to configure the connection by providing GitHub credentials.
- *Personal access token*: Pass the personal access token (PAT) created in GitHub to Azure DevOps to configure the connection. Refer to https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token to understand how to create a PAT.
- *GitHub Enterprise Server*: If an enterprise has GitHub Enterprise set up, use this option and use the client ID and client secret from the GitHub Enterprise configuration to connect. Refer to the Microsoft documentation to understand more about the GitHub Enterprise Server configuration: https://docs.microsoft.com/en-us/azure/devops/boards/github/connect-to-github?view=azure-devops#connect-azure-devops-services-to-github-enterprise-server.

Once the GitHub credentials or PAT or enterprise configuration is passed, the system will ask to authorize a few permissions to share data between the GitHub account and Azure DevOps.

The GitHub integration now completes and is added to the Azure DevOps settings, as shown in Figure 2-37.

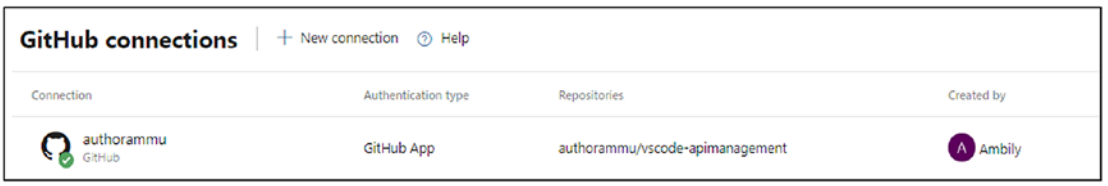


Figure 2-37. *GitHub integration, completed*

Repos ➤ Repositories

The Repositories section presents the single code project that resides in your Azure DevOps project. By default, one repository is configured at the time of project creation, as shown in Figure 2-38. If you are working with a project consisting of multiple

microservices, each service can be configured with a separate repo to handle the codebase separately. Also, granular repos help in defining different build and release pipelines for each of the repos.

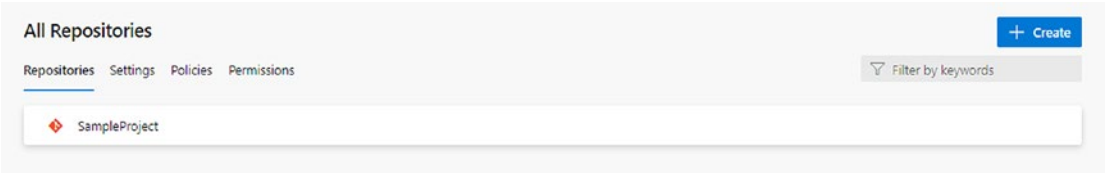


Figure 2-38. *Repos default*

Add a new repo to the existing project using the Create button. Repos can use different repository types such as Git or Team Foundation Version Control (TFVC), as shown in Figure 2-39. Git allows you to add a ReadMe file to the blank repo, which explains the usage of the project. This file captures the description and usage of the project.

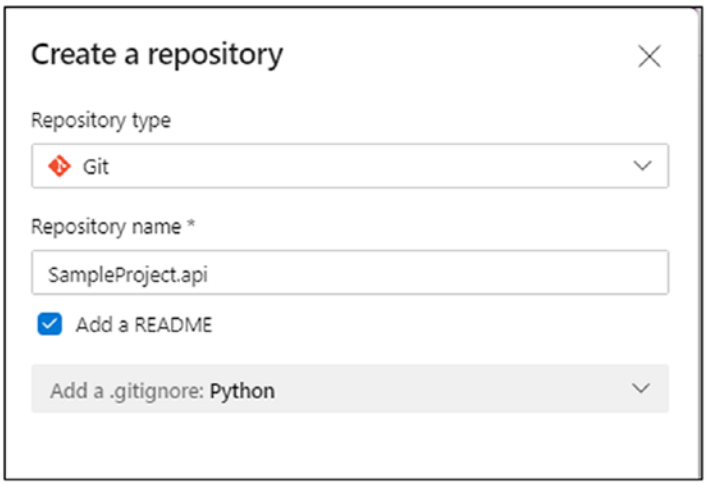


Figure 2-39. *Creating a new repo with Git*

Also, this dialog allows you to set up the `.gitignore` file, which explains which file types should be ignored as part of version control. For example, in a C# project, we may not do the version control of the DLLs generated as part of the build.

When it comes to a TFVC-based repo, there's no need to configure the ReadMe and `.gitignore` file, as shown in Figure 2-40.

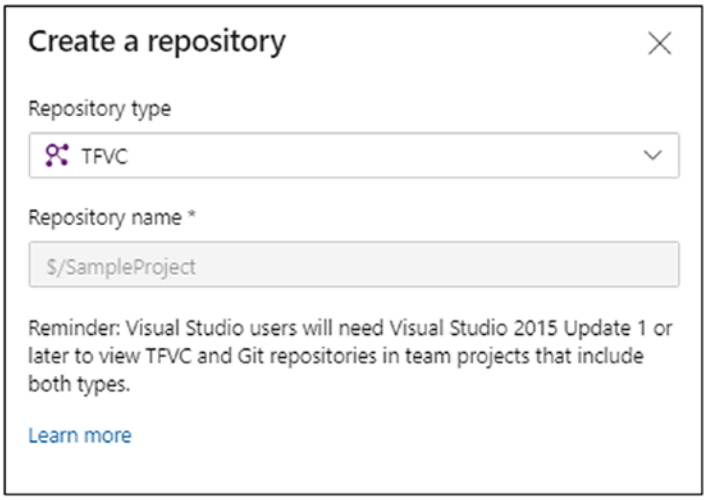


Figure 2-40. *Creating a new repo with TFVC*

Repo Settings

Select the newly created repo to configure the settings. The Settings tab provides options to disable forking and automatic link creation for commits and related work items, as shown in Figure 2-41.

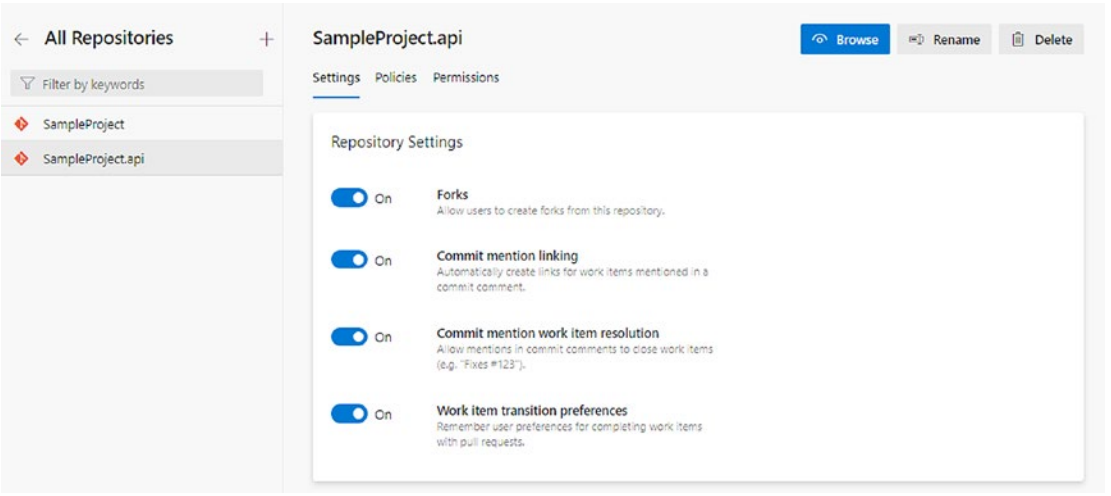


Figure 2-41. *Repo settings*

Repo Policies

Select the Policies tab to set different policies for the repo and branches. Turn on the required policies for the repo to protect the code quality.

The policy in Figure 2-42 restricts commits to team members with emails ending in `@org.com` and prevents commits from `service@org.com`.

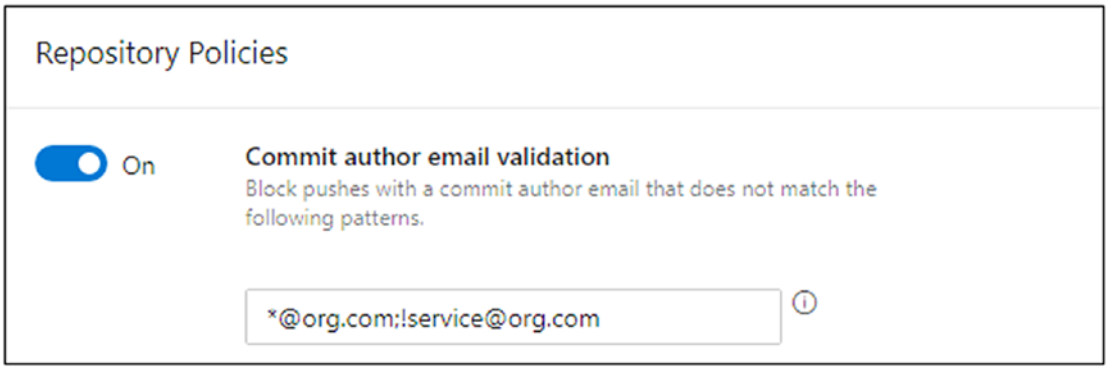


Figure 2-42. *Repo policies*

In configuration in Figure 2-43, file size is restricted to 100MB. Files larger than 100MB will be rejected from the commit.

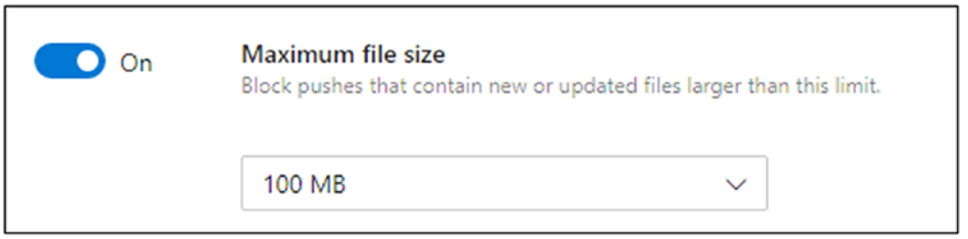


Figure 2-43. *Repo policy file size*

Branch Policies

Branch policies control the policies related to a branch. Every repo is configured with a single branch called *master*, as shown in Figure 2-44. Based on branching and merging decisions, the project administrator can configure different branches.

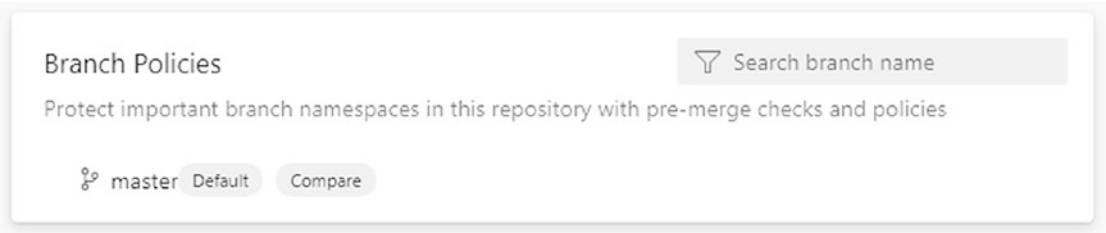


Figure 2-44. *Branch policy settings*

Select the branch for configuring the settings, as shown in Figure 2-45. There are four policy setups available for each branch: Branch Policies, Build Validation policies, Status Checks, and Automatically included reviewers.

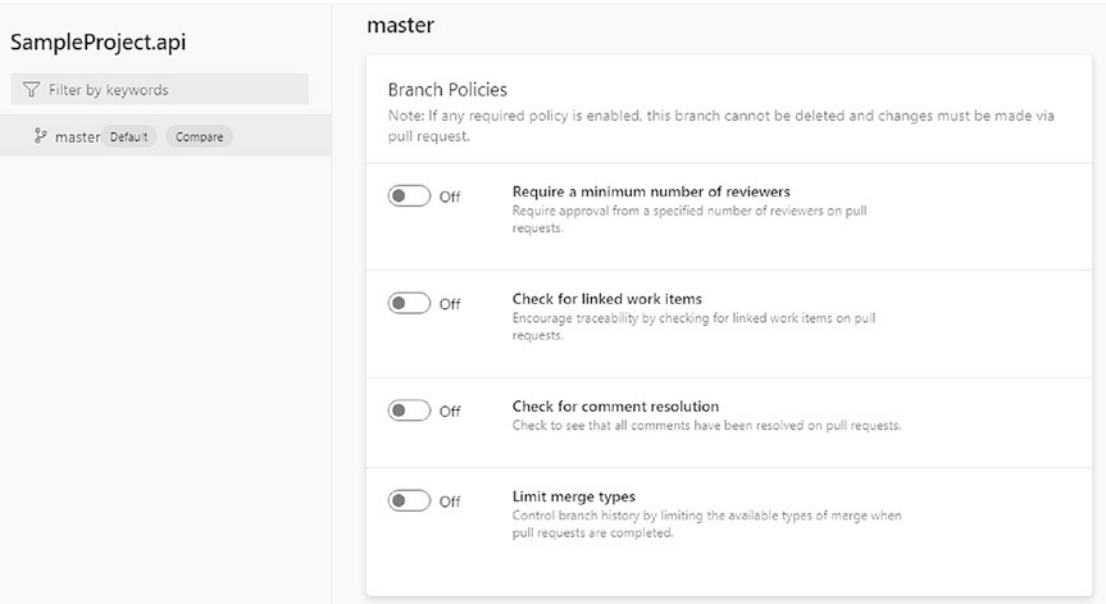


Figure 2-45. *Branch policies*

"Require a minimum number of reviewers" defines the number of reviewers that can approve a pull request and the actions related to the pull request. In Figure 2-46, the policy is configured to have a minimum of two approvers to complete the merge operation. If one of the reviewers selected Wait, the operation still completes due to the setup shown here in the policy.

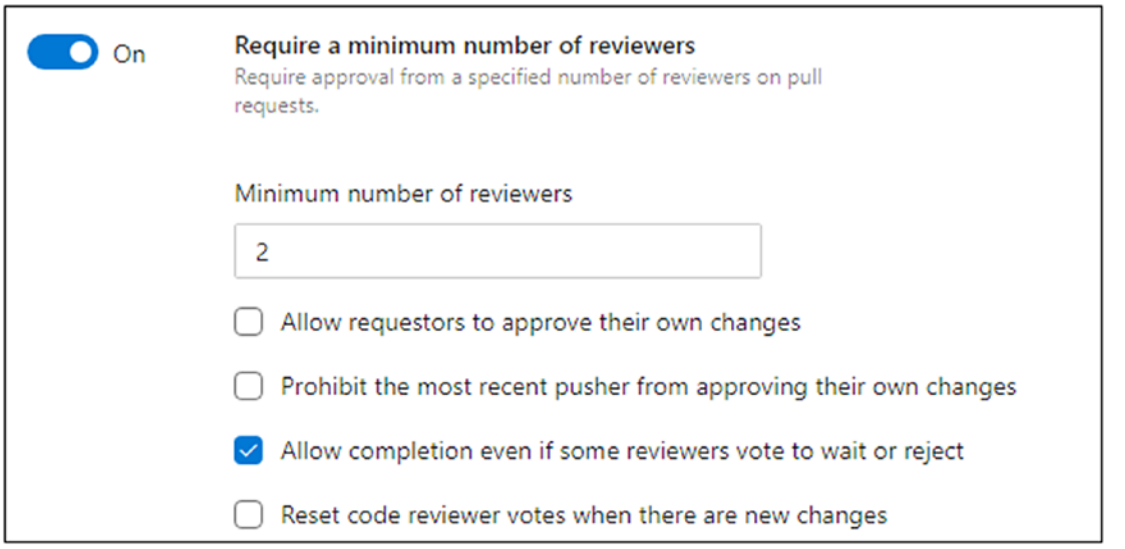


Figure 2-46. *Requiring a minimum number of reviewers*

It is a good practice to link the code changes to one or more work items; this will enable the traceability of the application. *Traceability* means the identification of code changes related to a specific requirement. It also traces the bugs raised against the requirement and the test cases executed against the code changes and establishes an end-to-end view from requirements to release. Select the Required option under "Check for linked work items" to enforce the linking of a minimum of one work item against the code changes, as shown in Figure 2-47.

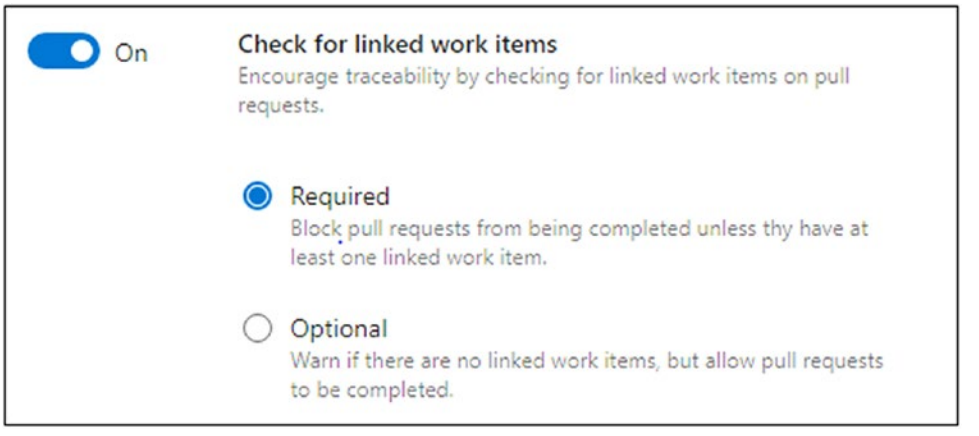


Figure 2-47. *"Check for linked work items" options*

The "Check for comment resolution" policy will be discussed after understanding the usage of comments as part of pull requests. Review comments provide an option for collaboration and pass messages to another person about the Git commits.

The next policy to limit the merge types controls the supported merge types between two branches, as shown in Figure 2-48. You'll learn more about merge types in Chapter 4.

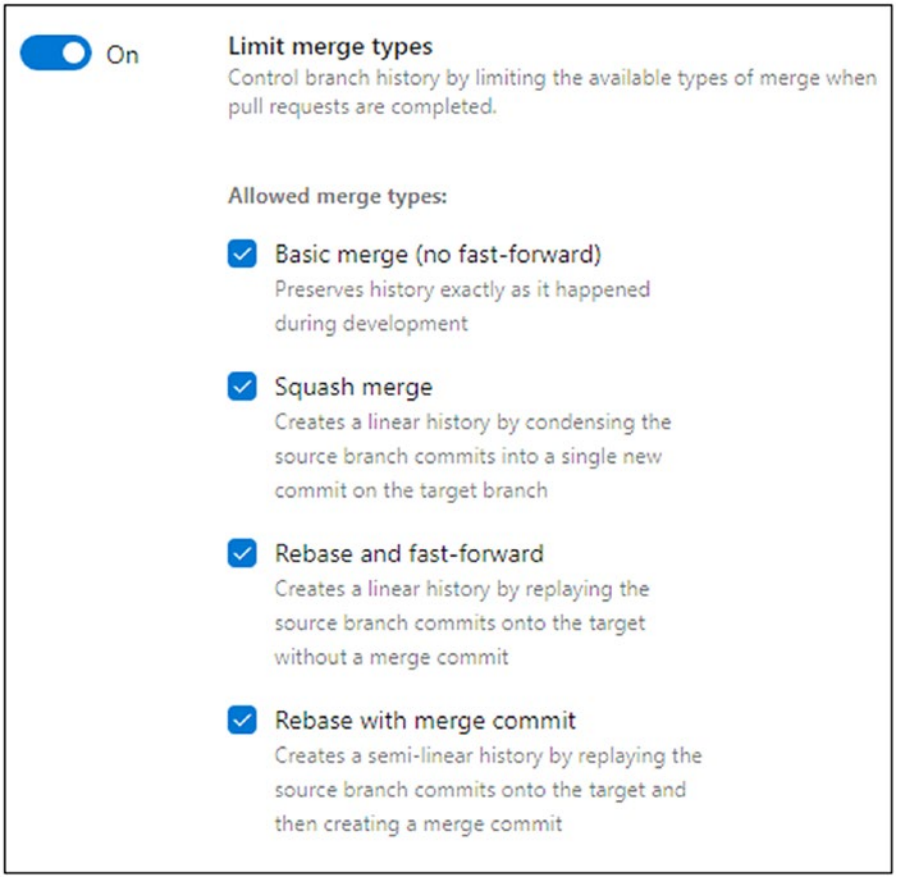


Figure 2-48. *Merge types*

By default, Azure DevOps provides four types of merges.

- *Basic merge*: This creates a merge commit to the target by pointing to the source and target as parents. This preserves all the history information. When the number branches increase, then the commit history in the target or parent branch is more complex and will be difficult to manage.

- *Squash merge*: Instead of adding all the commits from the source branch, squash merging takes all the changes and adds them as a single commit to the target branch. This will reduce the commit history in the target or parent branch.
- *Rebase and fast-forward*: Rebase rewrites the changes from one branch to another without creating any commit.
- *Rebase with merge commit*: After rebasing the target branch, do a merge commit to complete the merge operation.

Scroll down to view the policy setup related to the build validation, status checks, and automatically included reviewers, as shown in Figure 2-49.

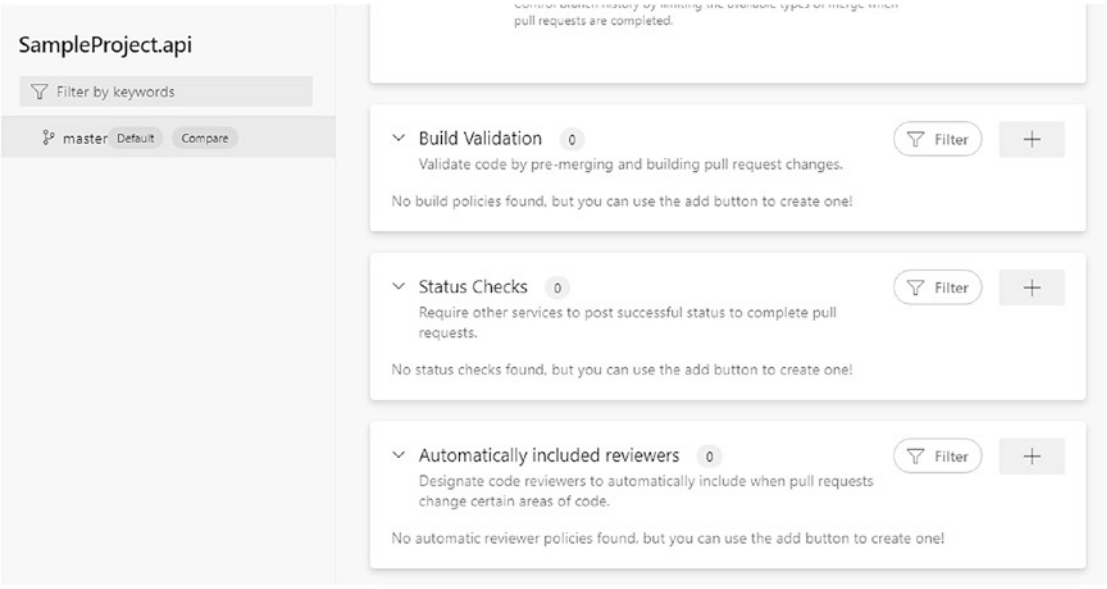


Figure 2-49. *Branch policies, expanded*

Add a new "Automatically included reviewers" policy to control the pull requests. For example, pull requests to the master should be reviewed by two reviewers to ensure the proper code quality and logic implementation, as shown in Figure 2-50.

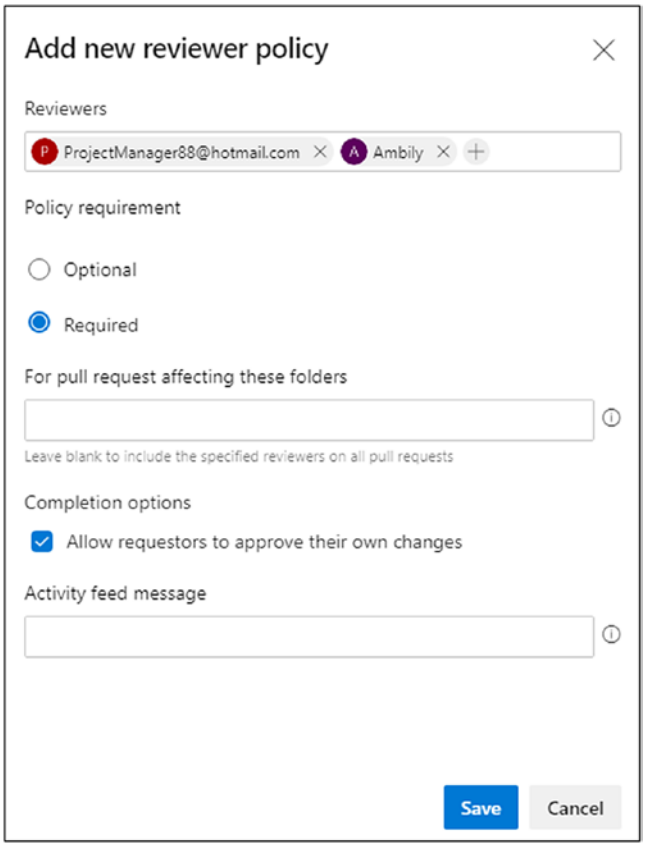


Figure 2-50. *Branch policies, reviewer policy*

Repo Permissions

The repository permissions section displays the permissions against each of the defined groups and users. Also, this section provides an option to configure the permissions related to tags and branches, as shown in Figure 2-51. Different supported permission values are Deny, Allow, and Not set. Deny will deny the specific permission, whereas Allow will explicitly allow the permission. If the value is "Not set," then it can be overridden by the permissions of a parent group, which will be displayed as Allow (inherited) or Deny (inherited).

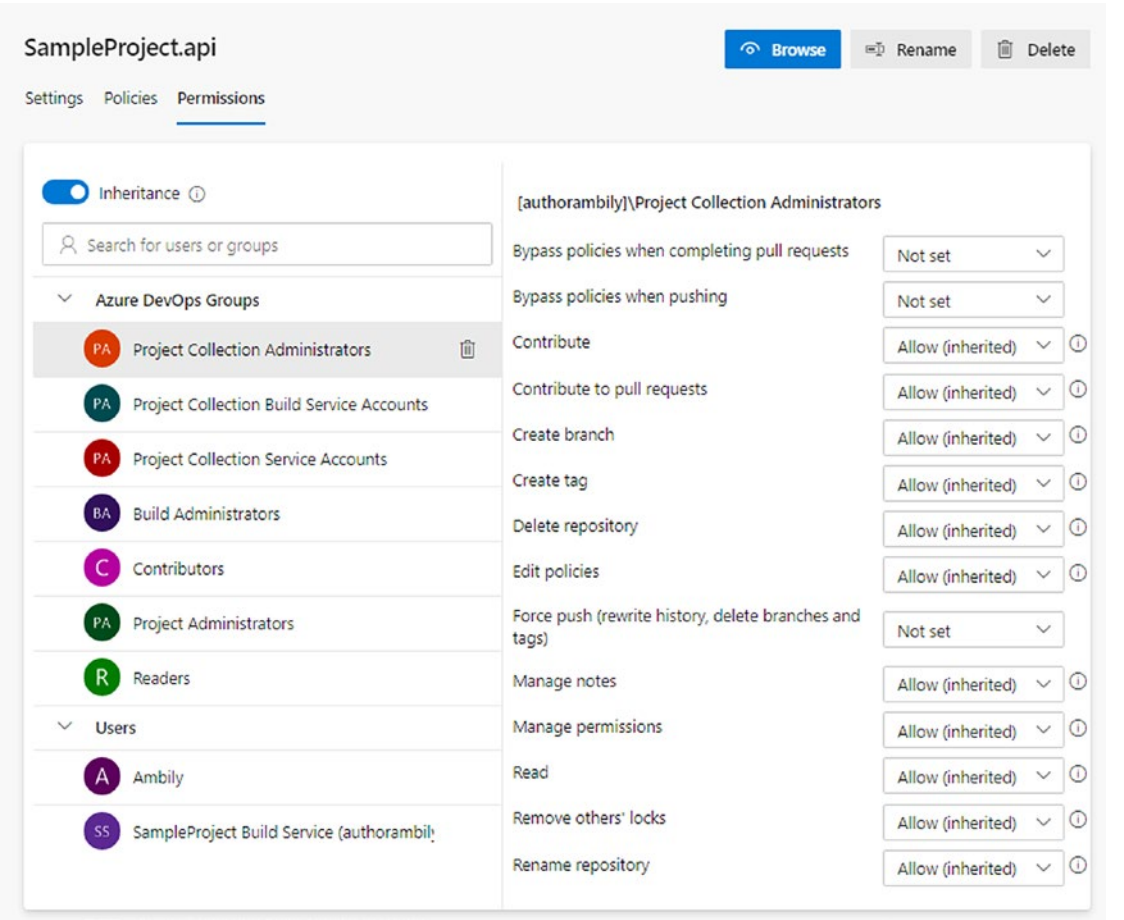


Figure 2-51. *Repo permissions*

Scroll down to view the permission section against the Git repo, as shown in Figure 2-52.

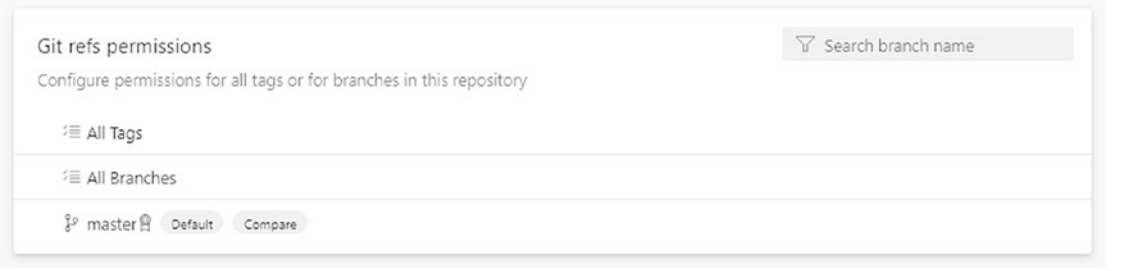


Figure 2-52. *Repo permissions: Git refs permissions*

Select a branch to modify the permissions associated with the branch. Similarly, permissions related to the tags can be modified here.

All Repositories: Settings

The Settings section related to all repositories allow you to configure the Gravatar images for external users outside the organization, as shown in Figure 2-53.

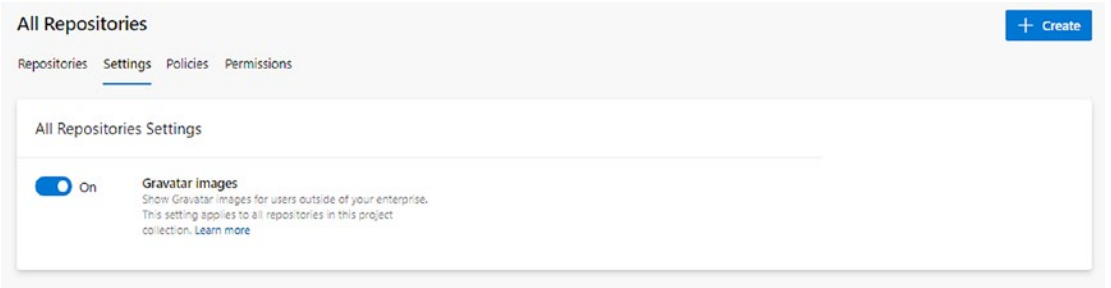


Figure 2-53. *All Repositories settings*

All Repositories: Policies

The Policies section under All Repositories allows similar configuration as for repo policies. In addition to the normal policies, this section allows you to protect namespaces associated with all the repositories.

You can add new branch protection using the plus sign available in the "Branch policies" section. See Figure 2-54.

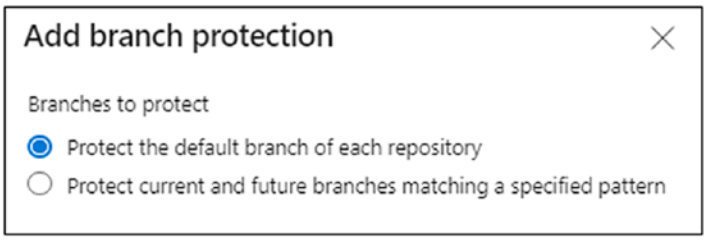


Figure 2-54. *Adding branch protection*

Select the type of protection required for the branch to continue. Once you select the protection type, select different branch policies discussed earlier in the "Branch Policies" section. Users can protect the current and future branches using a pattern

such as `master`, `releases/*`, `dev`, etc. The user can either specify the branch name or provide a pattern indicating the branch name. For example, * indicates all branches in the repo.

Pipeline ➤ Agent Pools

Pipeline jobs are executed by agents. Agent pools are collections of agents grouped together based on the location or features. By default, Microsoft provides a hosted agent pool to run the jobs. Different agent configurations are as follows:

- *Microsoft-hosted agents*: Agents are hosted by Microsoft and available based on your subscription mode.
- *Self-hosted agents*: Users can host or configure agents on their own systems or in a cloud subscription.
- *Azure Virtual Machine Scale Set (VMSS) agent*: These are self-hosted agents, autoscaled to support the demand.

Each agent pool shows the jobs running on the pool, agents configured against the pool, owner, and permissions. We will look at agent pools in Chapter 6.

Pipeline ➤ Parallel Jobs

Parallel job configuration was explained in detail as part of organization settings. The project-level settings help you control the usage of agent pools and parallel job availability.

Pipeline ➤ Settings

The Settings section under Pipeline configures the retention policies associated with runs and pipeline. Also, this section provides options to configure a few of the pipeline parameters such as publishing metadata from a pipeline, disabling anonymous access to a status badge API, and so on.

Pipeline ➤ Test Management

The Test Management section deals with the flaky test cases. These are test cases that result in different outcomes for the same set of data and without any code changes. These kinds of test cases may result in developer focus on unnecessary areas.

Detecting and fixing flaky test cases reduces the build failures. Configure the different options to detect the flaky test cases in the "Test management" section, as shown in Figure 2-55.

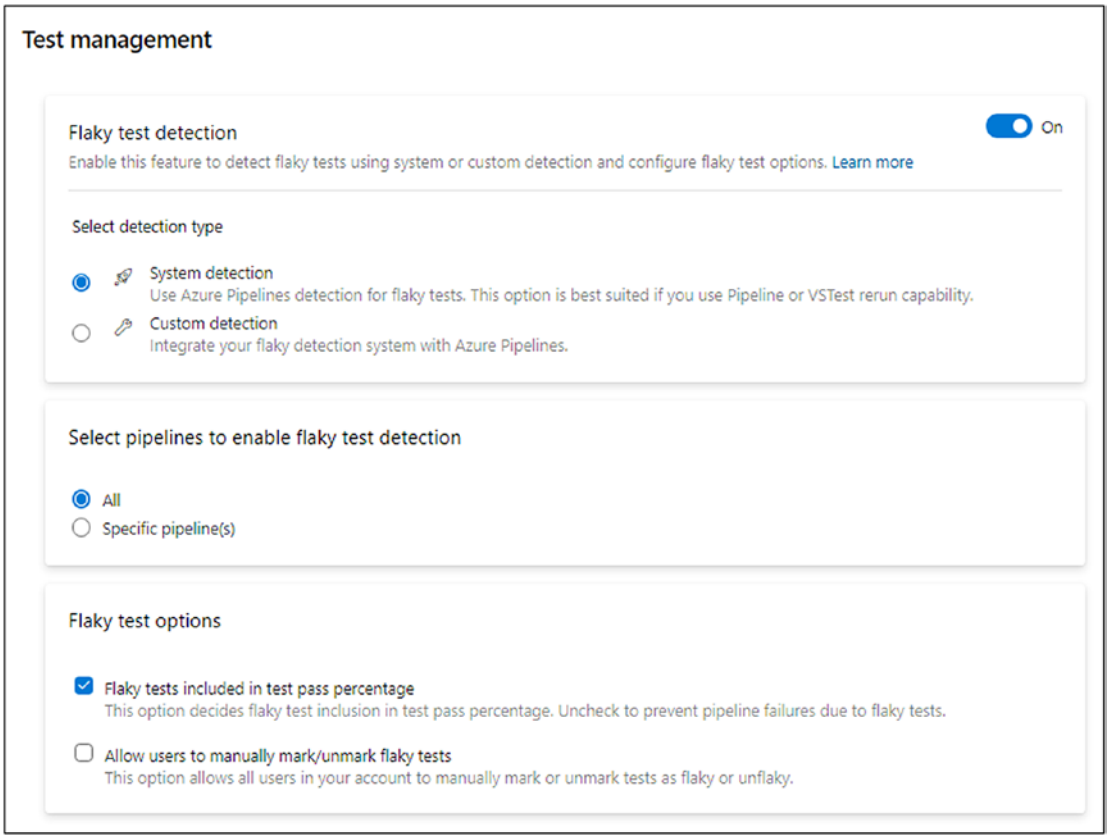


Figure 2-55. *Test management settings*

Pipeline ➤ Release Retention

The "Release retention policies" section, as shown in Figure 2-56, provides options to configure the retention of release and release artifacts.

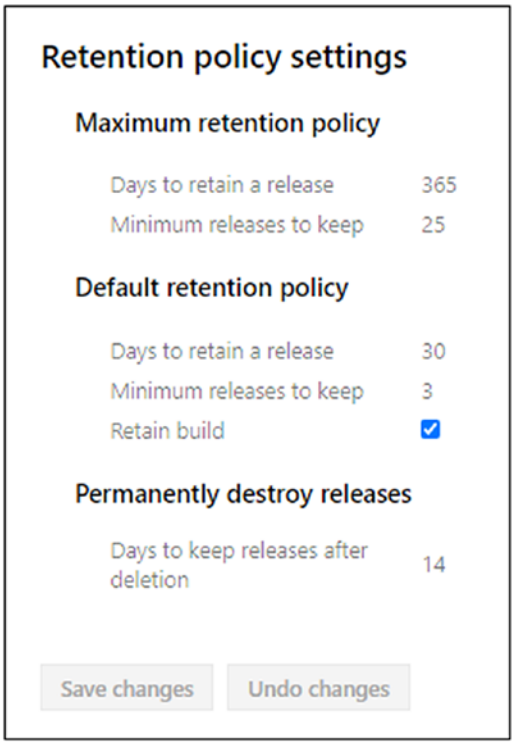


Figure 2-56. *Release retention policies*

Pipeline ➤ Service Connections

This section presents the connection settings to external systems. If the deployment of the project is in Azure, then define a service connection to the Azure subscription using the various supported mechanisms. The "New service connection" option shows the list of supported external systems including Azure, Chef, GitHub, Jenkin, SSH, and so on. You'll learn more about services connections and how to configure various connection in Chapter 6.

Pipeline ➤ XAML Build Services

XAML builds are the old build setup based on Windows Workflow Foundation (WWF). This section supports any legacy XAML builds available in your project.

Test ➤ Retention

This section defines the configurations related to the retention of test runs and associated results, as shown in Figure 2-57. Configure the retention period based on the enterprise requirements.

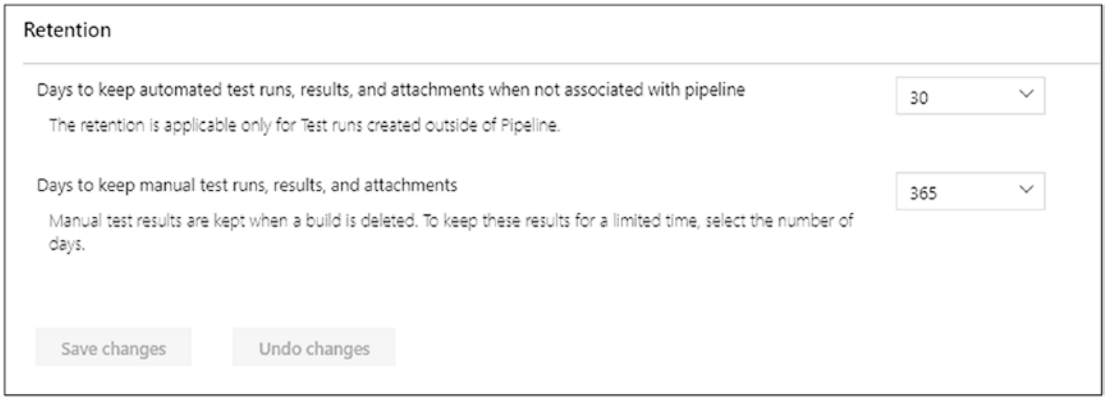


Figure 2-57. *Retention of test results*

Process Template Customization

Azure DevOps provides four process templates: Basic, Agile, Scrum, and CMMI. Based on the enterprise requirements, you may require basic customization of existing process templates. Complex process customization is not recommended.

Starting the Customization

Navigate to Organization settings ➤ Boards ➤ Processes to start the process of customization. Select one of the existing templates as the base template and select the option "Create inherited process," as shown in Figure 2-58.

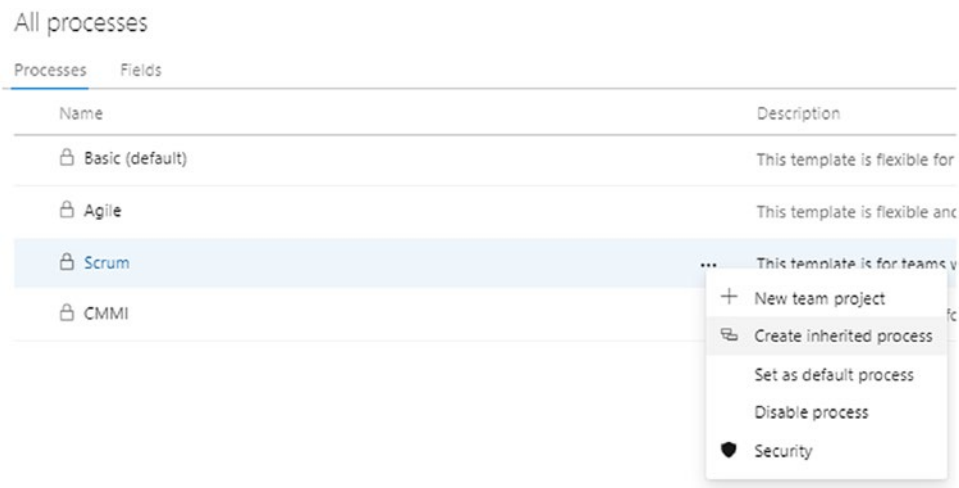


Figure 2-58. *Creating an inherited process*

Provide a name for the custom process template, say SampleScrum. Once it's created, select it to navigate to the work items associated with it. Initially the custom template will have the same set of work items as the base process template. You can add new work item types to the process template as well as edit existing process templates.

Adding a New Work Item

Select the option "New work item type" on the top of the work item types listed. Provide a new name for the work item and description as well. The user can select an icon and an icon color for the new work item type, as shown in Figure 2-59.

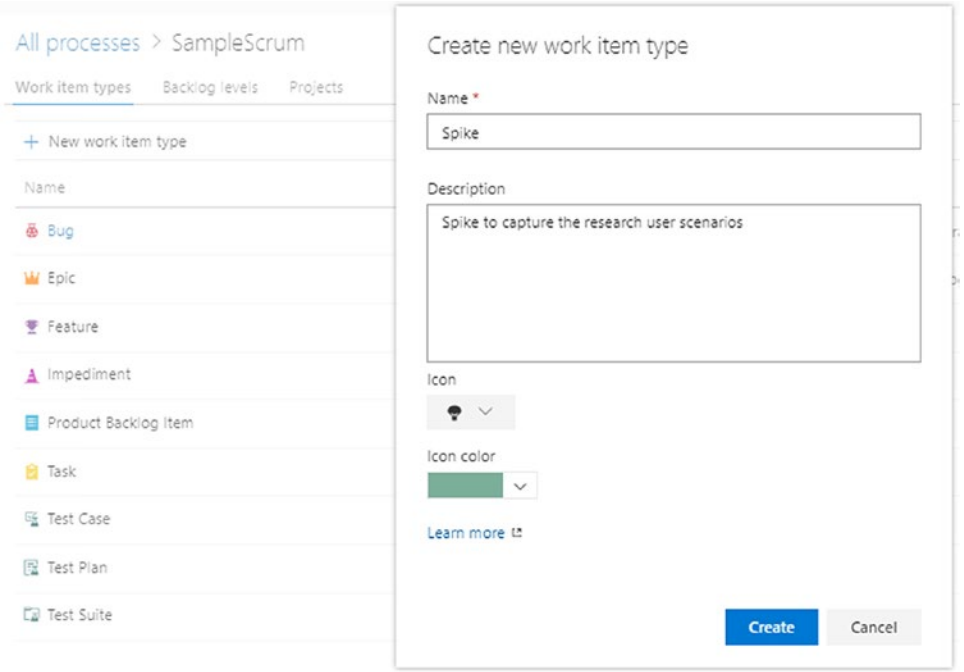


Figure 2-59. *Creating a new work item type*

The structure of the work item will be defined as three main components, called layout, states, and rules, as shown in Figure 2-60.

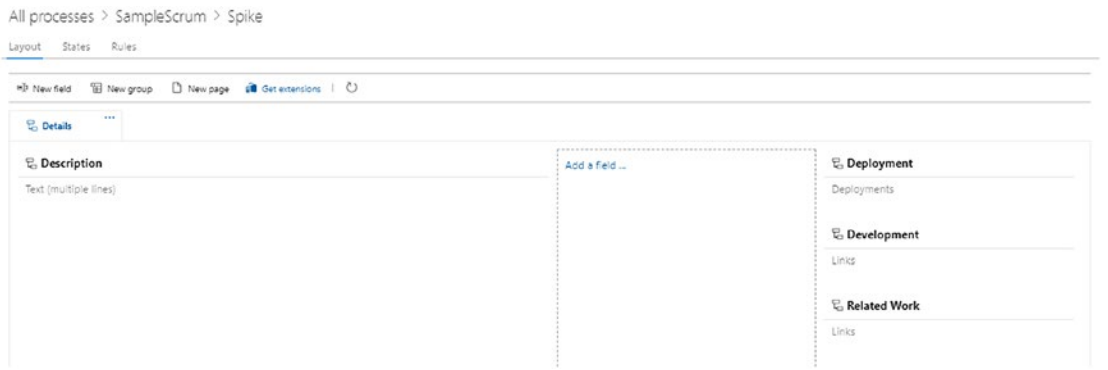


Figure 2-60. *Creating a new work item type*

- *Layout*: Layout defines the basic structure or elements of the work item. User can use the options "New field," "New group," and "New page." The "Get extensions" options helps in adding more complex controls available in Marketplace like the Multivalue control, color picklist control, and so on. The "New field" option opens the "Add a field to *new work item*" dialog, as shown in Figure 2-61. Select an existing field or add a new field. Specify whether the field is mandatory and specify the default value of the field using the Options tab. The Layout tab helps arrange the new field in the work item.

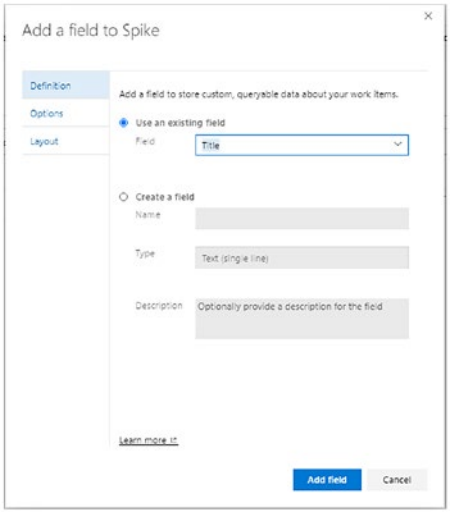


Figure 2-61. *Creating a new work item type*

- *State*: By default, there will be three states defined: New, Committed, and Done. The user can add new states under different predefined categories.
- *Rule*: The user can define rules associated with the work item. Figure 2-62 shows an example of a custom rule, which marks the priority field as mandatory when a new work item is created and marks the default value as 3.

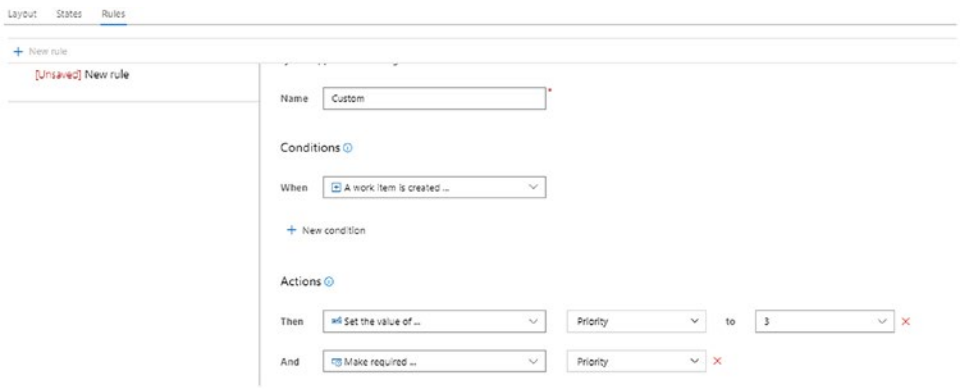


Figure 2-62. *Creating a new work item type*

> Rules will be defined as a combination of a set of conditions and actions.

Similarly, you can edit and disable an existing work item. Disable will remove unwanted work items from the list of available work items associated with the process template.

Summary

This chapter focused on the different organization and project configuration settings. These configurations and permission settings are used across the life cycle of the project execution. Each of the settings will be revisited based on its usage in other areas of the life cycle. With the organization and project configuration completed, we can explore how to manage the requirements using the requirements management features of Azure DevOps in the next chapter.

CHAPTER 3

Requirements Management Using Azure DevOps

Requirements management is important for the success of any project. A proper requirements management tool will help you gather the requirements from various stakeholders and from various perspectives, with an end-to-end traceability to different components of the software development. Azure DevOps provides a set of features to handle the requirements and associated functions. This chapter will deep dive into the requirements management features provided by Azure DevOps.

Process templates define the way the project gets executed and the different requirement categories like task, user story and so on. We discussed various process templates in previous chapters. The rest of this book will focus on scrum-based process templates. Other templates aren't that different except for the requirements handling.

Work Items

Work items capture the requirements in a systematic way. Based on the process template you have selected, different work items will be available for tracking the overall progress of the application development. When using the Scrum template, the following are the different work items available for tracking the requirements:

- *Epic*: This captures the high-level business requirements. If the epic is complex, it can be further defined into child epics to track the requirements properly.

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_3

- *Feature*: This defines the functional features being released as part of the application. Epics will be mapped to multiple features or product backlog items (PBIs) in a parent-child relationship, which will fulfil the expected business requirements.
- *Product Backlog Item*: This will track the user-specific actions.
- *Task*: This is the implementation work that needs to be accomplished to develop the product.
- *Bug*: This tracks any functional or nonfunctional defects in the implementation.
- *Impediment*: This tracks the challenges in executing the project.
- *Test Plan*: This captures the overall test strategy followed by the team.
- *Test Suite*: This is set of test cases is used for testing a specific feature, release, or module.
- *Test Case*: This is set of steps, that will be executed to validate the correctness of the feature implementations.

The "Work items" section is where you define new work items and associated functions. Let's explore the different features available in this section.

Work items

The New Work Item option is used to add new work items to the project. The Scrum process template allows you to capture the requirements using work item templates, as shown in Figure 3-1.

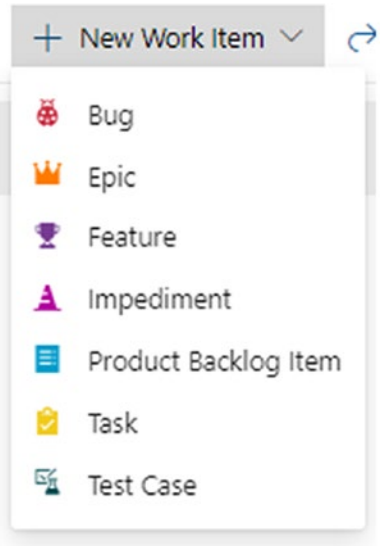


Figure 3-1. *New Work Item option*

Create an epic to explore the different fields and options available as part of a work item, as shown in Figure 3-2.

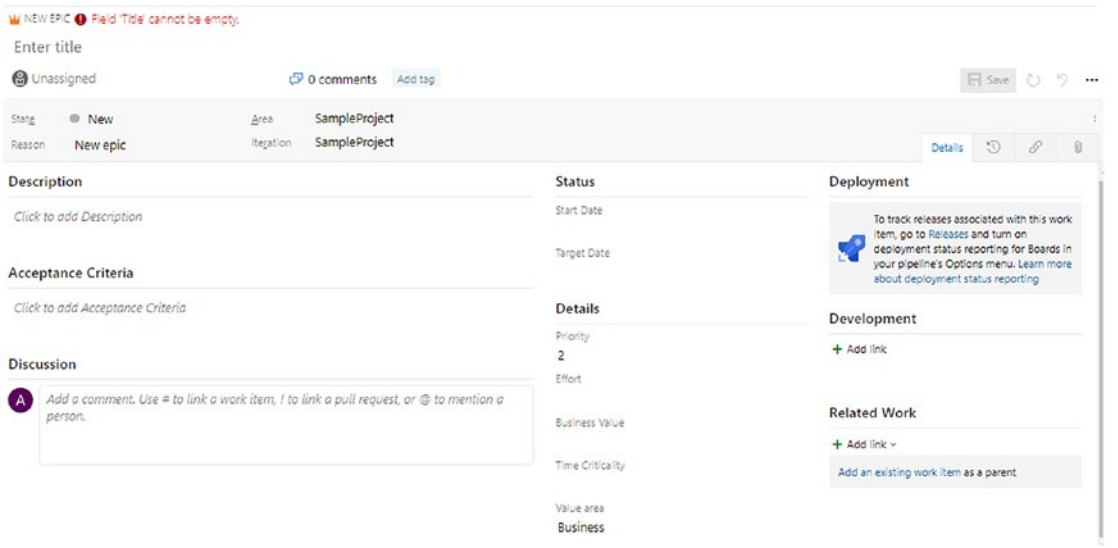


Figure 3-2. *New epic*

Work Item Fields: Details Tab

The Details tab captures all the relevant details associated with the work items such as the description, priority, and so on.

- *Title*: This is a brief description of the requirement. For example, the description "Mobile Experience" means that the business expects the application to run properly on a mobile device.
- *Assigned To*: Once the epic is ready to start working, it can be assigned to a team member to refine it.

Note *Backlog refinement*, formerly known as *backlog grooming*, is the process of consolidating different viewpoints, boundary conditions, risks and mitigations, option validations, and overall aspects of the requirement. Different stakeholders discuss the requirements and capture all the aspects, which helps the implementation team to fulfil the requirement. Another important concept is *backlog prioritization*, where teams prioritize the product backlog items to sequence the development and deployment of features.

- *State*: This defines the current progress of the work item. The State values for an epic are New, In Progress, Done, and Removed.
- *Area*: This lists the area mapped to the team in Project Settings ➤ Team Configuration.
- *Iteration*: Similarly, an iteration defines the iterations mapped for this team under Project Settings ➤ Team Configuration. If a project-level area or iteration is not visible to the team, then verify it under Project Settings ➤ Team Configuration.
- *Description*: This field explains the requirement in detail, which provides enough details for the implementation team to meet the requirement.

- *Acceptance Criteria*: This defines the verification condition, which verifies the completeness of the work item implementation. For the task level, the acceptance criteria can be the technical implementation of that task. From an epic level, this can describe how you verify the implementation of a business or architectural epic.
- *Discussion*: This field provides an option to collaborate with other team members to discuss various details about the work item, as shown in Figure 3-3. You can ask questions related to the work item to specific people by adding the member name with an @ symbol, you can refer to another work item with the # work item ID, and you can provide feedback about the description or acceptance criteria of the work item.

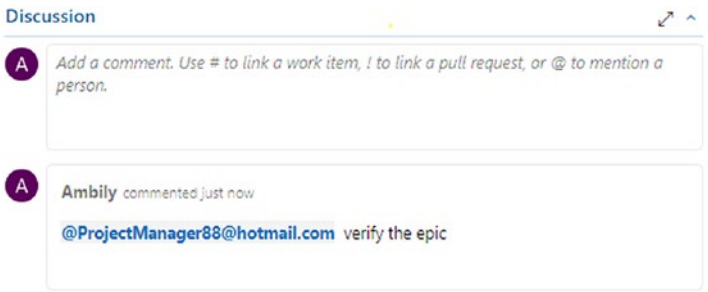


Figure 3-3. *Discussion section*

- *Status*: Under Status, there are options to specify the start date and end date of implementing the specific epic. This data will help the implementation team to prioritize and calculate the overall effort for developing the epic.
- *Details*: Under Details, data is captured for multiple fields.
  - *Priority*: This defines the priority of the epic.
  - *Effort*: This is the estimated effort for implementing the specified epic. This field helps the team to define the relative efforts for the features, product backlogs, and tasks derived from this epic.

    - *Business Value*: Here, you specify a priority that captures the relative value of an epic, feature, or backlog item compared to other items of the same type. The higher the number, the greater the business value. Use this field when you want to capture a priority separate from the changeable backlog stack ranking in Azure Boards.
    - *Time Criticality*: This is a subjective unit of measure that captures how the business value decreases over time. Higher values indicate that the epic or feature is inherently more time-critical than those items with lower values.
    - *Value Area*: This is the customer value addressed by the epic, feature, or backlog item. Values include the following:
        - *Architectural*: Technical services to implement business features that deliver a solution
        - *Business* (Default): Services that fulfill customers or stakeholder needs that directly deliver customer value to support the business
- *Deployment*: This section relates to the release pipeline and shows which release contains the implementation of this epic. This will establish an end-to-end traceability for each of the business requirements. This field will be discussed further in Chapter 6.
- *Development*: This section links the epic to the branch that holds the implementation, the build that contains the changes, or the commits that capture the implementation. Select the "Add link" option to link to an entity explaining the implementation of the epic. This process can be automated using the repo policies defined in the previous chapter.
    - *Link Type*: This shows the linked item type. This can be as follows:
        - *Build related*: This links to the build containing the implementation of the specific epic using the following options: Build, found in build, or Integrated in build.

    - *Code related*: This links to the implementation code using the following options: Branch, Changeset, commit, Pull Request, or Versioned Item. In large projects, epics may hold a completely new feature, and the development team may create a new branch to handle the feature development. In such a scenario, the epic will be linked to the corresponding branch to establish the end-to-end traceability. Commits and pull requests are used to link to Git-based version control, whereas Changeset and Versioned Item are used for TFVC.
    - *GitHub:* This is the new link type introduced to handle the GitHub integration with Azure DevOps. If your version control is based on GitHub, use this option to link the implementation with an epic. The options available are GitHub Commit and GitHub Pull Request.
- The remaining fields change based on the Link Types selection. If this is set to Branch, then the options allow you to select a repository associated with the projects in the organization. Once the repository is selected, it displays all the branches related to that repository for selection. Provide a comment to complete the linking of an epic with the implementation code or build. If Link Type is set to Build, the system provides an option to select one of the build number. Similarly, all other selections will provide respective fields to link to it.
- *Related Work*: Before proceeding to define a related work item, save the current epic. Once the epic is saved and an ID is generated, go to Related Work to define the related work items. There are two options to add an existing item or a new item. Existing Item provides an option to link an existing work item with a relation.

Select New Item to explore more about the related work item.

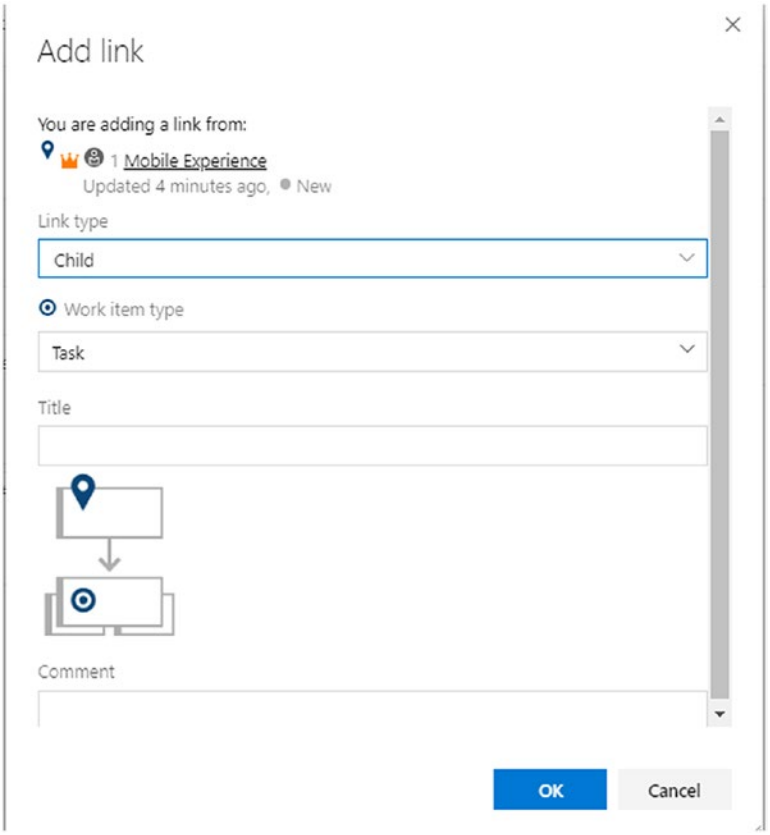


Figure 3-4. *Adding a link*

New Item provides the option to define the link type, as shown in Figure 3-4, (which is different from the previous link type) to handle the relationship with the epic. The available link types are as follows:

- *Child*: This defines a parent-child relationship. In the case of an epic, it can be further defined into features, product backlogs, tasks, and so on.
- *Duplicate*: This allows you to duplicate the same epic to handle another variation or to add to another project to implement similar feature.
- *Duplicate Of:* This specifies that the current epic is a duplicate of another one defined in the same project or a different project.

- *Parent*: A new item is going to be the parent of the current work item.
- *Predecessor*: This establishes a sequential relationship to execute the dependent requirements in sequence. A new item should be created before implementing the current epic.
- *Related*: Both the items are related, but they may not have any specific relationship like parent-child or sequential execution.
- *Successor*: A new item will be the next item to be implemented.
- *Tested By*: A new item implementation will be tested by the epic description or used for verification.
- *Tests*: A new item consists of the test cases corresponding to the epic, as shown in Figure 3-5.

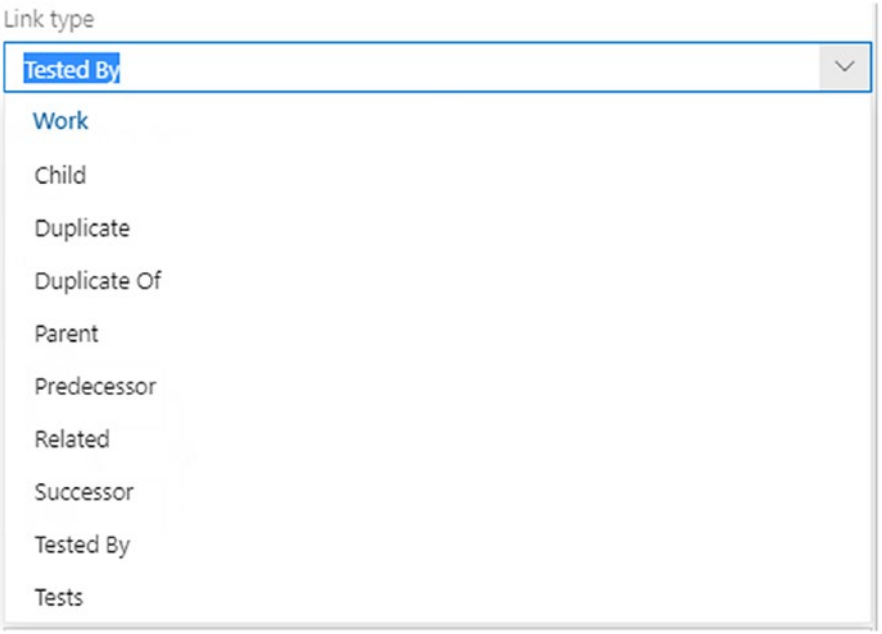


Figure 3-5. *Link types*

Add a child item to proceed. Once you provide the title information and click OK, the system will open the new item in a dialog so you can fill in the required information. Fill in the required details and select Save & Close to navigate back to the epic.

Work Item Fields: History Tab

In a work item, the History tab captures all the changes over time, as shown in Figure 3-6. All the operations are tracked by Azure DevOps for audit purposes. This tab shows the changes done by whom and when. Also, it captures the change details such as adding a new child item.

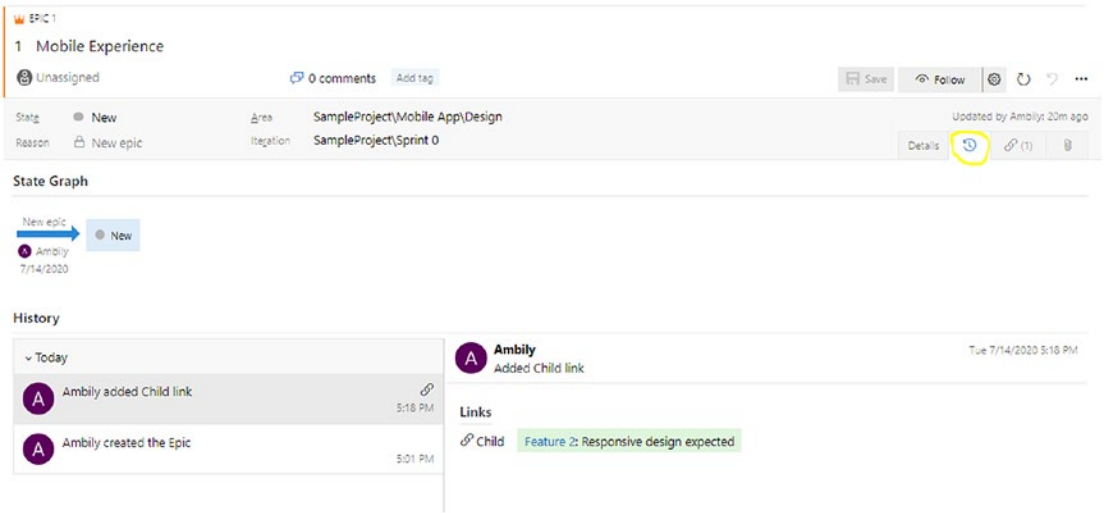


Figure 3-6. *History*

Work Item Fields: Links Tab

Navigate to the Links tab to view all the related items, as shown in Figure 3-7. The system provides an option to add new or existing work items from this tab as well.

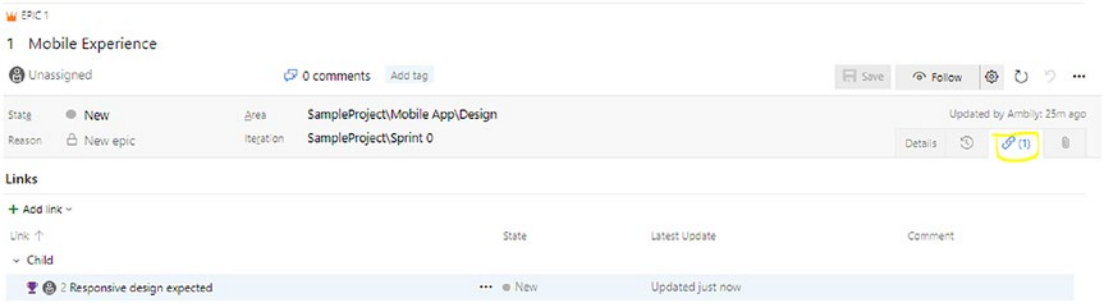


Figure 3-7. *Links tab*

Select the ellipsis icon (…) to view the options available. For each linked item, the system provides two options.

- *Edit comment*: You can edit the comment entered as part of linking an item.
- *Remove link*: You can remove the relation or link between two work items.

Work Item Fields: Attachments Tab

The Attachments tab provides an option to attach the related documents such as the architecture diagrams, wireframes prepared for UI design, technical design decisions, or any other artifacts related to the epic, as shown in Figure 3-8.

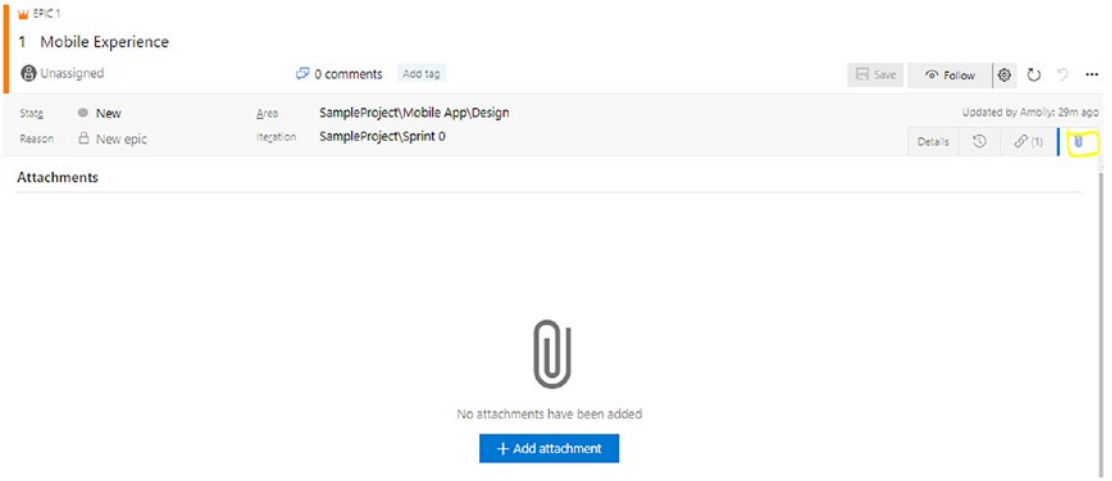


Figure 3-8. *Adding an attachment*

Once it's added, the Attachment tab provides different views.

- *Grid View*: This lists the attachments in grid format.
- Click the ellipsis icon (…) to view the actions available for the added attachments.
  - *Edit comment*: This allows you to edit the comments entered as part of the attachment upload.
  - *Preview attachment*: This is a preview of the attachment based on its type. If it is an image, it will display a preview. But, if the attachment is a document, it will not show any preview.

- *Download attachment*: This allows you to download the attachment.
- *Delete attachment*: You can delete unwanted or wrongly attached documents.

- *Thumbnail View*: This lists the attachments in card format. Hover your cursor over the top of the card to view the context menu icon (…) and click it to view the menu options.

Work Item Options

Now, look at the options available on top of the epic defined. You have a set of options available in the top-right corner, which includes the Save button.

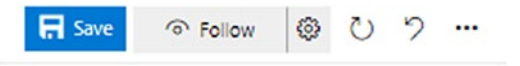


Figure 3-9. *Work item options*

Other options available are as follows:

- *Follow*: This allows you to be notified of the changes made to a work item.
- *Settings*: This includes notification settings at a work item level. If any of the work items are so important or really impacting your activities or have high business value, you can track them using these settings. The options available are as follows:
  - *Not Subscribed (Default):* This is not subscribed at the work item level. But, if the work item will be part of any of the project or organization notification settings, this will be displayed.
  - *Subscribed*: You can receive all notifications related to a specific work item.
  - *Custom*: This allows you to select specific notification options such as sending a notification only when there is a state change.

- *Refresh*: Refresh the item to reflect the changes made by another user or in another system; for example, a commit linked to the work item will get reflected immediately.
- *Revert*: Revert the unsaved changes. If there are no unsaved changes, then both the Save and Refresh icons are disabled.
- *Options (…):* This provides additional actions and configurations for a work item. The options available are as follows:
    - *New linked work item*: This is another option to add a linked work item.
    - *Change type*: Change the type of a work item; for example, you can change an epic to a feature or change a task to the product backlog based on detailed analysis. The "Change work item type" wizard provides an option to select the target type and provide a reason for the change. This information will be tracked as part of the history of the work item and can be audited in the future to understand the reason for the change.
    - *Move to team project*: Move the existing work item to another project within the organization. Also, this option allows you to change the work item type as part of the movement, if required. The "Move work item" dialog provides an option to select the target project, type and provide the reason for the movement for tracking purposes.
    - *Create copy of work item…*: Create a copy of the existing work item in the same or different project. This option also allows you to copy the work item to a different work item type. At the time of the copy, the system allows you to include related artifacts such as the attachments, links, and child work items.
    - *Email work item*: Share the work item over email. This option opens the email construction page with options to add to email IDs, specify a subject, add details on a note field, and attach the main details of the work item. Work item data can link to the work item by adding a hyperlink to the work item ID. The receiver of the email can open the work item directly using this link.

- *Delete*: Delete the current work item.
- *Templates*: Create a template to define the default values for a set of fields. For example, you can define a task template with a predefined area path for a team and share it with them, as shown in Figure 3-10.

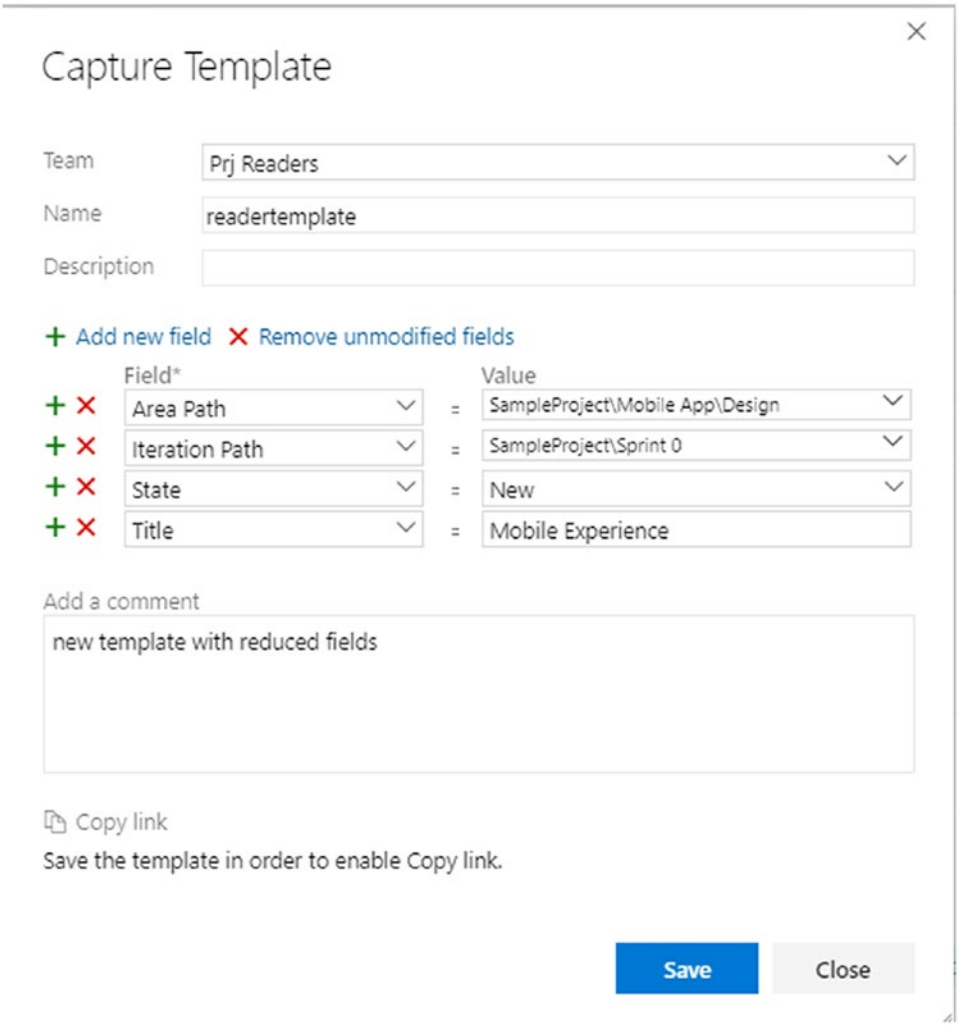


Figure 3-10. *Capturing a template*

Once it's saved, use "Copy link" to copy the link of the template, which can be shared with others or posted on your team configuration or dashboard to create new work items based on this template. The template can be applied by selecting the work item context menu as well.

- *New branch*: This option allows you to create a new branch out of the existing branch and link to the current work item. When the epic represents a feature and a feature branch strategy is followed by the team for the development of features, then this option supports the quick branching and linking operations.
- *Request Feedback:* You can request a team member or stakeholder verify the completeness of the current work item or provide feedback on the work item. This feature is used to fine-tune the work items before considering them as part of sprint planning or can be used to get different viewpoints or sign-off from concerned stakeholders, as shown in Figure 3-11.

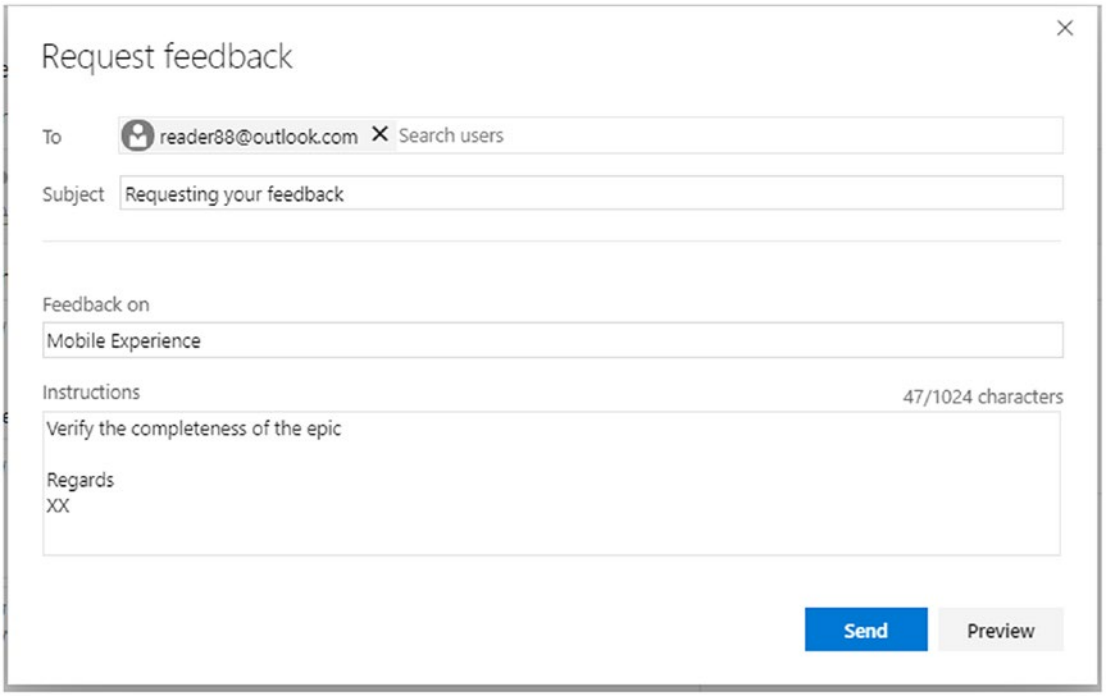


Figure 3-11. *Requesting feedback*

The "Request feedback" dialog allows you to specify the target members of the feedback request and instructions to provide the feedback, as shown in Figure 3-12. The Preview option shows the preview of the mail shared with the team.

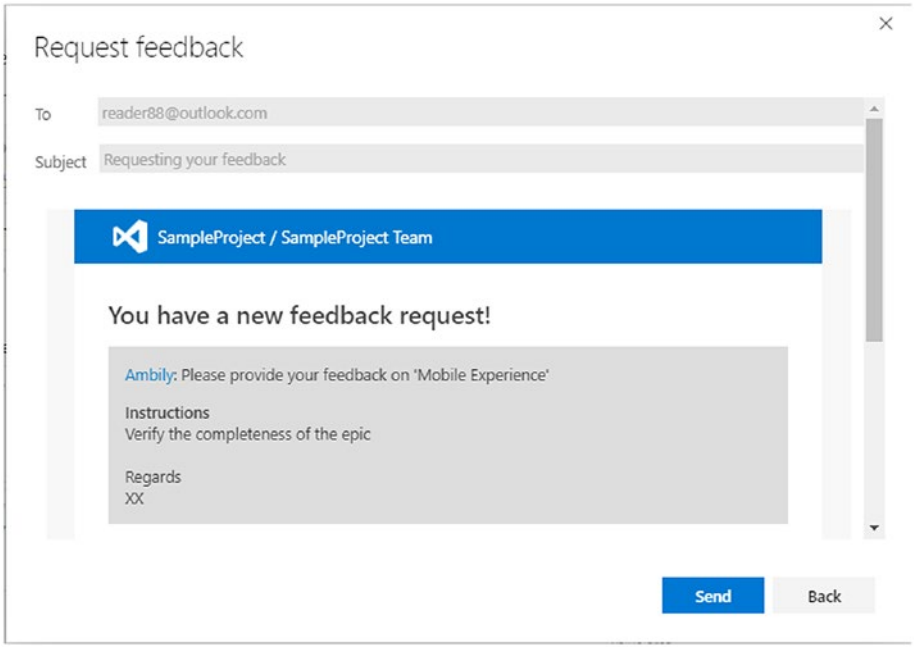


Figure 3-12. *Preview feedback request 1*

Figure 3-12 shows the preview of the request feedback email. The email captures the project name and the team that the requestor belongs to. Scroll down to view the "Provide feedback" button. Clicking this button takes the user to the specified work item, where the user can provide the feedback in the Discussion section.

- *Customize*: This will take the user to Organization Settings ➤ Boards ➤ process customization. Process template customization is covered in Chapter 2.
- *Keyboard shortcuts*: This option highlights the important keyboard shortcuts used in Azure DevOps. By clicking the link "Full list of keyboard shortcuts," you can refer to the complete set of shortcuts.

The remaining two options available on top of the work item are comments and Add tag.

- *Comments*: This provides the number of discussions added to the work item and a navigation link to the Discussion section in the work item.
- *Add Tag*: This option provides a facility to add tags to a work item. Tags help filter and query the work items.

Filters

Navigate back to the "Work items" section to view all the work items created so far, as shown in Figure 3-13.

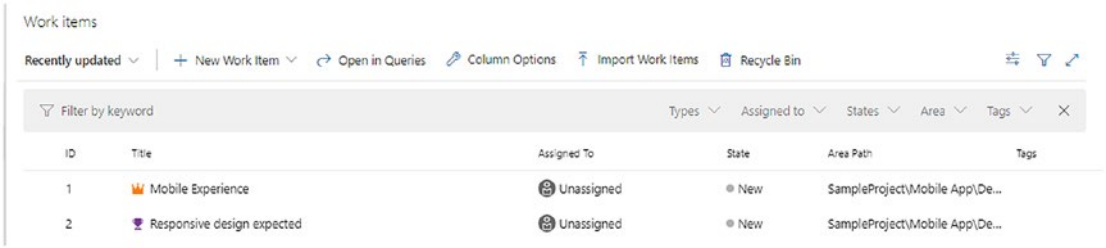


Figure 3-13. *Work items*

As you can observe in Figure 3-13, a set of filtering options is available just above the list of work items. These options help to filter the required work items. The filters available are as follows:

- *Filter By Keyword*: Filter the list of work items by using keywords such as *feature* or *epic* based on the requirement.
- *Filter by Type*: Filter the work items based on type. This filter option lists all the work item types available based on the process template. Multiple types can be selected to view the work items of various types.
- *Filter by Assigned to*: Filter based on the assignment of work item to team members.
- *Filter by State*: Filter based on the various states of the work item.

- *Filer by Area*: Filter by the areas defined.
- *Filter by Tags*: Filter by the tags. This option allows you to combine the tags using the OR and AND operators.

View Options

Figure 3-14 shows the view options available on the top-right side of the work items.

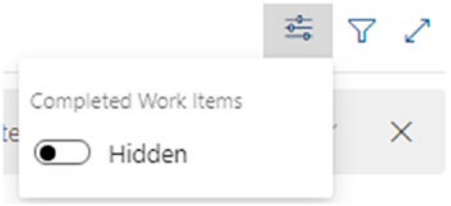


Figure 3-14. *View options*

- *View*: This will allow you to view completed work items in the list or not.
- *Filter*: This option allow you to view filter options, discussed in the previous section.
- *Full Screen*: The last icon allows you to view the work items list in full-screen mode.

Options

The following are the options available on top of the work item listing:

- *Recently Used*: This drop-down provides options to filter based on a few parameters such as the work items assigned to the user, work items the user is following, and items where the user's name is mentioned in discussions, related to his activities, and based on recent state updates.
- *New Work Item*: This option supports the addition of new work items to the project and was discussed already earlier in the chapter.

- *Open in Queries*: This opens the filtered work item list in the Query window, where you can add more filters and query elements. We'll discuss queries in more detail later in this chapter.
- *Column Options*: This option allows you to add new columns or remove an existing column from the grid view of the work items, as shown in Figure 3-15. Also, you can set the default sort order of work items based on specific columns.

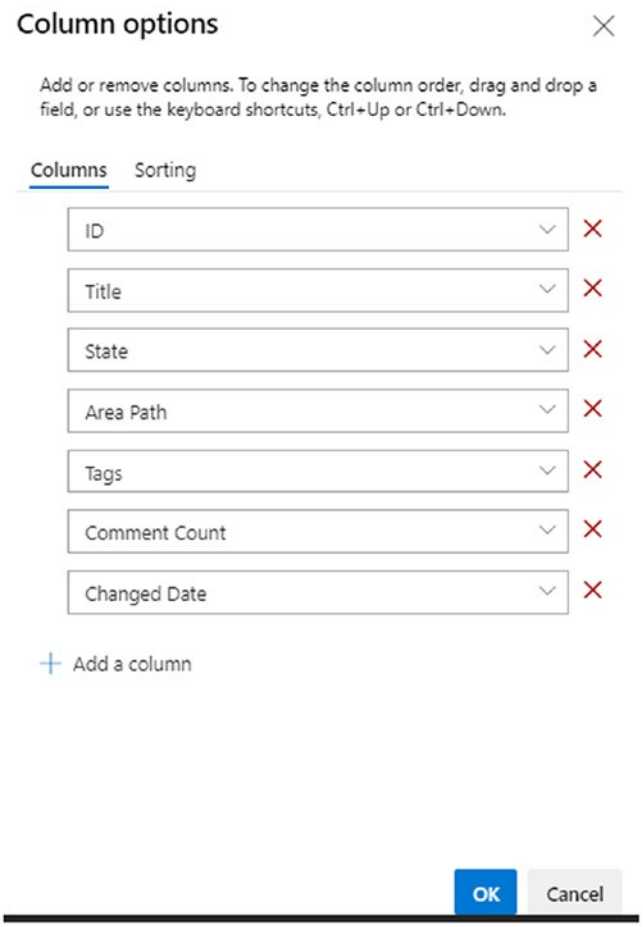


Figure 3-15. *Columns*

- *Import Work Items*: Work items can be prepared offline using CSV files and imported into the system in bulk.

  Explore this feature further using the sample CSV file content shown in Figure 3-16, with one task and one bug. If required, we can add more columns such as Description, Area Path, etc., and capture the requirement details.

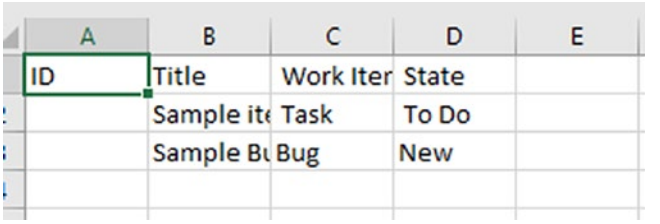


Figure 3-16. *Data*

Select the Import Work Items option to upload the file to Azure DevOps.

The system will parse the data based on the columns defined and shown to the user to confirm, as shown in Figure 3-17. The user can either edit the data or save it as is.

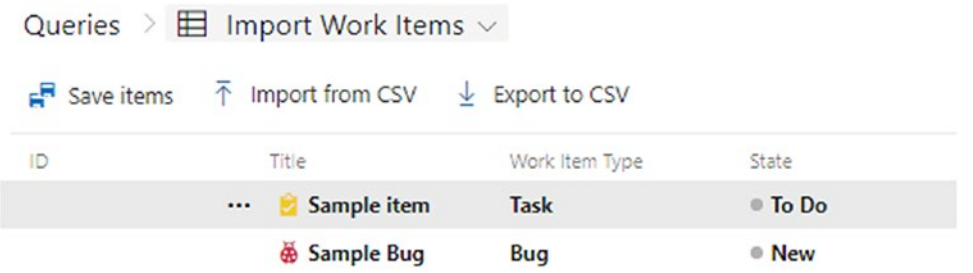


Figure 3-17. *Uploaded data*

Select Save Items to create the work items and generate the IDs for newly created work items, as shown in Figure 3-18.

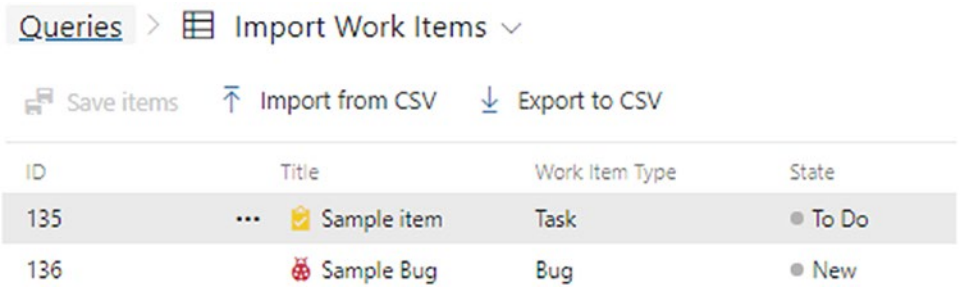


Figure 3-18. *Saving items*

- *Recycle Bin*: Azure DevOps allows you to remove the unwanted work items by changing the status to Remove. Once the work items are marked as Remove, they will be moved to the Recycle Bin, where the user can permanently delete the work item or restore it by changing the state.

Boards

Boards support another view of the work items and provide additional options to perform different activities related to the work items. By default, there will be one board configured for every team. The user can mark one of the boards as their favorite.

Select one of the boards to navigate to the specific board and view the work items, as shown in Figure 3-19. Each of the work items is displayed as a card with specific fields.

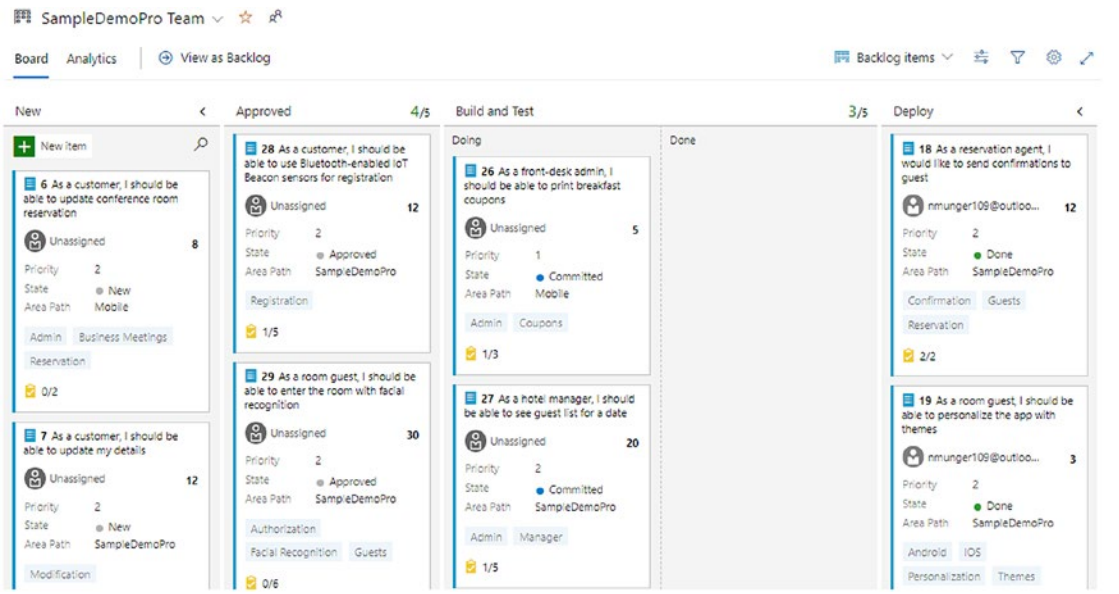


Figure 3-19. *Selected board*

Board Selection

There are multiple options available to process the board and work items. Let's start exploring the features from the top level. In the top-left corner, there are options to select a different board. Select another board from the drop-down list to navigate to a different board, based on your permissions, as shown in Figure 3-20.

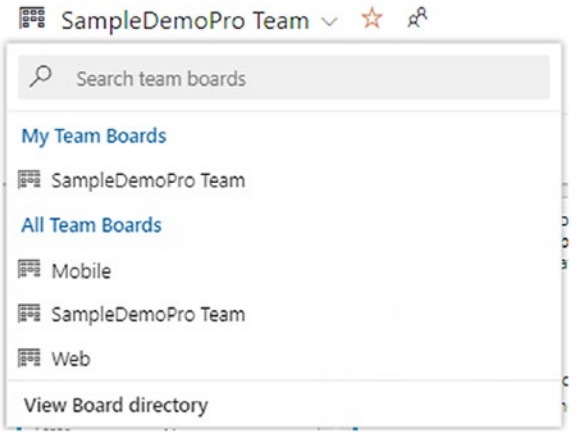


Figure 3-20. *Team board selection*

Use the star symbol on or near the team board selection to mark a specific board as your favorite. Select the people icon next to the star symbol to view the Team Profile page. This view lists the board, backlog, sprints, and dashboard corresponding to the team. Also, this view shows the members who belong to the team.

Analytics

The next set of options under the team board name consists of the following options:

- *Board*: Work items will be displayed as cards in a board.
- *Analytics*: This shows the analytics report in terms of cumulative flow diagram and velocity.

  The Cumulative Flow Diagram: Get insights about a selected team's workload, throughput, and potential bottlenecks, as shown in Figure 3-21. To customize the report, set the columns, swimlanes, and timeframe.

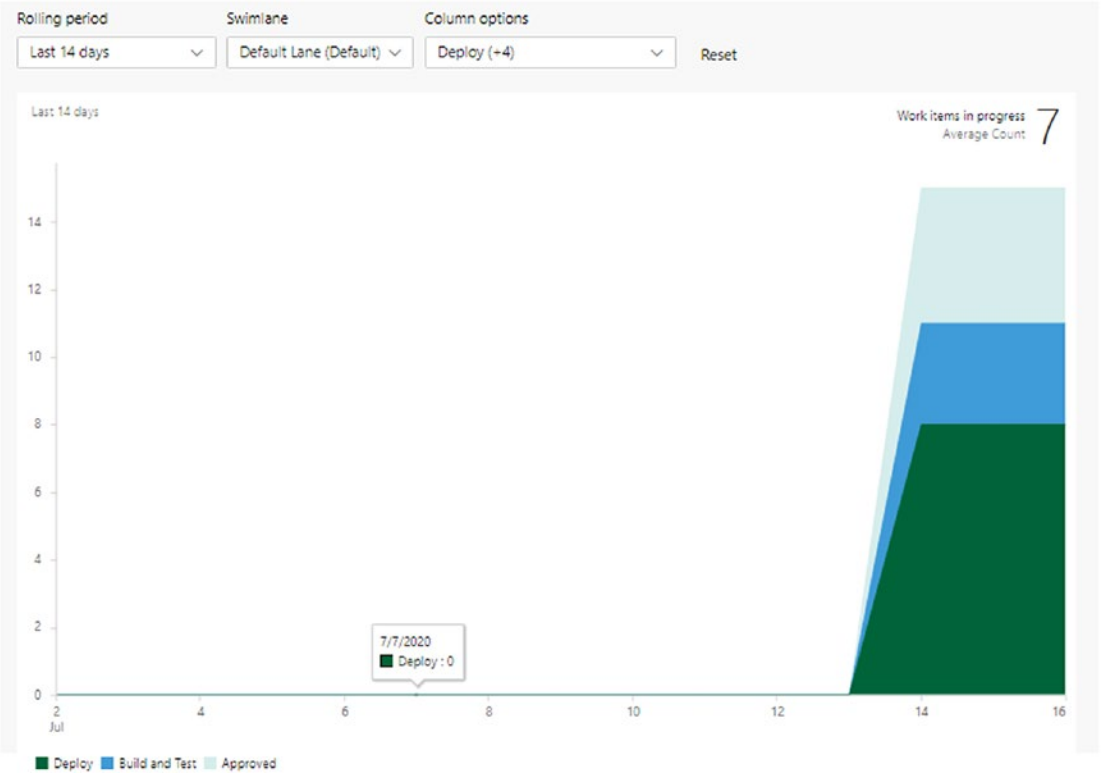


Figure 3-21. *The cumulative flow diagram*

Velocity Report: This shows the team's average speed in completing work across several sprints, as shown in Figure 3-22. Team velocity is a measure of how much work a team can complete during a sprint based on different metrics such as effort based on PBIs, story points based on user stories, and size based on requirements. Use this information to plan future sprint capacity.

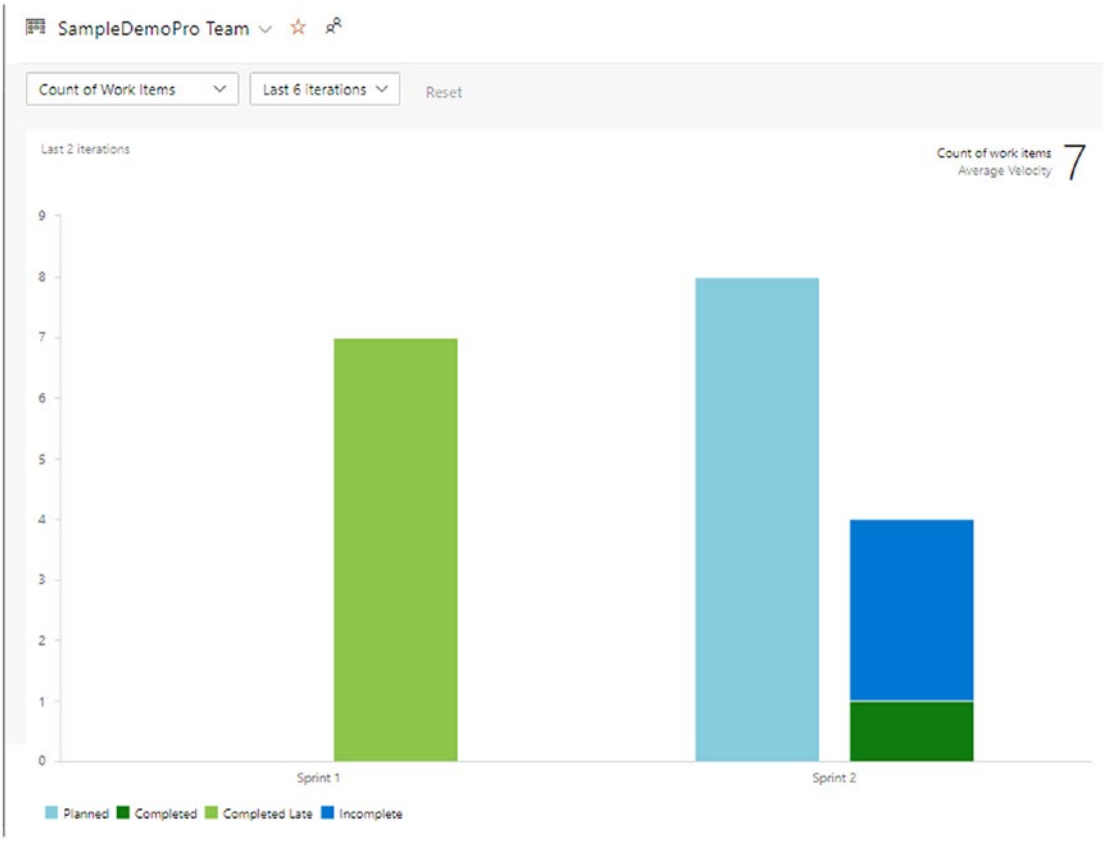


Figure 3-22. *A velocity report*

This report can be customized based on the effort, remaining work, business value, and count of work items along with the number of iterations to be considered.

- *View as Backlog*: Display the work items in a backlog view, which will be discussed in the next section.

View Options

Similar to the "Work items" section, boards also provide different view options at the top-right corner, as shown in Figure 3-23.

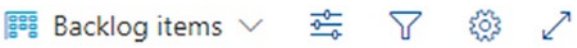


Figure 3-23. *View options for boards*

The first option allows you to view the board of backlog items, epics, and features. Based on the selection, the board changes the columns displayed and the work item types. Figure 3-24 shows the epics view of the board, where the columns have changed to New, In Progress, and Done.

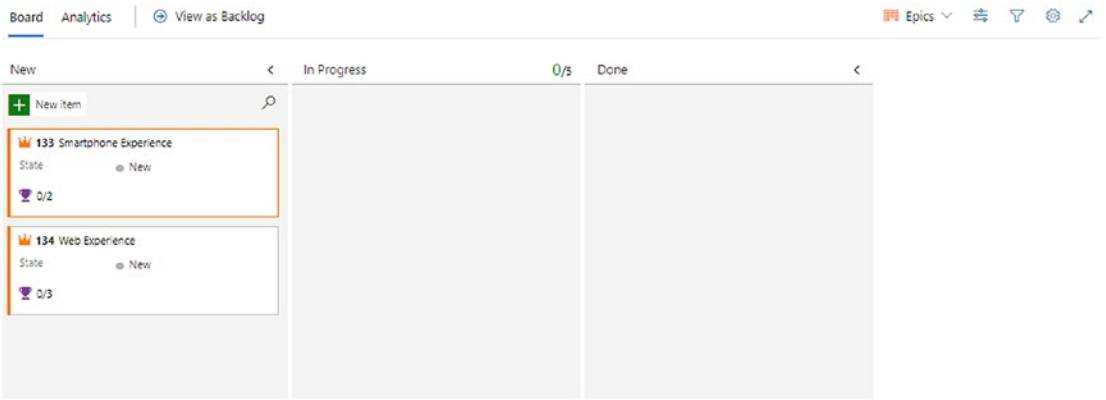


Figure 3-24. *Epics view*

The second icon ( ) shows an option to toggle the live update on and off. The third icon ( ) toggles the filter options on the top of the board. The last icon ( ) allows the user to navigate to full-screen mode.

Board Settings

Clicking the gear symbol brings up the team configuration, especially the board-level configurations. The Settings dialog supports various options to customize the cards and board.

Cards Configuration

The settings in this section help customize the card data and its look and feel.

- *Fields*: This section lists the fields displayed in the card. The user can configure the fields of importance here.
- *Styles*: Define a style rule to make the important content stand out. In Figure 3-25, the selected style defines the bugs with a different background color; this makes the bug stand out in the board.

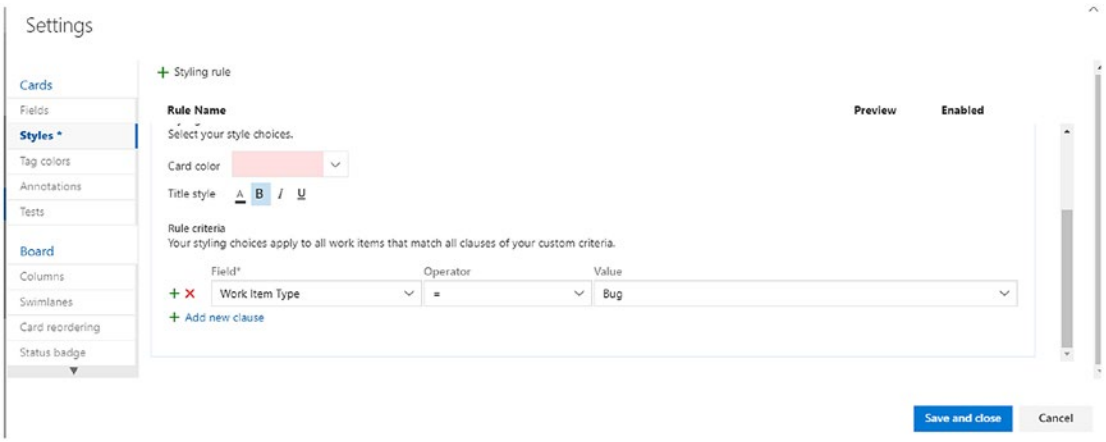


Figure 3-25. *Cards Style 1*

Once this style applied, the board changes its display of bugs, as shown in Figure 3-26.

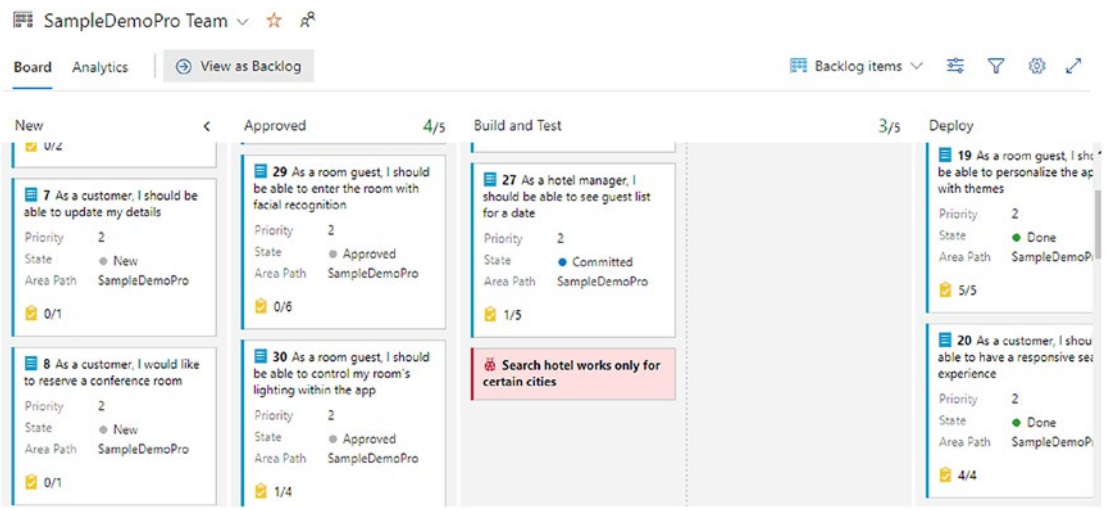


Figure 3-26. *Card style 2*

- *Tag Colors*: Tag colors support the configuration of different tag colors. Select a tag and configure the color against each tag to highlight some of the important tags in the board.

  Once the tag colors are applied, the cards will change the display of the tags, as shown in Figure 3-27.

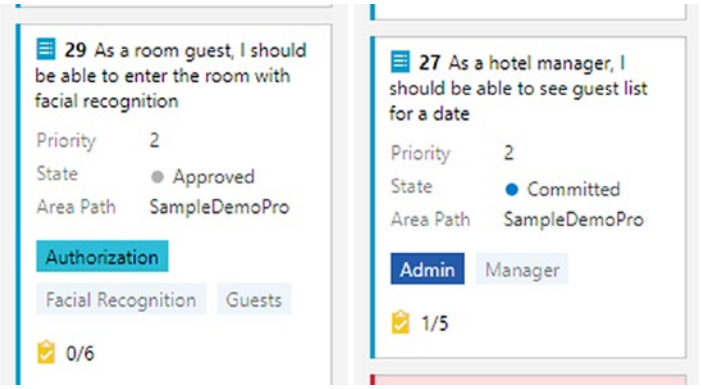


Figure 3-27. *Tag color*

- *Annotations*: Annotations show the number of tasks, tests, or GitHub linked to the current work item, as shown in Figure 3-28. By default, all these visual icons are selected and shown on the card.

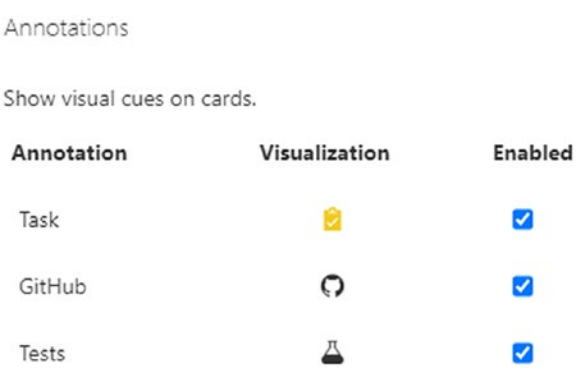


Figure 3-28. *Annotations 1*

In Figure 3-29, there are three tasks associated with the current work item and one test case. Out of the three tasks, one is completed, and other two are in progress.

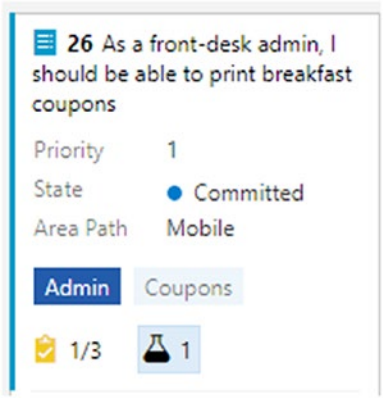


Figure 3-29. *Annotations 2*

- *Tests*: This configuration defines how the tests should be displayed on the card.

Board Configuration

The board configuration section defines the customization options available to configure the board. The following are the options available:

- *Columns*: This defines the columns available in the board. By default, there are four different columns: New, Approved, Build and Test, and Deploy.

Each column section shows the column name and status mapping for the bug and product backlog item. This setting is related to backlog items. If we open the settings related to an epic or feature, different options will be listed.

The Approved column has additional options to configure the work in progress limit, indicating the number of work items that should be displayed or shown as in progress. This section allows you to split the column into doing and done as well. The definition of done is another configuration available to define the criteria for qualifying a work item marked as Done. The Build and Test columns have similar options to configure the WIP, definition of done, and splitting of the column.

Moreover, this tab enables you to add a column to the board either using the plus sign on top or using the options on the header of the column names, as shown in Figure 3-30.

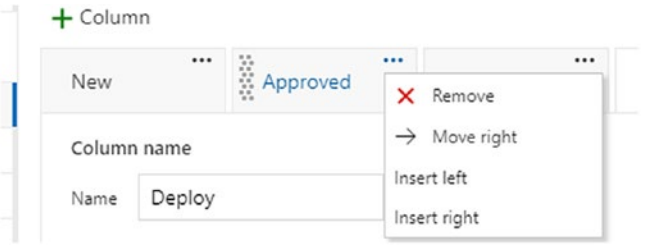


Figure 3-30. *Column movement*

- *Swimlanes*: Swimlanes allow you to group the work items horizontally. By default, there will be only one swimlane available. This setting allows you to add more swimlanes to group the work items horizontally on the board. In Figure 3-31, two new swimlanes are configured: Bugs and Product Backlog.

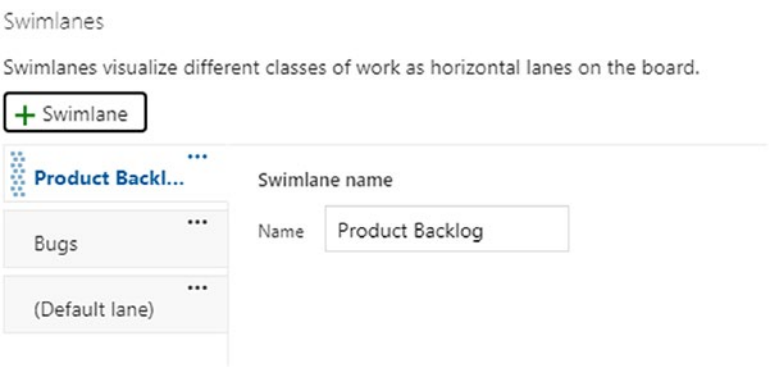


Figure 3-31. *Swimlane 1*

By default, newly created swimlanes will be blank. The user can drag and drop the work items to arrange the same horizontally using different swimlanes, as shown in Figure 3-32.

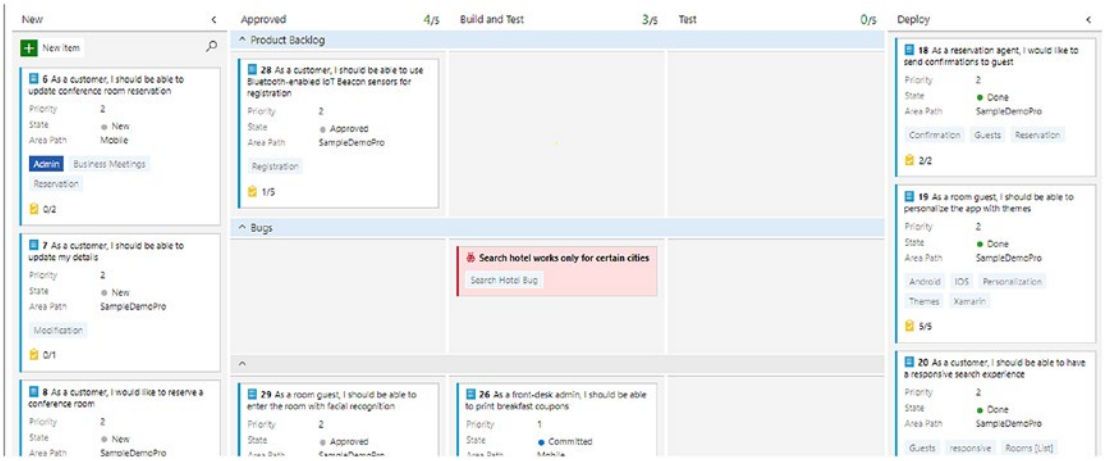


Figure 3-32. *Swimlane 2*

- *Card reordering*: This configuration defines the behavior of a card when dragging and dropping to a new column. There are two options available to order the card: either to place the new card in the drop location or to push it to the beginning and keep the order of the other cards as is, as shown in Figure 3-33.

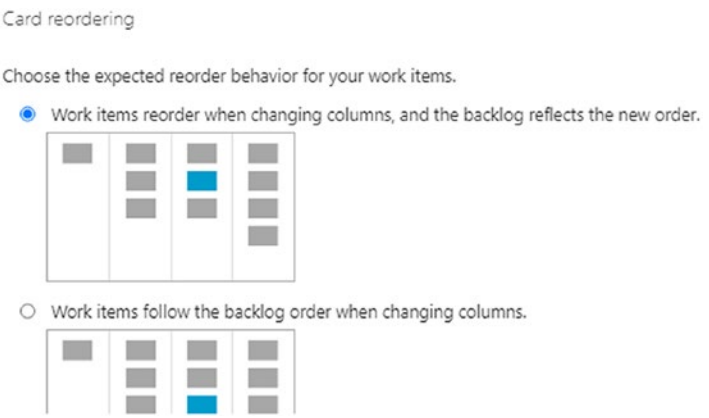


Figure 3-33. *Card reordering*

- *Status Badge*: Badges will be used to display the current status of the team on the dashboard. Badges can be configured using this section and copying the image URL, which can be inserted into any custom application to display the status information, as shown in Figure 3-34.

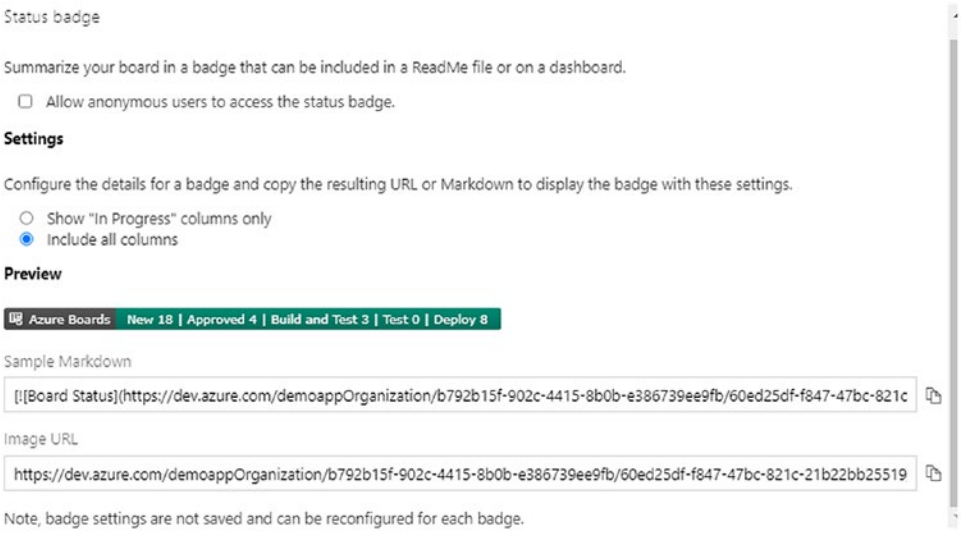


Figure 3-34. *Status badge*

General Configuration

This section contains the general settings related to the team board:

- *Backlogs*: The backlog configuration allows you to configure the backlog navigation levels available in the team board.
- *Working days:* This configuration allows you to configure the working days of the team. This information supports the capacity planning of the team.
- *Working with bugs*: This configures how the bugs will be handled by your team. Some teams consider bugs along with tasks and include them in sprint planning; others treat bugs differently and will not include them in the backlog and sprint planning area.

Cards

Each card in the board represents one work item. If the view selected is for backlog items, then the card shows backlog items associated with the current team. At the beginning of the card, there is an option to add a new item using this icon: 

. This option supports the creation of a new product backlog or bug. If the view changes, then this option allows you to create a new feature and epic.

Every card has the feature to drag and drop across the board, which will change the status of the item based on the drop location or column. Cards will have a badge indicating the type of work item and the title, as shown in Figure 3-35. The remaining fields are optional and can be configured based on the card settings.

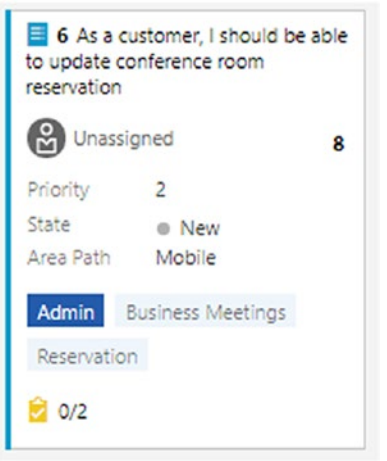


Figure 3-35. *Card*

The card settings provide the option to inline edit the content such as the assignment, effort, priority, state, and area path. When selected, each of these fields provides the list of valid values and allows a change in value, as shown in Figure 3-36.

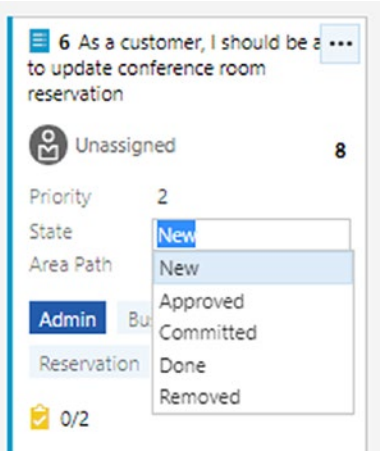


Figure 3-36. *State*

Each card has a top-right corner menu (…) to support additional operations, as shown in Figure 3-37.

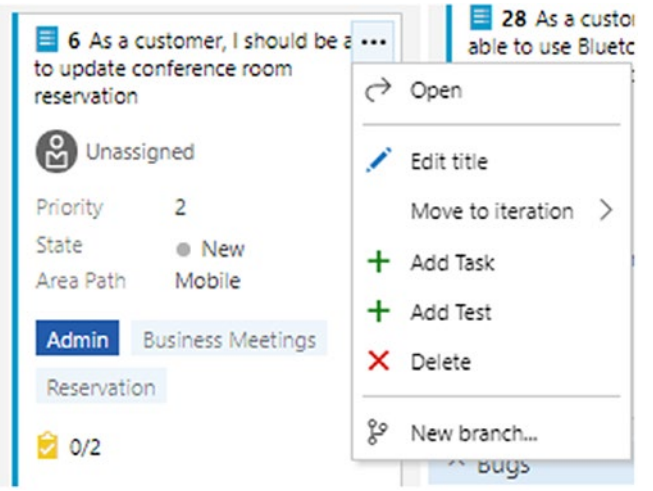


Figure 3-37. *Context menu*

The following are the additional options available in the Card menu:

- *Open*: This allows you to open the work item as a pop-up. A work item can be opened by clicking the title of the work item as well.
- *Edit Title:* This allows inline editing of the title.
- *Move to iteration*: Assign the work item to a specific iteration such as Sprint 2 or Sprint 3.
- *Add Task*: Add a subtask to implement the product backlog item fully or partially. One product backlog can have multiple tasks to implement it. The Add Task operation expands the card and provides an option to enter the title of the task.

  Once it's added, the user can navigate to the task by clicking the title and filling in the remaining details, if required.

  Also, this view allows you to drag to rearrange the order of the tasks. Newly added tasks can be moved up to complete them first or to set a higher priority for them.

Note If the user selects the task using the check box, this indicates that the task is completed and marked as done.

- *Add Test*: This is the same as adding tasks. The user can add test cases against a product backlog item to test the functionality.

  A test suite corresponding to the test cases can be opened using the right arrow next to "Add test." Figure 3-38 shows the test suite opened for some test cases. We'll discuss test suites in more detail later in the chapter.

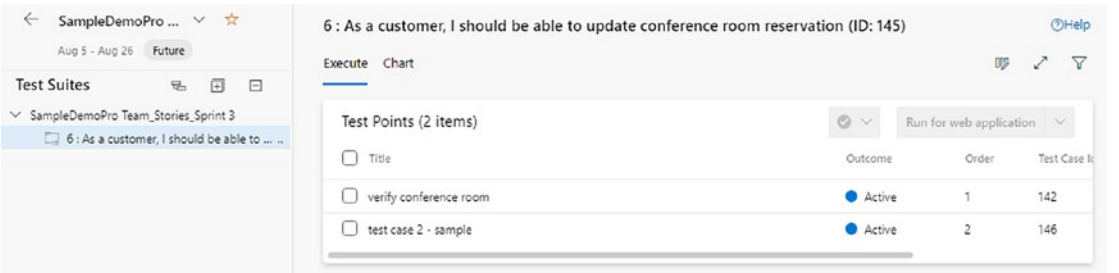


Figure 3-38. *Test suite*

- *Delete*: Delete the work item. This will move the work item into the Recycle Bin. The user can restore the work item, if required.
- *New Branch*: Create a new branch to implement the current work item and link to the work item.

Backlogs

The Backlogs section provides another view of work items and supports most of the options available in boards, as shown in Figure 3-39.

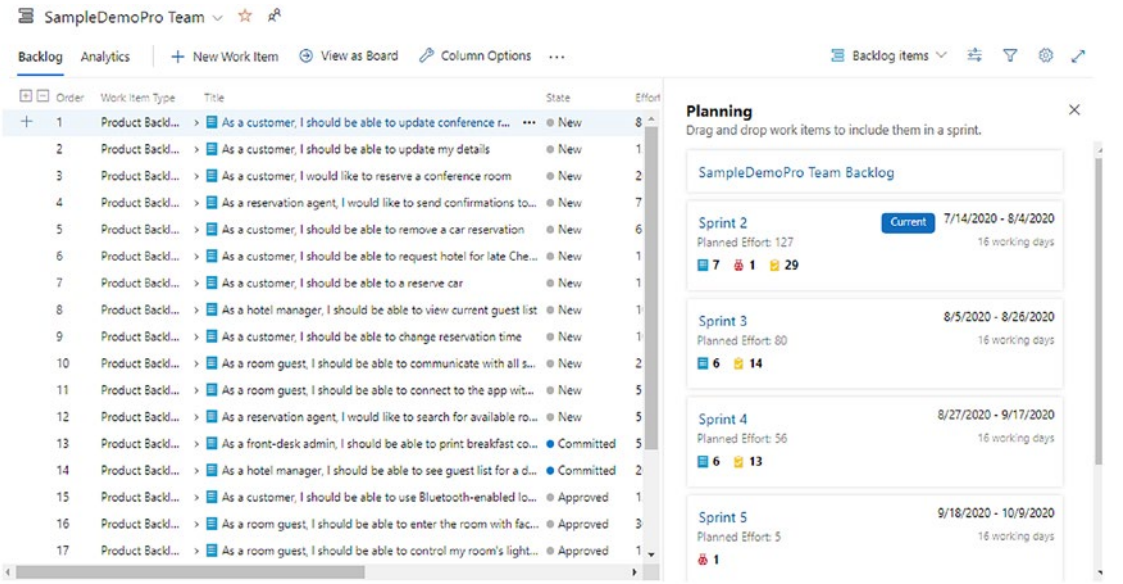


Figure 3-39. *Backlog Tab*

Instead of listing the work items as cards, they are displayed as a grouped list along with option to do iteration or sprint planning. Options such as the team backlog filter; favorite icon; team settings; filters; different views based on backlog, epics, and feature; filter options; analytics; and new work items are the same as boards. Additional options will be discussed in this section.

Backlog items can expand to view the tasks defined under each backlog. Use the + and – icons available on the header of the backlog list to expand and collapse all the items at once. For planning the activities for each of the sprints, backlog items can be dragged and dropped into the corresponding sprint displayed on the right, as shown in Figure 3-40.

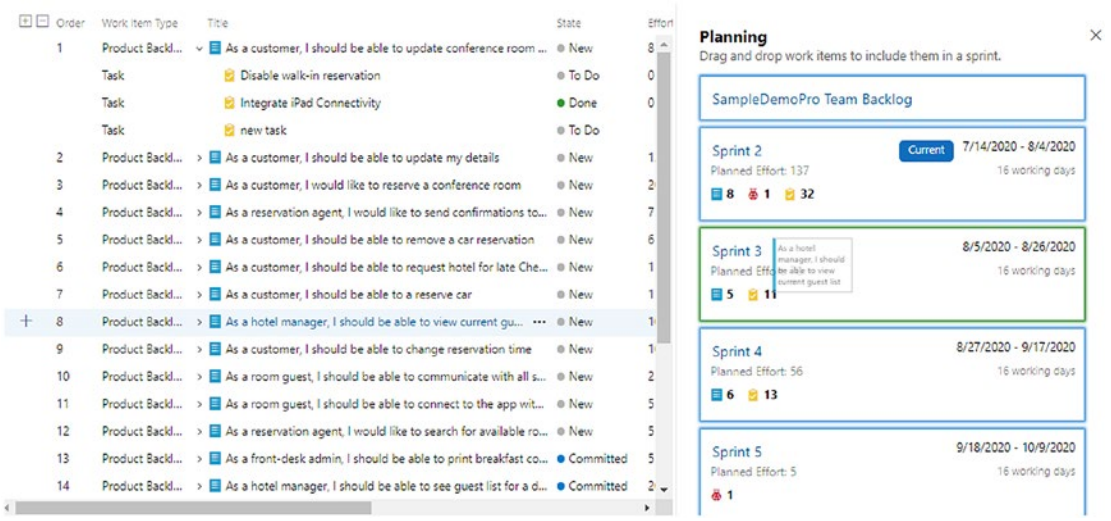


Figure 3-40. *Backlog planning*

Using the New Sprint option at the end of the Planning blade, you can add more iterations/sprints to the list. The Planning blade can be closed to view more fields of the work items.

Column Options

This option helps in configuring the columns required in the list of work items. There are two types of columns. One is related to the fields associated with the work item, and another is related to a rollup column. The rollup column is a calculated field based on the subtasks and linked items. In the case of a product backlog item, rollup columns consider the tasks and test cases linked to the backlog item.

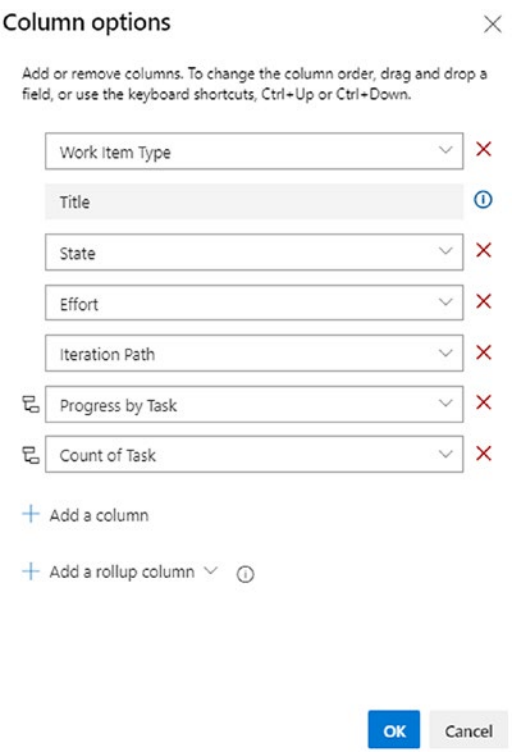


Figure 3-41. *Column options*

In Figure 3-41, there are two rollup columns configured: Progress by Task and Count of Task. Figure 3-42 shows the new view of the backlog.

Order	Work Item Type	Title	State	Effort	Iteration Path	Progress by Task	Count ...
1	Product Backl...	As a customer, I should be able to update conference r...	New	8	SampleDemoPro\Sprint 3	33%	3
2	Product Backl...	As a customer, I should be able to update my details	New	12	SampleDemoPro\Sprint 3	0%	1
3	Product Backl...	As a customer, I would like to reserve a conference room	New	20	SampleDemoPro\Sprint 3	0%	1
4	Product Backl...	As a reservation agent, I would like to send confirmations to...	New	7	SampleDemoPro\Sprint 3	0%	2
5	Product Backl...	As a customer, I should be able to remove a car reservation	New	6	SampleDemoPro\Sprint 4	0%	2
6	Product Backl...	As a customer, I should be able to request hotel for late Che...	New	15	SampleDemoPro\Sprint 4	0%	2
7	Product Backl...	As a customer, I should be able to a reserve car	New	15	SampleDemoPro\Sprint 4	0%	1
8	Product Backl...	As a hotel manager, I should be able to view current guest list	New	10	SampleDemoPro\Sprint 3	0%	3
9	Product Backl...	As a customer, I should be able to change reservation time	New	10	SampleDemoPro\Sprint 4	0%	1
10	Product Backl...	As a room guest, I should be able to communicate with all s...	New	23	SampleDemoPro\Sprint 3	0%	4
11	Product Backl...	As a room guest, I should be able to connect to the app wit...	New	5	SampleDemoPro\Sprint 4	0%	5
12	Product Backl...	As a reservation agent, I would like to search for available ro...	New	5	SampleDemoPro\Sprint 4	0%	2
13	Product Backl...	As a front-desk admin, I should be able to print breakfast co...	Committed	5	SampleDemoPro\Sprint 2	33%	3
14	Product Backl...	As a hotel manager, I should be able to see guest list for a d...	Committed	20	SampleDemoPro\Sprint 2	20%	5
15	Product Backl...	As a customer, I should be able to use Bluetooth-enabled lo...	Approved	12	SampleDemoPro\Sprint 2	20%	5
16	Product Backl...	As a room guest, I should be able to enter the room with fac...	Approved	30	SampleDemoPro\Sprint 2	0%	6
17	Product Backl...	As a room guest, I should be able to control my room's light...	Approved	15	SampleDemoPro\Sprint 2	25%	4

Figure 3-42. *Rollup columns*

Rollup fields provide a lot of information about the completeness of the task and the related work items.

The predefined rollup fields are as follows:

- *Progress by all Work Items*: Progress indicator by calculating the progress across all work items associated with the current backlog item
- *Progress by Bug*: Progress calculation only based on bug
- *Progress by Effort*: Progress based on effort
- *Progress by Task*: Progress by tasks completion
- *Count of Work items*: Count of associated work items
- *Count of Bug*: Count of bugs logged against the backlog
- *Count of Task*: Count of task
- *Sum of Effort*: Sum of the efforts
- *Sum of Remaining Work*: Sum of remaining work

In addition to these predefined rollup columns, Azure DevOps provides an option to define a custom rollup, as shown in Figure 3-43.

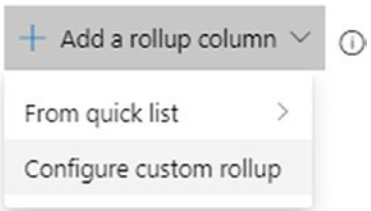


Figure 3-43. *Custom rollup 1*

Select "Configure custom rollup" to proceed. Select the rollup type from the options; you can select a progress bar or a total display, as shown in Figure 3-44. Also, select the work item type and column metrics (Count or Sum).

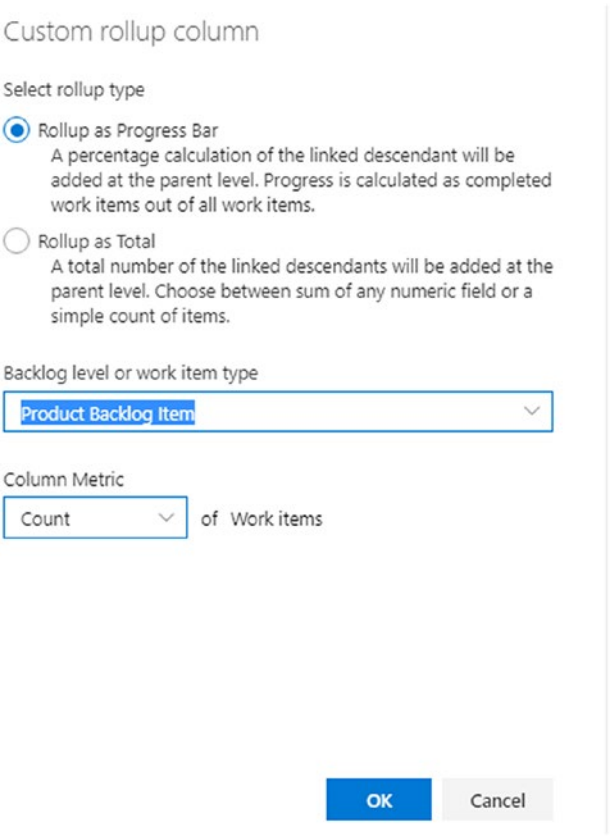


Figure 3-44. *Custom rollup 2*

Options

Multiple options are available in a backlog to handle the work items and maintain them. The More Commands (…) option next to the Columns provides the options Create Query and Email, as shown in Figure 3-45.

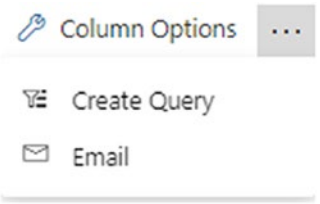


Figure 3-45. *Backlog options*

Create Query captures the filtered work items as a query for future usage. We'll discuss queries in more detail later in this chapter. The Email option will be used to share the listed work items over an email format.

The basic work item details along with a link to the work item will be captured in the email.

Another set of options is available for the work items listed in the view. Select the icon (…) next to the title of the work item to invoke other operations available, as shown in Figure 3-46.

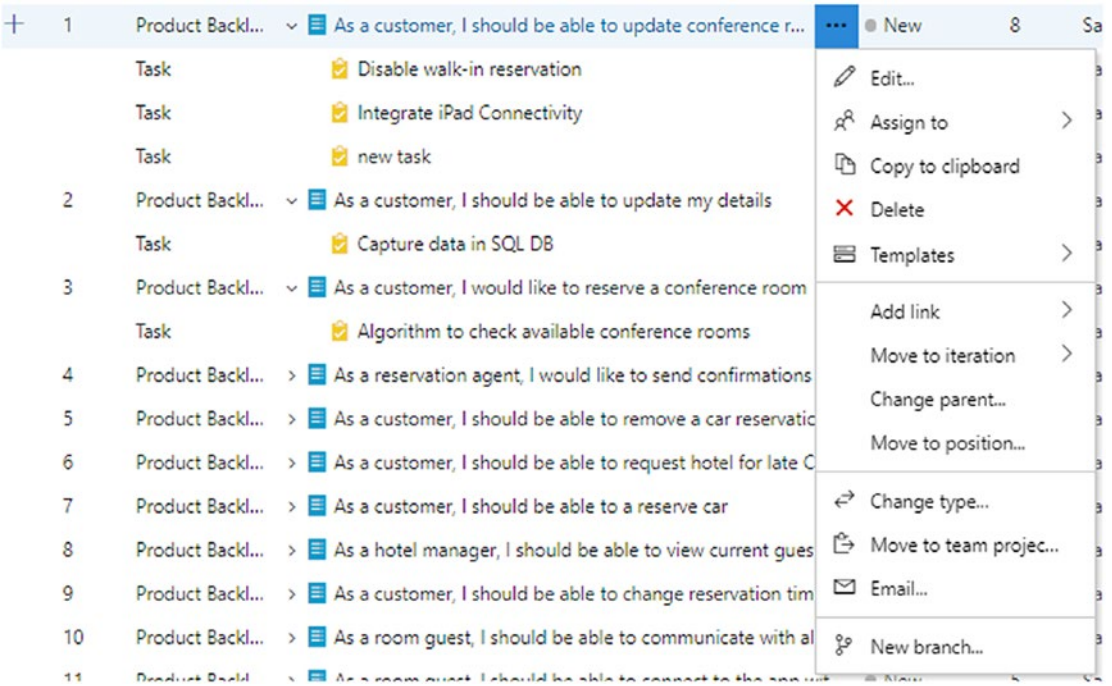


Figure 3-46. *Backlog options*

Most of these options were discussed in detail in previous sections. Backlog view supports bulk editing as well as applying an action to a set of work items at once.

- *Edit*: This allows you to edit the fields of the work item. The user can select multiple work items by using Shift and the arrow keys. After selecting multiple items, select the Edit option to edit a set of fields at once, as shown in Figure 3-47.

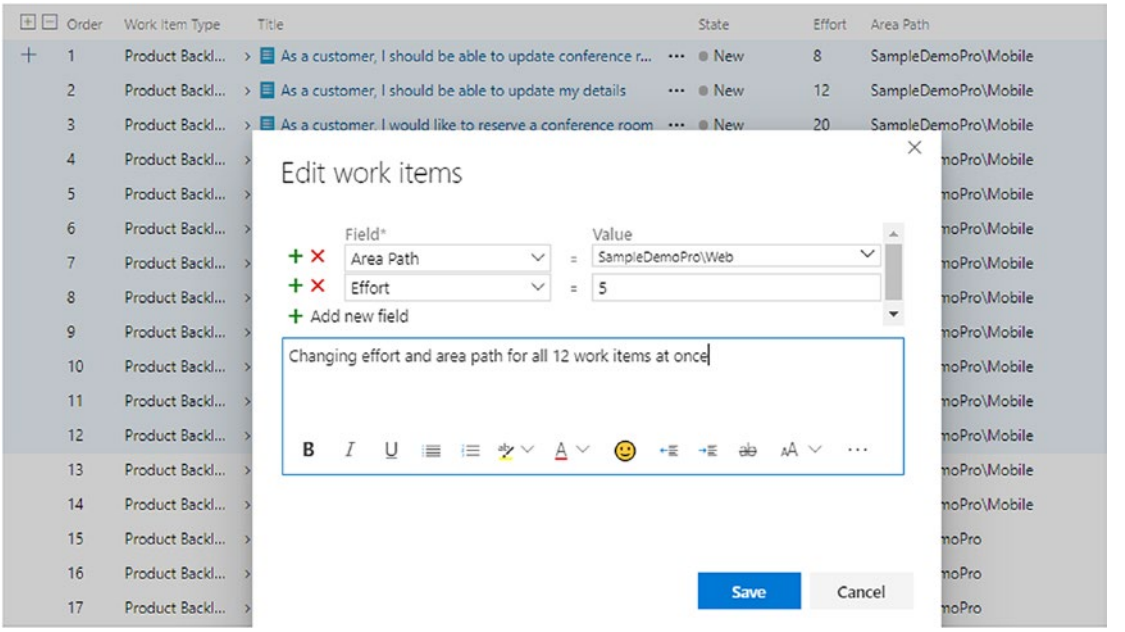


Figure 3-47. *Editing a work item*

In Figure 3-47, 12 items are selected with the area path set to `SampleDemoProj\Mobile` and different efforts. Change the area path and effort in the edit dialog and click Save. As shown in Figure 3-48, the effort and area path of the first 12 work items changed.

Order	Work Item Type	Title	State	Effort	Area Path
1	Product Backl...	As a customer, I should be able to update conference r...	New	5	SampleDemoPro\Web
2	Product Backl...	As a customer, I should be able to update my details	New	5	SampleDemoPro\Web
3	Product Backl...	As a customer, I would like to reserve a conference room	New	5	SampleDemoPro\Web
4	Product Backl...	As a reservation agent, I would like to send confirmatio...	New	5	SampleDemoPro\Web
5	Product Backl...	As a customer, I should be able to remove a car reserva...	New	5	SampleDemoPro\Web
6	Product Backl...	As a customer, I should be able to request hotel for late...	New	5	SampleDemoPro\Web
7	Product Backl...	As a customer, I should be able to a reserve car	New	5	SampleDemoPro\Web
8	Product Backl...	As a hotel manager, I should be able to view current gu...	New	5	SampleDemoPro\Web
9	Product Backl...	As a customer, I should be able to change reservation ti...	New	5	SampleDemoPro\Web
10	Product Backl...	As a room guest, I should be able to communicate with ...	New	5	SampleDemoPro\Web
11	Product Backl...	As a room guest, I should be able to connect to the app...	New	5	SampleDemoPro\Web
12	Product Backl...	As a reservation agent, I would like to search for availab...	New	5	SampleDemoPro\Web
13	Product Backl...	As a front-desk admin, I should be able to print breakfast co...	Committed	5	SampleDemoPro\Mobile
14	Product Backl...	As a hotel manager, I should be able to see guest list for a d...	Committed	20	SampleDemoPro\Mobile

Figure 3-48. *Bulk edit result*

- *Assign to*: Just like the Edit option, this option supports the bulk assignment of selected work items.
- *Copy to clipboard*: This selects a set of work items and copies the listed field details into a file.
- *Delete*: This deletes the selected work items.
- *Templates*: This captures the template out of a work item, as explained in the previous section.
- *Add link*: This adds a link to an existing or new work item. This operation supports the linking of multiple work items with a single work item. But the linking may fail based on the relationship or link type selected for the work item link. For example, in Figure 3-49 there are 12 work items selected and trying to link to an existing work item with the link type as Child.

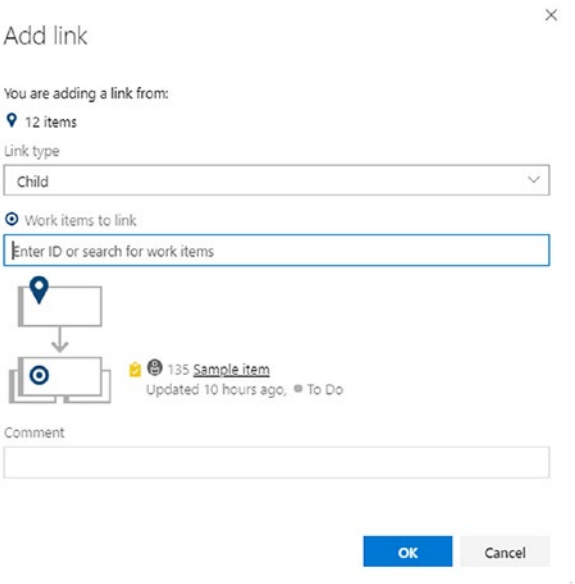
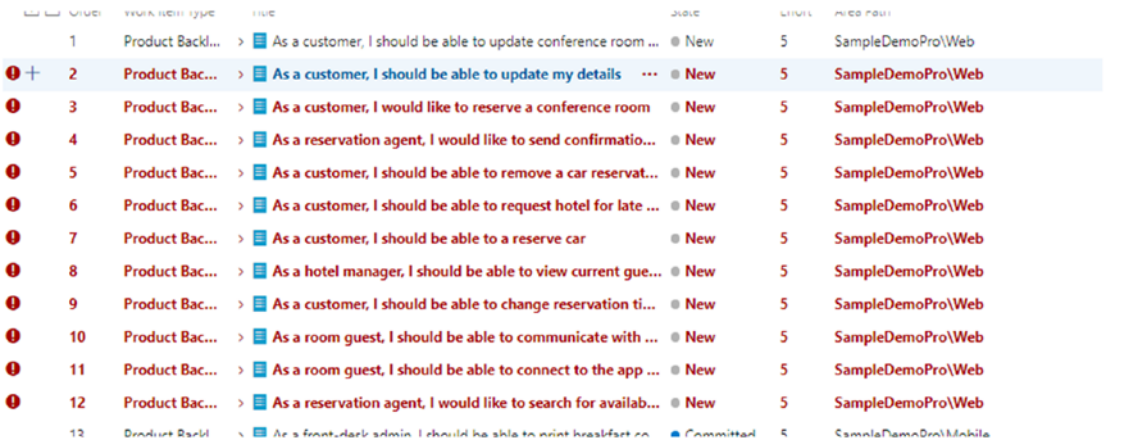


Figure 3-49. *Adding a link*

As the link type is Child, only one work item will be able to link with the selected work item. The remaining links will fail, as shown in Figure 3-50.



	Order	Work Item Type	Title	State	Effort	Area Path
	1	Product Backl...	As a customer, I should be able to update conference room ...	New	5	SampleDemoPro\Web
	2	Product Bac...	As a customer, I should be able to update my details	New	5	SampleDemoPro\Web
	3	Product Bac...	As a customer, I would like to reserve a conference room	New	5	SampleDemoPro\Web
	4	Product Bac...	As a reservation agent, I would like to send confirmatio...	New	5	SampleDemoPro\Web
	5	Product Bac...	As a customer, I should be able to remove a car reservat...	New	5	SampleDemoPro\Web
	6	Product Bac...	As a customer, I should be able to request hotel for late ...	New	5	SampleDemoPro\Web
	7	Product Bac...	As a customer, I should be able to a reserve car	New	5	SampleDemoPro\Web
	8	Product Bac...	As a hotel manager, I should be able to view current gue...	New	5	SampleDemoPro\Web
	9	Product Bac...	As a customer, I should be able to change reservation ti...	New	5	SampleDemoPro\Web
	10	Product Bac...	As a room guest, I should be able to communicate with ...	New	5	SampleDemoPro\Web
	11	Product Bac...	As a room guest, I should be able to connect to the app ...	New	5	SampleDemoPro\Web
	12	Product Bac...	As a reservation agent, I would like to search for availab...	New	5	SampleDemoPro\Web
	13	Product Backl...	As a front-desk admin, I should be able to print breakfast co...	Committed	5	SampleDemoPro\Mobile

Figure 3-50. *Adding a link result*

- *Move to iteration*: This maps the selected work items to an iteration or sprint.
- *Change parent*: This changes the parent of the work item to a different existing work item.
- *Move to position*: This option helps in reprioritizing the work items in the Backlog view. This can be done by dragging the work items to different positions as well.
- *Change type*: This changes the type of the work item, as explained in the previous section.
- *Move to team project*: This moves the work item to another team project.
- *Email*: This emails the work item to a group or stakeholder to review the content.
- *New Branch*: This creates a new branch for implementing this work item.

Sprints

The Sprints section provides a view of the work items planned for an iteration or sprint, duration of the sprint, progress of work completion, team working on the sprint, and other sprint-related information, as shown in Figure 3-51. Using boards and backlogs,

the teams decides on the work items for a sprint. Once the work items are finalized for a sprint, the team can start implementing them based on the details available and not entertain any change in plan.

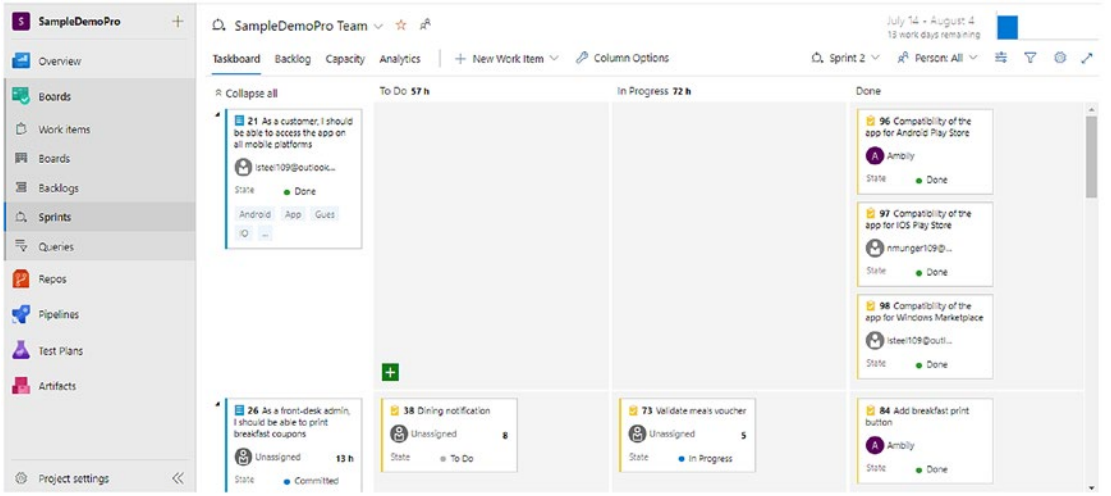


Figure 3-51. *Sprints*

The sprint view provides various options for executing the sprint activities. This view will be used as part of daily standup meetings to track the progress. The top-left corner has options that allow navigation to the sprint directory and other teams' sprint views, as shown in Figure 3-52. The next two icons show the favorite and team profile, the same as the Boards and Backlogs sections.

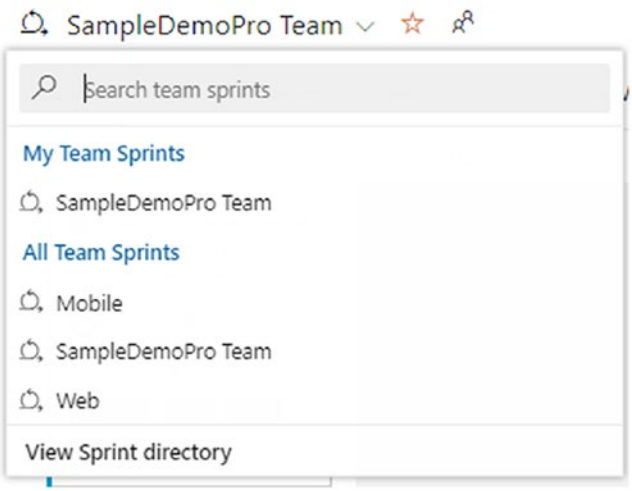


Figure 3-52. *Sprints options*

The top-right corner shows the sprint start and end dates along with the burndown trend, as shown in Figure 3-53. Sprint start and end dates can be edited, but this is not recommended generally. Changes to sprint cadence or work items are not recommended after a sprint starts.

Figure 3-53. *Sprints: Top options*

The burndown trend shows the progress of the sprint activities. Over time, the activities should be completed, and the trend line should go down. If the trend line does not show a steady decline, then there will be a chance of not completing the activities on time. We'll discuss different sprint dashboards later in this book. By selecting the Burndown Trend icon, it opens up the Burndown Trend report, as shown in Figure 3-54.

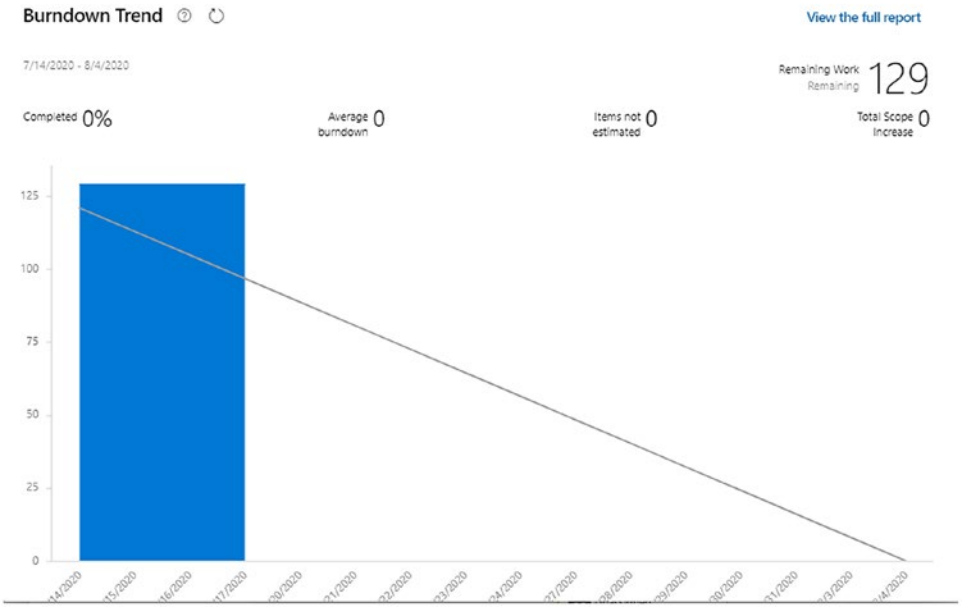


Figure 3-54. *Burndown Trend report 1*

You can drill down into this report further, and other details are available on the Analytics tab or by selecting the "View the full report" link at the top of this report, as shown in Figure 3-55.

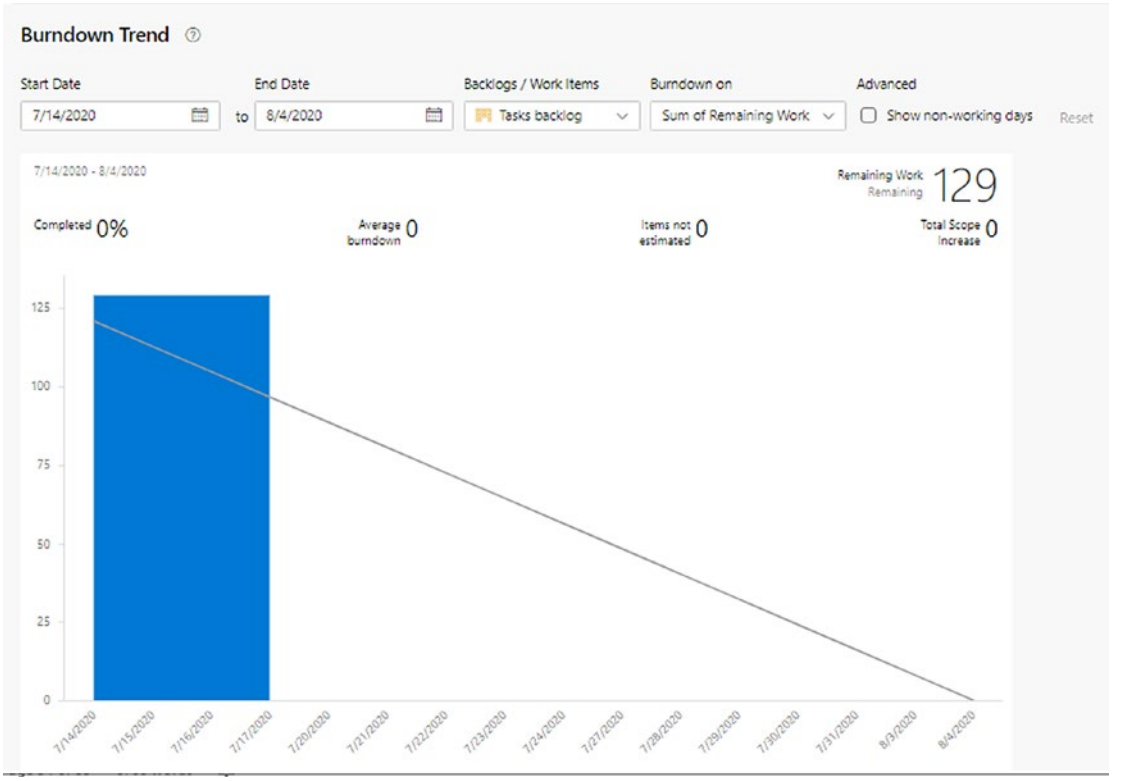


Figure 3-55. *Burndown Trend report 2*

The line shows the ideal trend, whereas the column shows the actual progress. Various filtering and search options are available on top of the report to extract the required information from this report.

Figure 3-56 shows the next set of options available just below the team name.

Figure 3-56. *More sprint options*

- *Taskboard*: This shows the sprint work items in the board, just like the Boards section. Work item cards provide similar options like the cards in the Boards section, which allows you to change the look and feel of the card and do limited operation on the work items.
- *Backlog*: This is a similar view as the Backlog section, but limited with work items mapped for the selected sprint.

- *Capacity*: This section helps in configuring the team capacity. Based on team availability, the total effort available for a sprint will be calculated. The team can enter the day off and partial availability data, which helps the sprint planning.

  Team can enter the day off using the link provided in the Capacity tab.

  Team can also specify the kind of work or activity one can perform, as shown in Figure 3-57.

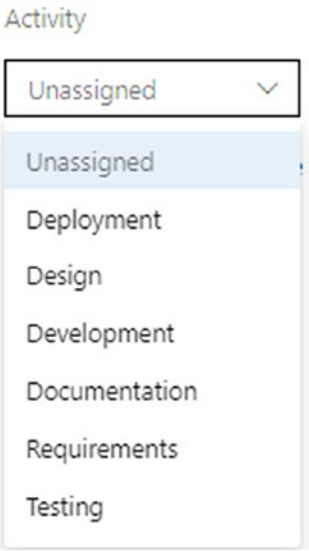


Figure 3-57. *Capacity*

Moreover, this tab helps in adding multiple activities for one team member and distributes that user's availability or effort across multiple activities, as shown in Figure 3-58. The scrum master can use all these details to plan the sprint.

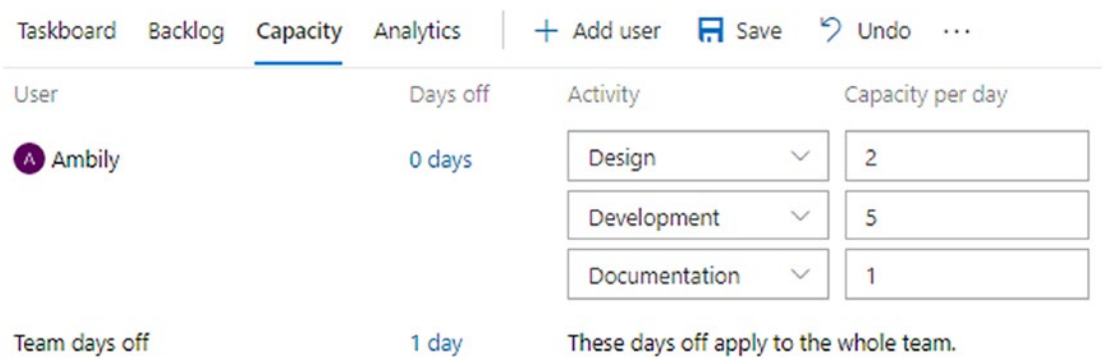


Figure 3-58. *Sprints: Capacity 2*

- *Analytics*: This shows the Burndown Trend report discussed in previous section.
- *New Work Item:* This shows another option to add new work items to the list.
- *Column Options*: This shows the different column options for the board. The user can add or remove columns based on the sprint requirements using this option.
- *Sprint*: Select different sprints to view the work items planned for each sprint or the items completed in previous sprints.
- *Person All*: You can filter the view with the assigned team member, as shown in Figure 3-59. This view will expand and shows only the work items assigned to the selected team member; the remaining work items will be in collapsed mode.

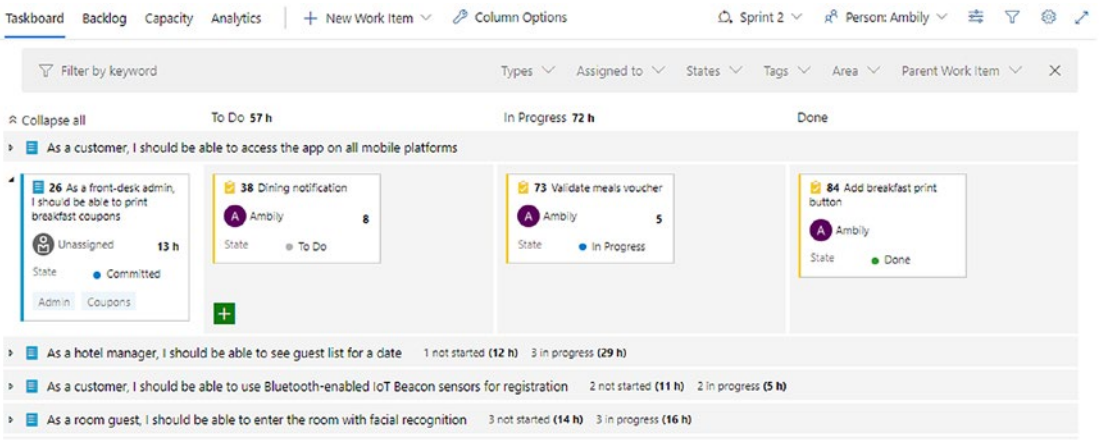


Figure 3-59. *Person view*

- *View Options*: By default, the taskboard displays work items grouped under backlog items. You can change this view to group them under the assigned team member using the People option. This will be helpful in driving the daily standup call and tracking the progress of individuals.

  Also, the user can select the side pane to display work details or planning. "Work details" shows the workgroup by activity and team member and shows the pending work or effort required to complete the work, as shown in Figure 3-60.

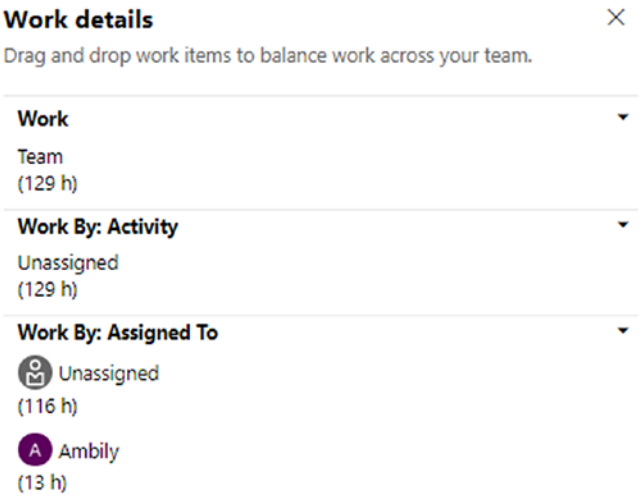


Figure 3-60. *Work details*

Planning displays the iteration or sprints with a number of work items planned, planned efforts, start and end dates, and other details, as shown in Figure 3-61.

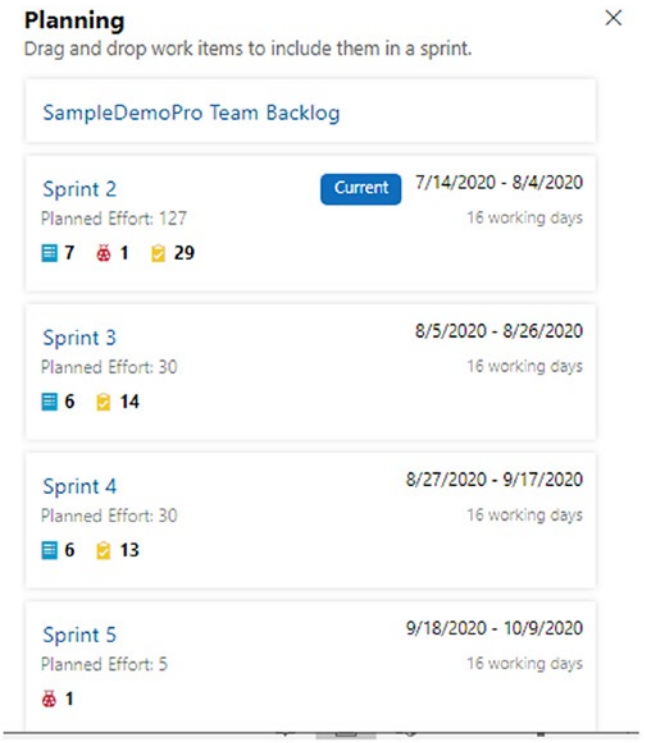


Figure 3-61. *Sprints: Planning*

- *Filter*: This toggles the filter options available for the board.
- *Settings*: This is a subset of settings in the Boards section to decide on the card data and general settings.
- *Full Screen:* This displays the section in full-screen mode, which provides additional screen space for users to work on the work items.

Queries

The Queries section supports the filtering of required work items based on various criteria. The initial view lists the saved queries under My Queries and Shared Queries, where My Queries will be available to only the user, and the Shared Queries will be shared across the team.

For better management, the user can create subfolders under My Queries and Shared Queries and move the appropriate queries into those folders. Use the option New folder on the top of the query list to create a new folder under any of the existing folders. The user can either define new queries or drag an existing query to the folder. In Figure 3-62, two existing queries were dragged to the new folder called Bug Related under Shared Queries.

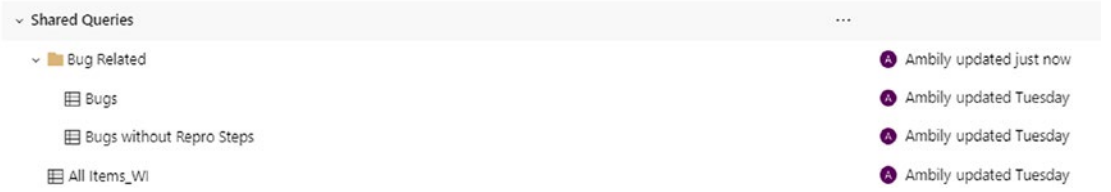


Figure 3-62. *Folders*

Import Work Items provides the option to import the CSV file, explained in the previous section. Filtering by keywords allows you to filter the query based on specific filter conditions such as a bug, as shown in Figure 3-63.

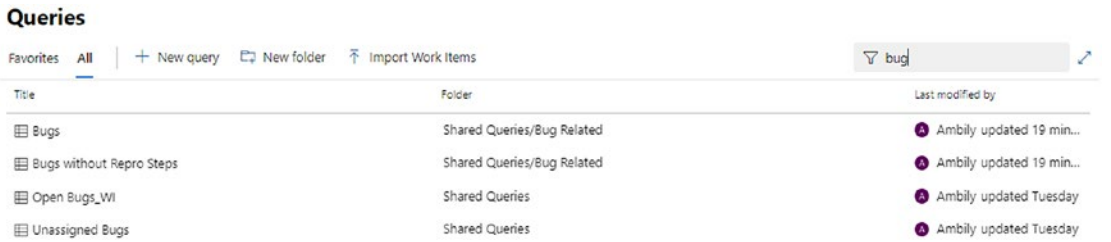


Figure 3-63. *Filtering queries*

For each of the queries, there is an option to add the query to a favorite list by selecting the star icon. Moreover, there are multiple options available as a context menu for the folders and queries.

- *Run Query*: This runs the query and shows the results page, as shown in Figure 3-64.

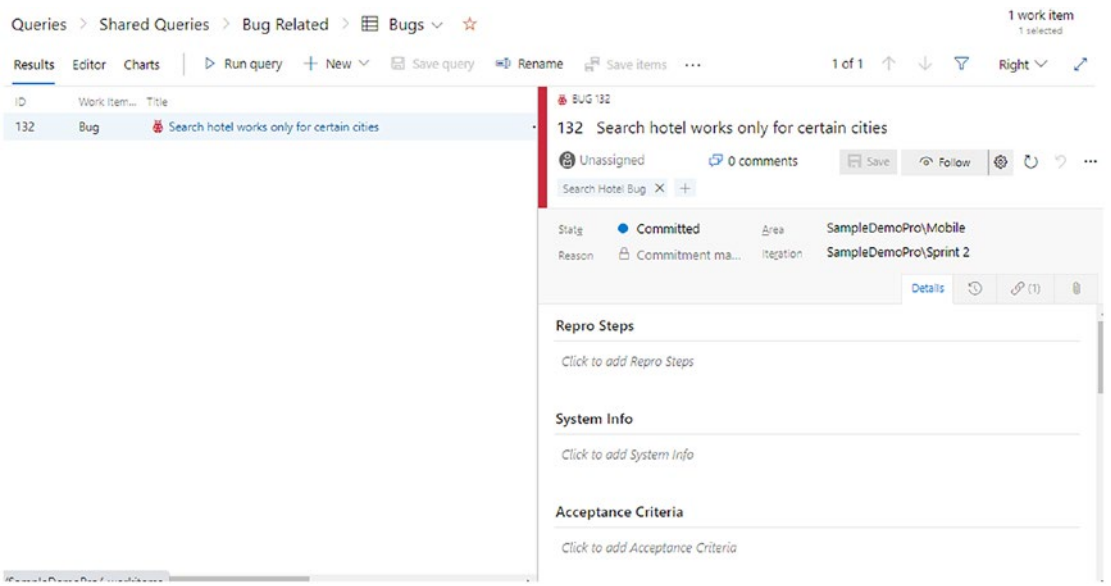


Figure 3-64. *Running a query*

- *Edit*: This allows you to edit an existing query.
- *Rename:* This allows you to rename an existing query.
- *Delete*: This deletes the query.
- *Add to Team Favorites*: This allows you to add the query to a team's favorite list.
- *Security*: This defines the security and permission for the queries. Select the user or group and allow permissions such as Contribute, Delete, Permission management, and Read. The user can set permissions such as Allow, Deny, and Not set.
- *Add to dashboard*: Add the query result in different forms to the team's dashboard.

New Query

New Query will open the page with a default query condition, as shown in Figure 3-65, which lists all the work items with all states. Select the Run Query option to execute the default query, which displays the query result below the query itself.

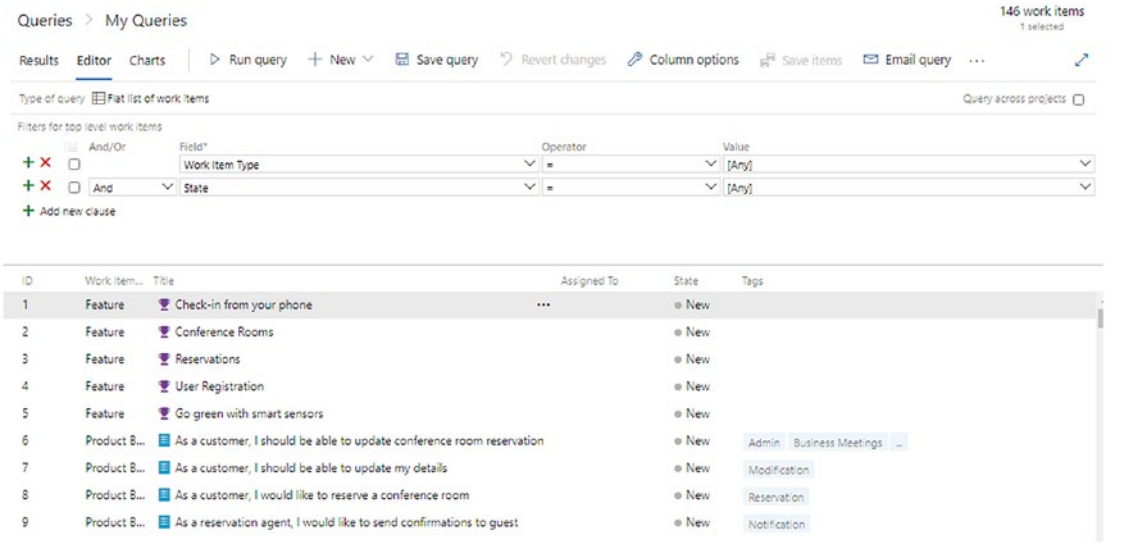


Figure 3-65. *Default query conditions*

By default, only a few columns are listed in the query result view. The user can add more columns using the Column options. The "Save query" option will be used to save the query into different folders.

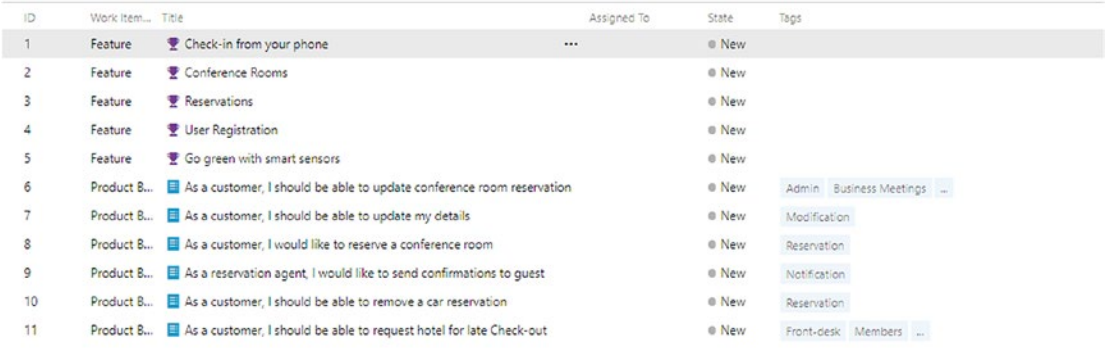
Once the query is saved, new options to save the query with a different name (Save as) and Rename will appear on the top option list. The following are some of the other options available on top of the query editor:

- *The Results tab*: This shows the query result on a separate tab.
- *The New option*: This helps create new queries or work items.
- *Revert Changes*: This will be enabled to revert changes that are not yet saved.
- *Email query:* Email the current query result.
- *Copy query URL*: This copies the current query URL, which will allow the team members to share it to other stakeholders to refer to a specific list of queries.
- *Export to CSV*: This exports the result into CSV format for offline reference. After updates, this can be imported back to the system as well.

Type of Queries

There are three types of queries available.

- *Flat list of work items (Default)*: This is a list of work items, as shown in Figure 3-66.
- *Work items and direct links*: These are work items with one level of child elements.
- *Tree of work items*: These are work items with a multilevel view.



ID	Work Item...	Title	Assigned To	State	Tags
1	Feature	Check-in from your phone	...	New	
2	Feature	Conference Rooms		New	
3	Feature	Reservations		New	
4	Feature	User Registration		New	
5	Feature	Go green with smart sensors		New	
6	Product B...	As a customer, I should be able to update conference room reservation		New	Admin Business Meetings ...
7	Product B...	As a customer, I should be able to update my details		New	Modification
8	Product B...	As a customer, I would like to reserve a conference room		New	Reservation
9	Product B...	As a reservation agent, I would like to send confirmations to guest		New	Notification
10	Product B...	As a customer, I should be able to remove a car reservation		New	Reservation
11	Product B...	As a customer, I should be able to request hotel for late Check-out		New	Front-desk Members ...

Figure 3-66. *Flat list of work items*

The "Work items and direct links" item shows the work items with direct links based on the filter options selected and the type of links selection, as shown in Figure 3-67.

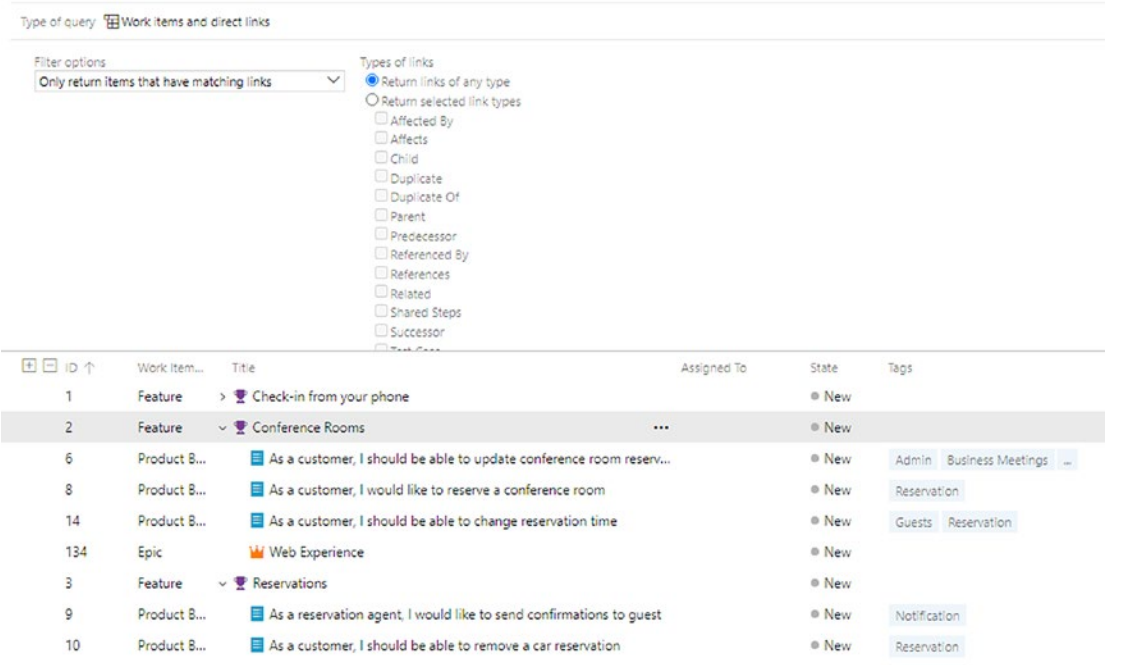


Figure 3-67. *Work items and direct links*

Different filter options are available, as shown in Figure 3-68.

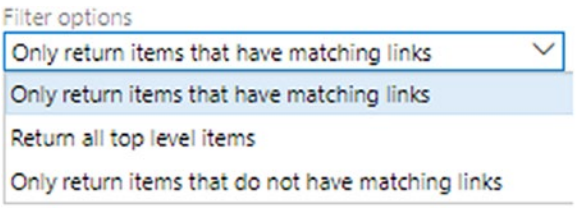


Figure 3-68. *Filter options*

"Tree of work items" is also controlled by different filter options and types of trees. There are two filter options available: Match top-level work items first and Match linked work items first. There are two values for the type of tree as well: Duplicate Of/Duplicate and Parent/Child, as shown in Figure 3-69.

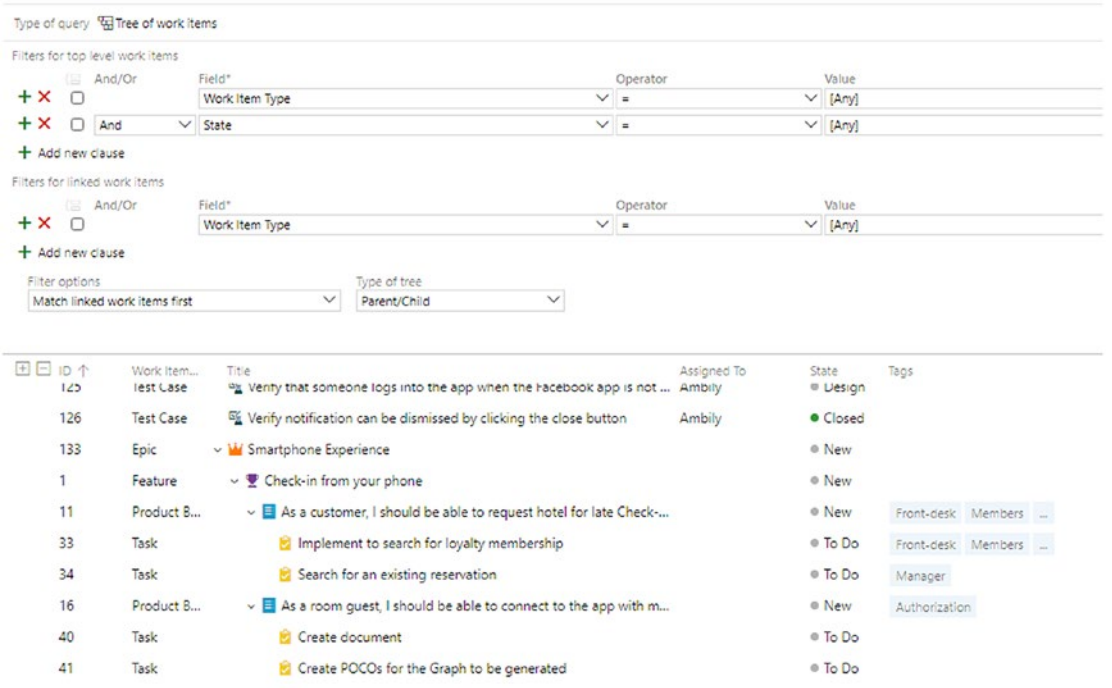


Figure 3-69. *Tree of work items*

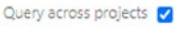
Moreover, in both cases, there are filters corresponding to top-level work items and linked work items. Filters define the filter conditions or the requirements of the work items. New filters can be added using the plus sign available at the beginning of the filter row and deleted using the cross icon next to it. The third icon, the check box, represents the grouping of the filters based on a logical operation like AND or OR.

For example, Figure 3-70 shows the filer to view only the tasks and product backlogs with any state. Select the filters using the check box and click the group icon on top to group them.

Figure 3-70. *Filters*

The next three columns show the field, operator, and value combination to define the core filter condition.

The user can query across all projects in the organization to view the work status across the organization. Select the "Query across projects" option to query the work items across the organization.



Work Offline

The Export to CSV option allows you to download the result of the query into CSV format.

The downloaded data will have the columns selected as part of the query result only. If you need more columns, select the column before downloading to include it in the result. If the result contains multiple Title fields, like the one shown in Figure 3-70, then the query will be either work items with a direct link or a tree of work items. In Figure 3-70, Title 1 shows the product backlog item's title, whereas Title 2 shows the title of the task, the child of product backlog item.

To add or delete the work items offline to Azure DevOps, install and configure the Azure DevOps Office Integration from `https://visualstudio.microsoft.com/downloads/`.

Now, open an Excel file and navigate to the Team ➤ New List option.

Add or select the DevOps server to continue. Once the server is selected, all the organizations and team projects are listed, as shown in Figure 3-71.

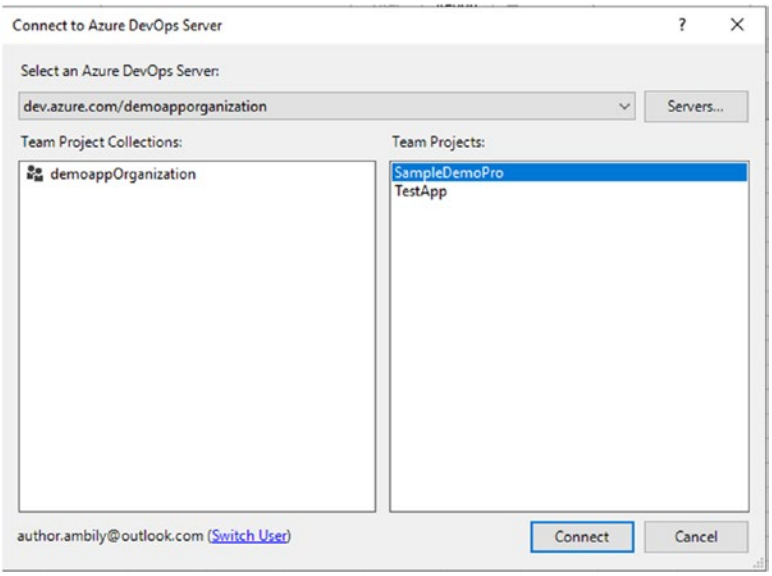


Figure 3-71. *Connecting to the Azure DevOps server*

Select the team project, connect to the project, and get the list of queries available, as shown in Figure 3-72.

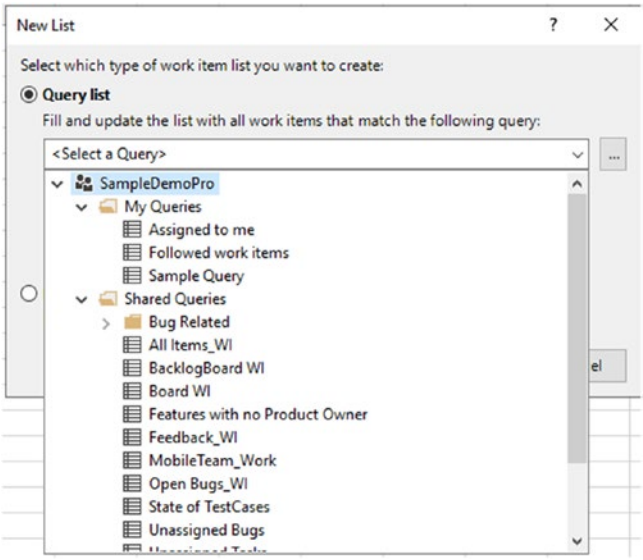


Figure 3-72. *Selecting a query*

Select the query and proceed to load the query result into the Excel sheet, as shown in Figure 3-73.

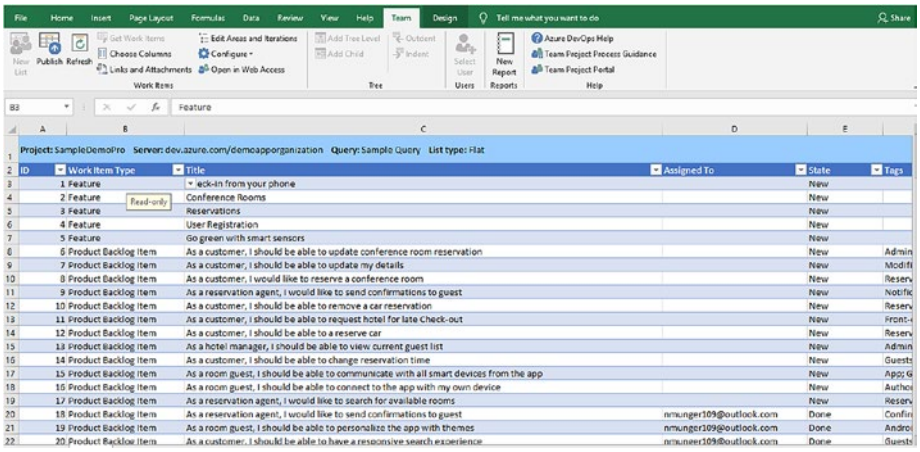


Figure 3-73. *Offline view of work items*

Add new work items without an ID field or edit existing work item details to work offline on the work items, as shown in Figure 3-74. Once the edits complete, click Publish to publish the changes to Azure DevOps.

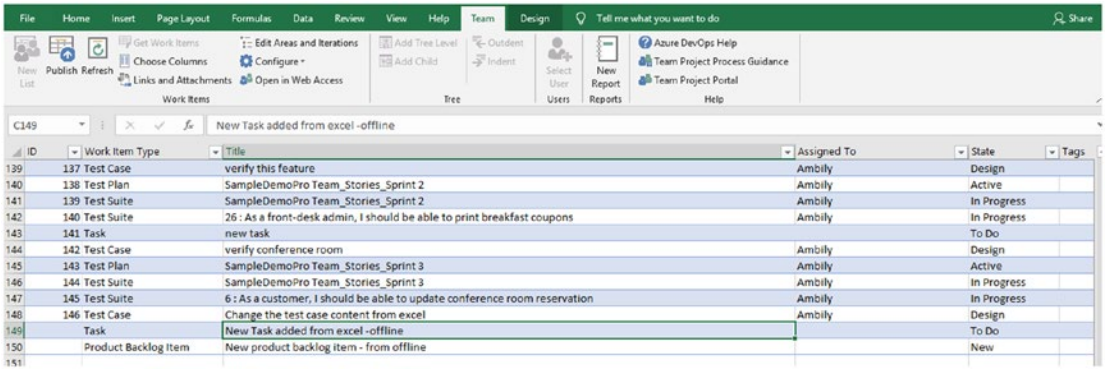


Figure 3-74. *Offline editing of work items*

Once published, newly added IDs will be populated in the Excel spreadsheet, as shown in Figure 3-75. Excel provides a few additional options to manage the work items such as the ability to select columns, edit areas and iterations, and so on.

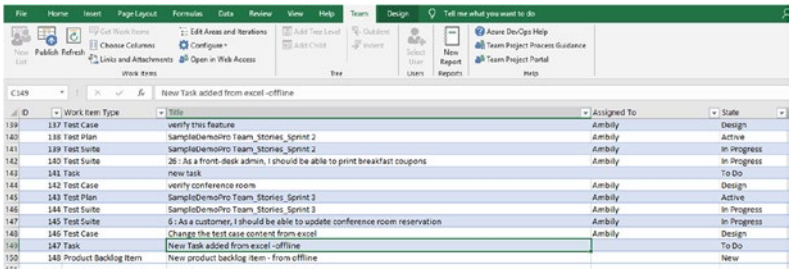


Figure 3-75. *Publishing work items*

Summary

The Azure DevOps features support the end-to-end management of requirements from all aspects. Business requirements can be captured using various epics and features, whereas the low-level implementation tasks will be captured using product backlog items and tasks. The requirements are not limited to functional and architectural requirements; Azure DevOps also covers the NFRs such as scalability testing, security testing, performance testing, and so on. The system supports the traceability of the requirements in different dimensions such as the code developed based on the requirement, test cases that are executed to verify the requirements, defects identified in the implementation, the build or release containing the requirements, and so on.

CHAPTER 4

Version Control Using Azure DevOps

The versioning of code is important in a DevOps implementation. Version control systems enable you to collaborate with team members by tracking and controlling changes within your code. Most version control systems provide the basic features required for working with team members in parallel on the same codebase or file. These features include versioning code, rolling back to previous versions, comparing versions to identify the changes, and so on.

This chapter focuses on version control systems, branching and merging strategies, and the associated functionalities.

Repos

Repositories define the source code management component of Azure DevOps. You can configure multiple repositories in a team project to manage the different source code related to a project. For example, a team can plan for a repo for the UI application and have another repo for API development.

Once created, this new repo will be empty. Initializing a README file is an optional step but industry best practice, as shown in Figure 4-1. The README file helps the reader to understand the usage of the project, language, and steps required for project setup.

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_4

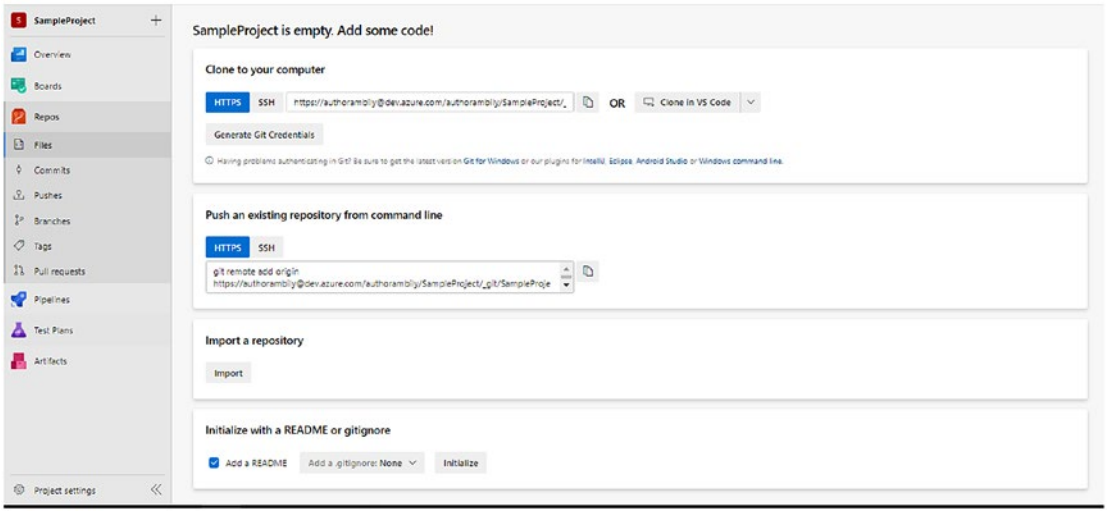


Figure 4-1. *Initial repo setup*

The team can set up the repo by importing an existing repo from another repository directly or by pushing an existing project from a system using the command line.

The following are the steps used to push the existing codebase from a local system:

1. Open a command prompt (Start ➤ cmd).
2. Navigate to the folder containing the source code of the project.

   Git should be installed before cloning or pushing the Git repository. Download Git from `https://git-scm.com/download/win` and install it to proceed.
3. Initialize the Git repository using this:

   ```
   git init
   ```
4. Once the repository is initialized, link the local Git setup to the Azure "remote" repo so they can communicate, by adding `origin` to the `git remote` command.

   ```
   git remote add origin https://authorambily@dev.azure.com/
   authorambily/SampleProject/_git/SampleProject
   ```

5. Once the repo is linked with Azure DevOps, push the existing code from the local version to the remote version using the `git push` command. You will see the code in the Azure repository called SampleProject.

   ```
   git push -u origin –all
   ```

   This may prompt you to provide your Azure DevOps credentials to continue.

6. If you get the following error message, set up the branch corresponding to our repo.

   No refs in common and none specified; doing nothing.
   Perhaps you should specify a branch such as 'master'.
   Everything up-to-date

   To fix the previous error, clone the Azure DevOps repository again using the `git clone` command.

   ```
   git clone https://authorambily@dev.azure.com/authorambily/
   SampleProject/_git/SampleProject
   ```

7. Navigate to our project folder using `cd SampleProject`.

8. Set the username and password, if they're not detected automatically, using the Git `global` settings commands.

   ```
   git config --global user.email sss.ambily@outlook.com
   git config --global user,name "Ambily"
   ```

9. Add the existing content to the local Git repository using this:

   ```
   git add .
   ```

10. Once the content is added, commit the changes to the local repository using this command:

    ```
    git commit -m "first file update"
    ```

11. Once the local codebase is ready to be merged with the centralized repository, push the code using the `git push` command.

    ```
    git push
    ```

Observe the merged codebase in Azure DevOps along with the change commands, as shown in Figure 4-2.

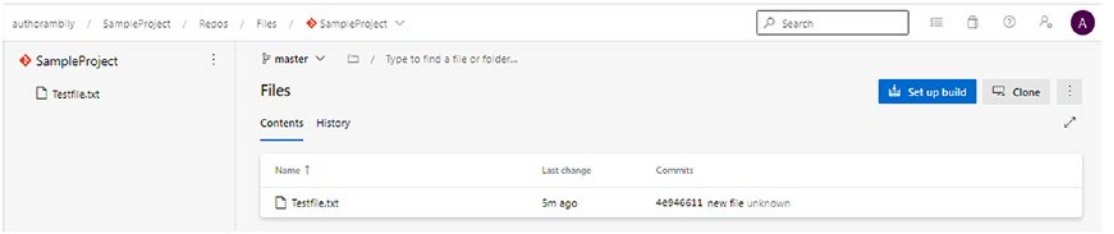


Figure 4-2. *Repo files*

The user can do minimum changes in the browser itself. The browser-based code editor supports a minimum level of IntelliSense, as shown in Figure 4-3.

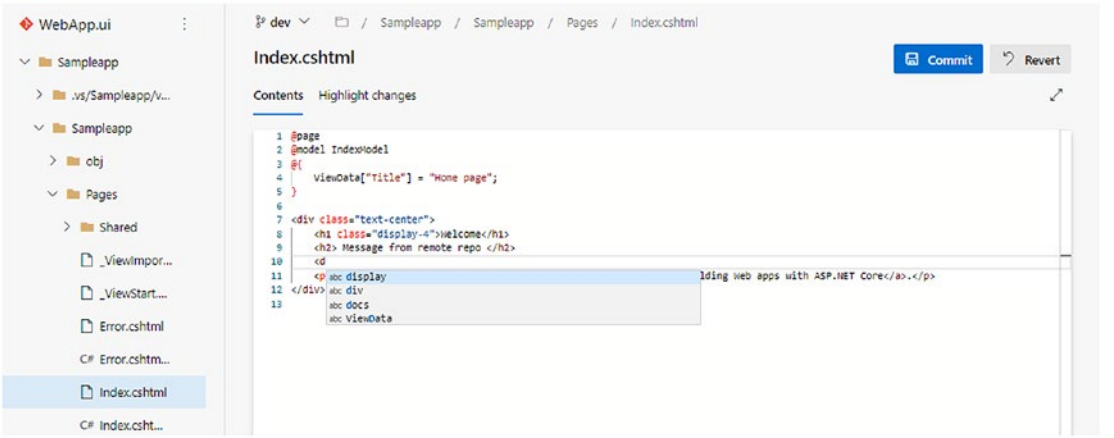


Figure 4-3. *Changes using the browser editor*

Branching and Merging

Before proceeding, let's explore the concepts associated with repos and branches. By default, every repo will have one branch associated with it: the master. As per industry practices, the master branch code should be the same as the code released to production. Developers will work in a separate branch created from the master, normally called the *dev branch*. Based on the branching and merging policy of the organization, the number and names of branches will vary. For example, some organizations use a separate branch called the *release branch* for deployment to production.

Branching strategies and merging should be defined and documented to control the release activities associated with the project. Along with branching and merging, tags can be used to track intermediate patch releases and hotfixes.

Branching allows the following:

- The development team to work parallel in different features
- Multiple teams like production support and developers to work independently
- Proper control on release management

You can create branches from any existing branch, unless protected by policy. As shown in Figure 4-4, in a normal project setup, there will be one master branch and a dev branch along with other feature development branches, if required. Developers clone the repository along with a branch to the local system to implement the features. Once cloned, updates from the remote repository will be pulled to the local repository and merged with the local changes. After the code implementation, the developer pushes the code back to the remote repository. Branches will be merged into the master branch or parent branch using *pull requests* (PRs).

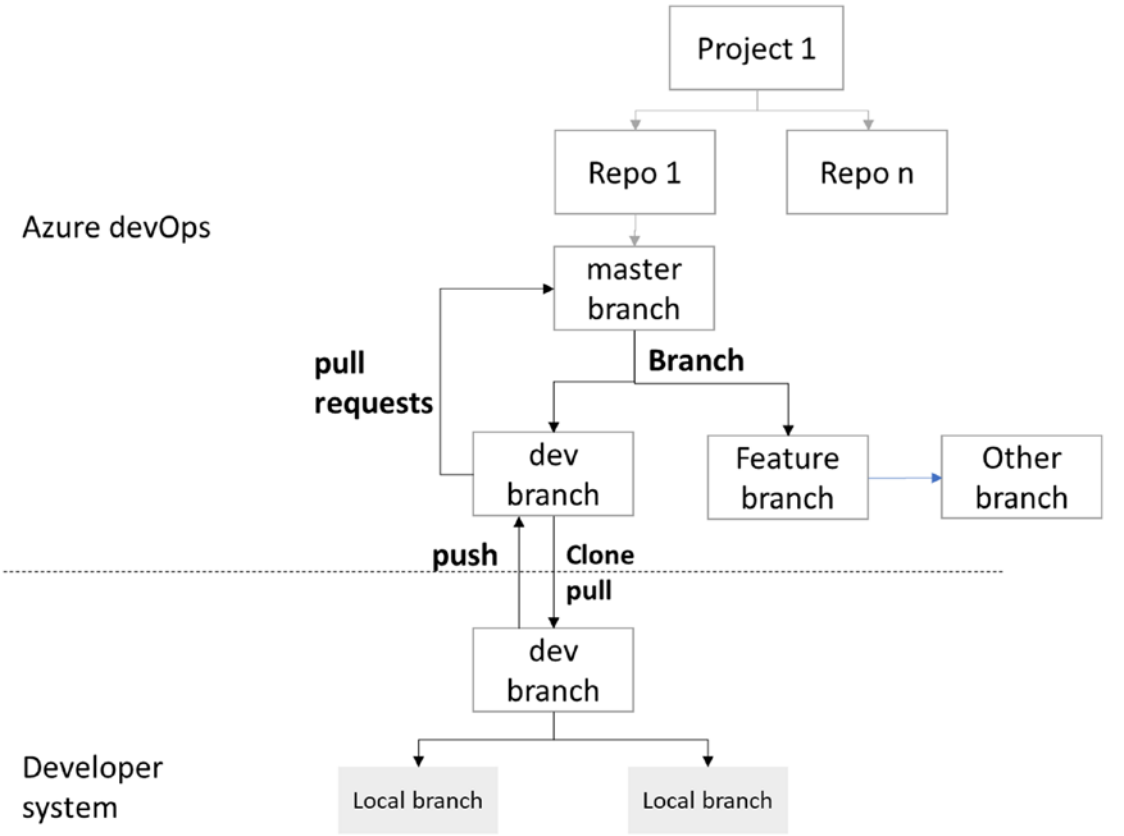


Figure 4-4. *Branch, push, and pull*

It is recommended that developers push the code to the remote repository every day to avoid a lot of conflicts as part of the merge changes. But, in some cases, developers will continue to work on the local copy for a longer duration or work offline in local Git branches to complete the implementation of the feature. Once the feature implementation is completed, the developers will push the changes to the remote repository. Developers use different branches in the local Git setup to manage the codebase and feature implementations. For example, the developer working on a feature may need to work on a production bug as part of a hotfix. In this case, the developer can configure a different branch to handle the requirement and bug fixes.

Microsoft's recommendation on a branching strategy is available at `https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops`.

Project Repo

Project repos in Azure DevOps provide a set of features to manage the repo efficiently. Figure 4-5 shows the options available at the repo level.

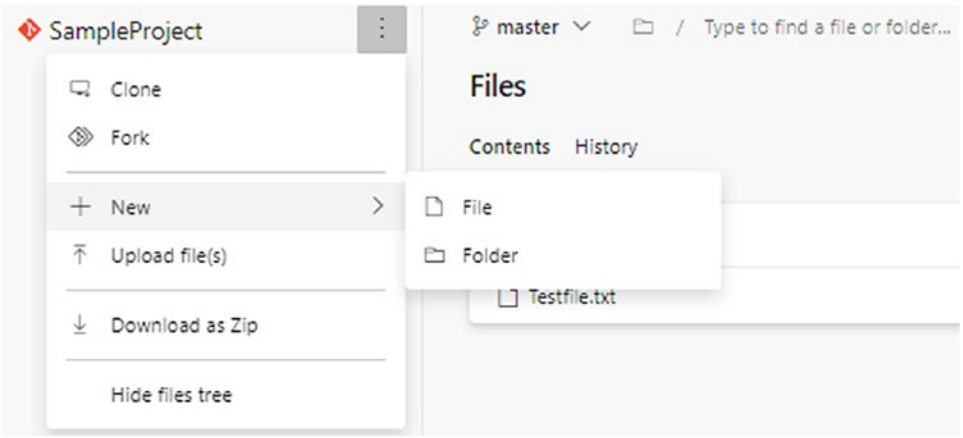


Figure 4-5. *Options at the repo level*

- *Clone*: This option allows you to clone the repo into the local system using HTTPS or SSH. Also, it provides an option to open the repository directly in Visual Studio.
- *Fork*: This allows you to create a copy of the existing repository in a current or different project.
- *New*: This option allows you to add new files and folders directly into the repo through the browser.
- *Upload file(s)*: This uploads one or more files from the local system to the Azure repo.
- *Download as Zip*: This downloads the repository as a zip folder to the local system.
- *Hide files tree*: This hides the file tree to get more screen space for the files list.

Other options available for each file listed in the file tree are Edit, Rename, Delete, and Download.

The History tab lists the changes or commits to the current repo under the selected branch, as shown in Figure 4-6. Also, this tab provides the option to filter the data based on the different views, author, and from and to dates. Every commit captures the merge operation through the Graph column and lists other details in the Commit column.

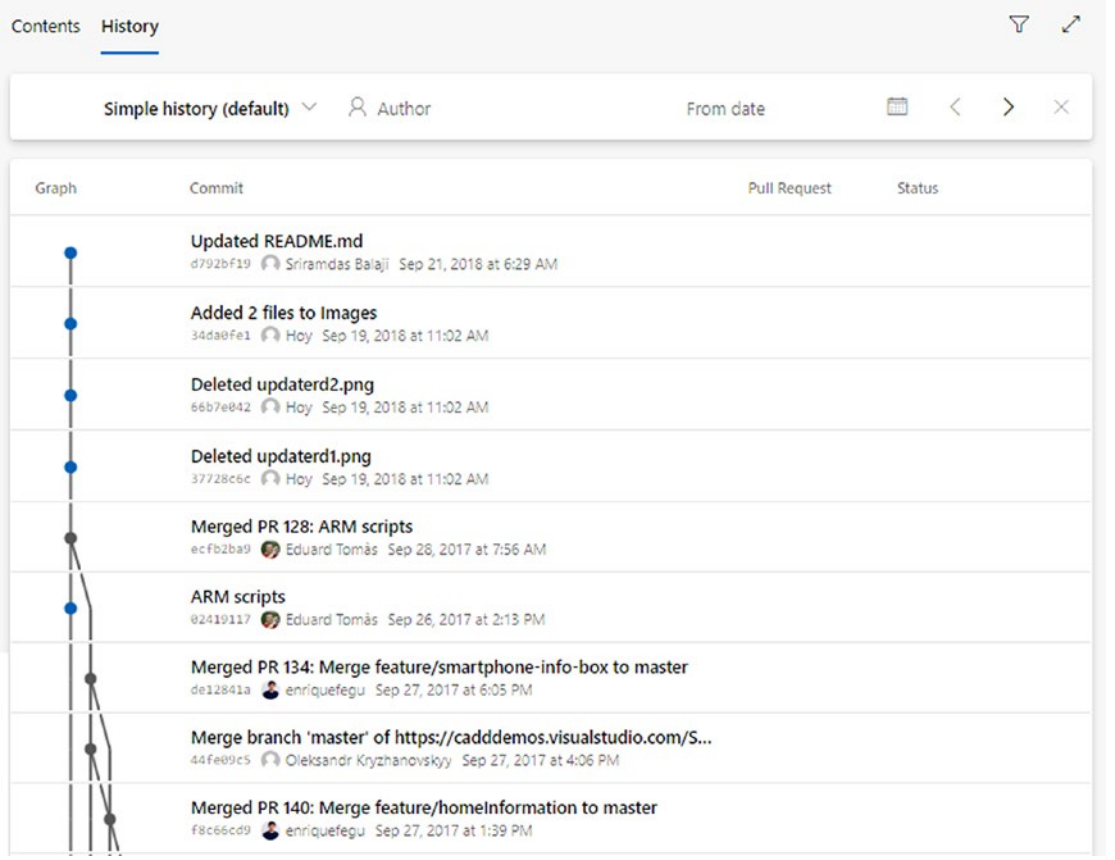


Figure 4-6. *File history*

Click the Commit entry to navigate to the files committed as part of the commit and to view the changes in each file level, as shown in Figure 4-7. New lines or code added as part of the commit will be highlighted in green, whereas the deleted lines will be highlighted in red.

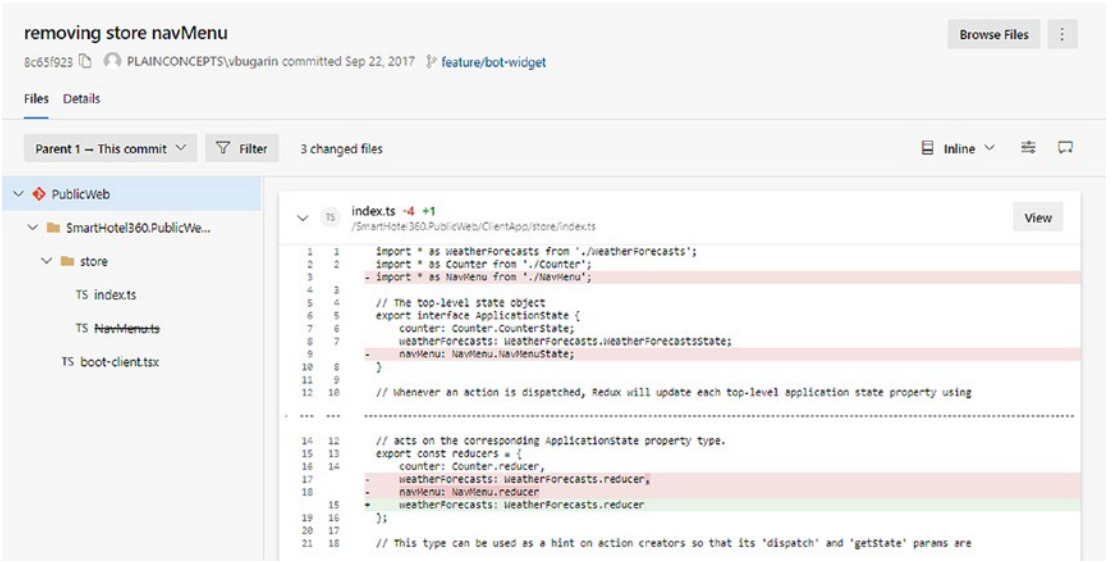


Figure 4-7. *Commit history*

There are number of filtering options available on top of the changes to drill down to the changes. Moreover, there is an option ( ) at the top right to add a global comment to the commit. This option can be used to provide review comments or changes to the commit.

Branch and Tags

Based on the branching and merging strategy, new branches will be created from a remote existing branch. Moreover, the user can use a tag to mark the current codebase with a label, which can be referred to later to get a snapshot of the codebase at that particular time. Tags are useful for versioning the code into specific points in the commit history. Once tagged, the same codebase can be retrieved using the tag in the future. For example, tags created as part of the QA releases can be used for debugging and reproducing bugs that are not shown in the latest codebase. See Figure 4-8.

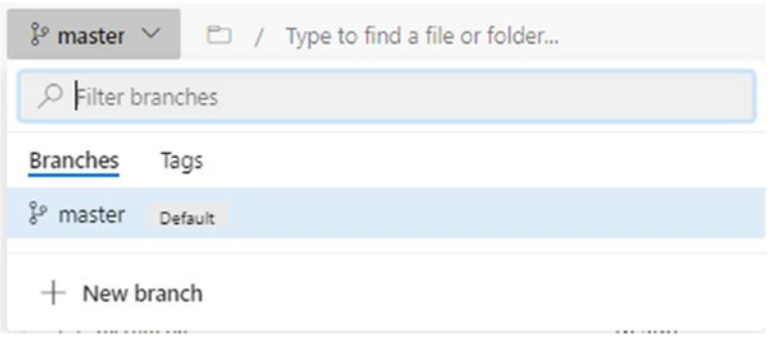


Figure 4-8. *Branch and tag options*

Create a new branch using the "New branch" option available in the branch menu, as shown in Figure 4-8. Provide a name for the new branch along with the base branch; the base branch will be your parent branch. See Figure 4-9.

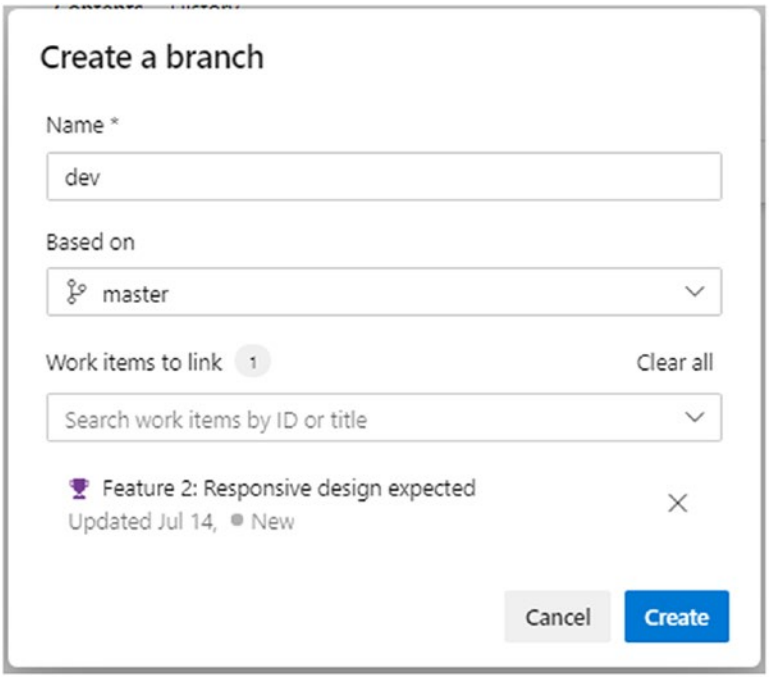


Figure 4-9. *Creating a new branch*

Moreover, the system allows you to link one or more work items to the new branch, as shown in Figure 4-9. If your branching policy is to create a new branch for each feature, then link the new feature to the new branch.

Similarly, you can add repositories using the "New repository" option available next to the existing repository, as shown in Figure 4-10.

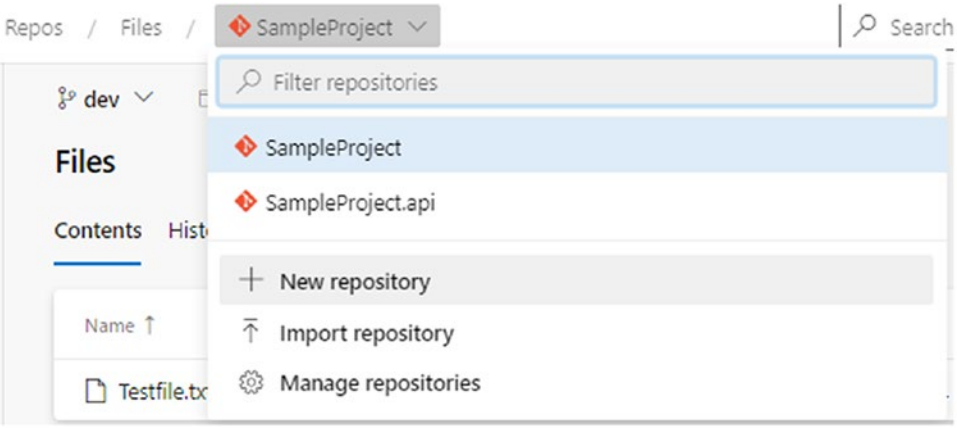


Figure 4-10. *Creating a new repo*

The "Manage repositories" option takes the user to Project settings ➤ repositories, which we discussed in Chapter 2. The "Import repository" options allows you to clone or import repositories from other systems such as GitHub or from another organization or project.

Commits

The Commits section shows the commit history with filters and options to drill down further, as shown in Figure 4-11.

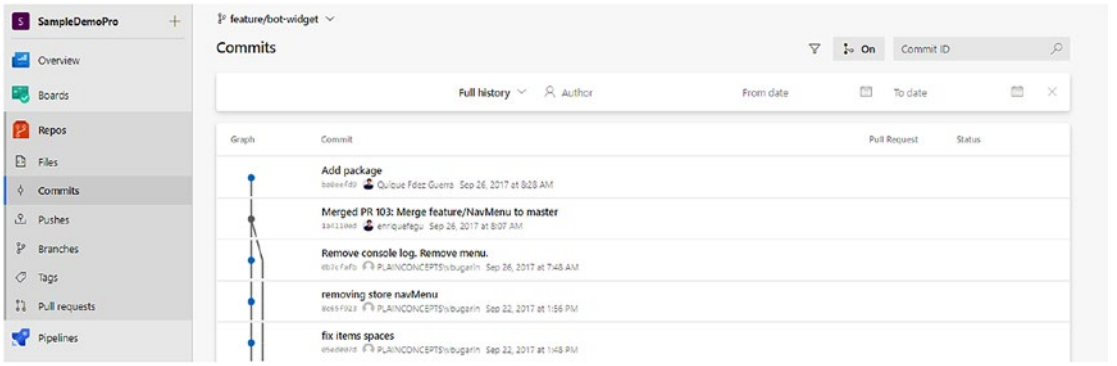


Figure 4-11. *Commits section*

Using the Graph option on the top, the user can toggle the commit graph. The user can search using the commit ID, if available. Click the commit to view the files and associated changes.

In addition to navigating through and understanding the changes, there are two important operations on the commit details page, Cherry-pick and Revert, as shown in Figure 4-12.

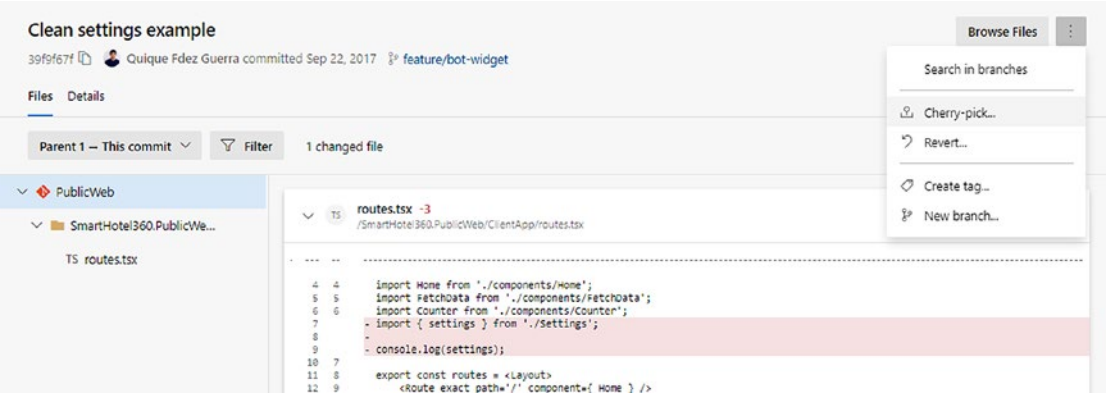


Figure 4-12. *Commit details*

- *Cherry-pick*: Git's cherry-pick operation is for applying the changes to a different branch that is not the parent of the current branch. For example, if you have made some changes in a release branch as part of a production issue fix and want to apply it in a feature branch, you can cherry-pick the commit from the release branch and apply it

directly to the feature branch. The system will not apply the change directly to the feature branch; instead, it creates a new branch out of the feature branch, applies the new commit, and asks for a pull request to merge it to a feature branch. This way, Azure DevOps protects the feature branch codebase from any unwanted cherry-pick merges. Pull requests can be further enhanced by multiple code reviewers or approvals.

- *Revert*: This option reverts the changes done as part of this commit operation. The Revert operation also creates a new branch from the current branch without the selected commit and asks the user to create a pull request to merge it back.

Pushes

The Pushes section shows the list of commits pushed to the remote repository (see Figure 4-13). Each push may have one or more commits done on the local repository. Commits are done in the local repository to track the changes locally. A set of commits associated with a task or feature implementation will be pushed and will get merged to the remote repository.

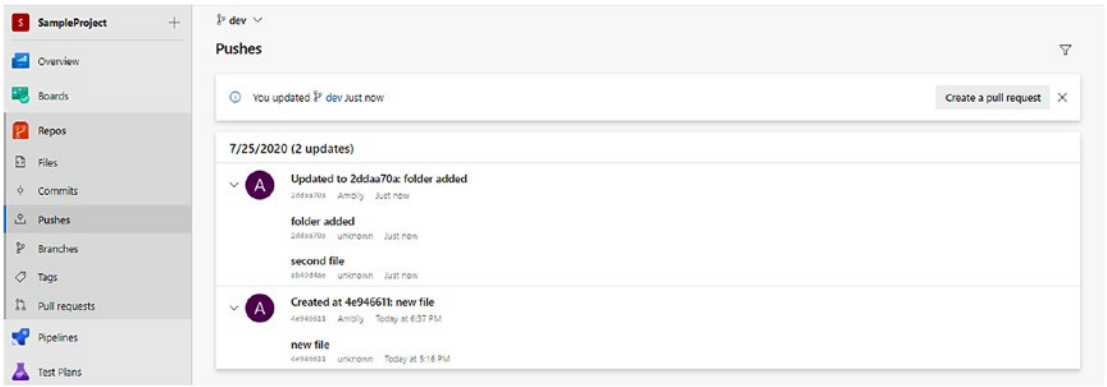


Figure 4-13. *Pushes*

In Figure 4-13, there are two pushes; one has one commit, and the other has two. The user can click the push item and view the changed files and corresponding changes. As discussed in the "Commits" section, you can revert the change, cherry-pick the commits, or create a new branch from the change.

Moreover, the Pushes view provides an option to raise a pull request to merge the commits with another branch. We will look at pull requests in the next section.

Branches

This section lists all the branches associated with the selected repo and related options. Branches are grouped under Mine, All, and Stale.

- *Mine*: This lists all the branches created by the user.
- *All*: This lists all the branches related to the repo.
- *Stale*: This lists the branches without any commit or activities in the last three or more months. This will help in tracking the unused repos and remove them. See Figure 4-14.

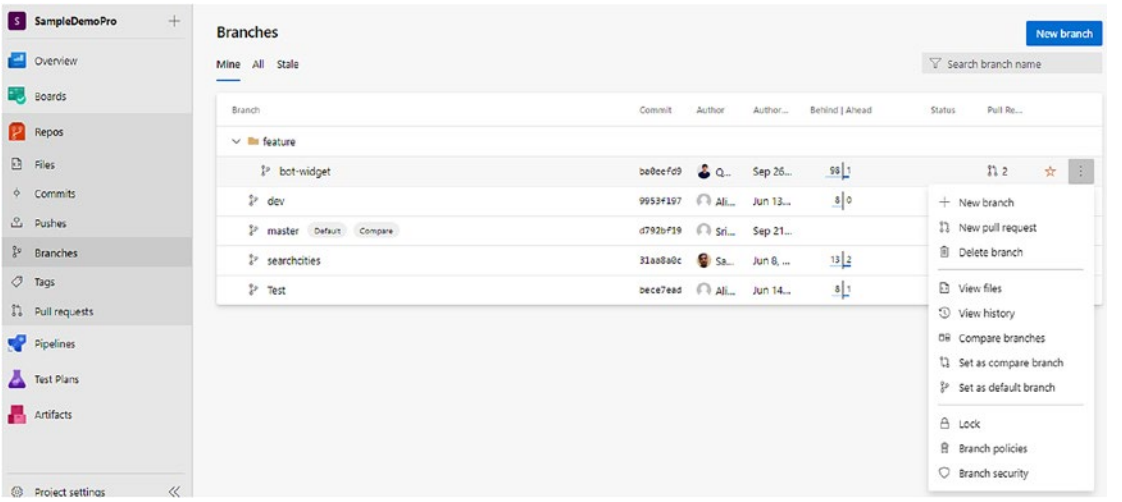


Figure 4-14. *Branches and options*

Along with the branch names, these other details are displayed: the last commit, who authored the commit, the date of the commit, and the pull request details. Moreover, the user can view the branch status with respect to the master branch; the Ahead data shows the number of commits that have not been merged with the master branch or parent branch, and Behind shows the number of commits that need to be merged into this branch. The "Add to favorite" option is available for each branch so the user can select a favorite branch.

The other options, as shown in Figure 4-14, are as follows:

- *New Branch*: This creates a new branch based on the existing branch.
- *New pull request*: This creates a new pull request to merge the changes with another branch.
- *Delete Branch*: This deletes the branch.
- *View files*: This lets you view files associated with the branch.
- *View history*: This lets you view the commit history.
- *Compare branches*: This compares two branches.
- *Set as compare branch*: This marks one base branch for comparison.
- *Set as default branch*: This sets the default branch.
- *Lock*: This locks the branch and prevents any updates to the commit history or any new updates from others. Only the locked user will be able to perform activities on the branch. This feature can be used to freeze a feature branch to complete the feature implementation.
- *Branch policies*: This defines the branch policies using Project settings ➤ repositories ➤ branch, as discussed in Chapter 2.
- *Branch security*: This defines the branch security corresponding to each group and users. This option also allows you to receive an email with a detailed report about the branch security.

Tags

The Tags section shows the set of tags and related options, as shown in Figure 4-15. Tagging helps to version the codebase to a specific commit and as required extract the tagged codebase for multiple purposes. We can use the `git checkout` command and tag name as parameters.

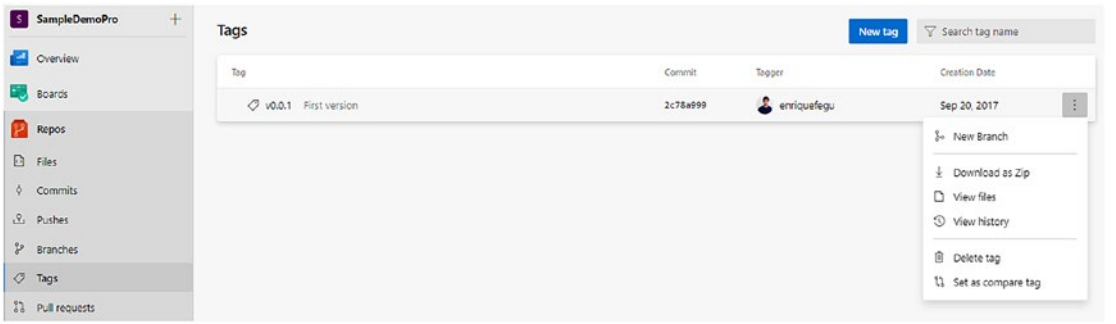


Figure 4-15. *Tags*

The options available are as follows:

- *New Branch*: This creates a new branch out of the tag.
- *Download as Zip*: This downloads the tagged codebase as a zip file.
- *View files*: This allows you to view the files marked with this tag.
- *View history*: This makes the history of the codebase tagged using the specified tag.
- *Delete tag*: This deletes the existing tag.
- *Set as compare tag*: This uses the tag as a base tag for comparison.

Pull Requests

The "Pull requests" section lists all the pull requests and options to create new pull requests, as shown in Figure 4-16. Pull requests are grouped under Mine, Active, Completed, and Abandoned.

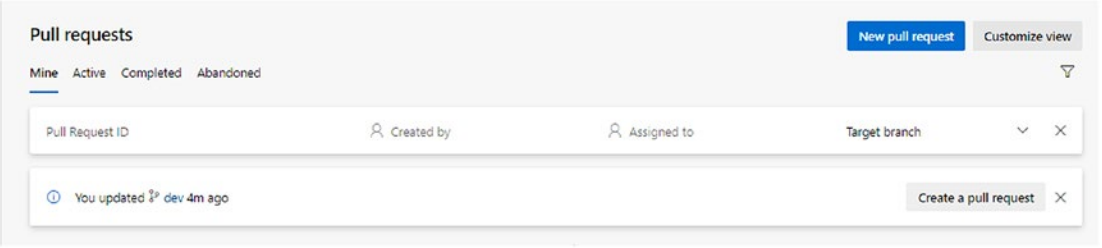


Figure 4-16. *Pull requests*

Select "New pull request" to create a pull request to merge the changes from any of the branches to the master or to another branch, as shown in Figure 4-17.

In Figure 4-17, the source branch is set to dev, and the target branch is the master, indicating that the commits from the dev branch will be merged into the master branch. Provide a title and description to refer to the pull request later. If the PR requires reviewers, add them to review the codebase before merging with the other branch. If it is mandatory that a certain person reviews the codebase, add the person as a required reviewer by using the link "Add required reviewers." The user has an option to link to one or more work items to establish the traceability of the implementation. Also, the user can add a few tags to tag the PR.

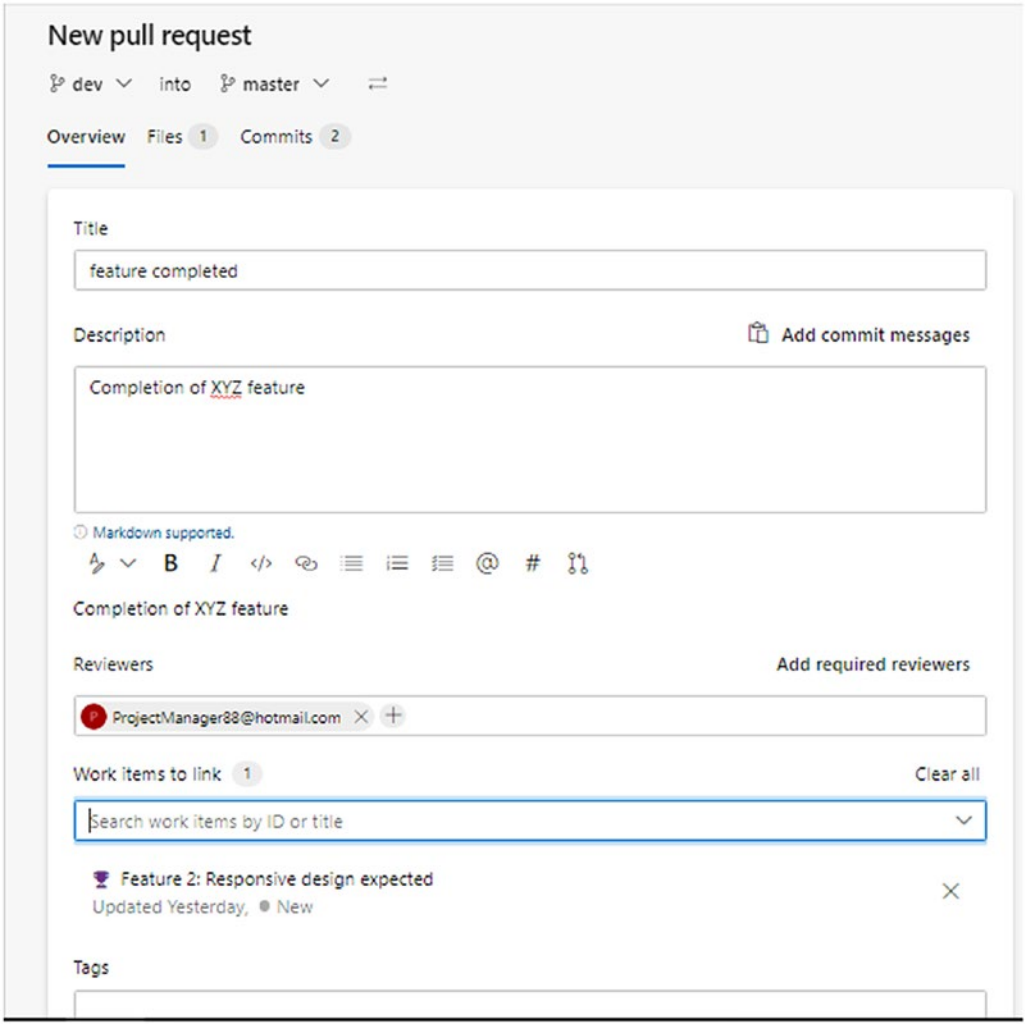


Figure 4-17. *New pull request*

Once the PR is created, reviewers can view it and select Approve. The user can select Set auto-complete, Complete, or Abandon for the PR based on the review.

In Figure 4-18, there are no required reviewers, so any team member with access can complete the PR and merge the changes to the target branch.

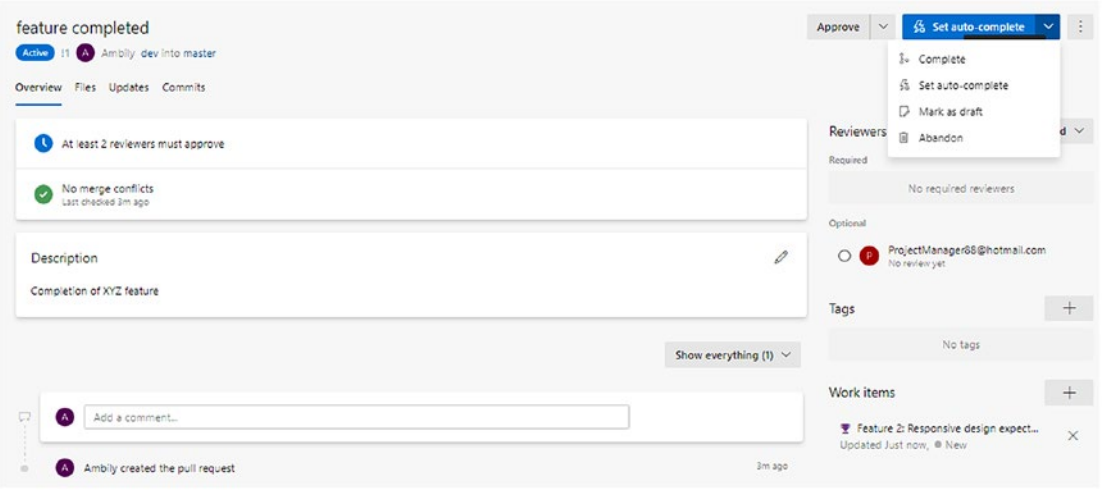


Figure 4-18. *Pull request: approval*

Figure 4-19 shows another set of options available as part of the PR to process.

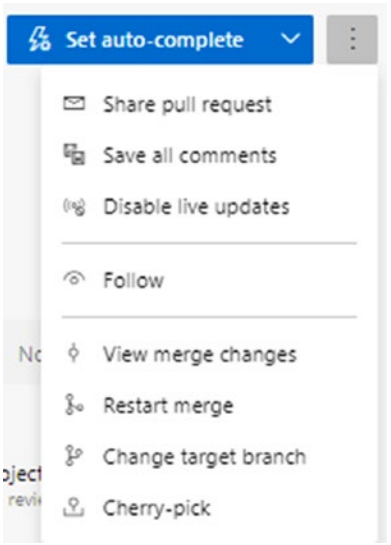


Figure 4-19. *Pull request options*

- *Share pull request*: This allows you to share the PR over email. This option helps you share additional information with reviewers, if required.
- *Save all comments*: Review comments are added in the PR for tracking. By default the comments appear as Active with edit, delete, and like options, as shown in Figure 4-20. Users can respond to the comments or mark them with a status such as Pending, Resolved, Won't fix, and Closed.

Figure 4-20. *Pull request comments*

- *Disable live updates*: This disables the live updates.
- *Follow*: Follow the PR to get notified on new comments and activities.
- *View merge changes*: View the changes after the merge.
- *Restart merge*: Restart the merge operation based on the fixes done by the developer to incorporate the review comments.
- *Change target branch*: Change the target branch.
- *Cherry-pick*: Cherry-pick the changes to another branch.

On the "Complete pull request" screen, select the different merge options supported by Azure DevOps, as shown in Figure 4-21.

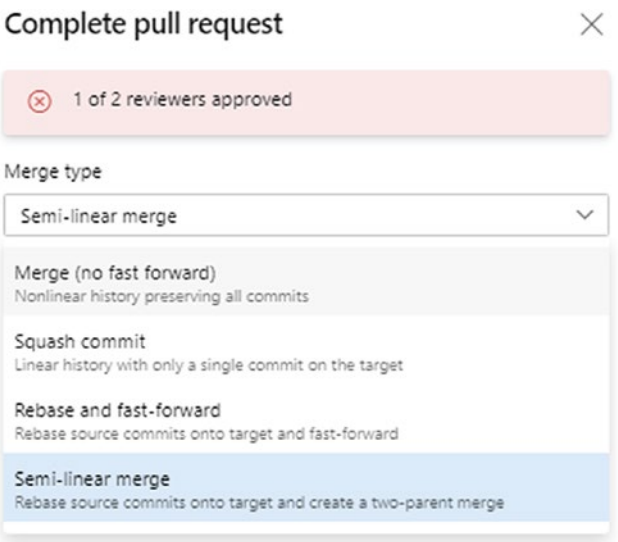


Figure 4-21. *Pull request: merge options*

- *Merge (default)*: This merges the codebase along with all the commits.
- *Squash commit*: This merges the codebase as a single commit to target branch. In this case, the target branch is not aware of the commits that happened to the source branch and isn't tracking the commit comments.
- *Rebase and fast-forward*: This merges the target changes to the source and rebases the target from the source and fast-forwards it.
- *Semi-linear merge*: This rebases the source commits onto the target and merges the two parents.

Working with Visual Studio

Most of the developers using the Microsoft technology stack for development use Visual Studio for development activities. To connect Visual Studio to Azure DevOps, open the Team Explorer using the menu View ➤ Team Explorer. Once the Team Explorer is opened, select Projects ➤ Manage Connections, as shown in Figure 4-22.

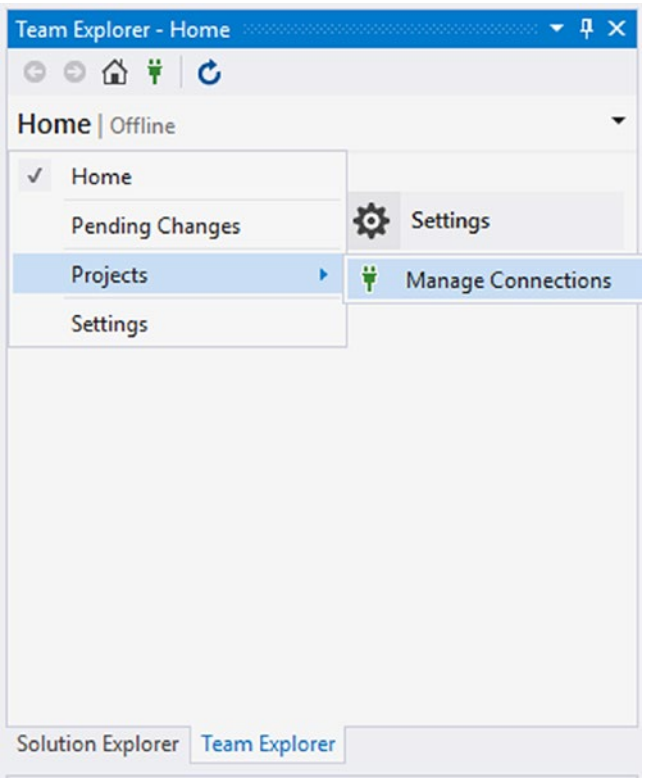


Figure 4-22. *Team Explorer*

This will show the Manage Connections menu with two options, Hosted Service Providers and Local Git Repositories, as shown in Figure 4-23.

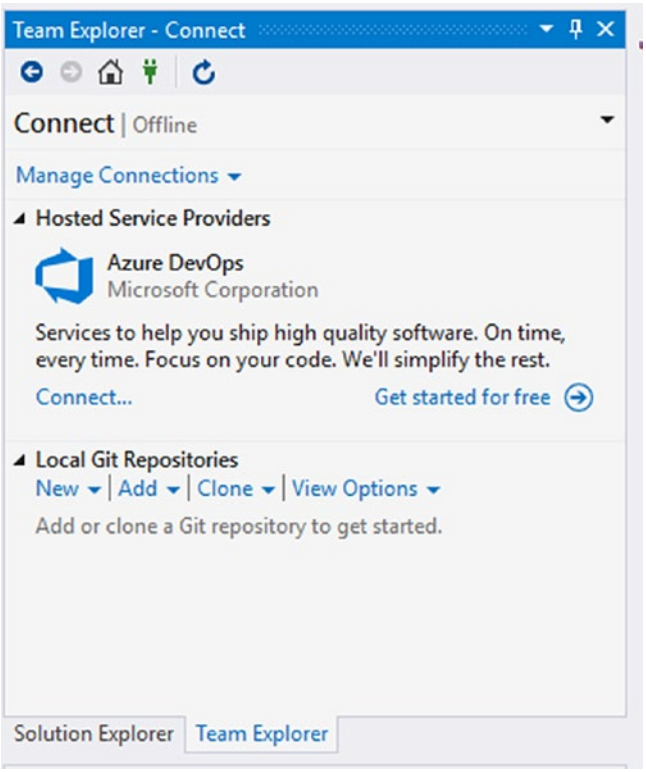


Figure 4-23. *Team Explorer, Manage Connections menu*

Click the Connect link under Azure DevOps to connect to the DevOps repo that is configured. The system will prompt you for your DevOps credentials to connect to the respective organizations. Once the credentials are validated, the system will populate all the organizations where the user has access to connect.

Select the repo to proceed, as shown in Figure 4-24.

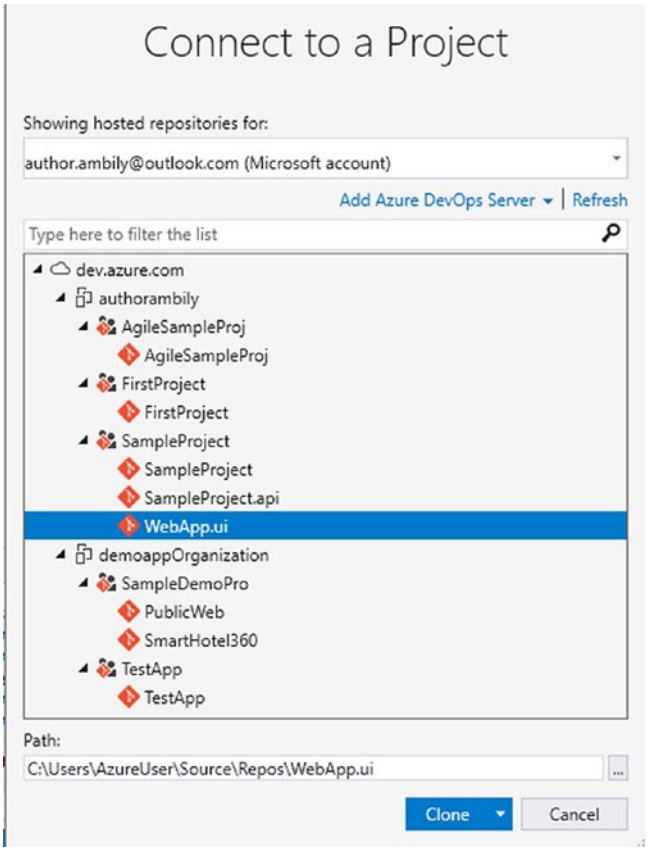


Figure 4-24. *Connecting to a project*

Visual Studio will establish a connection to the selected repo and provide all the services associated with the repo to the developer to perform from within Visual Studio. See Figure 4-25.

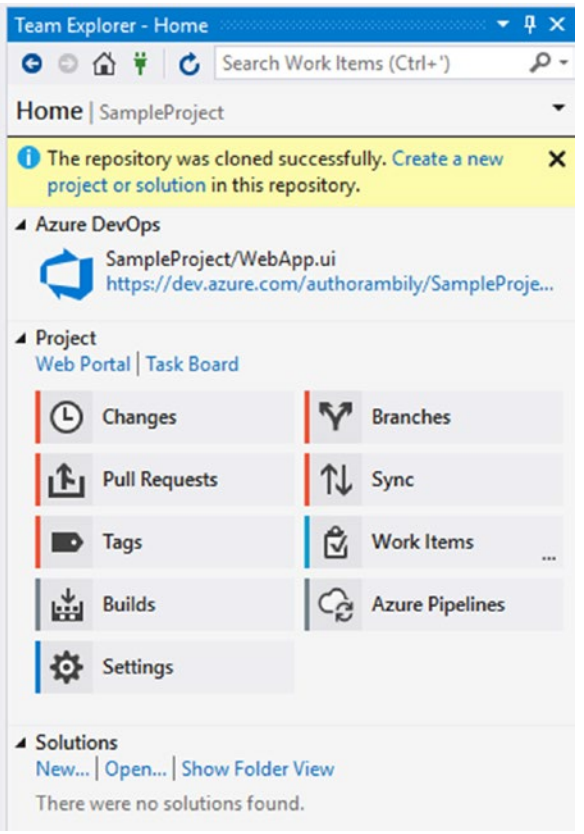


Figure 4-25. *Azure DevOps operations*

All the operations discussed with source code can be performed directly within Visual Studio, as shown in Figure 4-25. For example, select Branches to view the options associated with the branch and the list of existing branches. See Figure 4-26.

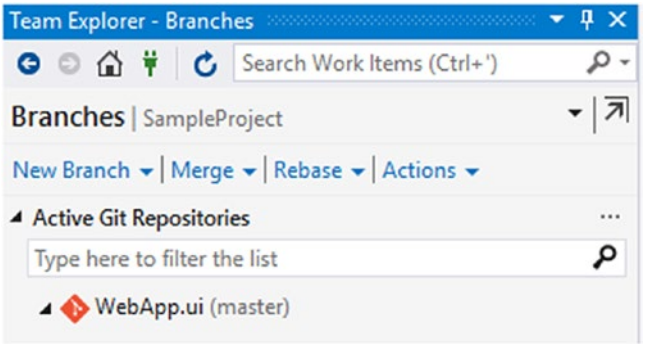


Figure 4-26. *Team Explorer, Branches*

The New Branch option helps in creating a new branch, whereas Merge allows you to merge the code from two branches. Rebase will override the existing branch code with the selected branch. Figure 4-26 shows these branch-related options.

Navigate to the Changes section to configure the username and email for committing the changes to the repo. Perform a commit and push of the codebase from the local repo to the Azure DevOps repo. In a remote execution mode, there are four methods: pull, push, fetch, and sync. Figure 4-27 shows all the options related to change tracking.

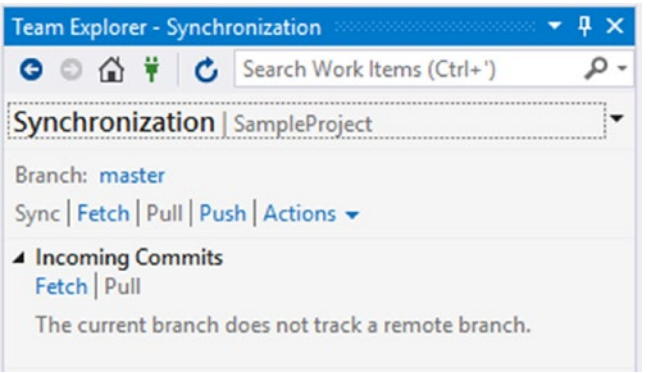


Figure 4-27. *Team Explorer, Synchronization menu*

- *Fetch*: This fetches the remote changes but does not apply them to any local branch. After selecting Fetch, invoke Merge to merge the remote changes with the local repo.
- *Pull*: This is a combination of Fetch and Merge, where the system gets the changes from the remote repo and merges them with the local copy.
- *Push*: This pushes the local changes to the remote server.
- *Sync*: This is again a combination of Pull and Push, where the system pulls the remote changes and pushes the local changes back.

Commit is another activity that can be confusing for developers. Commit will mark the local repo with a new change and version the changes. Commit will not impact the remote repository. Commits are used to protect the local changes in a version control system. Once the codebase is ready to be merged with the remote repository or with the code changes of other developers, then use any of the previously discussed methods to merge the changes with the centralized repository. Figure 4-28 shows the different Git operations.

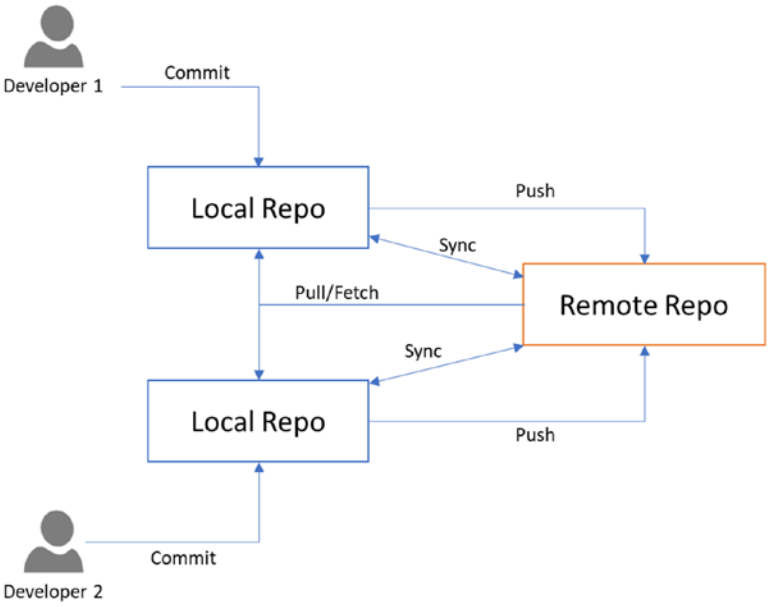


Figure 4-28. *Git operations*

Working with Visual Studio Code

To connect Visual Studio Code with Azure DevOps, install the Azure Repo extension from Extensions, as shown in Figure 4-29.

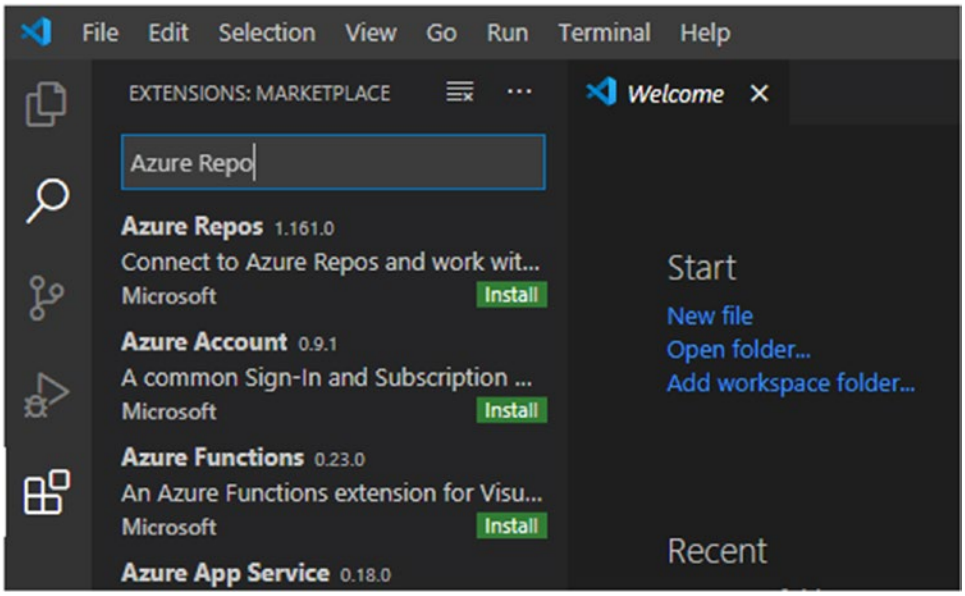


Figure 4-29. *Azure Repo extensions*

Once it's installed, ensure that the system has been set up with the TFVC command-line client (`TF.exe`). If the user has the Visual Studio Community edition, `TF.exe` will be available in `C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer\tf.exe`. Set this location to the tfvc.location setting under VS Code Settings, as shown in Figure 4-30.

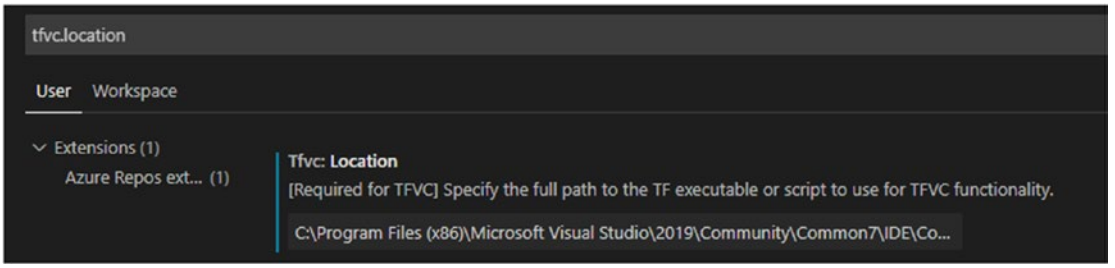


Figure 4-30. *TFVC location*

Now, select the Team option in the status bar of VS Code, as shown in Figure 4-31, to connect to Azure DevOps.

Figure 4-31. *VS Code status bar*

You have two options to connect with Azure DevOps; either provide the access token or follow the instructions to connect and generate the token, as shown in Figure 4-32.

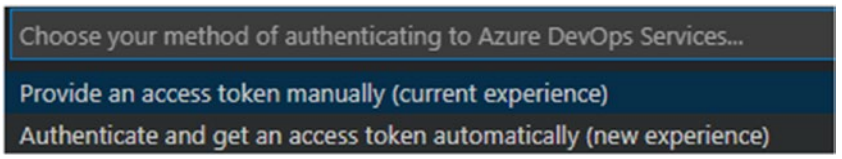


Figure 4-32. *Azure DevOps connect options*

Follow the steps to connect to Azure DevOps. Note that this extension or TFVC support is limited to local workspaces. This option supports local branch management and also has direct integration with the Azure DevOps servers.

Working with Git Bash

Git Bash is similar to the standard command prompt, where Git commands can be executed. Along with Git command support, Git Bash has a wrapper over the standard command line and supports Linux commands. If you are from a Linux background and want to use similar commands, then use Git Bash to manage the Git repos.

Some of the Git commands used for managing the repo are discussed at the beginning of this chapter. A few of the most useful branch-related Git commands are as follows:

- Use this command to list the branches available:

  ```
  git branch
  ```

  or

  ```
  git branch -a
  ```

  The first command will list all the remote branches, whereas the second command will list the remote and local branches available, as shown in Figure 4-33.

Figure 4-33. *Git branches*

- Use this command to change to another branch:

  ```
  git switch <<branch>>
  ```

- Use this command to check out a branch:

  ```
  git checkout <<branch>>
  ```

- Use this command to show the status of the working branch:

  ```
  git status
  ```

Sample Application

We have discussed the various version control features offered by the Azure DevOps service. Let's explore these features through a web application developed using Angular and the .NET Web API, as shown in Figure 4-34.

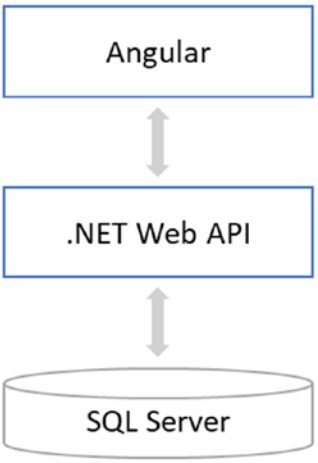


Figure 4-34. *Sample app*

The Angular development will be done using the Visual Studio Code IDE, and the .NET API will use Visual Studio as the integrated development environment (IDE). SQL Server database can also be added to the source repo using the database project templates available in Visual Studio.

Version Controlling an Angular App

As mentioned, the Angular development will be done using a Visual Studio Code IDE. As of now, this IDE will not support direct integration with the Azure DevOps service. Follow these steps to add the Angular app into the Azure DevOps service:

1. Create a team project in the Azure DevOps service using Git as the version control system.
2. Configure a repo for adding an Angular app.
3. Clone the empty repo to the local system using the `git clone` command.

   ```
   git clone https://authorambily@dev.azure.com/authorambily/
   SampleApp/_git/angularapp
   ```

Note Instead, you can push an existing repository from your system using the following:

   ```
   git remote add origin https://authorambily@dev.azure.com/
   authorambily/SampleApp/_git/angularapp

   git push -u origin –all
   ```

4. Navigate to the newly created repo, as shown here:

   ```
   cd angularapp
   ```

5. Create a new Angular app using ng, as shown here:

   ```
   ng new <<appname>>
   ```

6. Add all the newly created files into Git using the following:

   ```
   git add .
   ```

7. Commit the newly added files or changes, as shown here:

   ```
   git commit -m "first version"
   ```

8. Push the changes to the remote repo, as shown here:

   ```
   git push
   ```

Note If the repo was initiated with `ReadMe.md`, then do a pull before push, as shown here:

```
git pull
```

The first version of the Angular app is updated to the remote repository. The first commit will be done in the master branch, which get created along with the repo creation. Any further changes or modifications will be done in a separate branch such as the dev or feature branches. Follow these steps to update the Angular app:

- Create the dev branch from the master branch, using the Azure DevOps portal.
- Check out the dev branch.

  ```
  git checkout dev
  ```

Note This may lead to the following error:

error: pathspec 'dev' did not match any file(s) known to git

This is due to the new branch being added to the remote repo. Pull the newly added branch details before checkout the same as when using the `git pull` command.

```
git pull
git checkout dev
```

- Verify the branch.

  ```
  git branch
  ```

- Modify the application to implement a feature or bug fixes.
- Commit the changes properly to the local repository. to stage the changes

  ```
  git add .
  git commit -m "commit comment"
  ```

- Once the implementation completes, push to the remote repository.

  ```
  git push origin dev
  ```

Based on the remote repo modifications and local updates, there may be scenarios where merge conflicts can happen, which the system tries to resolve using the default merge strategy. If it fails to resolve the conflicts, it shows the following kind of error in `git push`:

> To https://dev.azure.com/authorambily/SampleApp/_git/angularapp
>
> ! [rejected] dev -> dev (fetch first)
>
> error: failed to push some refs to 'https://dev.azure.com/authorambily/SampleApp/_git/angularapp'
>
> hint: Updates were rejected because the remote contains work that you do
>
> hint: not have locally. This is usually caused by another repository pushing
>
> hint: to the same ref. You may want to first integrate the remote changes
>
> hint: (e.g., 'git pull ...') before pushing again.
>
> hint: See the 'Note about fast-forwards' in 'git push --help' for details.

To resolve this, pull the changes using `git pull`. This brings the remote changes to the local version and displays both the changes in Visual Studio Code along with options to resolve them, as shown in Figure 4-35.

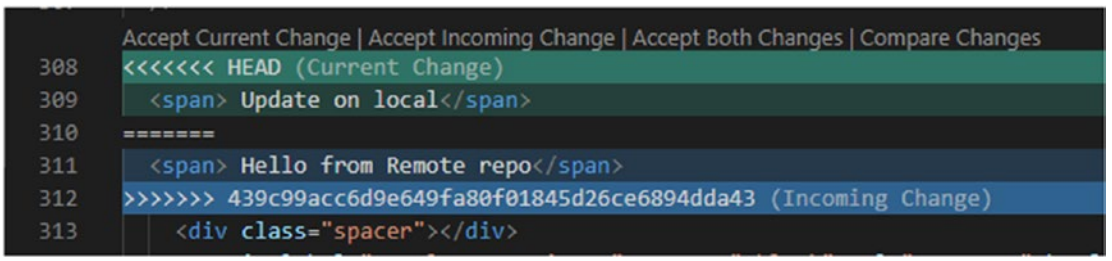


Figure 4-35. *Merging conflicts*

- *Accept Current Change*: This option accepts the local changes.
- *Accept Incoming Change*: This option accepts the remote changes.
- *Accept Both Changes*: This option accepts the remote as well as local changes.
- *Compare Changes*: This option compares the changes and makes a decision manually.

Another way to merge the conflicting changes is to use the `git merge` command.

```
git merge -s <<strategy>>
git merge -s ours
```

Refer to the merge strategies at https://git-scm.com/docs/merge-strategies to understand the different strategies and their usage.

Version Controlling the .NET API

APIs or the middle layer of an application will be developed using any of the IDEs such as Eclipse or Visual Studio. .NET APIs are generally developed using Visual Studio, which supports direct integration with Azure DevOps. Refer to the section "Working with Visual Studio" to version control the .NET API using Azure DevOps and VS.

Summary

Version control and branching strategies define the development and release management of a project. Based on the branching strategy, the team works on different branches in parallel to complete the development, production support, and feature releases. The build and release setups are also dependent on the way the version control management system is configured with the branching strategies.

CHAPTER 5

Test Management Using Azure DevOps

The Azure DevOps service has all the test-related features needed to support end-to-end test management. By default, the test management features are not available, so you must enable them. You can enable the test plan in the Billing section on the Organization Settings screen of Azure DevOps to start exploring features such as test planning, tracking, execution, browser-based test annotations, centralized reporting, and so on (Figure 5-1). This chapter will explain the various test management features available in the Azure DevOps service.

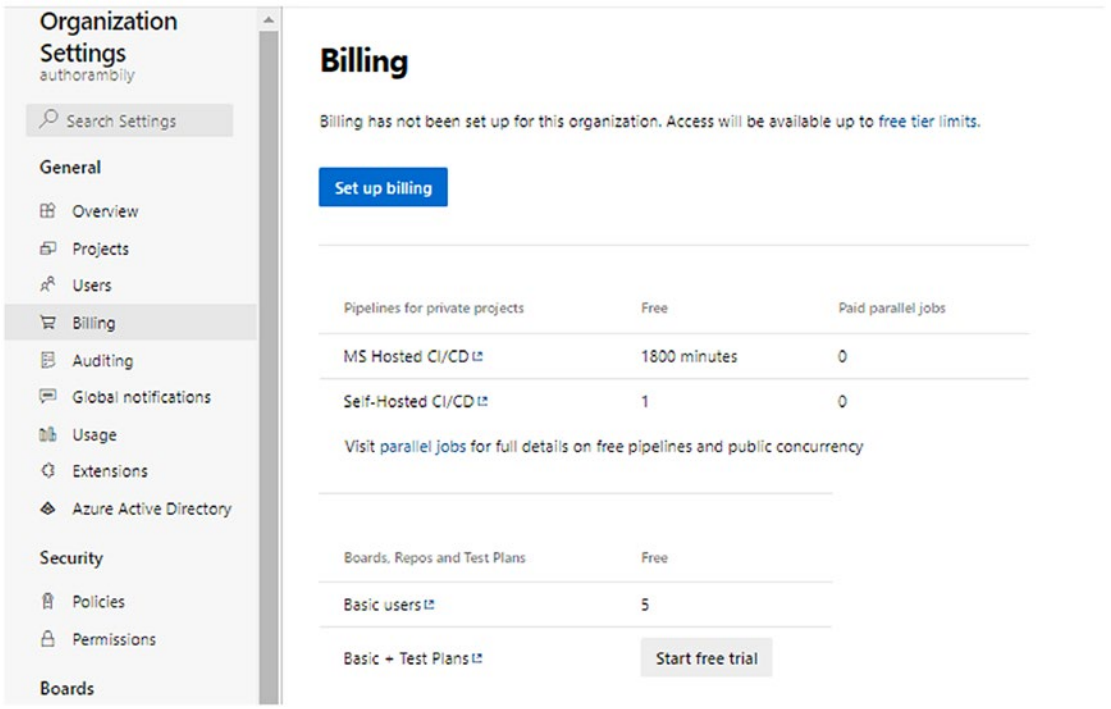


Figure 5-1. *Billing page*

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_5

Azure DevOps provides three different test management features: test plans, test suites, and test cases.

- *Test plan*: This is a group of test suites and test cases to be executed in an iteration or in an area.
- *Test suite*: This is a group of test cases to validate a scenario or use case. It can be a static test suite, a requirement-based test suite, or a query-based test suite; we will discuss more about test suites in the "Test Suites" section of this chapter.
- *Test cases*: These validate a single functionality of the application. Test cases can be added to test suites or test plans and can be mapped to multiple suites or plans at a time.

Test Cases

Test cases are the basic test elements to validate the functionality of an application or the nonfunctional requirements associated with an application. In test management, test cases are categorized based on their usage. The following are a few of them:

- *Unit test case*: This will be used by developers to verify the completeness of their implementation. There are many tools and technologies available associated with each of the implementation technologies to automate the unit test cases. There are many unit test frameworks and libraries such as Junit for Java-based applications, NUnit for .NET, Jasmine and Karma for Angular, and Mocha and Chai for React.
- *Functional test case*: This is a test case used to validate the functional implementation of the system.
- *Security test case*: This verifies the security constraints with the system. For example, security test cases may verify any cross-site scripting issues with a web application.
- *Performance test case*: This verifies the performance of the system with different parameters such as data volume, user load, stress testing, and so on.
- *User acceptance test*: This is acceptance testing by the business stakeholders.

There are many test cases such as regression testing, integration testing, smoke testing, sanity testing, system testing, BVT test cases, and so on. As a good practice, test cases will be reviewed by test leads or the functional SME to align the test cases with the expected functionalities. Most of the NFR-related test cases will be reviewed by the architect.

In most cases, the test cases will have few predefined fields or information such as the steps for verifying the functionality, acceptance criteria for each of the steps, etc. Azure DevOps provides a test case template, which can be customized to fit the individual project requirements. Figure 5-2 shows the default template.

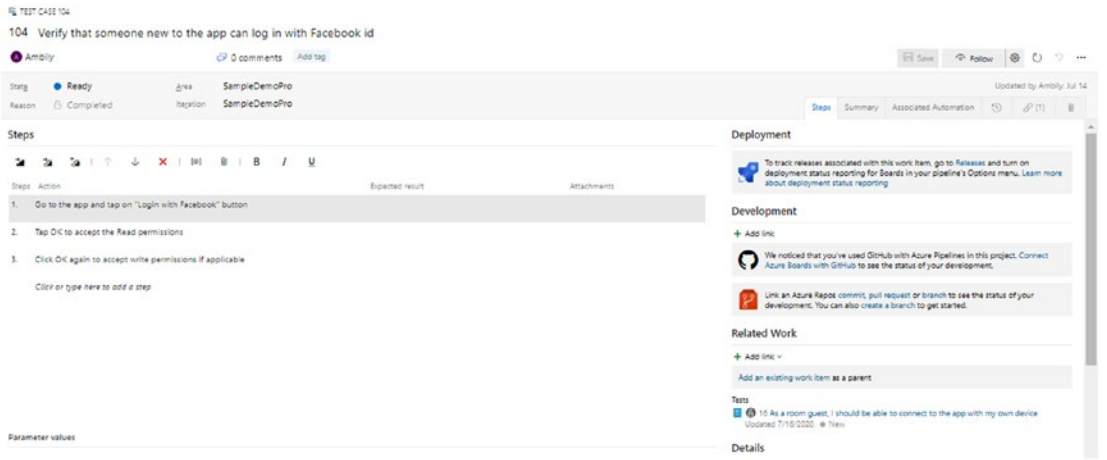


Figure 5-2. *Test case template*

The main fields are the steps defining the the actions and expected result, and the parameters. A step defines an action by the user to validate a user interaction in the system. For example, a user can provide credentials in a login form and click the Login button to login to the system. When the user performs an action such as the login, user input is required. When a tester executes such steps, they should pass various values to identify whether the system is working properly and responding to different user input values. These values define the positive cases, negative cases and boundary conditions. If a test step requires a user value that can vary and return different results, then such fields are marked as parameters.

Shared steps are another important concept; these are where a set of steps will be shared across multiple test cases. For example, the login test steps will repeat across most of the test cases that require the user authentication. The user can define such

steps along with actions, results, and parameters as a shared step and reuse it in other test cases to avoid duplication of the same content. Follow the Don't Repeat Yourself (DRY) principle to avoid duplicating the same steps in another test cases.

Shared Steps and Parameters

As mentioned earlier, a shared step is a set of test case steps that can be reused in multiple test cases. For example, the steps for logging in to the system can be defined as a shared step and included in many other test cases.

Define the steps, select them to group them as shared steps, and click the "Create shared steps" icon on the top of the steps (third icon), as shown in Figure 5-3.

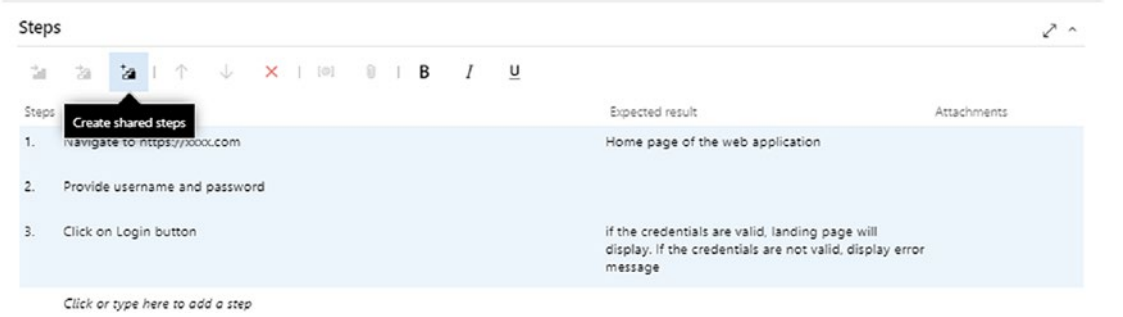


Figure 5-3. *Shared steps*

Provide a name for the steps to continue. This will replace all the existing steps with a single step referenced by the name of the shared steps, as shown in Figure 5-4.

Figure 5-4. *Shared steps inserted*

Once you double-click the step name, it opens the newly added "Shared steps" work item, as shown in Figure 5-5.

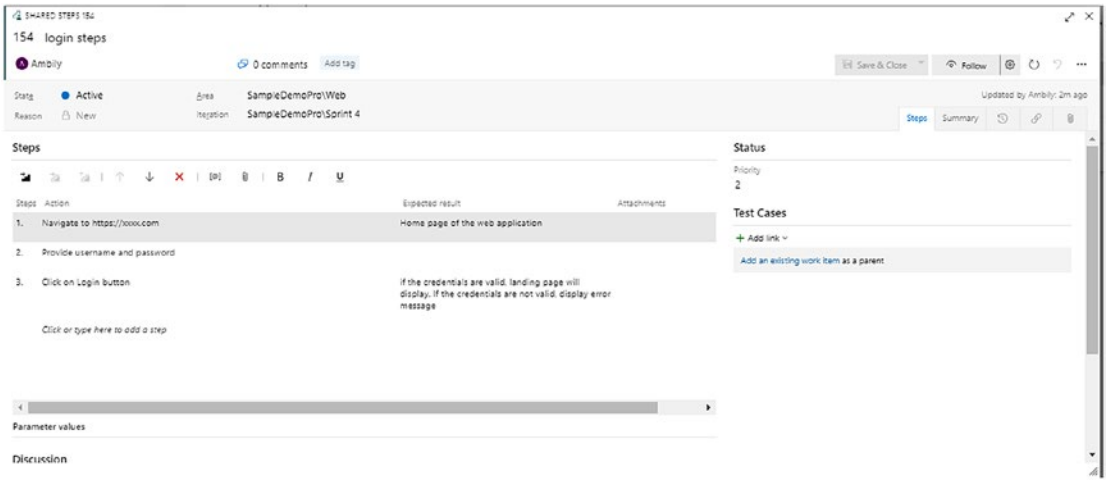


Figure 5-5. *"Shared step" work item*

Users can further customize the steps by adding parameters, which expect data from users at the time of execution. For the previous scenario, add the parameters *username* and *password* and bind to the steps.

Parameters can be added by selecting the parameter option ([@]) or by adding variables with @, as in `@username`. Once they're added, navigate back to the test case where the parameters will be listed in the "Parameter values" section. Provide the set of values expected for the parameter, as shown in Figure 5-6.

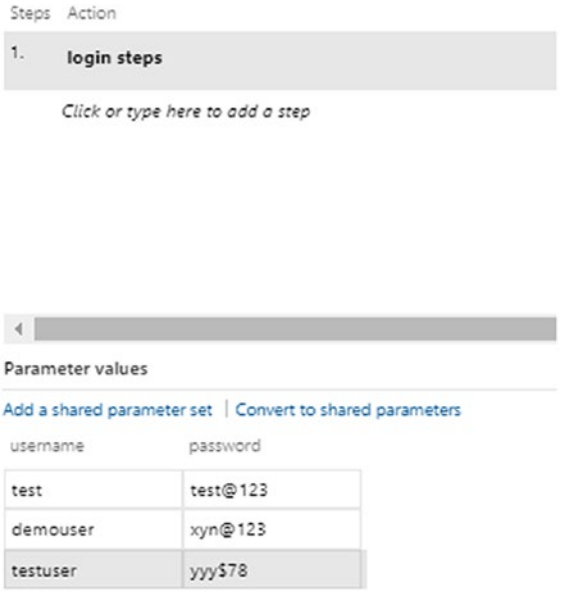


Figure 5-6. *Parameter values*

If the parameters are used across multiple test cases, they can be added as a shared parameter set. Just like shared steps, a shared parameter set will be available for all test cases to reuse the same set of data across multiple test scenarios.

In Figure 5-6, select the parameters and click the "Convert to shared parameters" link to convert it to a shared parameter set. The "Parameters" section will cover more about the parameters.

Test Suites

Test suites are a set of test cases executed to validate the implementation of a feature or component. We can define the test suites using a static suite, a requirement-based suite, and a query-based suite, as shown in Figure 5-7.

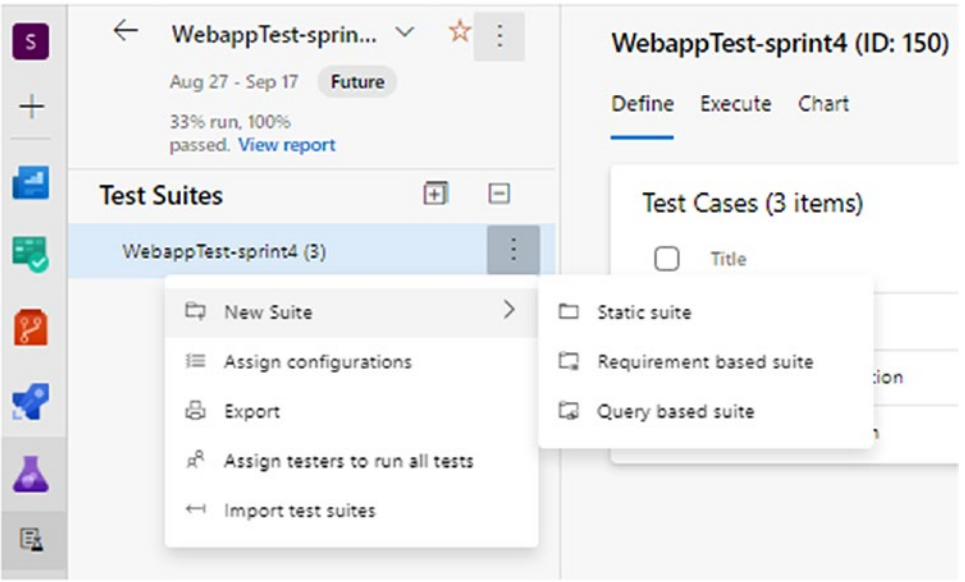


Figure 5-7. *Test suites*

Here are the differences between the three options:

- *Static suite*: Add the test cases manually to the suite.
- *Requirement based suite*: Select the test cases based on the requirements.
- *Query based suite*: Select the test cases using a query.

Test Plans

A test plan defines the test suites and test cases corresponding to an area or iteration. It is used for grouping the test cases based on the features prioritized in an iteration or related to a specific area.

This section displays the existing plans and allow us to create new test plans, as shown in Figure 5-8.

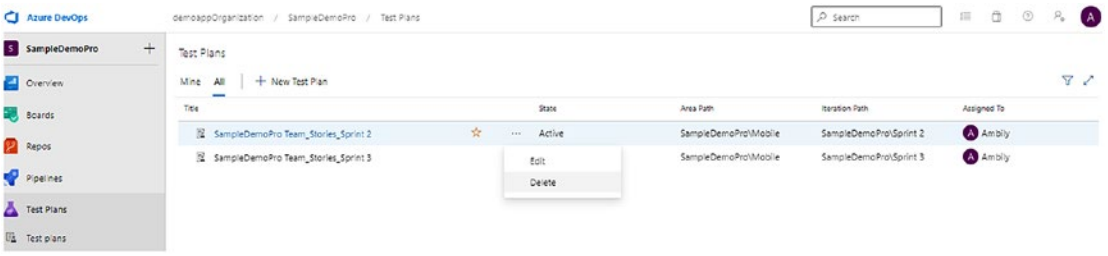


Figure 5-8. *Test Plans section*

Select the New Test Plan option to create a new test plan, as shown in Figure 5-9.

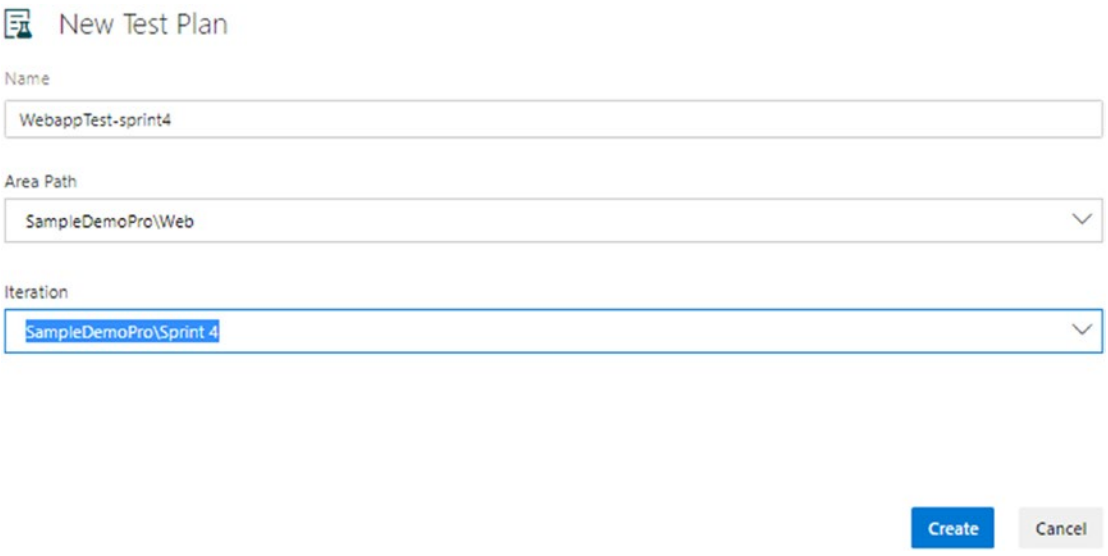


Figure 5-9. *Creating a new test plan*

Once the test plan is created, the user will be redirected to the details page to add new test cases. Test cases can be added to the test plan using the following two options:

- *Add existing test cases*: This adds an existing test case. This option opens a query window to select the test cases based on a query, as shown in Figure 5-10. The user can edit the query to filter the test cases and select the specific test cases from the result set.

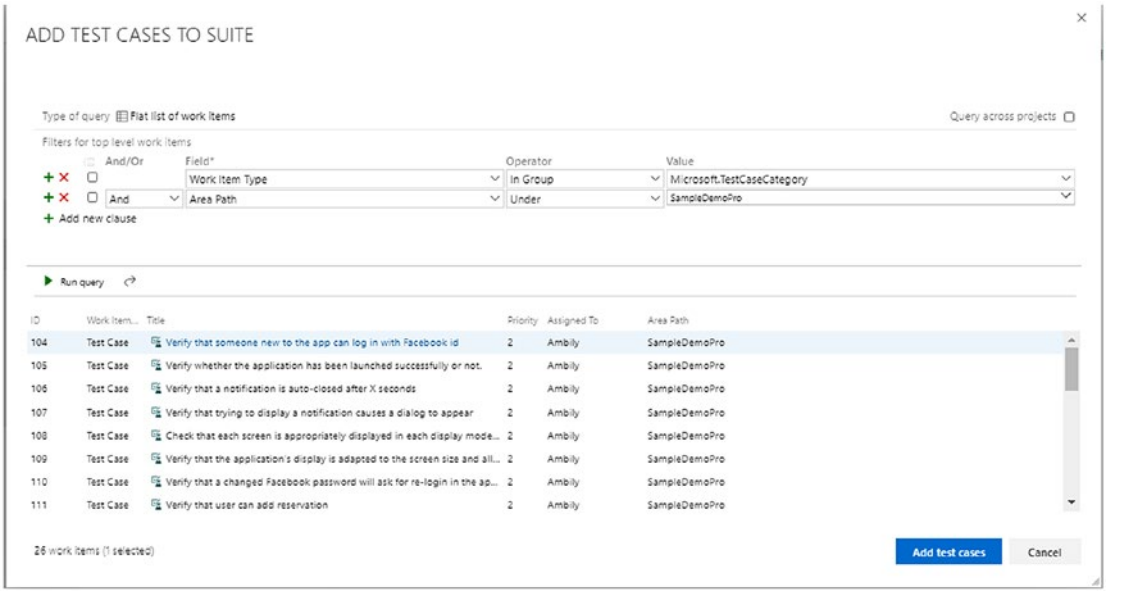


Figure 5-10. *Adding the test cases to suite*

- *Add test cases using grid*: This option supports the entry of multiple test cases using an Excel or table format. The user can create test cases in Excel or CSV format, and then testcases can be copied directly to the grid view to insert the test cases. This feature helps the testers to work offline.

  Enter the test case title in the first column and define the steps associated with the test case in the second column. Each step should be specified in different rows along with the expected result, as shown in Figure 5-11.

ID	Title	Step Action	Step Expected Result	Assigned To	State
	verify user login	1. login to web site	able to login to site	Ambily	Design
		2. navigate to home page	home page loaded		
		3. verify the existence of user name on top right corner	user name exists on top right corner		
	Verify user notification	1. login to system	user will be able to login with valid credentials		
		2. notification icon on top right corner	user view the notification icon with number of notifications		

Figure 5-11. *Adding test cases using a grid*

The Assigned To and State columns can be filled based on the requirements or left empty. The default option is for the system to add the state Design to every new test case. Once the test cases are added, click the Save icon ( ) on top of the grid to save the test cases.

Once they're saved, the system will create new test cases and display the ID for each test case, as shown in Figure 5-12.

ID	Title	Step Action	Step Expected Result	Assigned To	State
151	verify user login			Ambily	Design
		1. login to web site	able to login to site		
		2. navigate to home page	home page loaded		
		3. verify the existence of user name on top right corner	user name exists on top right corner		
152	Verify user notification			Ambily	Design
		1. login to system	user will be able to login with valid credentials		
		2. notification icon on top right corner	user view the notification icon with number of notifications		

Figure 5-12. *Test cases, grid view*

Define Tab

The Define tab shows the existing test cases corresponding to the selected test plan, as shown in Figure 5-13. The user can change the list view into a grid view, which allows for multiple test case editing. Also, the system provides the options to reorder the test cases using the move up and move down icons, to select columns, and to filter the test cases using various filter options.

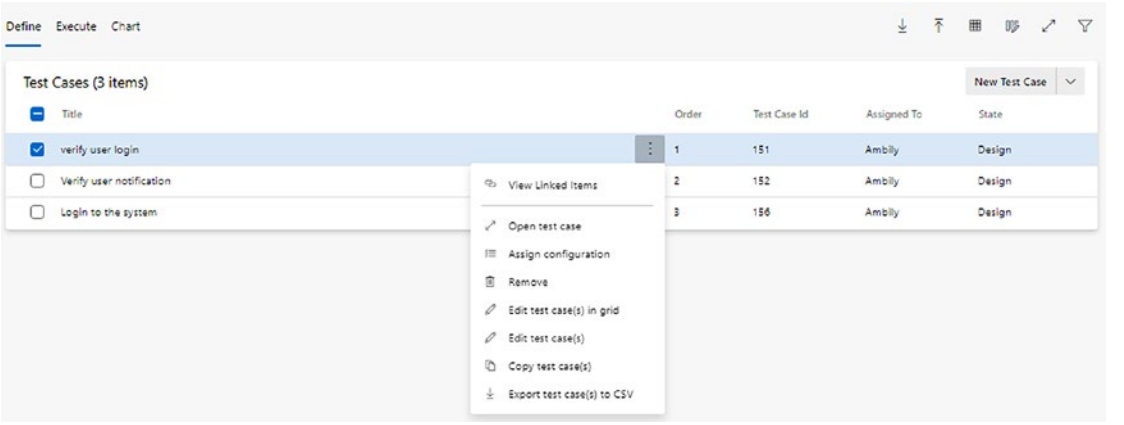


Figure 5-13. *Define tab*

For each of the test cases, there are a set of options available.

- *View Linked Items*: This option shows the linked test suites, requirements, and bugs. If the system establishes proper traceability, the user can identify the requirements and bugs associated with the current test case, as shown in Figure 5-14.

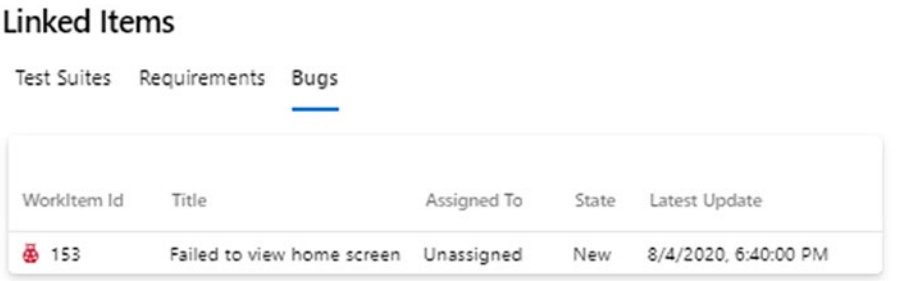


Figure 5-14. *Linked items*

- *Open test case*: This option opens the test case.
- *Assign configuration*: This specifies the system configurations required to execute the test cases; refer to the "Configurations" section for more details, as shown in Figure 5-15.

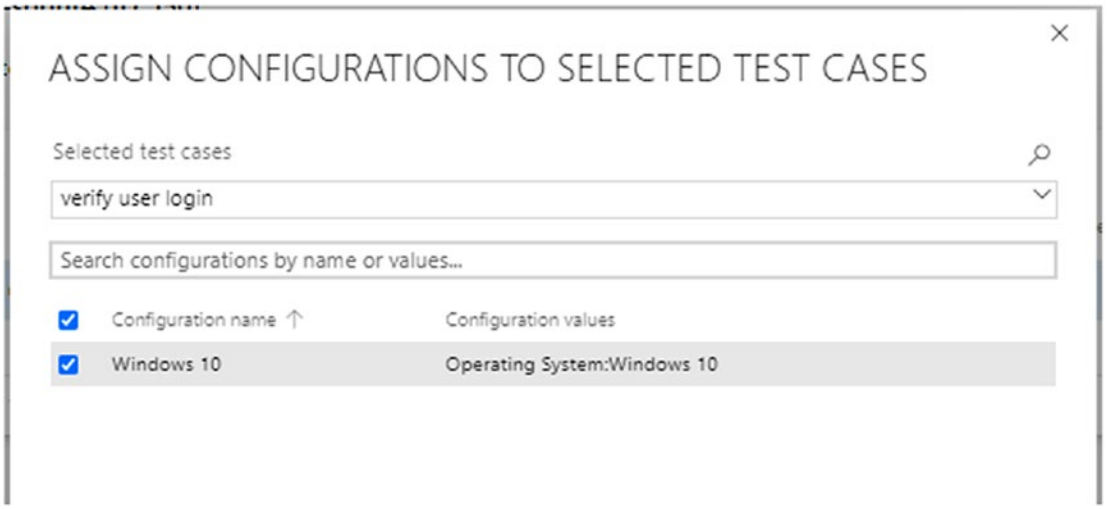


Figure 5-15. *Assigning configurations*

- *Remove*: This option removes the selected test case.
- *Edit test cases(s) in grid*: This allows you to edit multiple test cases in a grid.
- *Edit test case(s)*: This allows you to edit the test case or test cases by selecting specific fields, as shown in Figure 5-16.

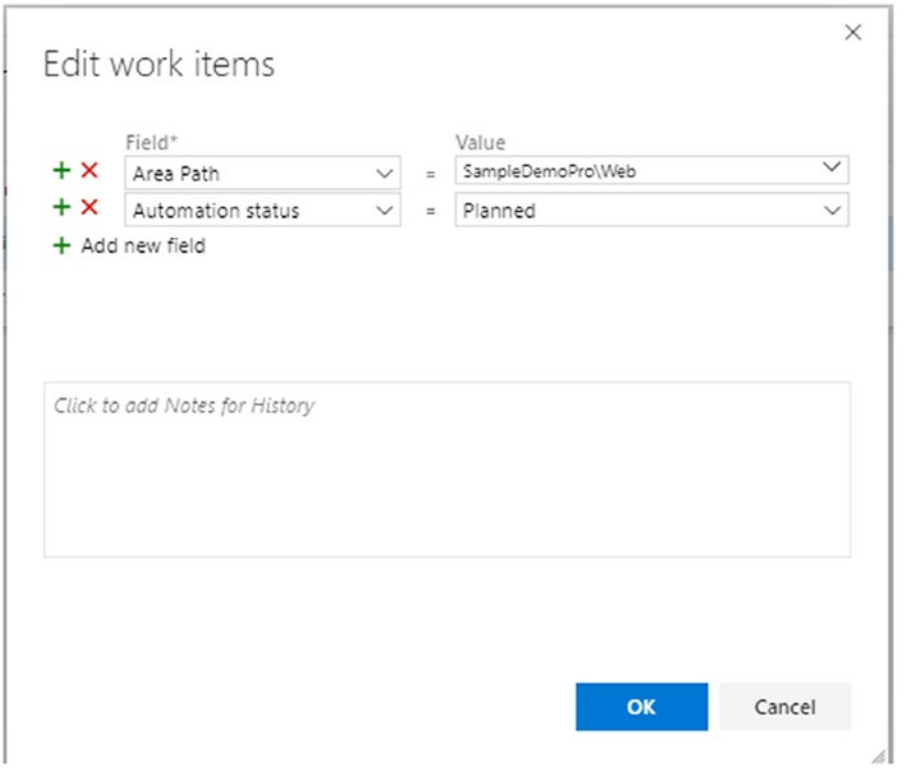


Figure 5-16. *"Edit work items" screen*

- *Copy test case(s)*: This copies the test case.
- *Export test case(s) to csv*: This allows you to export the selected test cases to CSV.

Execute Tab

Once the test plan with a few test cases is ready, then navigate to the Execute tab to execute the test cases. The user can select one or more test cases and mark them as Pass Test, Fail Test, Block Test, and Not Applicable, as shown in Figure 5-17.

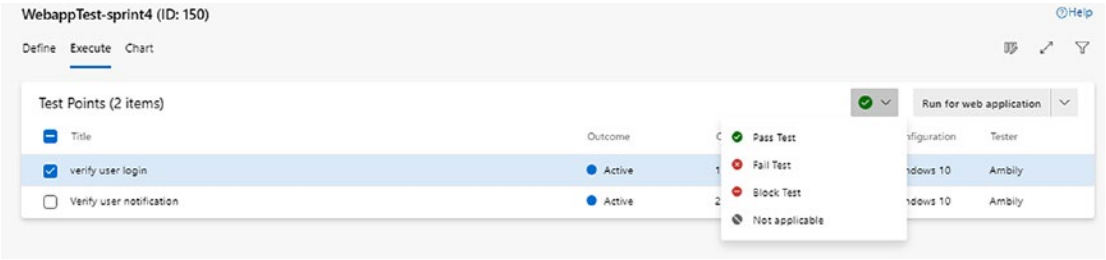


Figure 5-17. *Executing a test plan*

You'll see a list of run options available to execute the selected test cases. The run options available are as follows: Run for web application, Run for desktop application, and Run with options.

Also, the user can select other columns using the columns option ( ) available on top of the tests. There is another option to filter the list of test cases using various parameters, as in Figure 5-18.

Figure 5-18. *Filtering option*

Run for Web Application

Select one or more test cases and click the "Run for web application" option. This will open the Runner window with many options and a list of steps associated with the test cases, as shown in Figure 5-19.

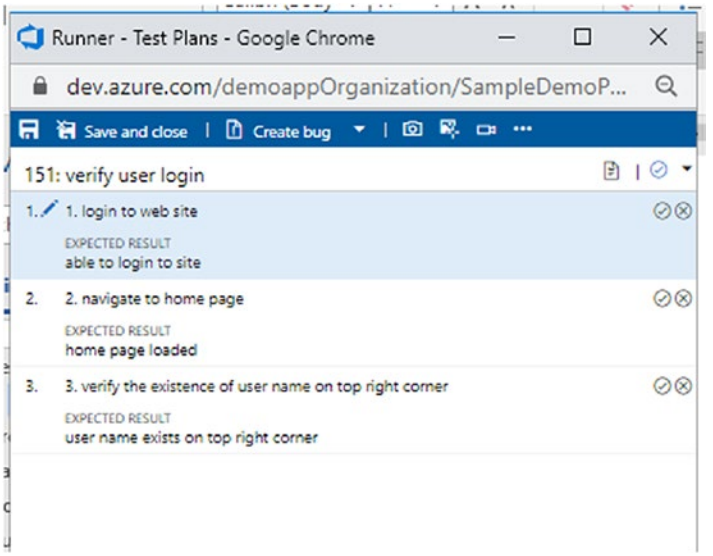


Figure 5-19. *Runner window, test plans*

The test steps provide three options: edit( ), Pass test step( ), and Fail test step ( ). Click the edit icon, which allows you to edit the step and the expected result associated with the test step, as shown in Figure 5-20.

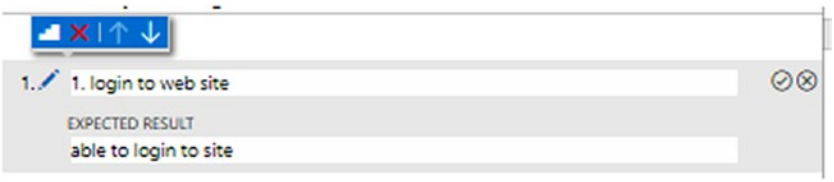


Figure 5-20. *Runner window, edit step*

The user will get further options to add new test steps, delete the existing step, and move down and up the selected step in the test case. Pass and Fail mark the test step status as part of this execution.

The options available near the test case title enables the user to mark the entire test case with pass, fail, pause, block, and not applicable, as in Figure 5-21.

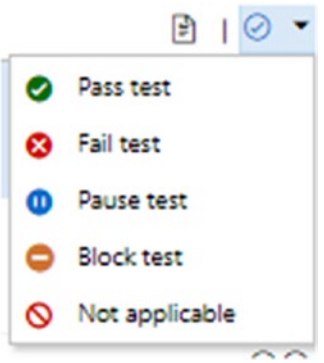


Figure 5-21. *Runner window, mark status*

The other options available on the top of the Runner window are as follows:

- *Save*: Save the changes.
- *Save and Close*: Save and close the Runner window.
- *Create new Bug/Add to existing bug*: From the Runner window, the user can create a bug related to the current test case execution.
- *Capture screenshot*: Capture the current screen to provide more details for bug fixing or feedback.
- *Capture user actions as image action log*: Capture all the user actions as image logs, which will help the team to reproduce the execution or defects.
- *Record screen*: Add a video recording of the execution of the test case, which will help provide further details about the user actions.
- *Add Comment*: Add comments as part of the test case execution.
- *Add attachment*: Add additional information as an attachment such as the log files.
- *Record screen with audio*: Add a screen recording with audio, where the user can explain the observations clearly.

These options help the QA engineers or testers to provide enough details to the development team to reproduce the defects. Before proceeding with screen recording or image capture, the user should install the Test & Feedback extension for Google Chrome, if the user is using Chrome for browsing the web application, as shown in Figure 5-22.

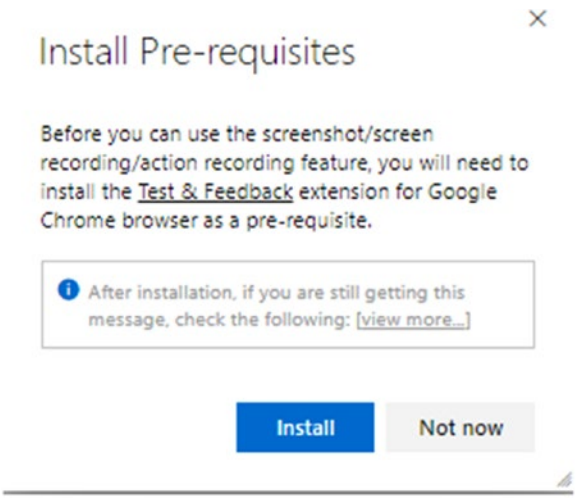


Figure 5-22. *Installing the Test & Feedback extension*

Once the installation completes, configure the Test & Feedback extension to connect with the Azure DevOps organization, as shown in Figure 5-23.

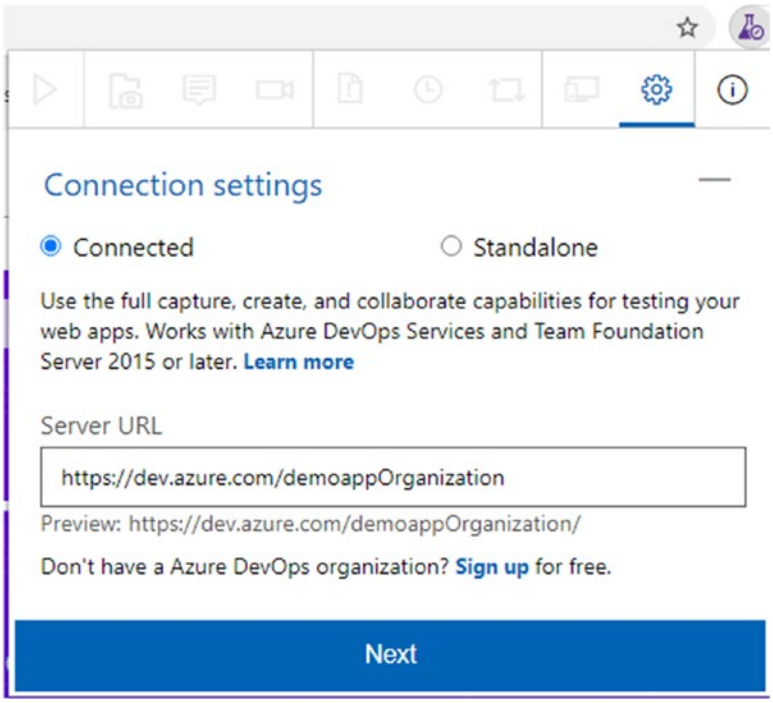


Figure 5-23. *Test & Feedback extension*

Once integrated with Azure DevOps, relaunch the Runner window to execute the test case with an image or screen recording.

The user can record separate operations against each step of the test cases. In Figure 5-24, the user has passed the first step and failed the second step and tries to create a new bug against the failed step.

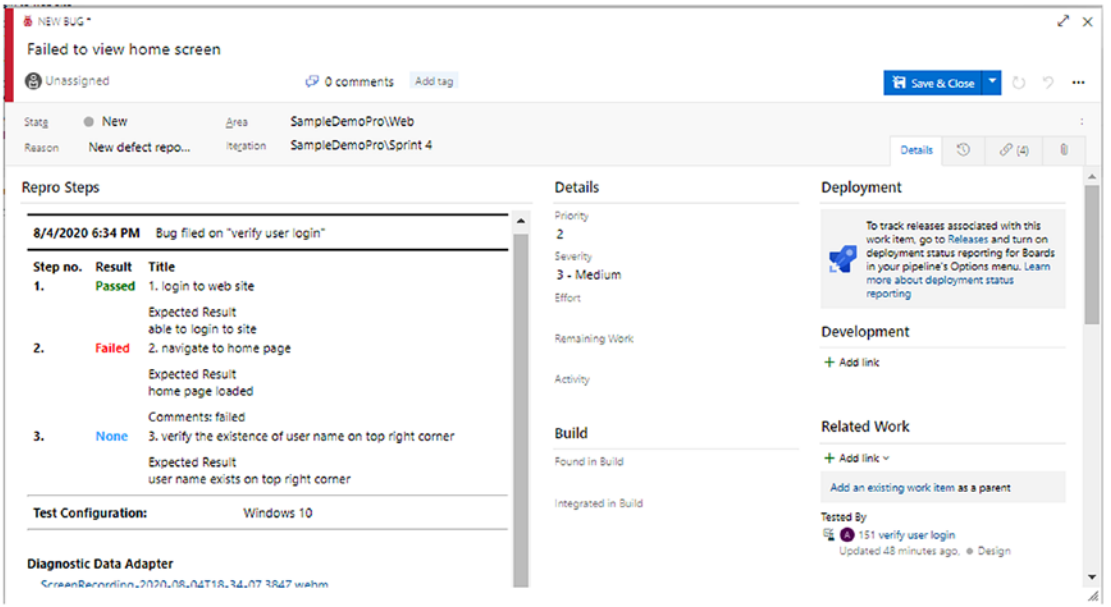


Figure 5-24. *Runner window, new bug*

The system automatically captures the steps, status of each test step, and video or image logs. Moreover, the system captures the current system info, which helps the developers to reproduce the issue in a similar setup. Figure 5-25 shows the sample system info captured.

System Info

Browser - Name	Google Chrome 84
Browser - Language	en-US
Browser - Height	666
Browser - Width	432
Browser - User agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
Operating system - Name	Windows NT 10.0; Win64; x64
Operating system - Architecture	x86_64
Operating system - Processor model	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz
Operating system - Number of processors	2
Memory - Available	2948272128
Memory - Capacity	8589463552
Display - Pixels per inch (X axis)	0
Display - Pixels per inch (Y axis)	0
Display - Height	768
Display - Width	1366
Display - Device pixel ratio	1

Figure 5-25. *System info captured*

All details related to the bug will be attached to the attachment tab of the bug, as shown in Figure 5-26.

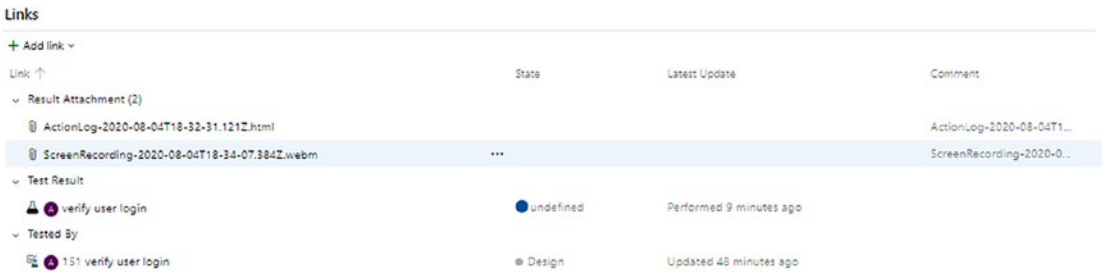


Figure 5-26. *Attachments*

Creating a bug based on test case execution provides enough information such as the steps followed, screen recording, image captured, comments, etc., to reproduce the defects in a similar setup.

The Test & Feedback extension will be used for exploratory testing, where the test cases are not defined in the system. Exploratory testing is the random testing of the system to identify the defects. This is explained in more detail in my article published in Simple Talk (https://www.red-gate.com/simple-talk/dotnet/net-development/exploratory-testing-chrome-plugin/).

The "Test Cases" section discussed the test cases, shared test cases, and parameterized test case and showed an example test case to verify the user login. This test case has parameters associated with it. When such test cases with parameters get executed, it prompts the execution of the test case steps for each of the parameter values. The test case execution for each parameter set will be added as an iteration in the test runner. In Figure 5-27, the parameter set contains three sets of values, indicating three iterations.

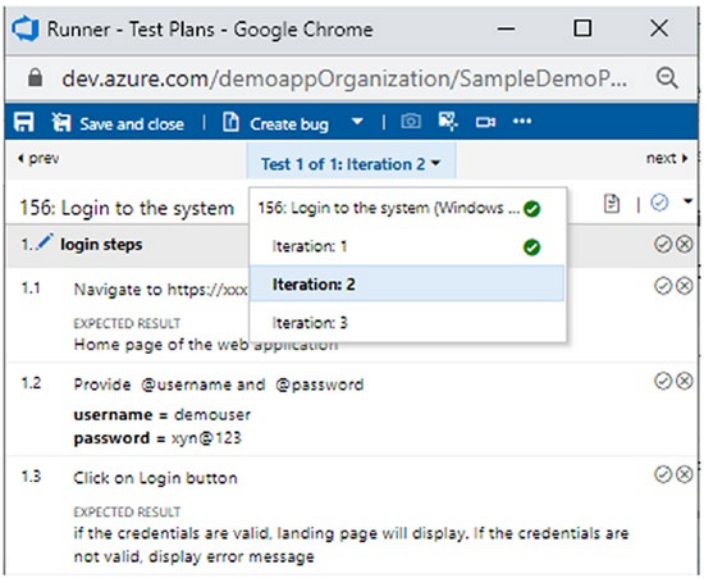


Figure 5-27. *Runner window, multiple iterations*

Run for Desktop Application

This option supports the user to run the test cases against the desktop application. Select the "Run for desktop application" option to run one or more test cases associated with a desktop application. If this is the first time you are executing testcases with "Run for desktop application" option, the system will prompt you to download and install Test Runner before executing the test cases, as shown in Figure 5-28.

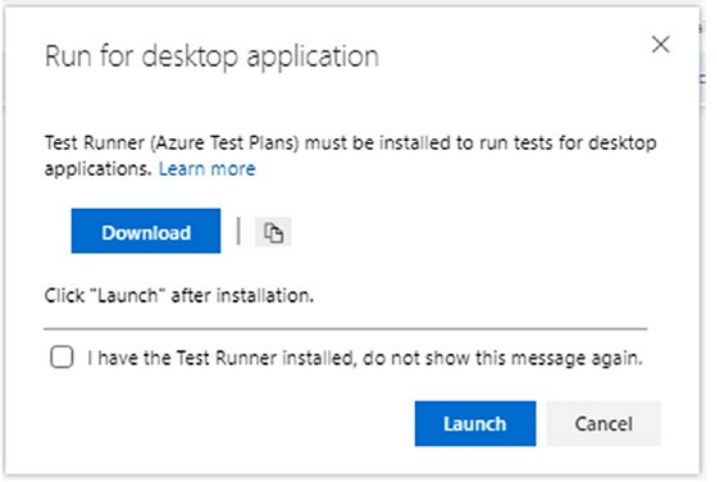


Figure 5-28. *Desktop runner*

Download and install Test Runner. Any new test case run against the desktop application will launch the desktop-based Test Runner and capture the desktop environment details while running an application installed in the system.

Run with Options

The "Run with options" dialog lists the options for executing the test cases based on various existing tools. Based on the selection of test type and runner, the actions will change. For example, if you are planning to execute an automated test case using a release stage, then the required input data will be something like shown in Figure 5-29.

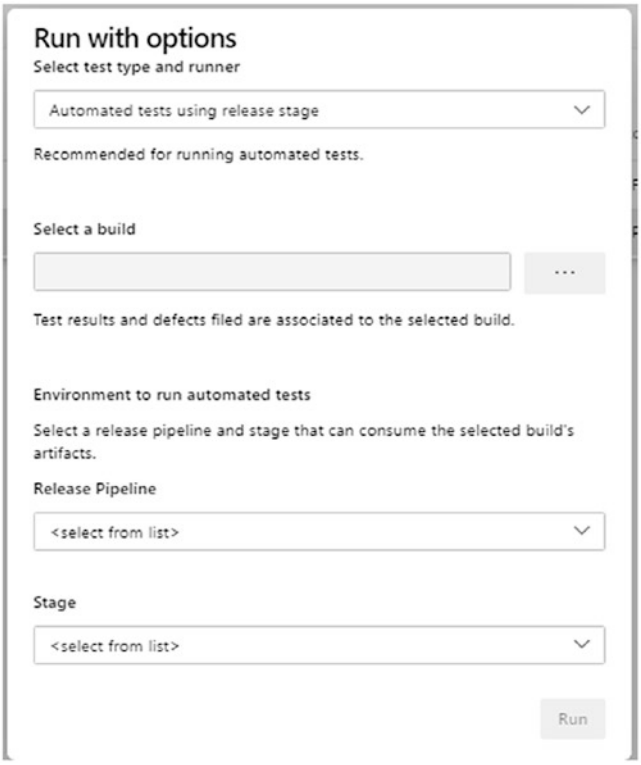


Figure 5-29. *"Run with options" window*

The different options available for the test type and runner are as follows:

- Manual tests using browser-based runner
- Manual tests using test runner client
- Automated tests using release stage
- Manual tests using Microsoft Test Manager 2017 client
- Manual tests using Microsoft Test Manager 2015 or earlier client

There are other options available against each of the test cases, as shown in Figure 5-30.

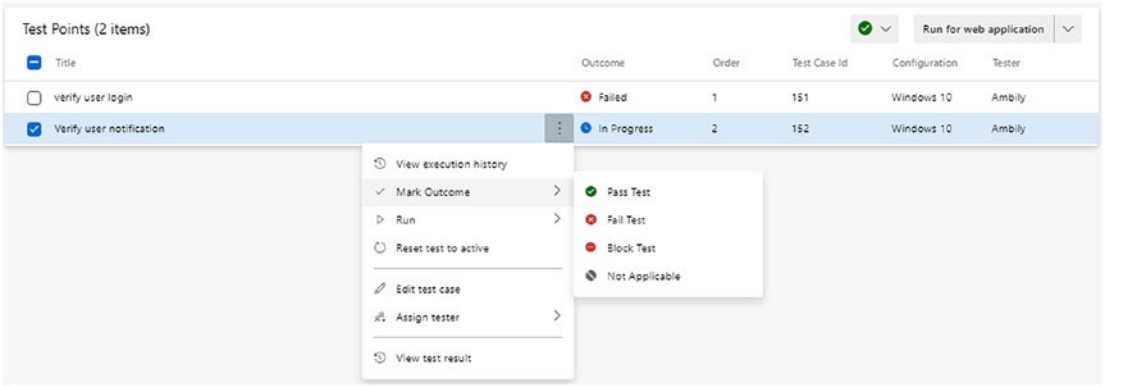


Figure 5-30. *Test case, options*

The options are as follows:

- *View execution history*: This shows the execution history of the test case.
- *Mark Outcome*: This marks the outcome as pass, fail, block, etc.
- *Run*: This runs the test cases using any of the previously mentioned three options.
- *Reset test to active*: This resets the test to active mode.
- *Edit test case*: This edits the test case.
- *Assign tester*: This assigns a tester.
- *View test result*: This views test results.

Chart

The Chart tab provides two different options to create charts based on test cases and results.

- *New test case chart*: Select this option to create a new chart based on the test cases grouped under this test plan, as shown in Figure 5-31. The user can select a specific chart type and associated parameters to define a chart that brings some insight.

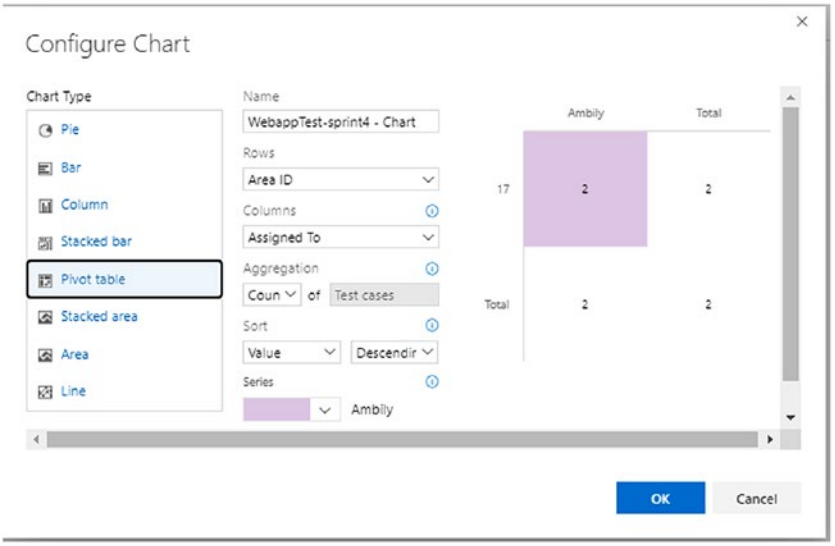


Figure 5-31. *New test case chart*

- *New test result chart*: This option supports the creation of a new chart based on the test case results, as shown in Figure 5-32.

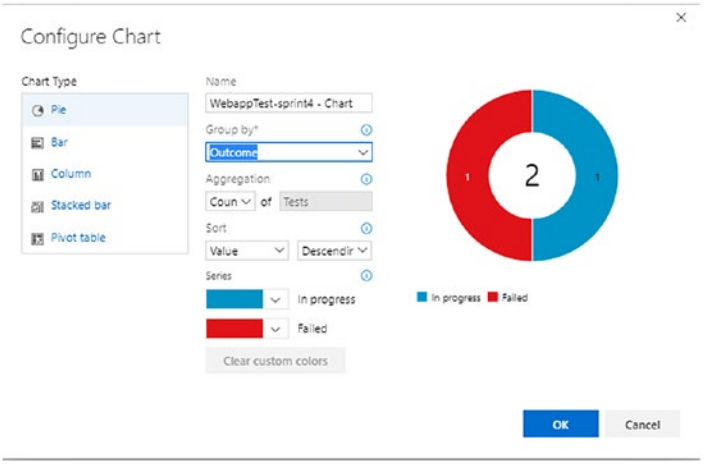


Figure 5-32. *New test result chart*

In this case, there are fewer chart types available compared to the test case charts. Moreover, the "Group by" parameter will be based on the test run and result only.

Once charts are added, they can be added to dashboard. The "Add to dashboard" option helps the user add the specified chart to the team dashboard to bring the insights to the dashboard, as shown in Figure 5-33.

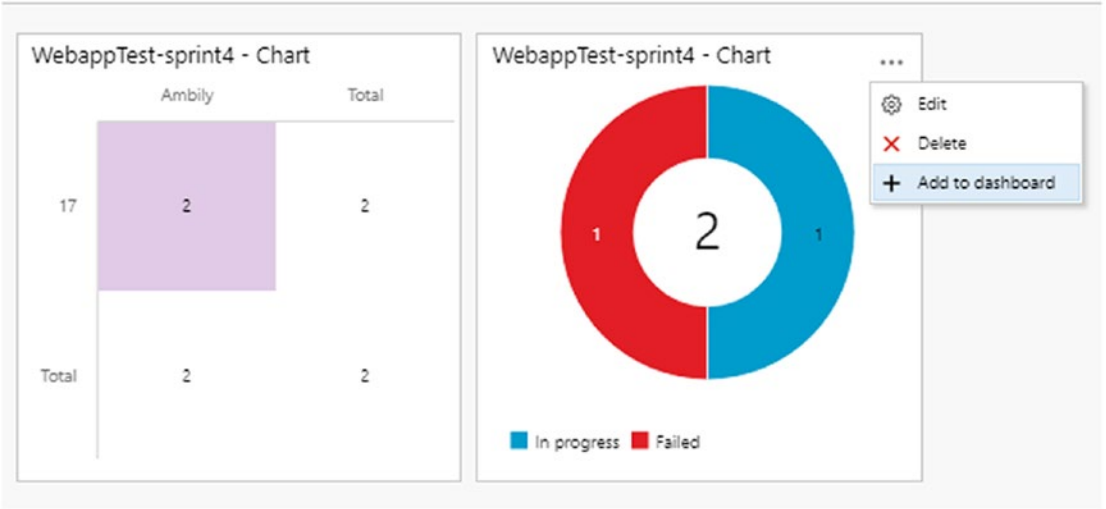


Figure 5-33. *"Add to dashboard" option*

The "Test plan settings" window, shown in Figure 5-34, is where you can set the test run settings and test outcome settings.

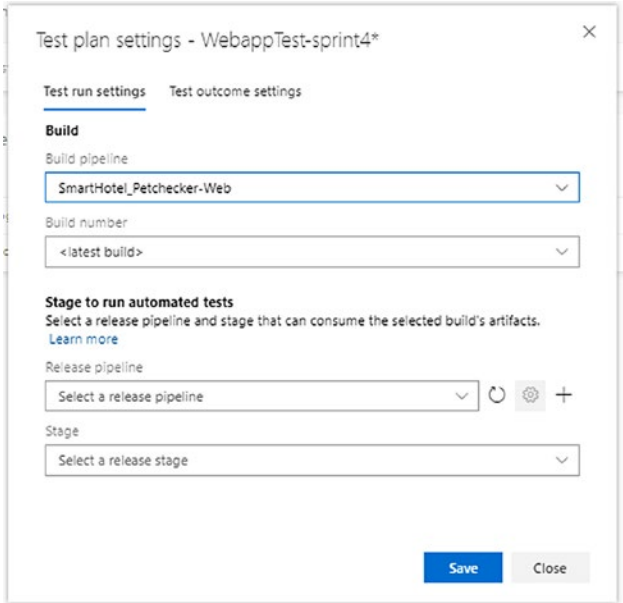


Figure 5-34. *Test plan settings*

Progress Report

This section shows the test plan, suite, and test case execution and results in a dashboard. The user can filter a report based on various criteria such as the test suite, date range, tester, etc. Figure 5-35 shows a sample progress report.

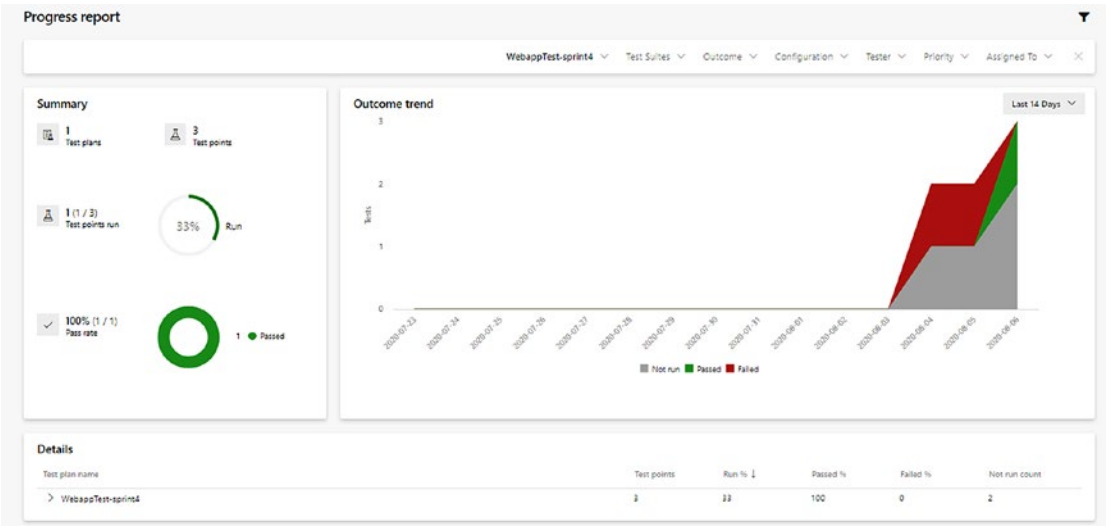


Figure 5-35. *Progress report*

Parameters

The Parameters section shows the shared parameters defined across the project along with their values and related test cases. Figure 5-36 shows the parameters defined in earlier sections.

Figure 5-36. *Parameters*

The user will be able to add new shared parameters along with the values. The user can turn on or off the related test case view.

Configurations

The Configurations section deals with the different test environment configurations and associated configuration variables.

- *Test configuration*: This defines a set of configurations that can be used for executing a particular test case. For example, when the user tests the platform, the configuration will contain the supported OS such as Windows, Mac, or Linux along with other configuration variables.
- *Configuration variables*: This is used for defining the variables, which can be configured for test configurations, as shown in Figure 5-37.

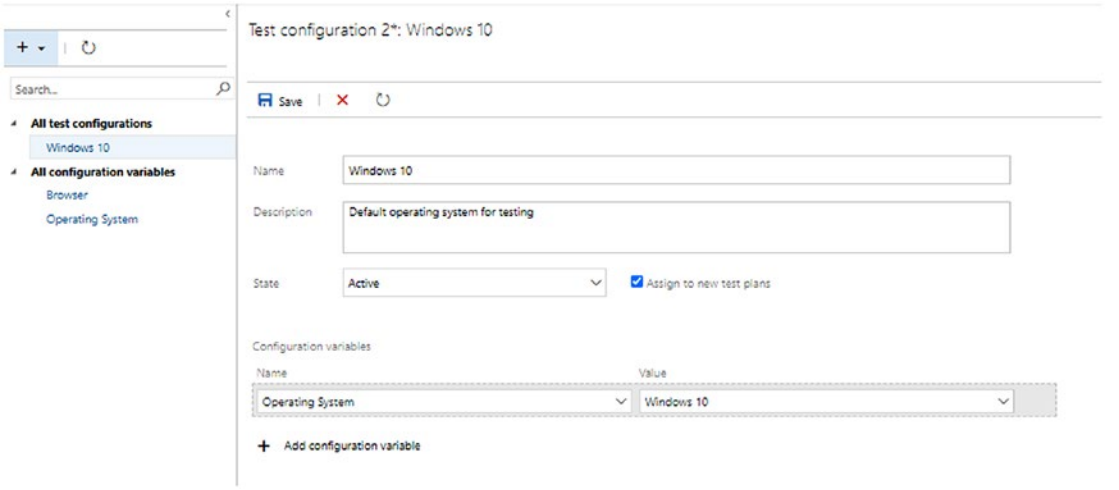


Figure 5-37. *Test configurations*

Runs

The Runs section shows the list of recent runs along with run result, as shown in Figure 5-38. The user can select a specific run using the run ID or using the Filter tab.

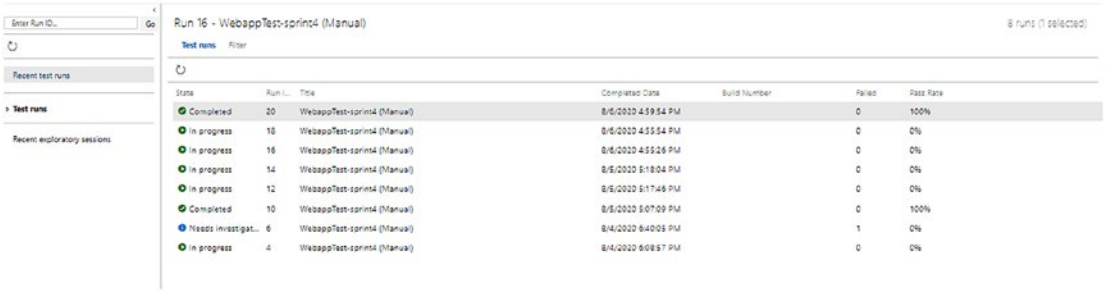


Figure 5-38. *Test runs*

Also, the user can select one run by double-clicking the run's row and view the complete details associated with it, as shown in Figure 5-39.

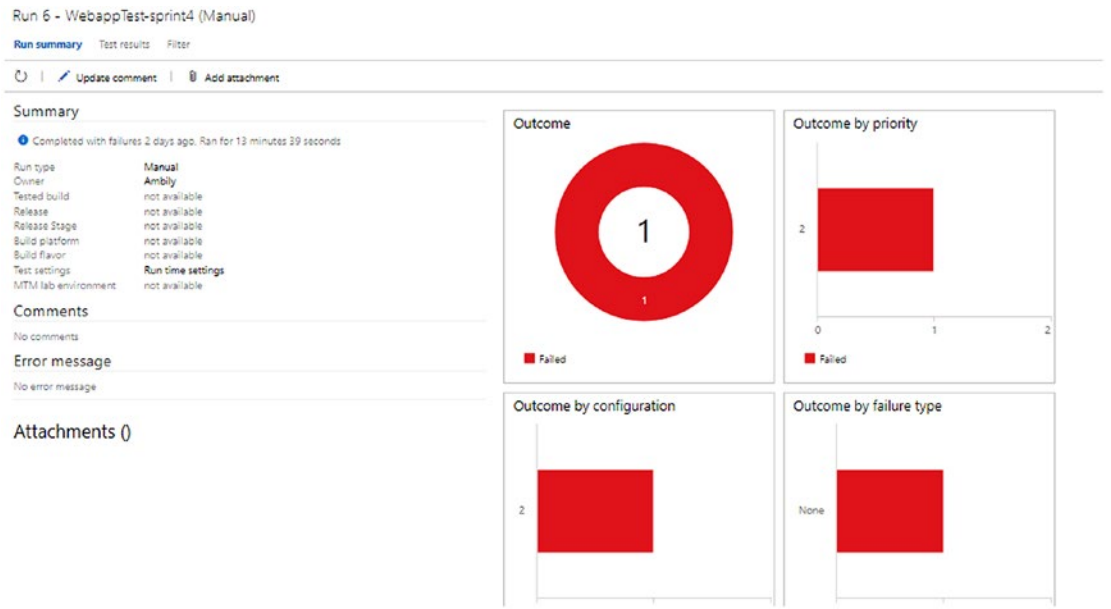


Figure 5-39. *Run details*

Load Test

The load testing feature allows you to conduct performance testing on an application. Understanding end-to-end load testing requires a good understanding of performance testing. The load testing feature in Visual Studio and in Azure DevOps is being deprecated; please refer to the Microsoft blog at `https://devblogs.microsoft.com/devops/cloud-based-load-testing-service-eol/` for more details.

You can also refer to my book on Amazon.

- *Kindle edition*: https://www.amazon.com/Dialogue-Visual-Studio-Performance-Testing-ebook/dp/B01LYSDNCN
- *Paperback*: https://www.amazon.com/Performance-Testing-Visual-Studio-product/dp/1522000712

Summary

Azure DevOps provides a set of test features to do the end-to-end test management. Test plans, test suites, and test cases are the fundamental elements for test management along with other supporting features. Moreover, the integrated platform provides an end-to-end integration between test cases and requirements, implementations, defects, and other artifacts, which enables full traceability.

CHAPTER 6

Build Automation and Release Management

Build automation and release management are the two core concepts in a DevOps implementation. Build automation involves the automation of not only the build process but also the integration of many other verifications such as unit testing, code coverage, code analysis, security verification, and so on. Doing all these verifications along with a proper build takes lots of manual effort otherwise. Similarly, release and deployment management involves many concepts such as the deployment to various environments, approval flows, pre- and post-conditions, verification gates, infrastructure provisioning, and so on. This chapter explains the build automation and release management processes in detail.

Build and Release Process

In DevOps, different build and release processes are involved and adopted based on the maturity of the DevOps process in the company. In general, there are two processes commonly used in DevOps: continuous integration (CI) and continuous deployment (CD).

Continuous Integration

The build or build automation of an application means the preparation of a deployable codebase. This differs from program to program. In the case of a desktop application, this may be the generation of a setup file or executable (EXE) file. When it comes to an API or web application, it involves the packaging of all dependencies, the compiled codebase, and the associated artifacts such as resource files, images, etc.

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_6

Preparing the build manually makes it error-prone. Before DevOps, the development, testing, and operations teams worked in silos. The activities of each team were isolated and not in sync, which impacted the overall delivery and quality of application.

Automated builds are one of the essential steps in the overall DevOps process. Builds are automated and configured to run manually or scheduled in the different phases of the DevOps process. Through the Pipeline menu, build definitions are created. The most used build definition types are as follows:

- *Gated check-in build*: In this type, builds get executed as per the branch policy setups and as code is pushed to that branch. These builds help you verify the quality of the code pushed to the origin branch. If the build succeeds, then the commit will be successful, and the changes will be available for other team members. If the build fails, the commit will automatically roll back. This kind of build helps in safe-guarding the code in distributed repositories; code in the distributed repo will always be good quality. There are different names for this build such as *commit verification build*, etc. One major point to note is that gated check-in builds do not create any artifacts or drop folders for deployment.
- *Scheduled builds*: In this type, builds are scheduled for a specific time frame. For example, sprint builds, which gets triggered at the end of the sprint, can be a scheduled build. Scheduled builds ensure a release pattern to different environments like dev, qa, staging and production.
- *Continuous integration builds*: This kind of build triggers when a commit successfully merges a remote origin branch to a specific branch. For example, a build administrator can configure a CI build for the dev branch to execute the build and run unit test cases associated with the application. This will ensure that the automated unit test scripts are validated in every commit. Gated check-in builds help in keeping the remote repo intact and create an executable codebase, whereas CI builds ensure the quality of code in every commit.

Not only are unit test scripts integrated as part of the build pipeline, other tools and utilities such as code analysis tools (SonarQube, Code analysis, etc.) are integrated as well, as shown in Figure 6-1, to ensure the code completion, quality, and so on.

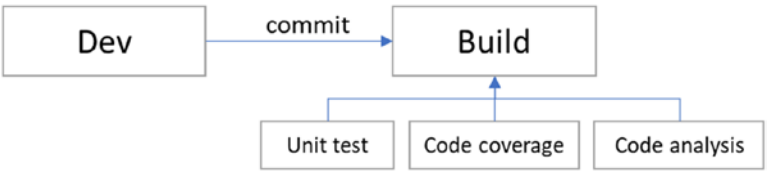


Figure 6-1. *Build process*

Continuous Delivery and Continuous Deployment

Once the build is ready, it should be deployed to an environment to verify its functionality. If there is an automated pipeline set up from the development environment all the way to production deployment, then we call it *continuous deployment*. In continuous delivery, anyone after completing the due checks or after completing the necessary communication can deploy and monitor a new release with a few clicks.

As part of the pipeline, many tools and technologies are integrated to ensure the application stability, functionality, and so on, as depicted in Figure 6-2. Deployment to each of the environments can be controlled through different quality gates and approval flows, which we will discuss in detail in subsequent sections. The following sections discuss some of the commonly used deployment models.

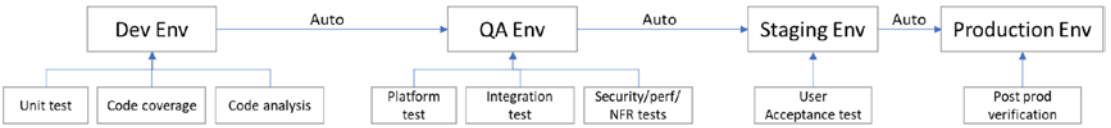


Figure 6-2. *Release process*

Blue/Green Deployment

In *blue/green deployment*, there are two identical production environments. At any one time, one will be active, and the other will not. New changes will be deployed to the second environment and tested. Once the validations and verifications are complete, you can change the router to the second environment. Now the first environment will be in an inactive state. Figure 6-3 shows this deployment.

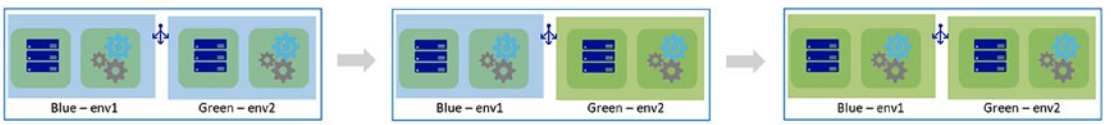


Figure 6-3. *Blue/green deployment*

Here are the advantages:

- Staggered deployment across multiple environments
- Reduced or zero downtime

Here are the challenges:

- Backward compatibility of the apps
- Increase in operational overhead
- Updating database schema requires caution

Canary Deployment

In *canary deployment*, new changes will be released to a subset of users, tested, and then deployed to everyone. This model helps incorporate early feedback from end users and identifies the feature issues before the final release. Figure 6-4 illustrates canary deployment.

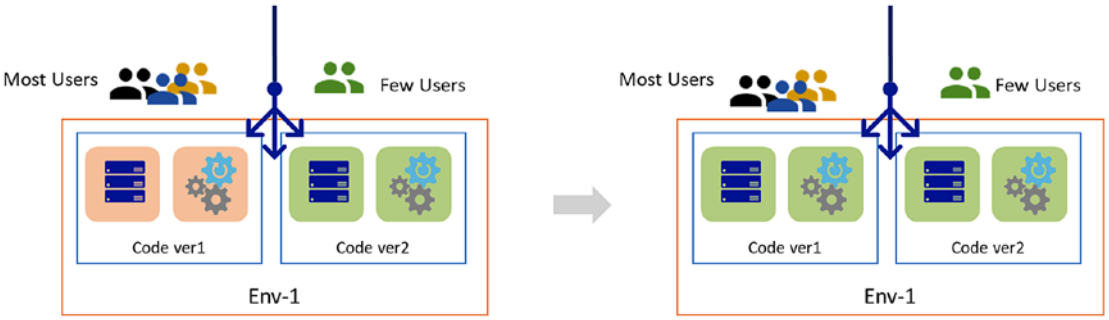


Figure 6-4. *Canary deployment*

Here are the advantages:

- Releases to a subset of users or servers
- Quick rollback with minimum impact
- Feature toggling or server options

Here are the challenges:

- Routing decisions

Rolling Deployment

In a *rolling deployment*, instead of deploying to all servers at once, new releases will be deployed to one server at a time; in other words, you slowly roll out the deployments server by server. Figure 6-5 depicts a rolling deployment.

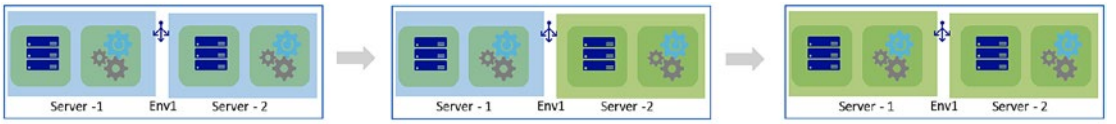


Figure 6-5. *Rolling deployment*

Here are the advantages:

- Reduced downtime
- Deploy to one server at a time

Here are the challenges:

- Backward compatibility of the apps
- Multiple versions co-exists in production

Pipelines

Pipelines define the build and release processes and associated integrations.

Creating a Pipeline

A pipeline is defined using four steps: connect, select, configure, and review.

The connect step helps in connecting to the repo in Azure DevOps or to an external repo, as shown in Figure 6-6. By default, pipelines use a YAML-based build setup. If the user is familiar with a task or activity-based setup, the user can click the link displayed at the bottom to shift to the classic editor. (The classic build steps will be explained after the YAML-based build configuration.)

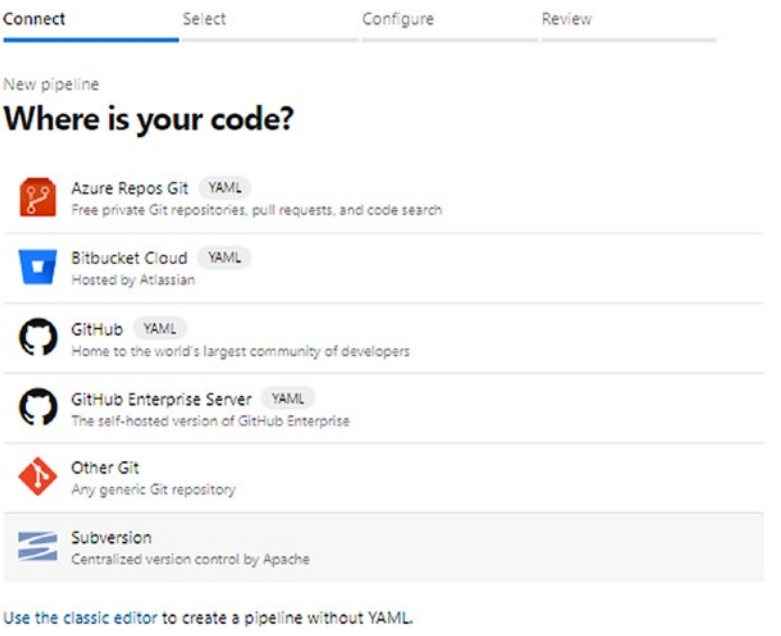


Figure 6-6. *Connect step*

Note that YAML is a human-readable data serialization language. It is mainly used for configuration files such as Docker files. It is one of the standards followed by lots of DevOps tools, and an understanding of YAML will help in handling multiple DevOps aspects such as infrastructure as a code.

If you select a repo other than Azure Repos Git, the Azure DevOps pipeline process will redirect you to the login page of the corresponding integration to proceed. Once the Azure Repos Git is selected, the pipeline process will list the existing repos available in the project, as shown in Figure 6-7.

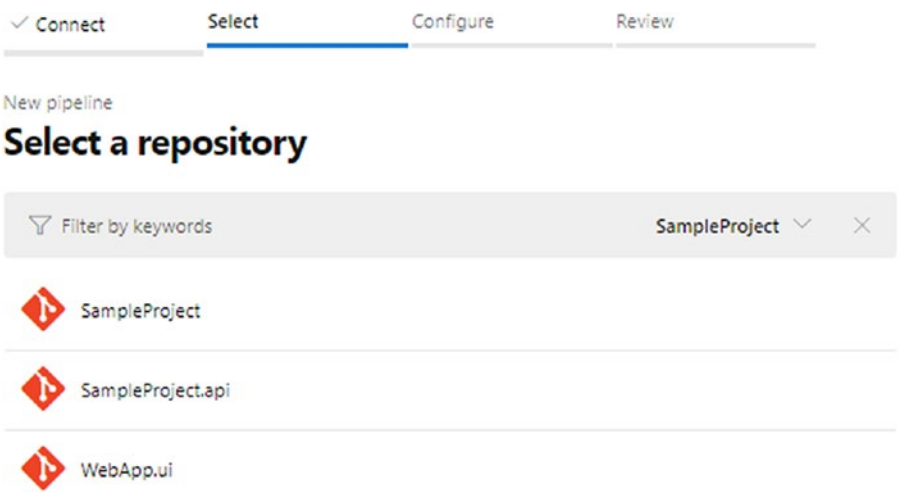


Figure 6-7. *Select step*

Select the repo for which the pipeline is getting configured and proceed to configure it. Select one of the predefined YAML templates listed in Figure 6-8 to build the application.

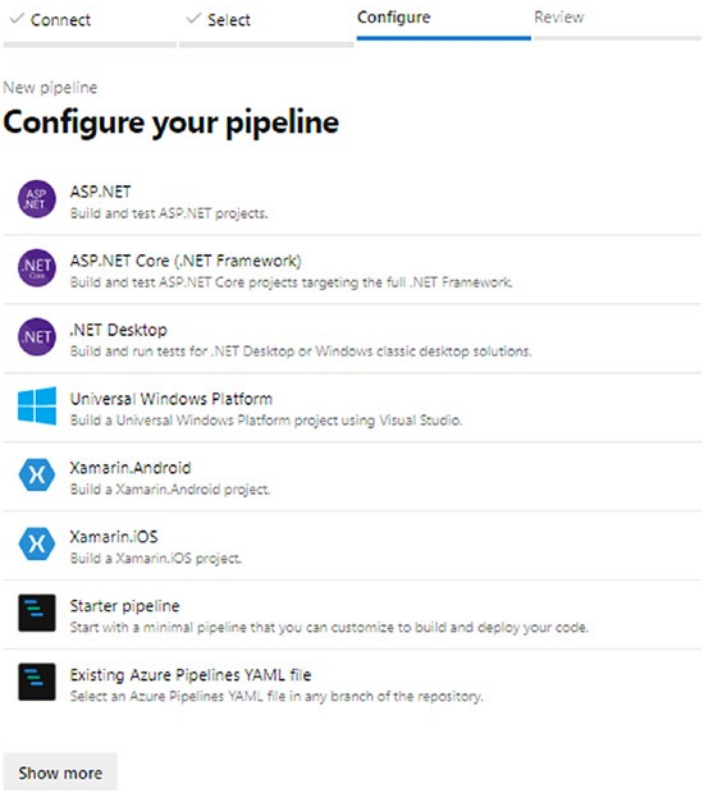


Figure 6-8. *Configure step*

Review the selected details such as the repo, branch, technology stack, etc., in YAML form and complete the pipeline, as shown in Figure 6-9.

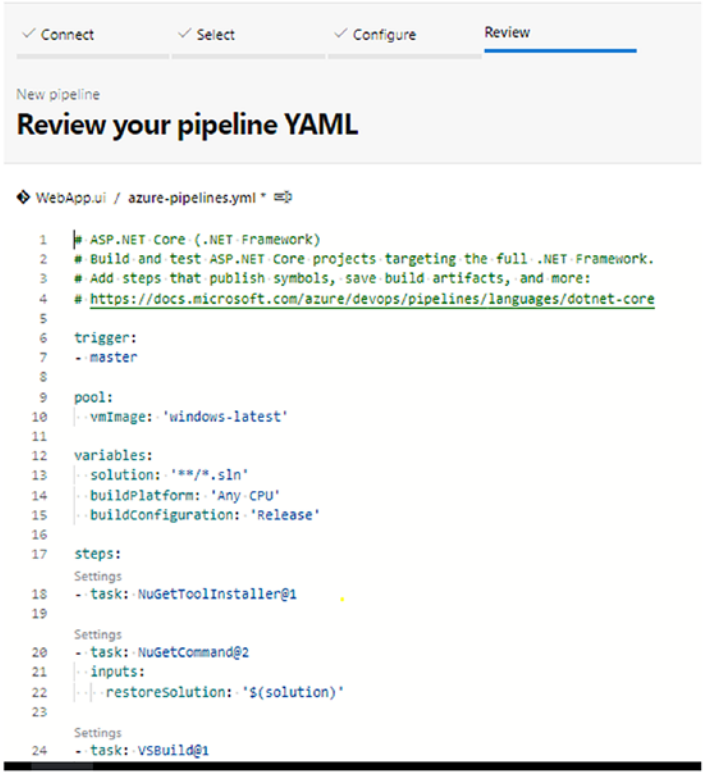


Figure 6-9. *Review step*

Variables

Before completing the pipeline setup, we can define a set of variables required to configure the build. Variables help in handling the dynamic or environment-dependent configurations or parameters. For example, in Figure 6-10, *branch* is defined as a variable with the default value as *master*, which is a secret, and users are allowed to provide the value while running the pipeline.

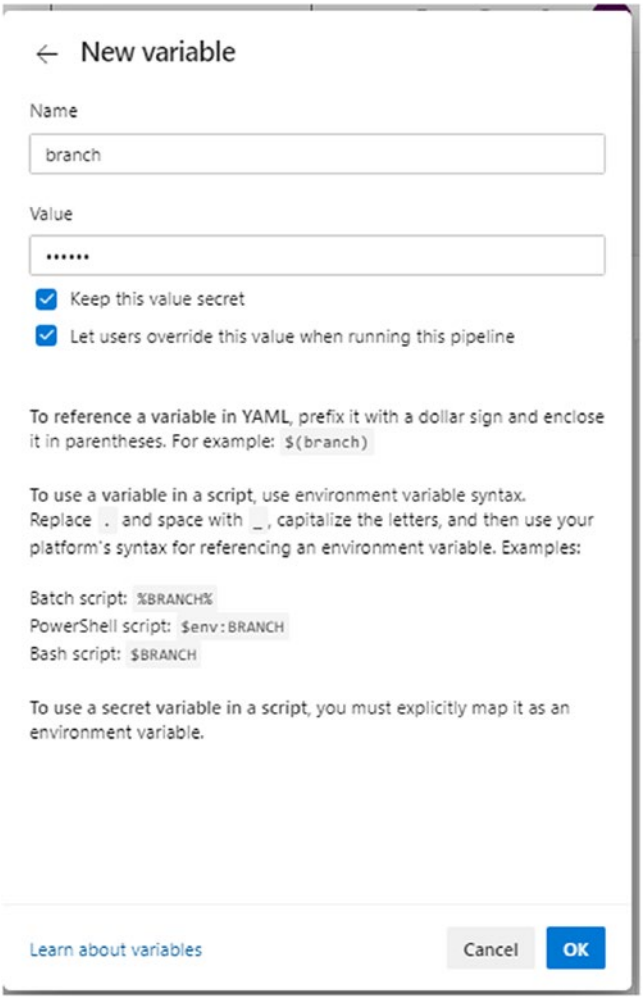


Figure 6-10. *Variables*

Variables can be inserted into the YAML or ARM template or into other build areas. If it is YAML, variable is referred using the `$(variable)` syntax, as shown in Figure 6-11.

```
trigger:
- $(branch)
```

Figure 6-11. *Variable usage*

Moreover, the YAML review screen provides an option to add tasks using the "Show assistant" link available on the right side. Once you click the link, you will see a list of steps available for integrating with the current YAML flow. We'll cover tasks in more detail in the "Classic Editor" section.

Settings and Triggers

The top-right corner menu shows the other options associated with the YAML build. The user can select Settings, as shown in Figure 6-12, to enable the new builds and automatic linking of work item options.

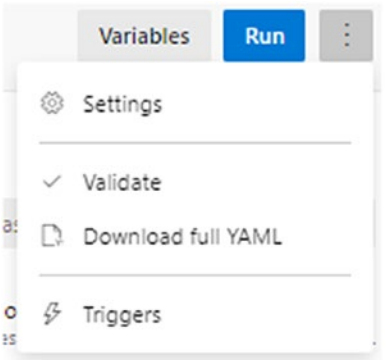


Figure 6-12. *Settings menu*

The Validate option validates the YAML build script. The "Download full YAML" option allows you to download and distribute the build definition. The Triggers option helps in setting up various build triggers such as CI, Scheduled, and Build completion, as shown in Figure 6-13. These triggers, along with various available options, are explained in the "Classic Editor" section.

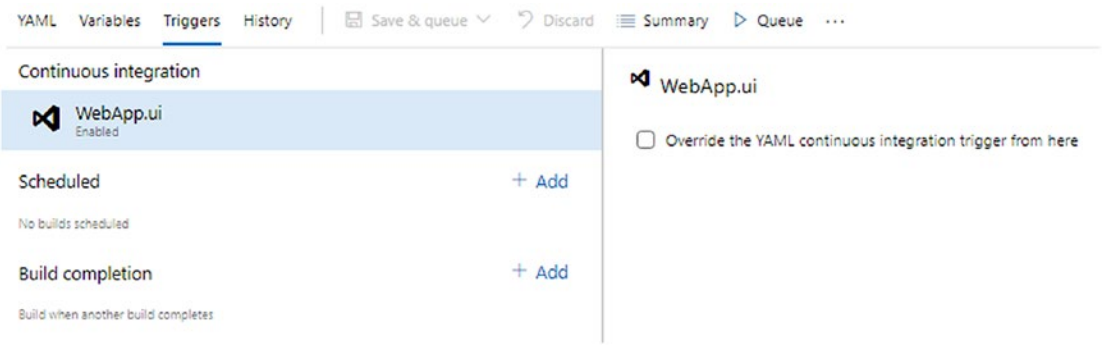


Figure 6-13. *Triggers*

Once the pipeline YAML modification completes, select the Save and Run option to queue the build. The user can observe the progress of the build and logs in the job details windows, as shown in Figure 6-14.

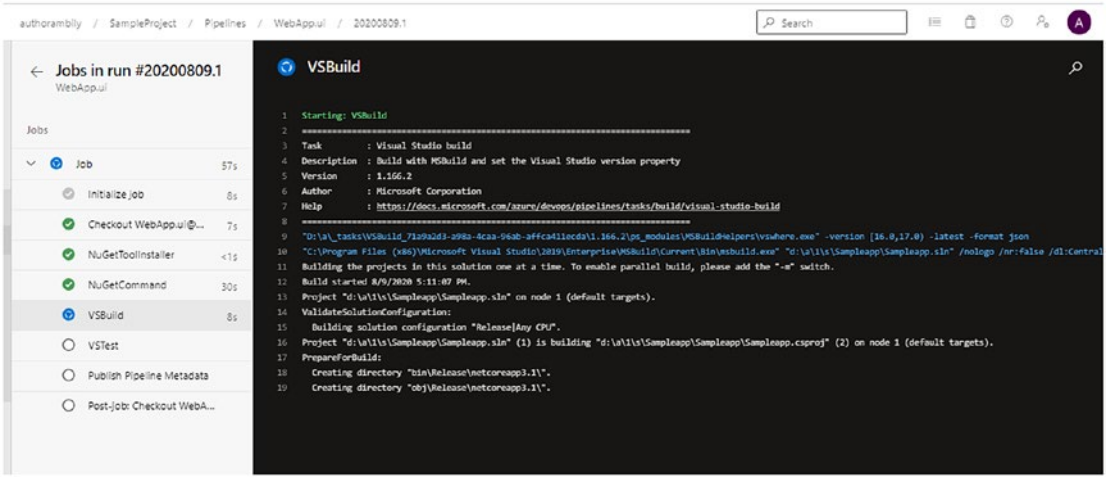


Figure 6-14. *Execution*

Classic Editor

The classic editor starts with source selection, as shown in Figure 6-15, along with the repo and branch.

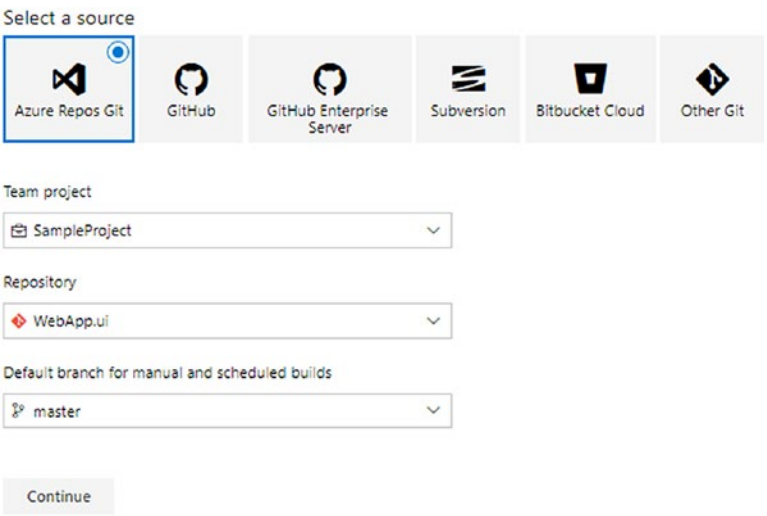


Figure 6-15. *Source selection*

Once the source, repo, and branch are selected, continue to select the build template. Either select one of the predefined templates, as shown in Figure 6-16, or start with an empty job.

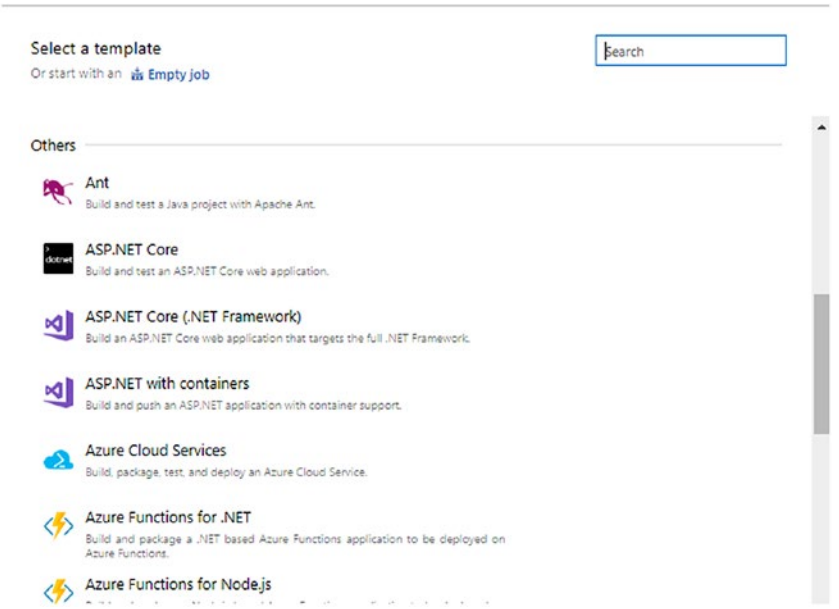


Figure 6-16. *Templates*

This will complete the pipeline creation and take the build engineer to the details of the pipeline. The user can configure all the build-related activities from here, as shown in Figure 6-17.

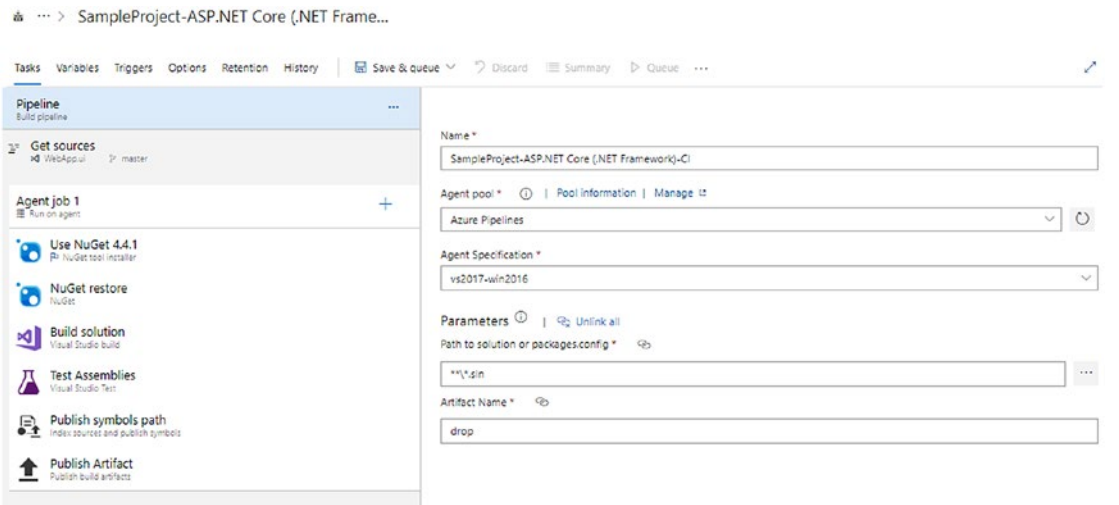


Figure 6-17. *Configuring all the build-related activities*

Tasks

The Tasks tab is the first tab in the build pipeline definition, where all the selected tasks are listed along with an option to add new tasks. In Figure 6-17, the user has selected an ASP.NET Core build template, which consists of a list of tasks to get the latest source, restore NuGet packages, build the solution, test the assemblies, publish symbols, and publish artifacts. The user can add or remove some of these steps. Click the Add icon for the agent job to add a new task to the list.

The user can select a step and drag it to rearrange the order of the steps in the pipeline. Moreover, the system provides the option to view the link settings, get a YAML view of the step, and remove the selected steps, as shown in Figure 6-18.

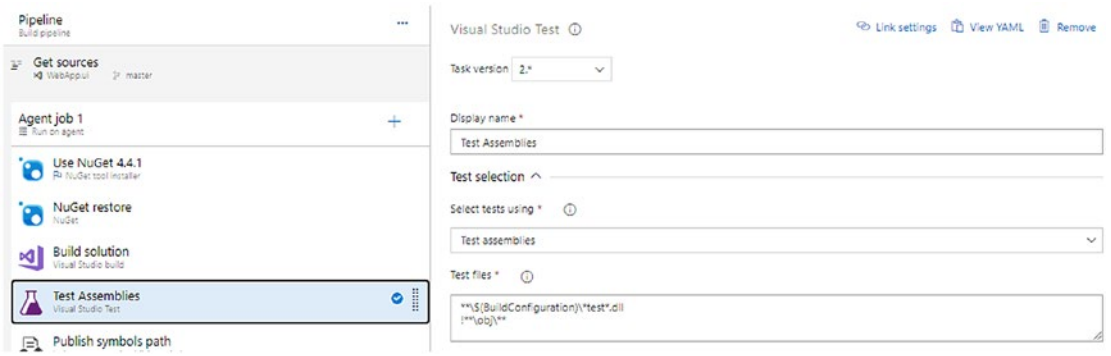


Figure 6-18. *Editing tasks*

In addition to task management, you'll see the options "Add an agent job" and "Add an agentless job" in the top corner of the Tasks tab, as shown in Figure 6-19.

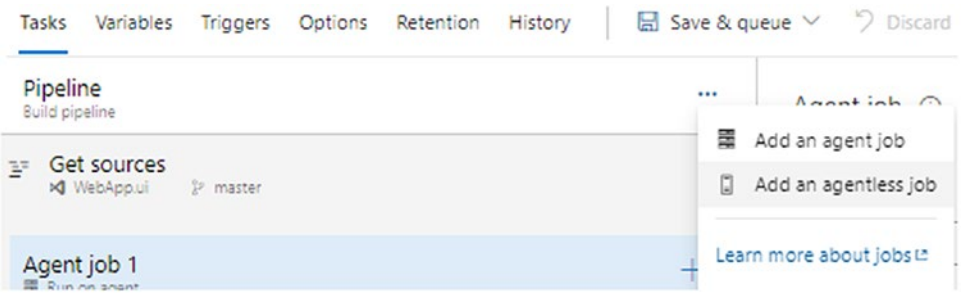


Figure 6-19. *Agent job options*

- *Agent job*: These are jobs running on an agent under the agent pool like building a .NET project, installing npm packages, and so on.
- *Agentless job*: These are jobs running in the server or Azure DevOps pipeline such as delaying execution, invoking Azure functions, querying work items, and so on.

Variables

The Variables tab shows the parameters or environment variables, as shown in Figure 6-20. Build engineers can define variables based on the environment requirements and mark them as *secret* and settable or mark them as configured at release runtime as *queue-time* properties.

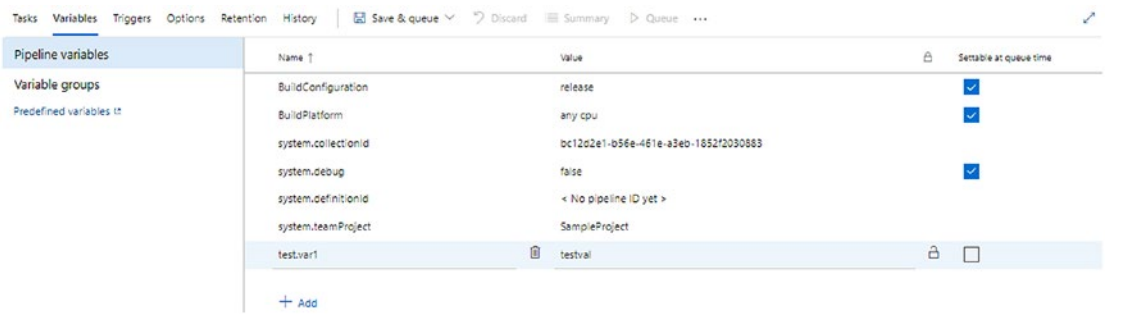


Figure 6-20. *Variables tab*

Variables can be grouped into different categories to manage them, such as environment-specific groups or build-specific groups. We'll learn more about variable groups in the "Library" section.

Triggers

The Triggers tab provides options to run the build or pipeline, as shown in Figure 6-21. By default, nothing will be selected under Triggers. Build engineers can configure the continuous integration build by selecting the option "Enable continuous integration." The system provides another option, "Batch changes while a build is in progress," to control the number of CI builds. If a build takes more time, then all the commits in the execution duration will be batched and triggered as a single build.

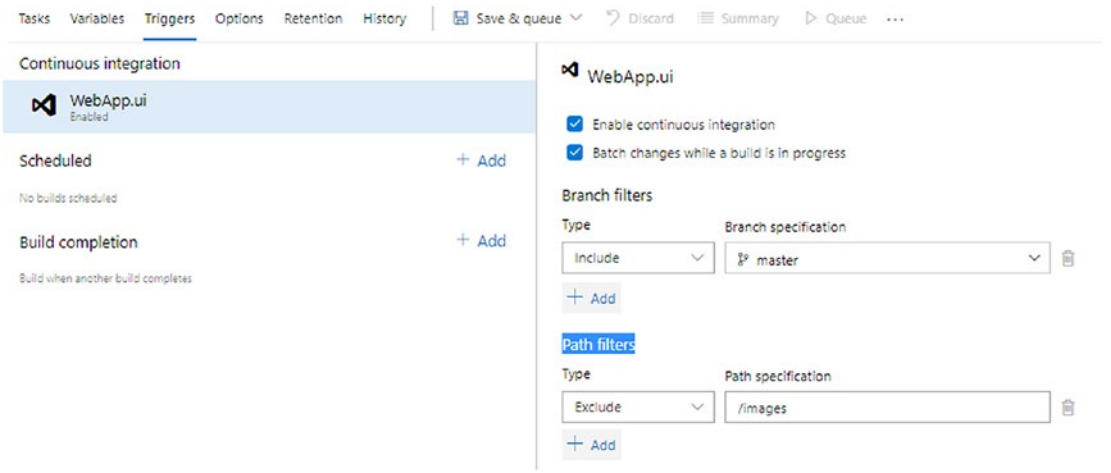


Figure 6-21. *Triggers tab*

In the CI build, the build engineer can include and exclude branches as well as specific paths. In the previous configuration, the master branch is included in the build. Also, the path `/images` is excluded. This indicates that this build will get triggered on any modification on the master branch except under the path `/images`. Any changes in `/images` will not trigger the build.

The build engineer can add a scheduled build by clicking the Add link for the build. One can configure the days, time, and branch filters as part of the scheduled build, as shown in Figure 6-22. Moreover, using the option "Only schedule builds if the source or pipeline has changed," one can control the build to be fired only if there are any changes after the last build.

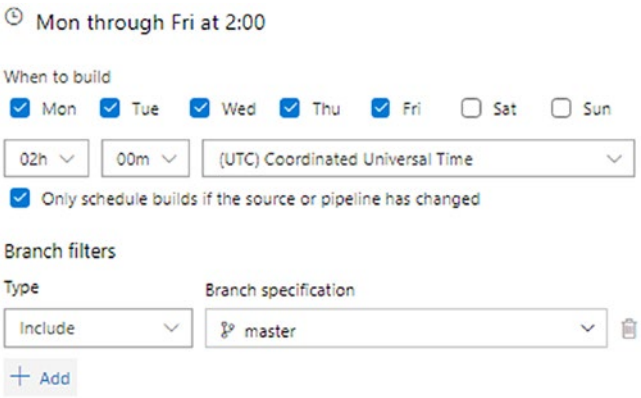


Figure 6-22. *Creating a schedule*

Another option available is to execute the build as a continuation of another build. If the previous build is successful, then execute only this build. This option helps in defining the related builds; for example, the UI build should be followed by all the API builds or the API build will be executed after the Database build, as shown in Figure 6-23.

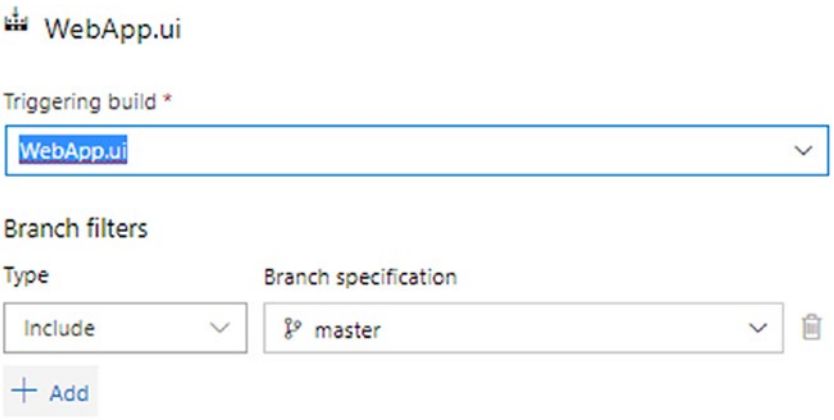


Figure 6-23. *Triggering the build*

Options

The Options tab lists the options associated with a build such as the build job timeout, build number format, linking work items with build, and so on, as listed in Figure 6-24.

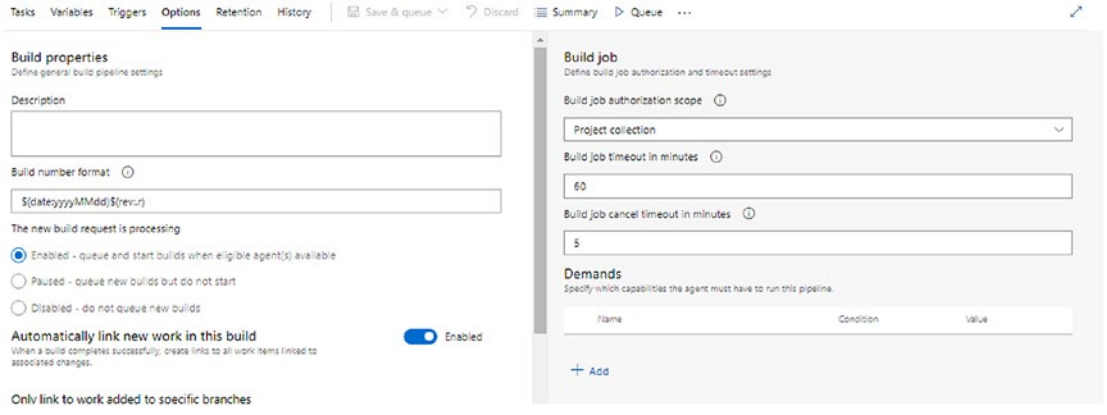


Figure 6-24. *Options*

The next tab, Retention, handled retention policies like the build result retention period in previous releases of the tool, but now this tab is centralized to the project settings. The History tab shows all the changes that have been made to the pipeline. The Summary tab shows the run summary of the build.

In More Options (…), the Security option plays an important role in controlling the build security. The build engineer can configure the security to ensure only authorized team members are able to edit or queue the builds. Permissions can be assigned to groups or individual users to manage the security of the build.

Once the pipeline configuration completes, we can queue the build to execute it.

Environments

The Environments section defines various environments required for deploying the application. You can configure virtual machines and Kubernetes as part of environments, as shown in Figure 6-25.

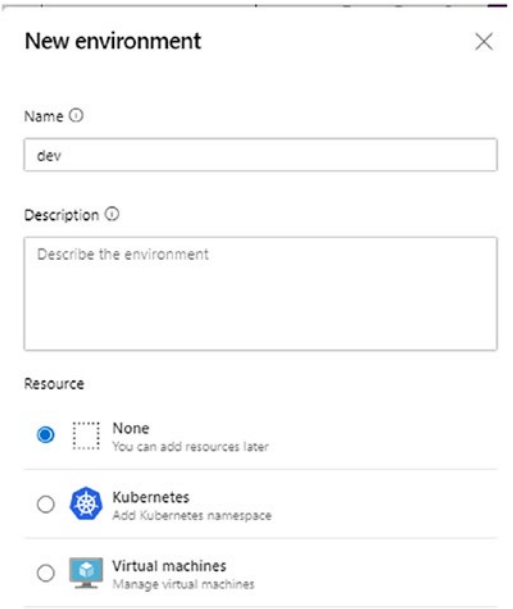


Figure 6-25. *Creating a new environment*

The Kubernetes-based environment will prompt you to select a provider: the Azure Kubernetes provider or the generic provider. If you select the Azure Kubernetes provider, the user should provide the Azure subscription details to connect to an Azure subscription. If you select the generic provider, provide the details related to the cluster name, namespace, server URL, and secret to connect to an on-premise Kubernetes cluster.

Adding virtual machines to an environment requires the execution of PowerShell scripts, which can be downloaded directly from Azure DevOps.

Select the operating system as Windows or Linux and copy the PowerShell script, as shown in Figure 6-26. Once the script gets executed in a VM, a corresponding VM will appear in this window for configuring as part of the environment.

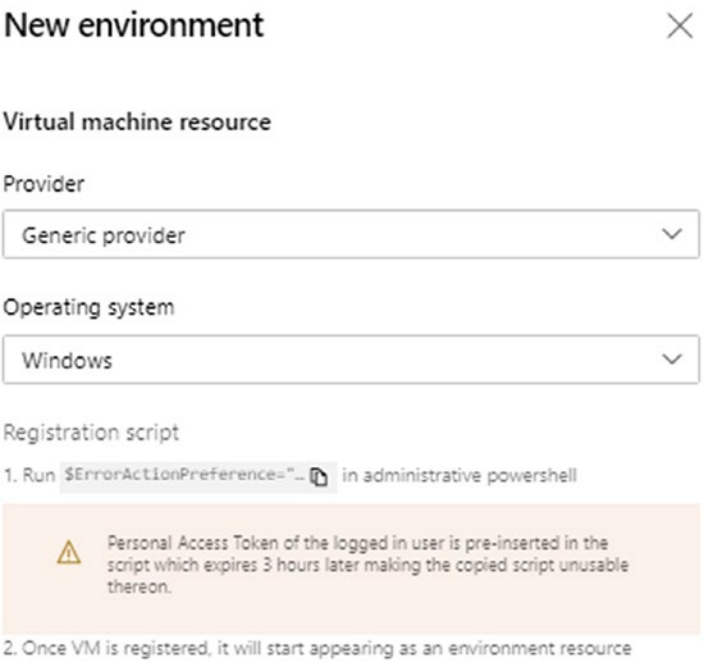


Figure 6-26. *VM settings*

Releases

Before getting into the details of the release process, let's talk about two words that are often used interchangeably: release and deploy. *Release* means the availability of a product to a wider audience, whereas *deploy* means the installation and configuration of

software or a product in a target system. Release management involves the deployment of different versions of the application into various environments. Moreover, the release pipeline defines the controls required for a proper and secure release of a product.

Creating a Release Pipeline

While creating a new release pipeline, you have an option to start with an empty job or select one of the predefined templates and configure it, as shown in Figure 6-27.

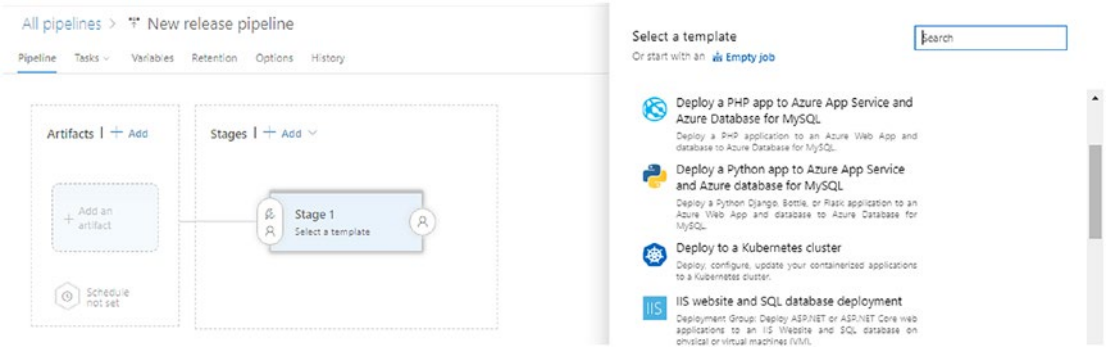


Figure 6-27. *Release pipeline options*

Every release pipeline starts with artifacts as an input. Then the pipeline defines various environments as stages to deploy the product along with the activities to be performed in each stage before and after deployment.

Artifacts

Artifacts define the deployment packages or files that are deployed into a target environment based on the stage-wise configurations. Currently, there are nine source types supported by Azure DevOps services, as shown in Figure 6-28. The user can select one from among the nine source types and provide the required configuration details.

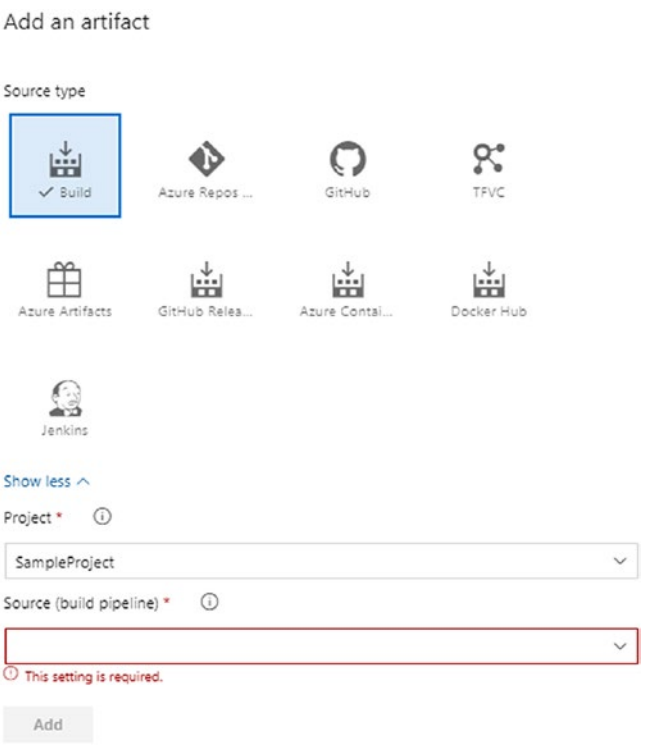


Figure 6-28. *Adding an artifact*

- *Build*: If the deployment is based on a build, configure the output of the build as an artifact for the current pipeline. Most of the time, this option is selected to integrate the build and release pipelines.
- *Azure Repos Git*: This option helps to deploy the files directly from the repo itself. Python-based models are not built using any build system; instead, they are deployed using the release pipeline.
- *GitHub*: Configure GitHub as an artifact repository to deploy the files from GitHub to different environments.
- *TFVC*: If the project is configured with Team Foundation Version Control (TFVC), this option will be used for deploying the files directly from the source control to the deployment environments.
- *Azure Artifacts*: The user can configure Azure artifacts as an artifact repository and use it in the build pipeline. If the build configuration uses the Azure Artifact repository, use it as the input for the release pipeline.

- *GitHub Release*: Configure the GitHub release output as the input for the current release pipeline.
- *Azure Container Repository (ACR)*: If the build pipeline deployed any images to ACR, configure them as the artifact repository for the release pipeline to deploy the container images into a target environment.
- *Docker Hub*: Just like Azure Container Repository, Docker Hub is also used to store the custom images generated for container-based deployment. From the build pipeline, we can deploy the images to either Docker Hub or ACR.
- *Jenkins*: If the release pipeline based on a Jenkins-based build, then use this to configure it.

Based on the release pipeline setup requirement, we can select any one that may be required as part of a deployment. For example, one release pipeline may contain the UI artifacts, API artifacts, and database artifacts for completing a single deployment.

Another option available is the scheduling of a release pipeline. We can add multiple times when the release pipeline should run, as shown in Figure 6-29. In Figure 6-29, the release will be triggered at 3 a.m. on Monday and at 1 a.m. on Friday and Sunday.

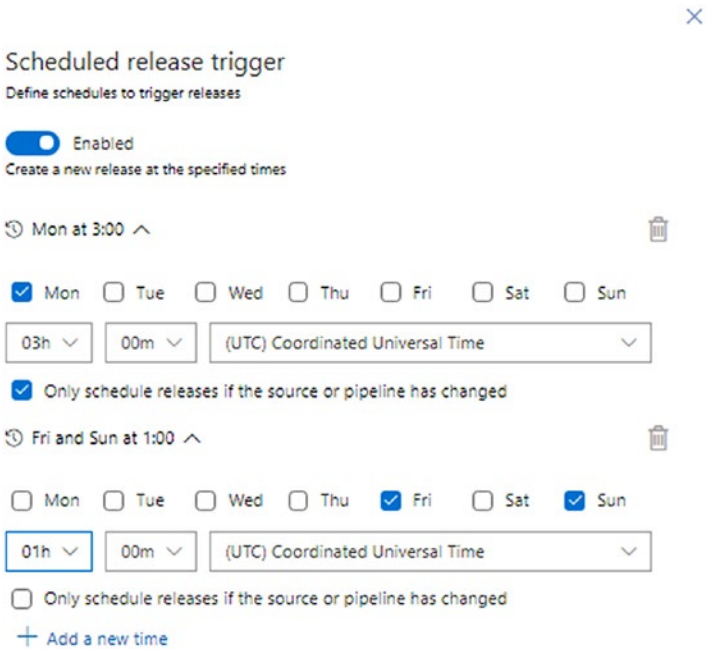


Figure 6-29. *Scheduling a release pipeline*

Stages

Each stage will define a deployment unit or environment setup. Selecting a stage name will allow the user to change the stage name, delete it, move it, and save it as a template. There are two conditions associated with a stage.

Pre-deployment Condition

The "Pre-deployment condition" section defines the conditions that should be evaluated or met before deploying to the current stage. There are multiple configuration items available in this section.

- *Triggers*: This section defines the trigger for the release; it can be Manual or After Release. If it is After Release, then the system will automatically deploy it to the stage after a release is created. If it is manual, even if there is a new release, the system will wait for a manual deployment. The following are other options to set:
  - *Artifact filters*: You can define an artifact condition to deploy to this stage. A release gets deployed to this stage only if the artifact conditions are met.
  - *Schedule*: This is a stage-wise schedule for deployment.
  - *Pull request deployment*: This triggers the deployment based on a pull request. But this option has a dependency on how the artifacts for the release pipeline are configured.
- *Pre-deployment approvals*: This section defines the approvals required to deploy any new artifact to this stage, as shown in Figure 6-30. If multiple users are added under Approvers, all approvers are required to approve the release for deployment. If a group is added to the Approvers list, then one member of the group can approve and proceed with deployment.

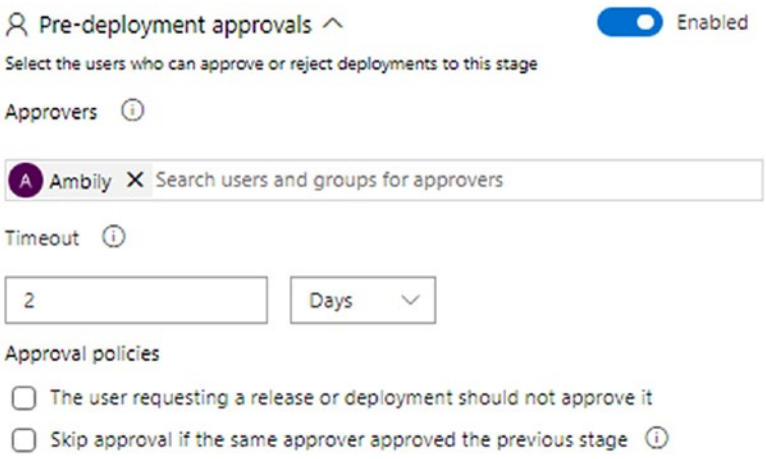


Figure 6-30. *Pre-deployment approvals*

The Timeout property allows you to set the maximum time to wait for an approval. If there is no action happening within this time limit, the system will automatically reject the deployment at this stage. In addition to these settings, the system allows you to configure two of the approval policies. The first one prevents the release or deployment requester from approving it. The second policy eliminates multiple approvals from the same team member; if that person approved a previous stage, then it is auto-approved to the current stage.

- *Gates*: This adds gates that evaluate health parameters. They are periodically re-evaluated in parallel. When all gates succeed for the success criteria duration, the deployment will proceed. If this does not occur before the timeout period, the deployment is rejected.

  As listed in Figure 6-31, the options available for adding a new gate are as follows: Check Azure Policy compliance, Invoke Azure Function, Invoke REST API, Query Azure Monitor alerts, and Query Work items.

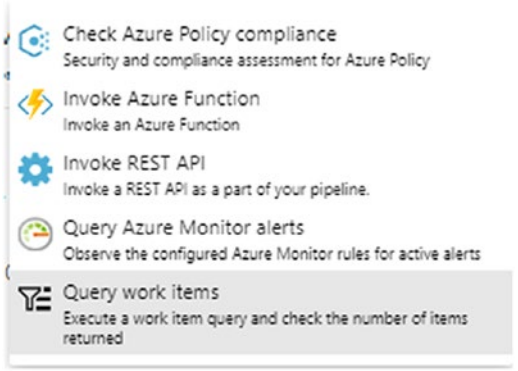


Figure 6-31. *Gates*

- *Deployment queue settings*: This section provides settings to execute multiple parallel deployments and control subsequent releases, as shown in Figure 6-32.

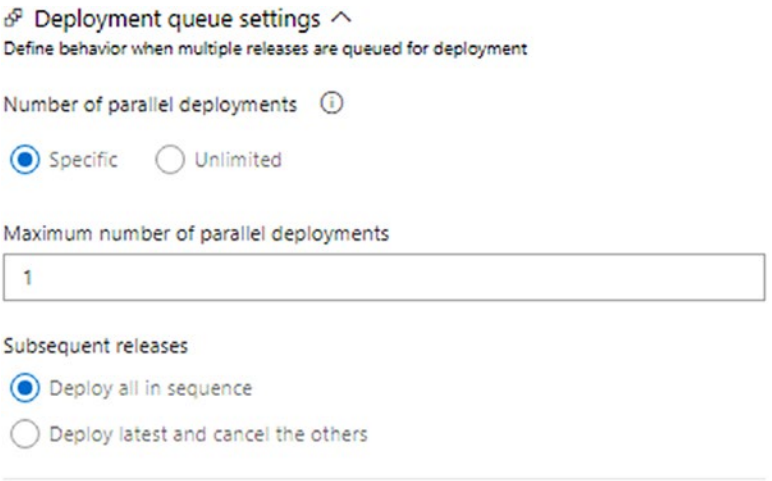


Figure 6-32. *Deployment queue settings*

Post-deployment Condition

Post-deployment conditions help in handling the required controls after a deployment to a stage. The following are the main settings available in this section:

- *Post-deployment approvals*: Just like pre-deployment approvals, post-deployment approvals handle the approval or rejection of a deployment to a stage. Similar properties such as timeout and policy configurations are available here as well.
- *Gates*: Similar to pre-deployment approval, gates protect the deployment quality and completeness.
- *Auto-redeploy trigger*: This configures the events that trigger the automatic redeployment to the stage, as shown in Figure 6-33. Select an event and corresponding action to configure the redeployment.

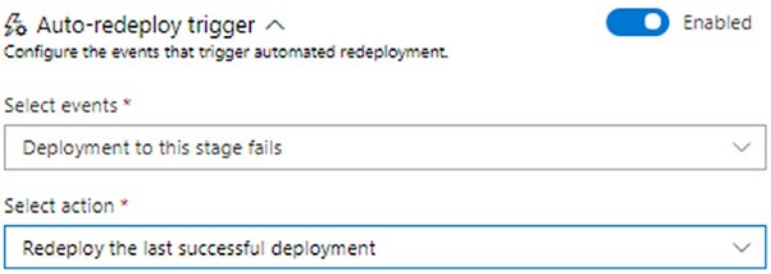


Figure 6-33. *Auto-redeploy trigger*

Jobs

Every stage will have one or more jobs associated with it. There are three types of jobs supported in classic mode and in YAML mode. The job types available in classic mode are as follows:

- *Agent job*: Run on an agent in an agent pool.
- *Agentless job*: Run on the Azure DevOps server.
- *Deployment group job*: Run on the machines in the deployment groups. Deployment groups make it easy to group the target machines, where the application gets deployed.

In a YAML-based release pipeline and build, there are three different jobs.

- *Agent pool jobs*: Run on an agent in the agent pool.
- *Server jobs*: Run on an Azure DevOps server.
- *Container jobs*: Run in a container on an agent in an agent pool.

Each of these jobs will have set of tasks associated with it to run the pipeline.

Multiple Stages

A build engineer can define multiple stages in a sequence or parallel form. In Figure 6-34, there are four stages, starting with the Dev stage and deploying to two QA stages in parallel. After that, the Prod stage deployment will quickly start.

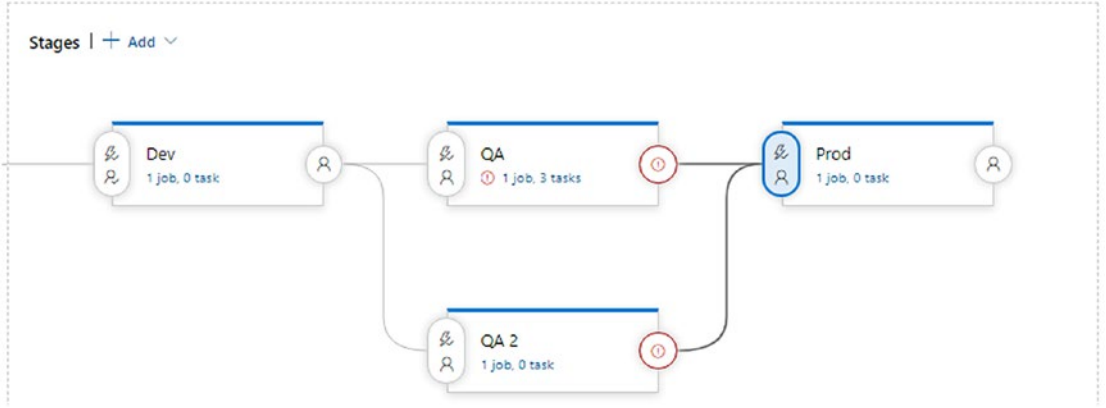


Figure 6-34. *Multiple stages*

Stages can be connected to each other using the "Stage selection" option under the pre-deployment conditions. The system allows the user to select multiple stages to ensure parallel deployment to various stages. If there is any circular dependency issue, the system detects it and prompts the user.

Tasks

Tasks are the low-level execution units that execute an activity. Tasks are added under jobs based on the execution environment. If the task execution is part of the build activity or artifact processing, then add it to the agent jobs. If it is independent of artifacts and executed at the server level, add it to the agentless jobs. If the tasks should be executed in target systems, then add them under the deployment group jobs.

Click the plus sign to add the new tasks to the agent job. The tasks, which can be executed in an agent, will be listed. The user can select the task from the list, as shown in Figure 6-35, and configure it.

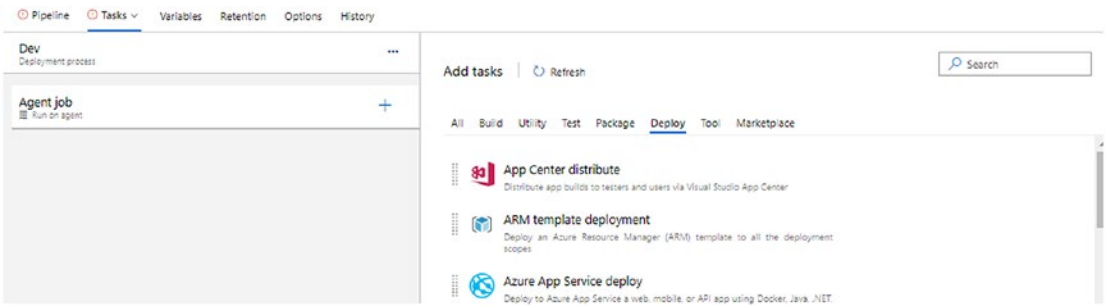


Figure 6-35. *Tasks, Deploy tab*

The task list is already filtered based on the job. If the user tries to add tasks under a deployment group job, different tasks will be listed. Figure 6-36 shows how the tasks are grouped under each of the job categories.

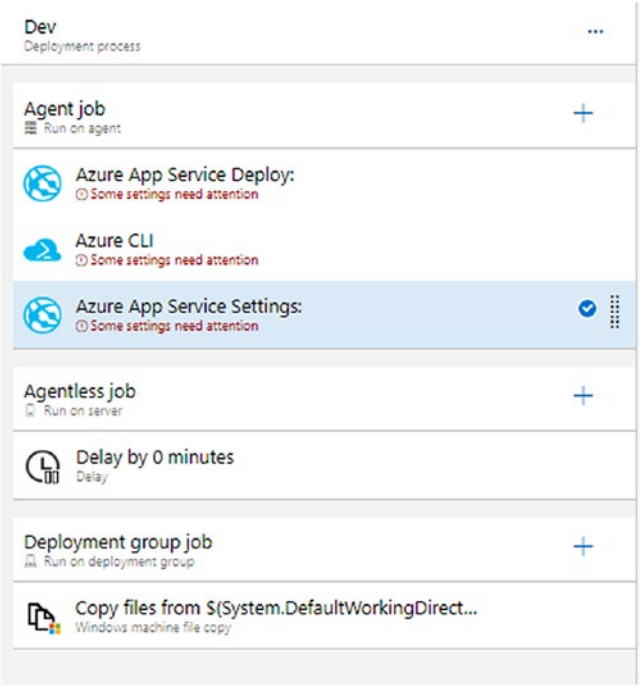


Figure 6-36. *Jobs listed by category*

Variables

The Variables tab shows the pipeline variables and variable groups. This is similar to the Variables tab in the build pipeline.

Retention

The Retention tab lists the retention configurations for each of the stages, as shown in Figure 6-37.

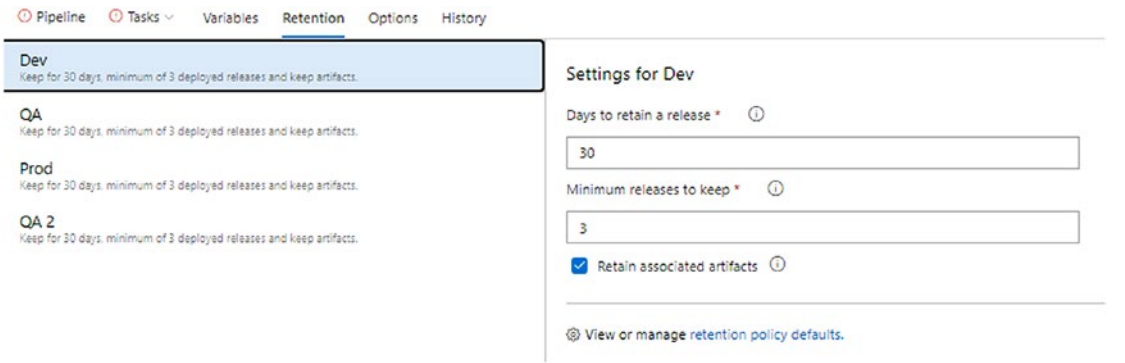


Figure 6-37. *Retention tab*

Centralized retention configurations are captured in Project settings ➤ Pipeline ➤ Release retention.

The user can override the default configuration for each of the stages. Retention configurations control for how many days the release should be maintained and how many releases to keep along with associated artifacts.

Options

Two different configurations are available on the Options tab. The first one is the release name format, which can be changed based on the project requirements. The default release name format is `Release-$(rev:r)`.

The second set of configurations help with integrations with external systems, as shown in Figure 6-38. Using these configurations, the system can notify an external system like JIRA about the deployment status of one or more stages.

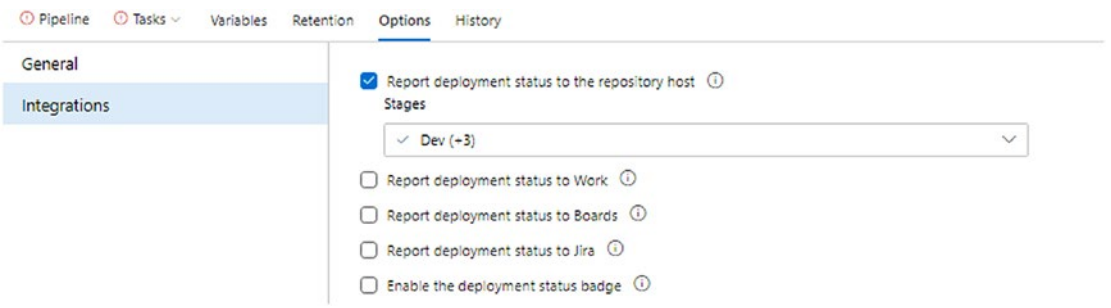


Figure 6-38. *Options tab*

Library

The Library section deals with the reusable components such as variable groups and secure files. Variable groups help build engineers to create a group of variables that can be shared across multiple pipelines.

Variable Groups

A build engineer can add a new variable group by providing a name and a brief description about the variable group, as shown in Figure 6-39, which will help others understand the usage. The system provides an option to allow access to all pipelines or restrict to a specific pipeline.

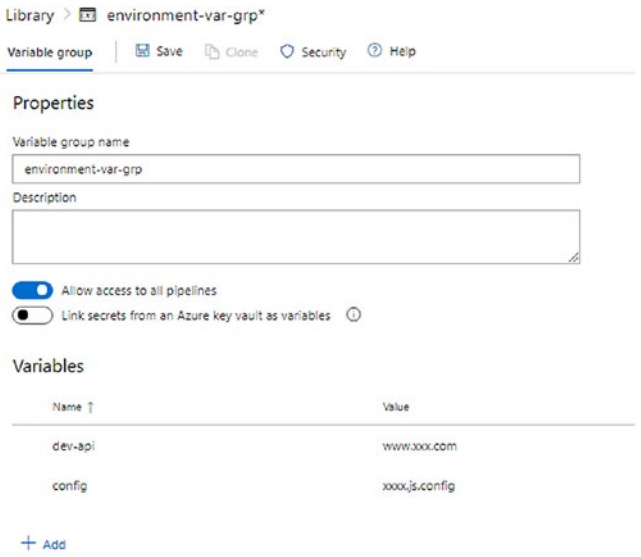


Figure 6-39. *Variable groups*

Moreover, the user can link a secret from Azure Key Vault as a variable and use it in the pipeline. Azure Key Vault is a secret or password management offering from Azure, which can be leveraged to securely use the secrets and passwords in an application. Once this option is selected, all the existing variables will be removed, and the system will prompt you to integrate the Azure service connection and Key Vault name to proceed.

Variable groups will appear as part of the pipeline definition on the Variables tab and be invoked by clicking the option "Link variable group," as shown in Figure 6-40.

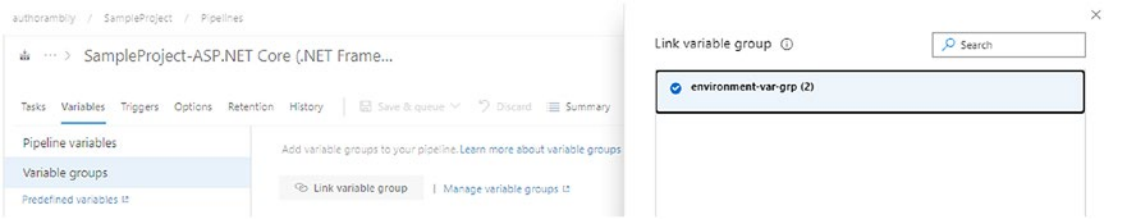


Figure 6-40. *Variable groups*

Secure Files

The "Secure files" section helps the build engineers to manage secure files such as certificate files or signing keys and use them across different pipelines. Moreover, one can control the permissions in each file level to ensure the security of each of the files getting added to the "Secure files" section using the Security option, as shown in Figure 6-41.

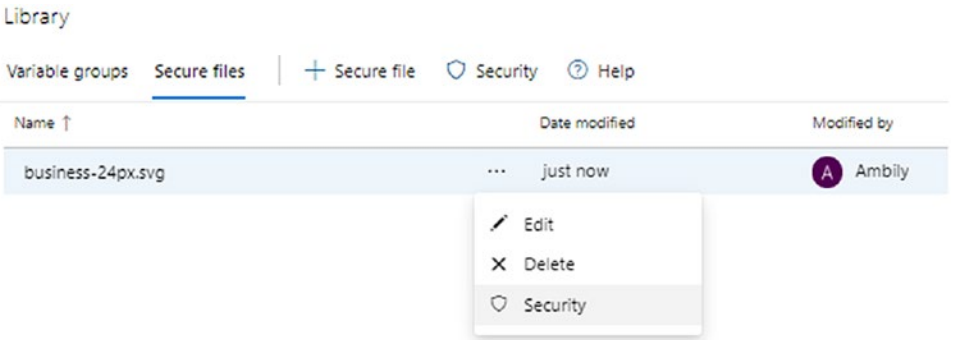


Figure 6-41. *"Secure files" section*

Task Groups

Task groups define the set of tasks commonly used across different pipelines and help in reusing the same configurations and variables across the pipeline. This reduces the overhead in maintaining the same configurations in multiple places. Also, this reduces the chance of missing the changes in one of the pipelines due to the duplicate existence of the same task groups.

Task groups are applicable for the classic build only. This is not applicable for YAML-based build templates, where we use YAML templates for reuse. Task groups will be created from the build or pipeline, after unlinking all the linked parameters.

Click the Unlink All link, as shown in Figure 6-42, available in the root level of the Tasks tab to unlink all the parameters.

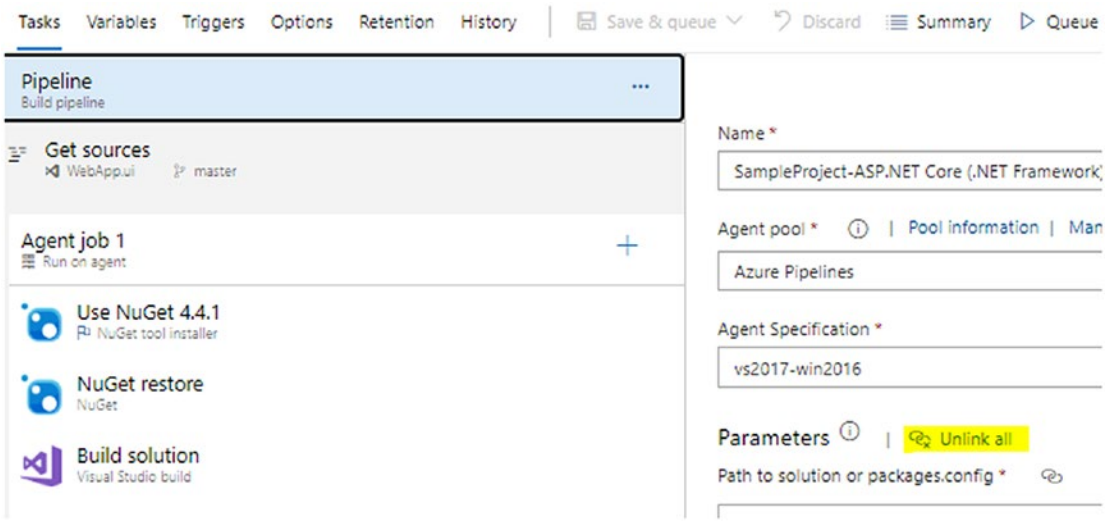


Figure 6-42. *Tasks tab*

Select the steps and right-click to select the option "Create task group," as shown in Figure 6-43.

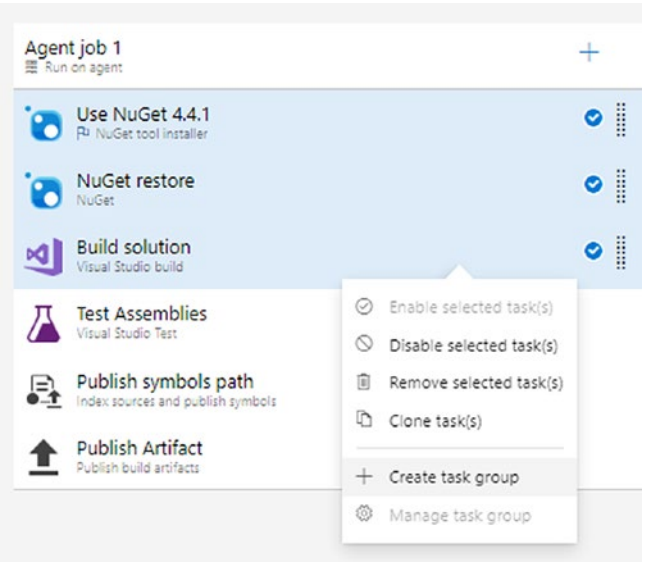


Figure 6-43. *Creating a task group*

Once prompted, provide the required details such as the task group name, description, and category. Also, the system will list the variables used as part of the task groups and the default values. The user can change the variable values and create a task group, as shown in Figure 6-44.

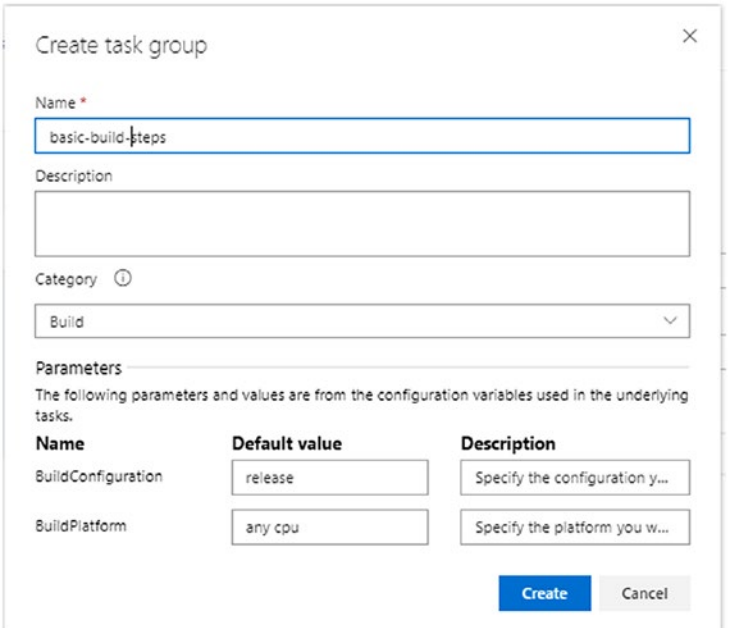


Figure 6-44. *Changing the variables of a task group*

This will replace the specified steps with the task group name in the build or pipeline. Notice the "Task version" option on the right side, which is set to 1.* now, as shown in Figure 6-45.

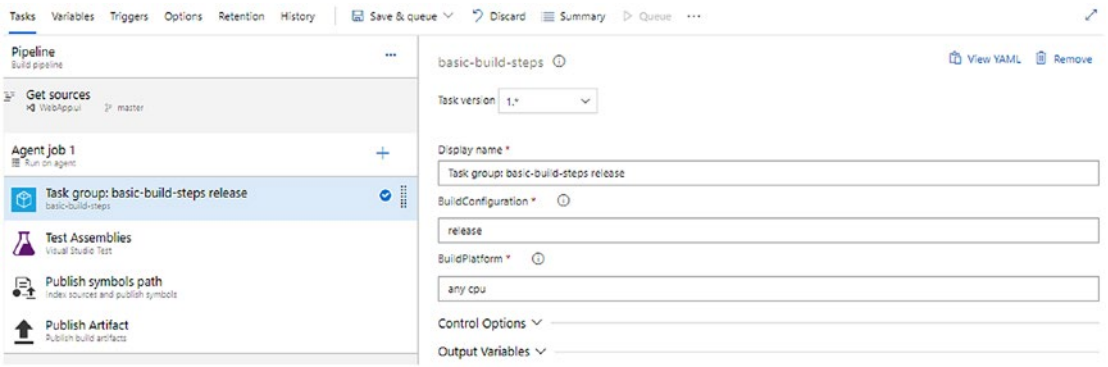


Figure 6-45. *Including task groups*

Navigate back to the "Task groups" section to view the newly created task group. Open the task group and edit some elements or configurations to publish a new version of it. After modification, click "Save as draft" to create a draft version of the task group. The system will add a new version with a suffix of `-test`, so it will be named `1.*-test`. Also, it provides a "Publish draft" option, which will make the changes a preview feature. This will add a new version called `2.*(preview)`, as shown in Figure 6-46.

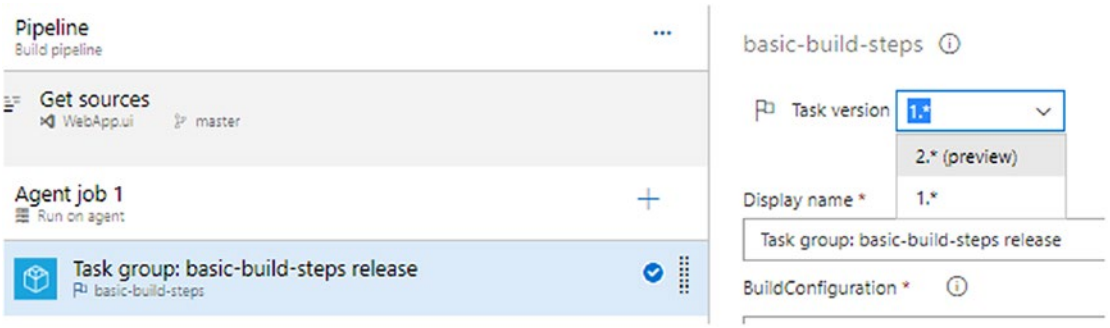


Figure 6-46. *Preview of task groups*

In the task group, it shows the option Publish Preview to publish the current preview changes as a new version. Once the user selects this, the system will publish a new version `2.*` of the task group. Versioning and management of the task groups are supported in the "Tasks groups" section. Moreover, the system captures the changes

and associated comments as history for future audit purposes. We can refer to the build pipelines, release pipelines, and other task groups, which refer to the current task group, in the References section.

Deployment Groups

Deployment groups define a logical group of target machines for parallel deployment. Add a new deployment group by specifying the name and a description. Once the target OS is selected, the system will prompt you for the PowerShell script, as shown in Figure 6-47, for configuring the systems as part of the deployment group.

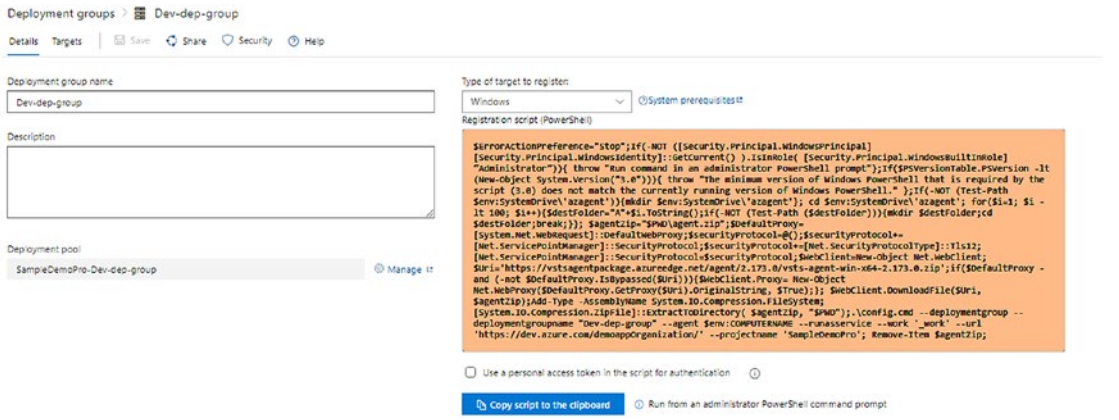


Figure 6-47. *Deployment groups*

The user can decide to include the personal access token in the script for authentication or provide it at the time of script execution. Copy the script and execute it in the target VMs to add them as part of the deployment group.

The Share option on the top of the definition allows the user to select other projects in the same organization to share the deployment group with. The Security option helps you configure the permissions for the deployment group.

Sample Build and Release Implementation

In Chapter 4, we discussed a sample application using Angular and the .NET Web API, as shown in Figure 6-48.

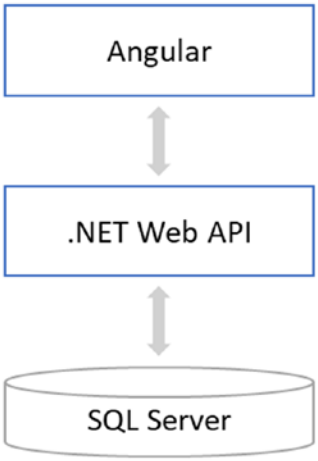


Figure 6-48. *Sample app*

In this section, we will configure the build and release pipeline for the application.

Build and Release of an Angular Application

The Angular application build involves the following steps:

1. *Install npm*: Install the dependencies using `npm install`.
2. *Build Angular*: Use the `npm run` command to build and package an Angular project. For production deployment, use the production build (`ng build --prod`) to minify the file before building. An easy way to configure it is in the `package.json` file of an Angular app, as shown in the following snippet:

```
"scripts": {
   "ng": "ng",
   "start": "ng serve",
   "start-qa":"ng serve -c=qa",
   "build": "ng build",
   "build-prod": "ng build --prod",
   "build-qa":"ng build -c=qa",
```

   Now invoke it in the npm task, as shown in Figure 6-49.

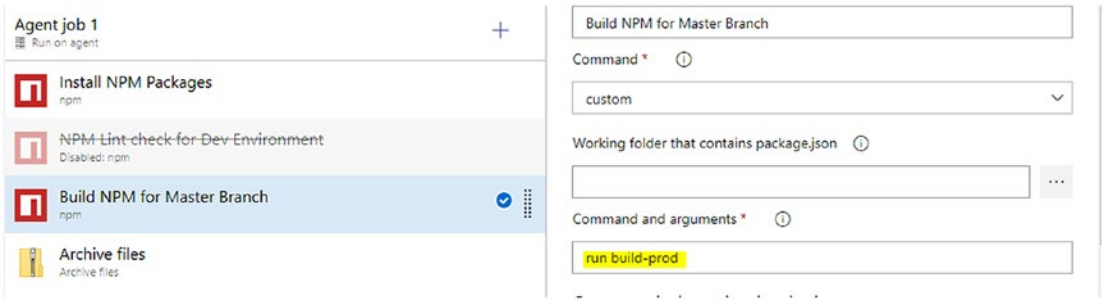


Figure 6-49. *npm task configuration*

3. *Archive files*: This task compresses the packaged files and other artifacts. Azure DevOps offers the archive options as zip, 7zip, tar, and wim.

4. *Publish artifacts*: Publish the artifacts to a repository like the Azure artifact repository or Nexus artifact repository.

In addition, the user can add many other tasks such as code analysis, integration with unit test scripts, and so on. Two such tasks are Lint for code analysis and WhiteSource for vulnerability analysis.

- *Lint*: Add the npm task to the list and specify `run lint` under "Command and arguments."

- *WhiteSource*: This detects and fixes security vulnerabilities and problematic open source licenses. Install the WhiteSource extension (free) from the Marketplace using Organization settings ➤ General ➤ Extensions. Add the WhiteSource Bolt task after the first step of npm package installation to scan for vulnerabilities. This will add a menu item in the Pipelines section with detailed reports.

Save the build pipeline and queue it up for execution. While queuing, the user needs to select the agent pool and agent specifications. By default, Azure DevOps provides an agent pool called Azure Pipeline consisting of a hosted agent with multiple agent specifications such as ubuntu-16.04, windows-2019, vs2017-win2016, etc. In addition, the user can set up self-hosted agents or agents in Azure Virtual Machine Scale Set (VMSS) using Project settings ➤ Pipeline ➤ Agent pools.

The user can add a new agent pool by providing a name, say, angular-agents, and setting the pool type to "Self-hosted." Now add the agents using "New agent," which opens the detailed step to set up the agent, as shown in Figure 6-50.

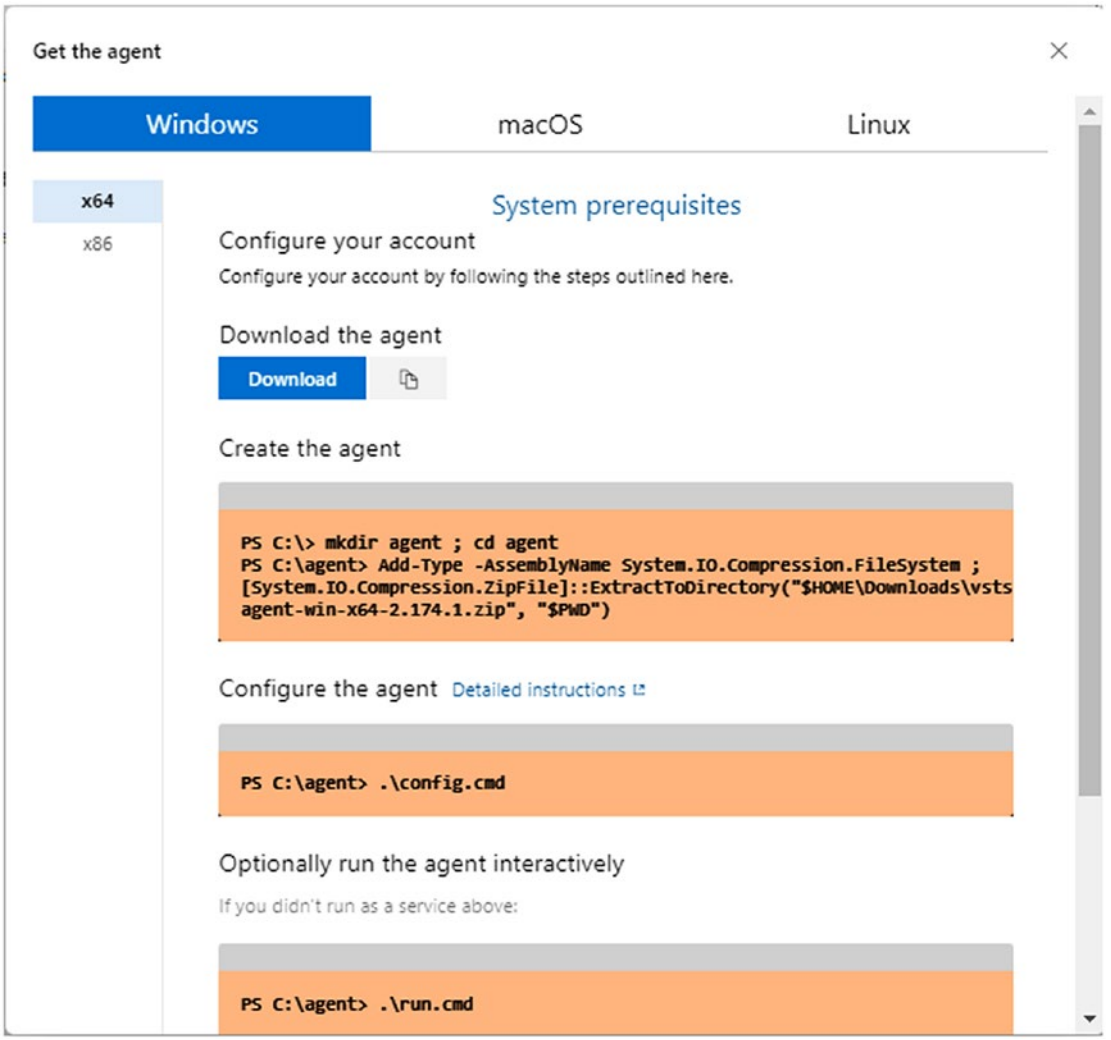


Figure 6-50. *New self-hosted agent setup*

Follow these steps to set up the agent:

1. Download the agent based on the OS in the agent machine.
    - Navigate to the agent location and run the commands given in the "Create the agent " section mentioned in the above Figure.
2. Either configure the agent as a service using the given command (`config.cmd`) or execute it in an interactive mode. We recommend using interactive mode for the initial setup and debugging; otherwise, run the agent as an agent.

Using the release pipeline, we will deploy the Angular application to the Azure App Service. Follow these steps to deploy it in a preconfigured app service:

1. Configure Azure Service Connection under Project Settings ➤ Pipeline ➤ Service connections.
2. Create an app service in Azure.
3. Define a release pipeline.
   a. Add the previous build output as an artifact.
   b. Enable a continuous deployment trigger, if required.
   c. Add the first stage with the following tasks:
      i. *Stop Azure App Service*: Stop the Azure app service before deployment.
      ii. *Azure App Service Deploy*: Select the Azure service connection, app service, and package/folder where the output of the build is copied. Specify `npx serve -s` as the startup command.

      Note: npx is npm's package runner, which makes it easy to run any sort of node executable.

      iii. *Start Azure App Service*: Start the app service after deployment.
4. Save and queue the new release and deploy it to verify the pipeline.
5. Once the pipeline is successful, add more stages and pre/post-conditions to the pipeline.

Build and Release of a .NET Application

Built-in templates support the normal build for any .NET applications. Here, we will build the container image for the .NET API. The main steps required are as follows:

1. Add the Docker file to the application. The sample Docker file used for the .NET Core API is given here:

```
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim AS base

WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build
WORKDIR /src
COPY ["Sample.TestApi.csproj", ""]
RUN dotnet restore "./ Sample.TestApi.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build " Sample.TestApi.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish " Sample.TestApi.csproj" -c Release -o /app/
publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", " Sample.TestApi.dll"]
```

2. Prepare the build pipeline with the following tasks:

   a. *Get Sources*: This is the initial step to get the sources.

   b. *Use NuGet*: This installs the NuGet packages referenced in the project.

   c. *NuGet restore*: This restores the NuGet packages.

   d. *Build an image*: This provides the container registry such as Azure Container Registry or Docker Hubor Docker file path, and an image name.

   e. *Push an image*: This pushes the image.

The containerized .NET Core API can be deployed to different Azure offerings such as Azure Container Instance (ACI), Azure Kubernetes Services (AKS), or Azure App Service. Follow the same three-step release process used in the Angular app to deploy it into Azure App Service. Specify the Azure Container registry and the image details in the second step, which is the Azure App Service deploy step, to deploy the container into the app service.

Summary

Azure DevOps offers many features and components to automate the build and release pipeline. YAML-based and classic task-based models support many third-party tool integrations to help you create the build and release ecosystem.

CHAPTER 7

Continuous Feedback and Other Features

Continuous feedback plays a major role in improving the quality of an application. A timely feedback mechanism improves customer satisfaction and reduces the time it takes to address any user concerns. A live dashboard displaying the status of the activities and a collaboration platform for requirements brainstorming are equally important. This chapter looks at some of these features available in Azure DevOps to meet these needs.

Dashboards

Dashboards provide a live report about the status of the project and related activities. The user can configure multiple dashboards such as a team-level dashboard, a project-level dashboard, and a dashboard for communicating some other status. Each of the dashboards consists of set of widgets displaying textual data, graphs, charts, badges, and so on. Figure 7-1 shows a sample dashboard.

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_7

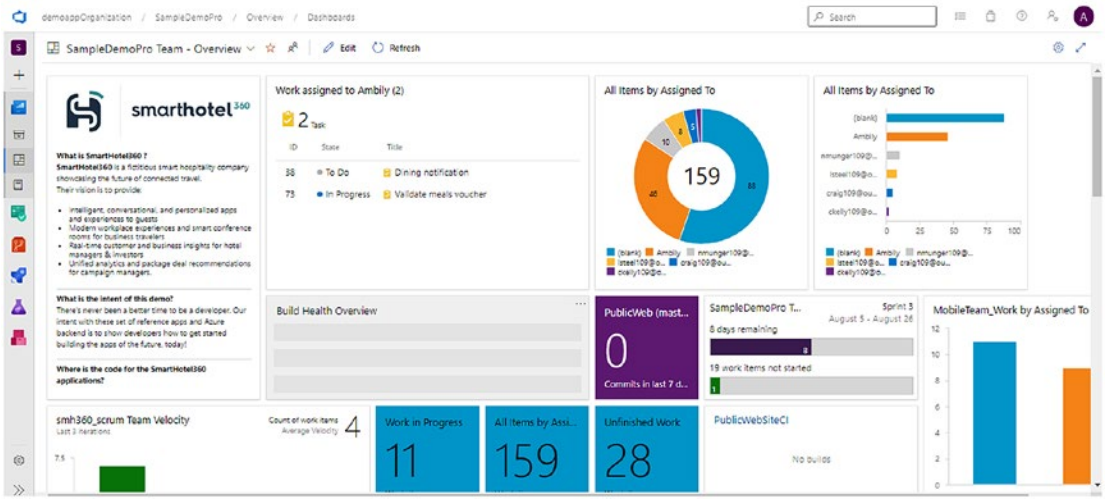


Figure 7-1. *Sample dashboard*

On top of the dashboard, there are options to select a different dashboard from the drop-down list of dashboards, to add a new dashboard, and to navigate to the list of available dashboards. Other options available are as follows:

- *Add to Favorite*: Use this option to mark the current dashboard as a favorite.
- *Team Profile*: Use this option to view the team profile, which lists the boards, backlogs, sprints, dashboards, and team members in the current team.
- *Edit*: Use this option to edit the current dashboard. The user can add new widgets from the Marketplace or realign the existing widgets. To realign the widgets in the dashboard, the user can drag each of them after selecting the Edit option and drop them in a target place. This will automatically realign the entire grid displaying the dashboard widgets. Edit opens the Add Widget list on the right side so the user can select from an existing list of widgets. The user can refer to the list of available widgets in the Visual Studio Marketplace (`https://marketplace.visualstudio.com/`).
- *Refresh*: Use this option to refresh the data displayed in each widget. Data refresh can be done automatically using the Settings option.

- *Settings*: Use this option to open the settings, which allows the user to change the name of the dashboard, provide a description, and control the security, as shown in Figure 7-2. Also, the user can set the automatic refresh of the dashboard to every five minutes.

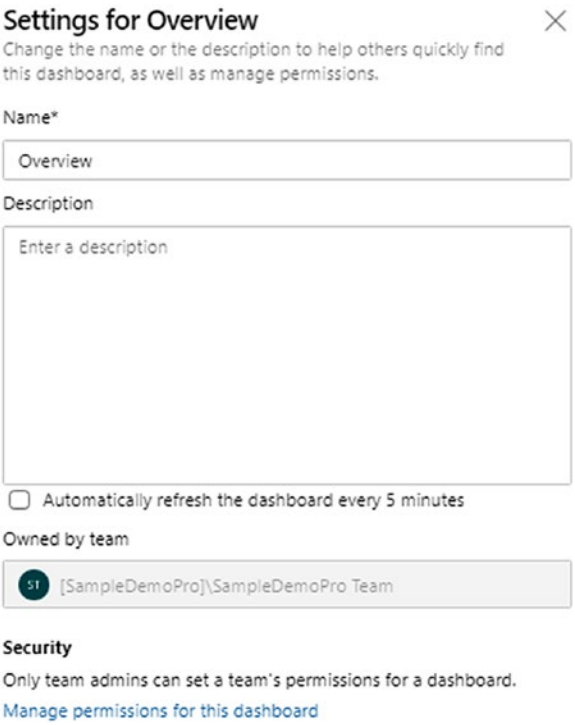


Figure 7-2. *Dashboard settings*

Every widget has a list of options available to configure the widget, as shown in Figure 7-3. The "Copy to Dashboard" option allows you to copy an existing configured widget from one dashboard to another. "About this Widget" takes the user to the Microsoft document explaining the functionality of the widget. "Delete" removes the widget from the dashboard.

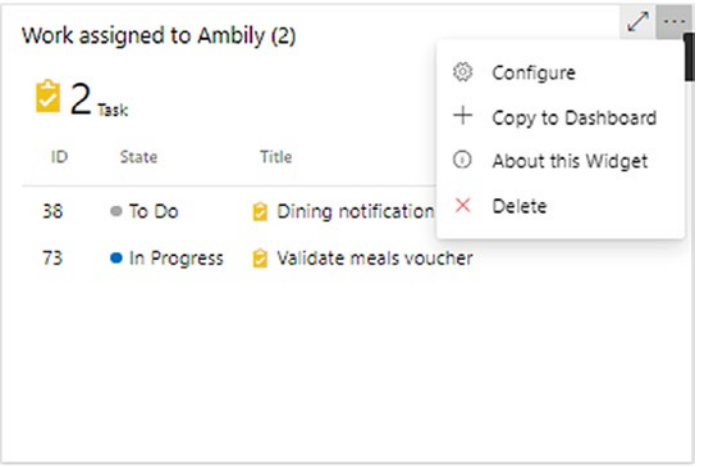


Figure 7-3. *Widget options*

The "Configure" option opens the different configurations associated with a widget. Figure 7-4 shows the settings for the State of TestCases chart configuration.

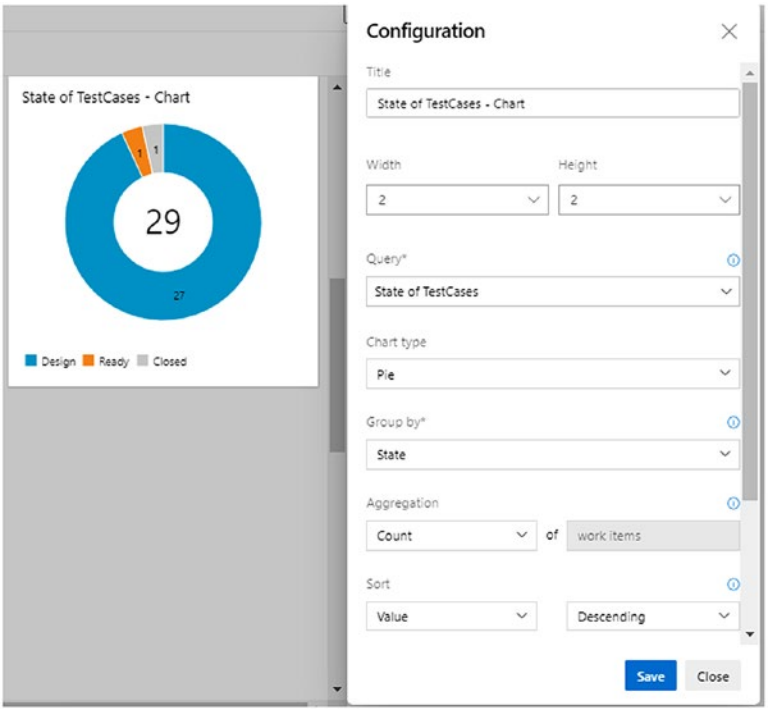


Figure 7-4. *Configuring a widget*

This is a query-based widget, and the name of the query is listed in the configuration. Click the widget chart to open the associated query and resulting work items in list view, as shown in Figure 7-5.

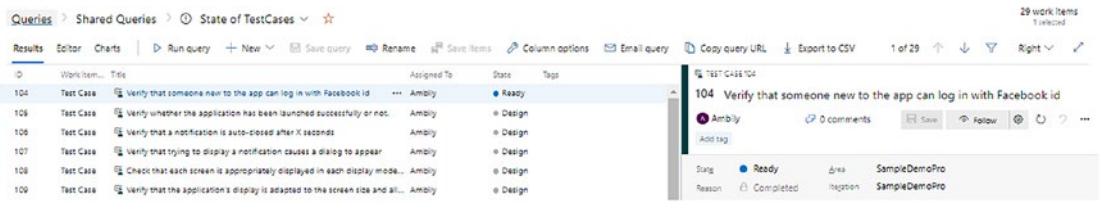


Figure 7-5. *Querying a widget*

Any modifications in the query will impact the dashboard automatically. Moreover, this view provides various options to export the query result, copy the query URL, and so on.

Another set of widgets is from the Marketplace; there won't be any queries associated with these kinds of widgets. For example, the Burndown chart widget shown in Figure 7-6 is from the Marketplace, which shows the burndown of the project activities.

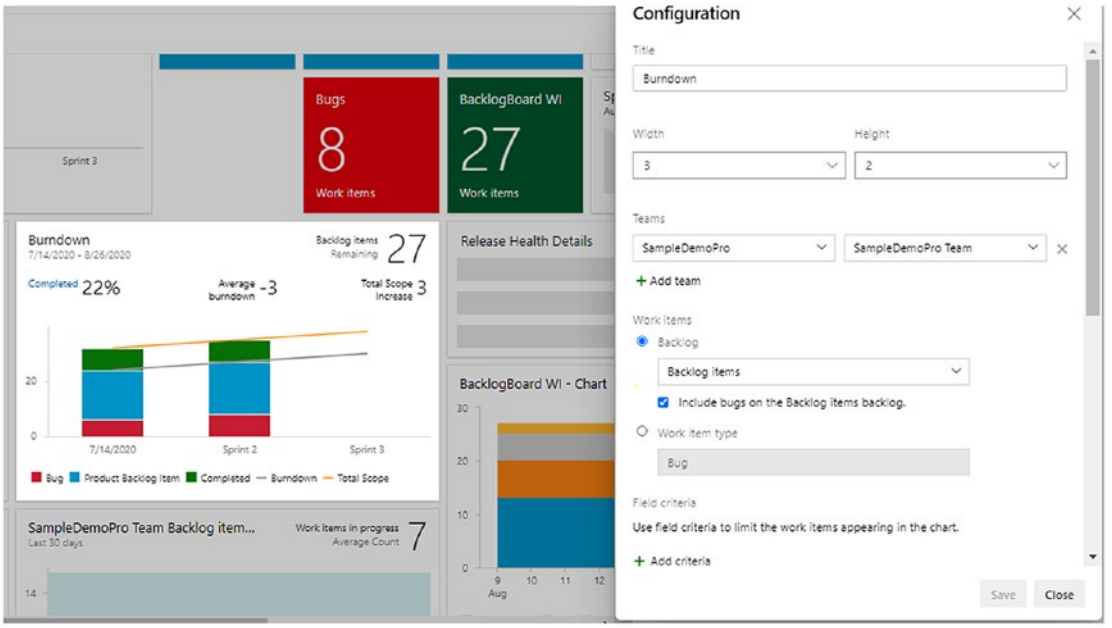


Figure 7-6. *Widget from Marketplace*

Even though the complete chart is not based on any specific defined query in the project, the user can navigate to a custom query by selecting one of the chart legends or blocks. This custom query will list the work items specific to that legend only.

Wiki

A wiki addresses the detailed documentation requirements of a project. A wiki can be used to capture the requirements in detail, architecture design documentations, coding standards, flow diagrams, minutes of meetings, new feature requirements, and so on. This collaboration platform enables users to collaborate on the wiki pages using the discussion boards.

A user can create a project wiki using one of the two options available: Create project wiki or Publish code as wiki.

Publishing Code as a Wiki

This option takes the Markdown pages from the selected repository and publishes them as a wiki. In Figure 7-7, the root folder is selected, which contains the `Readme.md` file. A new wiki will be constructed based on this file and then displayed.

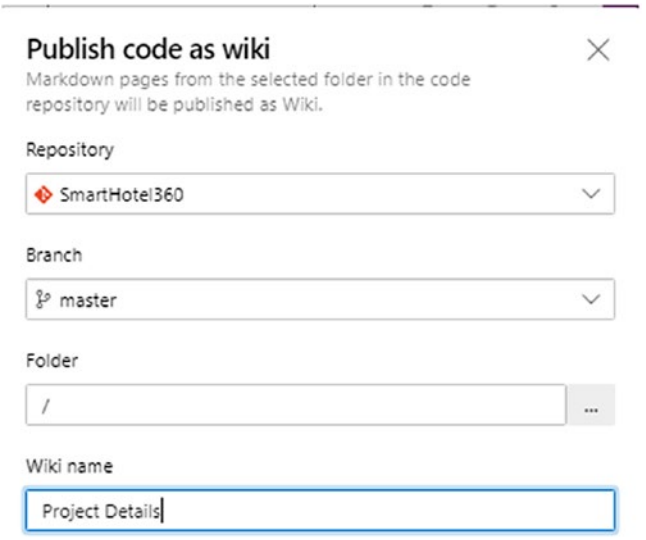


Figure 7-7. *Wiki creation*

Edits in the wiki will be reflected in the file, and vice versa. Once the wiki is published, the user can unpublish the wiki to remove it from the project. Also, the user has an option to rename a wiki to update the name.

Wiki pages are arranged in a hierarchical format, and the user can add subpages and main pages based on the requirements. See Figure 7-8.

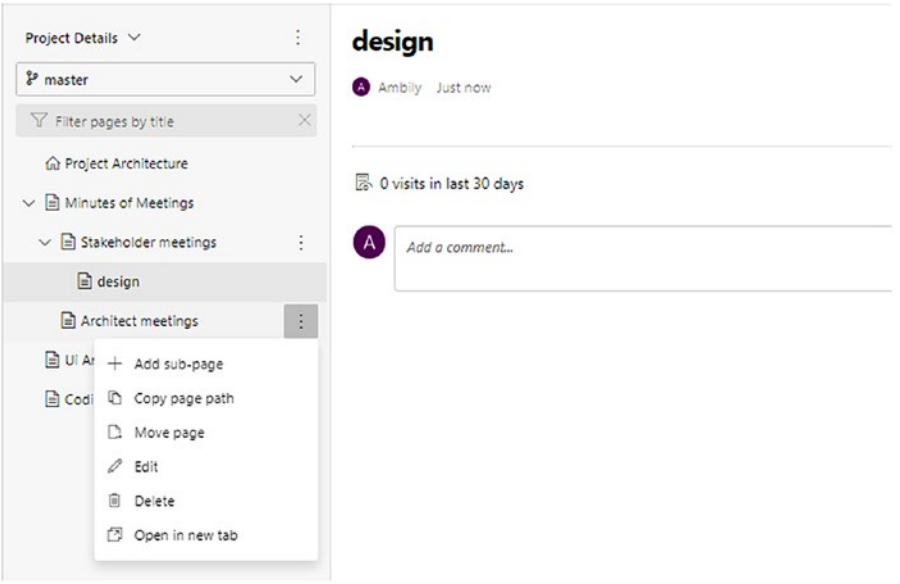


Figure 7-8. *Wiki pages*

The following options are available for each of the wiki pages:

- *Add sub-page*: Use this option to add a child page to the existing page.
- *Copy page path*: Use this option to copy the path of the wiki page, which can be shared over email and other mediums.
- *Move page*: Use this option to move the current page to another location.
- *Edit*: Use this option to edit the page.
- *Delete*: Use this option to delete the page.
- *Open in new tab*: Use this option to open the page in a new tab.

This view shows the selected page content on the right side along with the last edit detail. There is an option to add comments at the end of each page, where stakeholders can share their comments on the content and can have discussions about a specific point. Moreover, in addition to the content, there are options to edit the page, print the page, link work items, view revisions, and delete pages. If the wiki is linked to a requirement or is explaining a task, then the wiki page will be linked to the corresponding work items to establish proper traceability. Once they are linked, the corresponding work item details will appear on top of the wiki.

Editing a Wiki

Wiki pages are constructed using Markdown; you can learn about Markdown in the Microsoft documentation at `https://docs.microsoft.com/en-us/azure/devops/project/wiki/markdown-guidance?view=azure-devops`. As a user, it is not mandatory to learn Markdown; instead, users can use a browser-based editor, which supports the selection of styles available at the top of the editor. A live preview of the changes is available on the right side of the editor. See Figure 7-9.

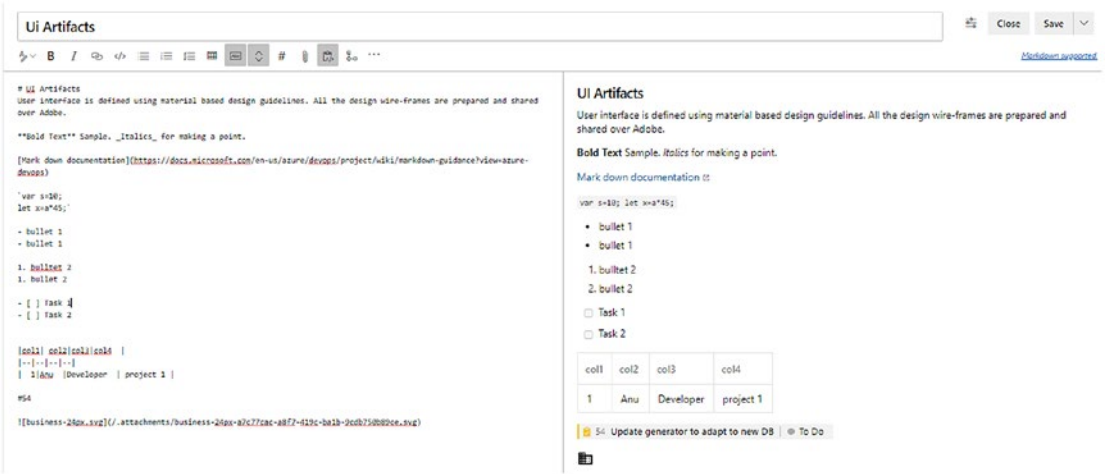


Figure 7-9. *Editing a wiki*

The following are the styling options used in Figure 7-9:

- *Headings*: The first icon on top of the editor provides three heading options.
- *Bold*: Use this option to bold the selected text.

- *Italics*: Use this option to apply italics to the selected text.
- *Link*: Use this option to add a link to the text.
- *Code*: Use this option to insert code into the wiki page.
- *Bulleted list*: Use this option to create a list of items using bullets.
- *Numbered list*: Use this option to create a list of items using numbers.
- *Task list*: Use this option to create a list of tasks.
- *Table*: Use this option to add a table by selecting the columns and rows.
- *Disable word wrap*: Use this option to disable word wrap in the editor; this will add a horizontal scroller for the editor, if there are any long lines added.
- *Disable synchronized scrolling*: Use this option to sync the scrolling of the editor and preview the view in a synchronized manner.
- *Add work item*: Use this option to add a work item reference using a numbered work item ID.
- *Insert a file*: Use this option to insert a file. If the file is a picture file, it displays the image in the editor. In Figure 7-9, the user inserted an image of a business icon.

In addition to these basic options, there are multiple other options available in the editor, which are explained here:

- *Paste as HTML*: This option allows the user to copy and paste the content as HTML. By default, the editor converts all HTML content and renders the result in the wiki. If the user wants to display the HTML tags, then use this option. See Figure 7-10.

Figure 7-10. *Pasting as HTML*

In Figure 7-10, the highlighted two lines of HTML are pasted without enabling the Paste as HTML option, and the following block of code shows the same two lines pasted by turning on this option. Observe the difference in the preview section.

- *Insert Mermaid Diagram*: Use this option to insert a diagram that explains the requirements in detail using a class diagram, sequence diagram, or flowchart. A Mermaid diagram supports a list of diagrams, which can be constructed using Mermaid Markdown; for more information, refer to `https://mermaidjs.github.io`.

  Figure 7-11 shows one of the sample flowcharts using Mermaid markup.

Figure 7-11. *Mermaid diagram*

- *Table of Contents*: Use this option to insert a table of contents based on the headings in the current wiki page. In Figure 7-12, there are two headings available on the page.

Figure 7-12. *Creating a table of contents*

- *Video*: Use this option to insert a video into the wiki from YouTube or a Microsoft stream. Copy the iFrame tag from YouTube and insert it. See Figure 7-13.

Figure 7-13. *Inserting a video*

- *Insert YAML tag*: This will insert the sample YAML tags at the beginning of the wiki, which the user can change. See Figure 7-14.

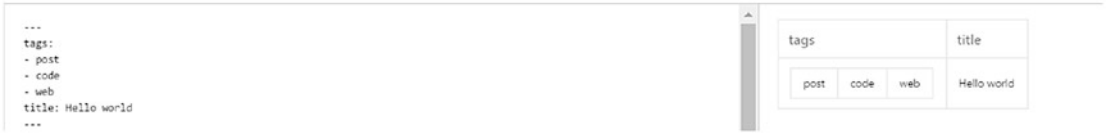


Figure 7-14. *Inserting YAML tags*

- *Formulas*: Use this option to insert formulas into the wiki page. See Figure 7-15.

Figure 7-15. *Formulas*

- *Mention*: Use this option to add reference to a team member using @.
- *Query results*: Use this option to include the result of a selected query as part of the wiki. See Figure 7-16.

Figure 7-16. *Adding query results*

Using all these options, users can define some great content and collaborate over the wiki using the discussion option available on each page.

Discussions or Comments

At the bottom of all wiki pages, there is an area for discussion and comments. Team members can collaborate using this comment area, as shown in Figure 7-17.

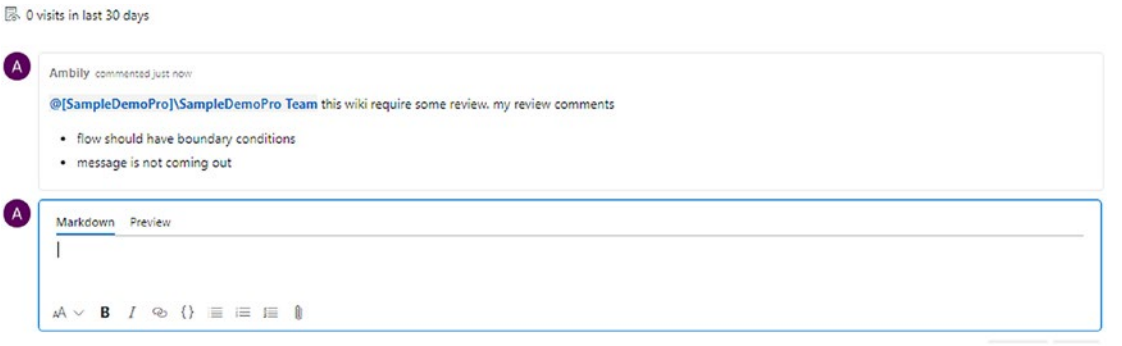


Figure 7-17. *Comment area on a wiki*

Markdown supports various text styling options, and a subset of them is available in the wiki editor. Team members and groups can be added as part of the comment using @ and can add work items using the # symbol. In addition, users can provide feedback using the reaction option available at the top-right corner of the comments. See Figure 7-18.

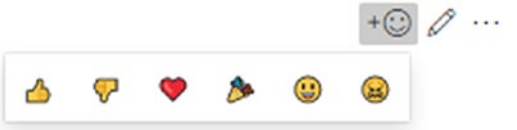


Figure 7-18. *Adding a reaction*

User Settings

The "User settings" section supports the customization of the overall DevOps system based on the user selections. A user can open these settings using the "User settings" icon ( ) at the top-right corner. See Figure 7-19.

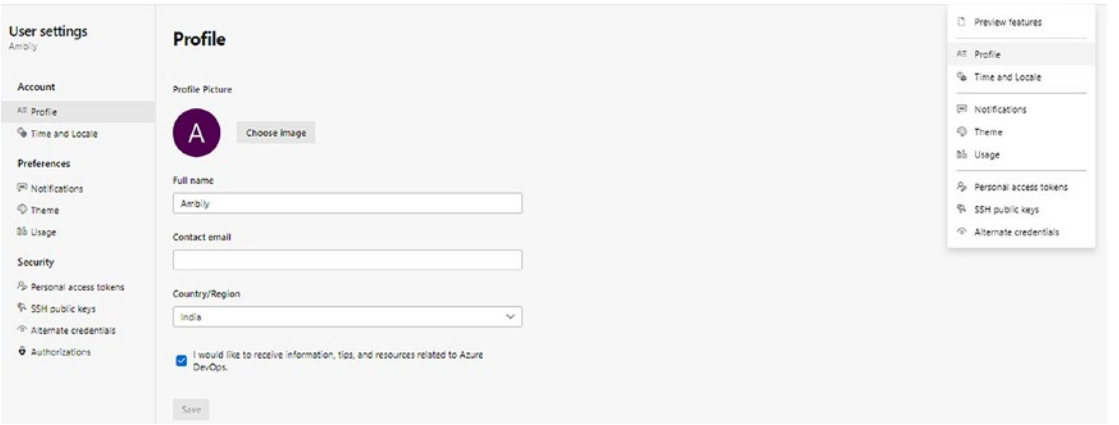


Figure 7-19. *User settings*

The following are the options available as part of the user settings:

- *Preview features*: Use this option to list all the preview features, enabling the user to enable and disable any of them.
- *Profile*: Use this option to update the profile name, email, and other details.
- *Time and Locale*: Use this option to modify the preferred language, data and time pattern, and time zone.
- *Notifications*: Use this option to list the notifications enabled for the current user. The user can disable and manage the notifications from this view.
- *Theme*: This provides two options: the Dark and Light themes.
- *Usage*: Use this option to view a usage report for the specific user.
- *Personal access tokens*: Personal tokens can be used instead of password-based integration from external systems such as Git.

- *SSH Public Keys*: Connect to your Git repos through SSH public keys when you can't use the recommended Git credential managers or personal access tokens to securely connect to Azure DevOps.
- *Alternate Credentials*: Use this option to create alternate credentials to access the Git repository.
- *Authorizations*: Use this option to give applications and providers access to the DevOps resources.

Continuous Feedback

Continuous feedback is one of the most important phases in a DevOps implementation. Continuous feedback ensures the proper flow of feedback from different stages of the application execution life cycle. In the implementation time, based on the maturity of the unit test scripts and feedback, feedback is shared with the developer as part of unit test execution. The quality assurance team, which is part of the sprint team, plays an important role in identifying the defects associated with the sprint implementations at an early stage and providing feedback in the form of bugs. *User acceptance testing* (UAT) means testing the sprint deliverables or release deliverables by end users or business stakeholders; this provides another level of feedback.

Once the application moves to the production environment, it is important to set up continuous feedback on the overall environment in the execution life cycle to improve the application quality. Different monitoring tools such as Splunk, App Dynamic, and so on, are used to monitor the production systems. These tools capture the application issues as well as the environment issues. Moreover, most of these monitoring tools support customization in terms of analyzing the logs and deriving meaningful insights, setting up automatic notifications, logging tickets directly to an integrated ticketing tool with detailed information, and so on.

Azure Application Insights

Application Insights and Azure Monitors are used mainly to monitor Azure-hosted applications and other resources. Application Insights can be integrated into Azure DevOps to log the observations as tasks or bugs in the system.

Application Insights can be integrated from the Azure release pipeline using the default template: "Azure App Service deployment with continuous monitoring".

This template internally will have three tasks to be configured for Azure connectivity. Figure 7-20 shows the three tasks that can be integrated into an existing release pipeline.

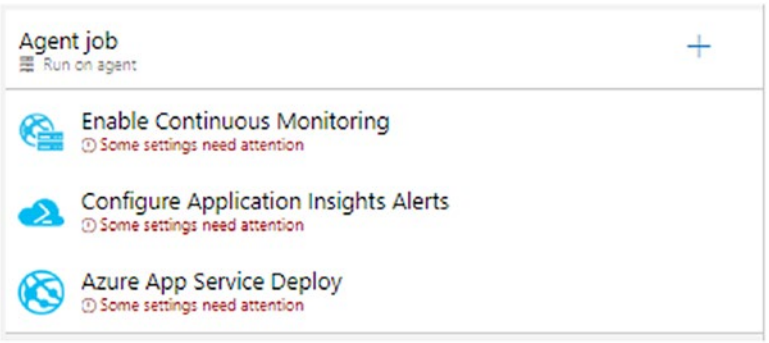


Figure 7-20. *App Insights, release tasks*

Another option is to configure the feedback mechanism in the Azure Portal. Open the Azure Portal and navigate to Application Insights. Navigate to the Work Items blade to configure the Azure DevOps connection. See Figure 7-21.

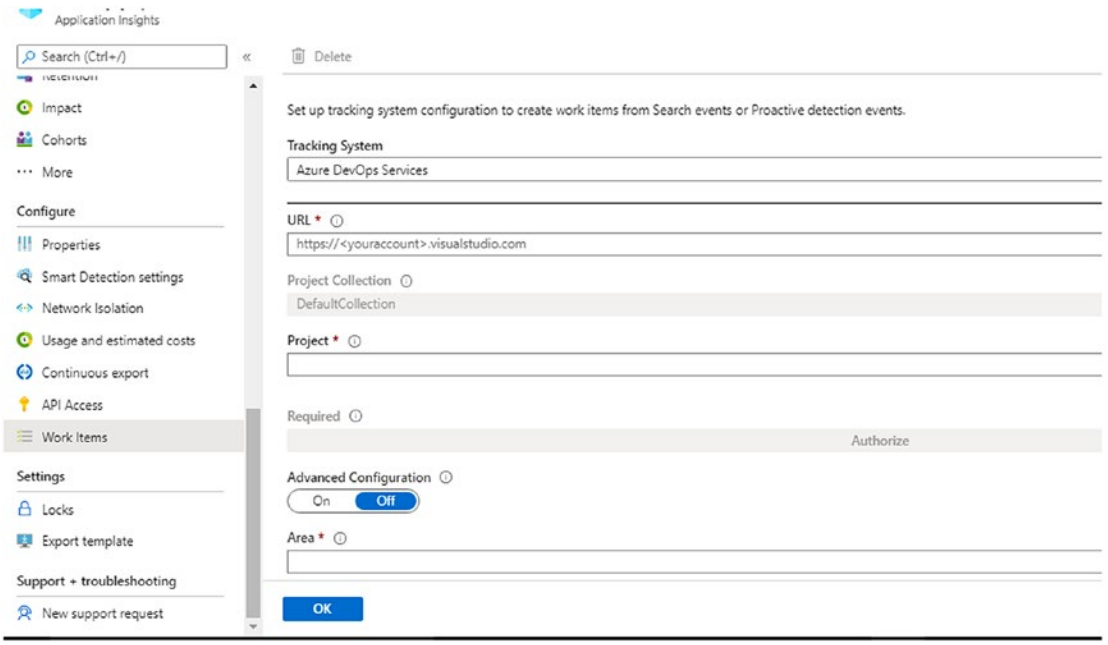


Figure 7-21. *App Insights, DevOps integration*

Summary

A wiki provides a collaboration platform for users to work together and drive new requirements and other documentations. Dashboards provide live updates about the project execution status. Continuous feedback plays an important role in fine-tuning the product quality. Azure DevOps provides a handful of features to support the collaboration, reporting, and integration needs as part of an Agile project execution.

CHAPTER 8

DevOps Architecture Blueprints

DevOps deals with the entire software development life cycle, starting from requirements gathering through post-production monitoring. The main DevOps concepts and approaches were covered in Chapter 1. In this chapter, we will look at various DevOps architectures.

The DevOps Approach for Web Applications

The DevOps approach for web application development follows the traditional model of requirements management, version control, branching and merging strategy, and continuous integration and deployment. Sometimes, this gets extended further into a continuous monitoring and feedback loop as well. In addition to the base components, most of the time the tool integration required in each process step varies from application to application.

The tools and technologies integrated across the DevOps pipeline are based on the technology stack and the requirements of the customer. XebiaLabs publishes a list of the leading DevOps tools used across the pipeline (`https://digital.ai/periodic-table-of-devops-tools`), categorized by different usages. This is a great reference for DevOps architects.

Consider a *n*-tier web application presentation layer developed in Angular and with microservices as the middle layer. The following are some of the tools that can be used for the DevOps implementation along with Azure DevOps:

- *Requirements management*: Azure Boards
- *Version control*: Azure Repo or GitLab

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6_8

- *Continuous integration*: Azure Pipeline for the build automation
  - Jasmine and Karma for unit testing
  - WhiteSource for open source vulnerability analysis
  - Nunit/SOAP UI/Postman for unit testing of the API
  - Sonar for code analysis
- *Continuous deployment*: Azure Release Pipeline
  - ARM/Terraform/Pulumi for infrastructure provisioning
  - Protractor and Karma for end-to-end testing
  - Selenium for functional testing
- *Continuous monitoring*: Azure Monitor or specifically Application Insight for monitoring

This is just the basic setup for modern application development, and the list of tool integration varies based on the customer requirements.

Microservices

Adapting a microservice architecture enables companies to release small independent units of work to production environments quickly. Specifically, Azure DevOps lets you create multiple release pipelines to automate the deployment of microservices in parallel and independent to each other. Figure 8-1 shows a high-level view of a microservices deployment approach.

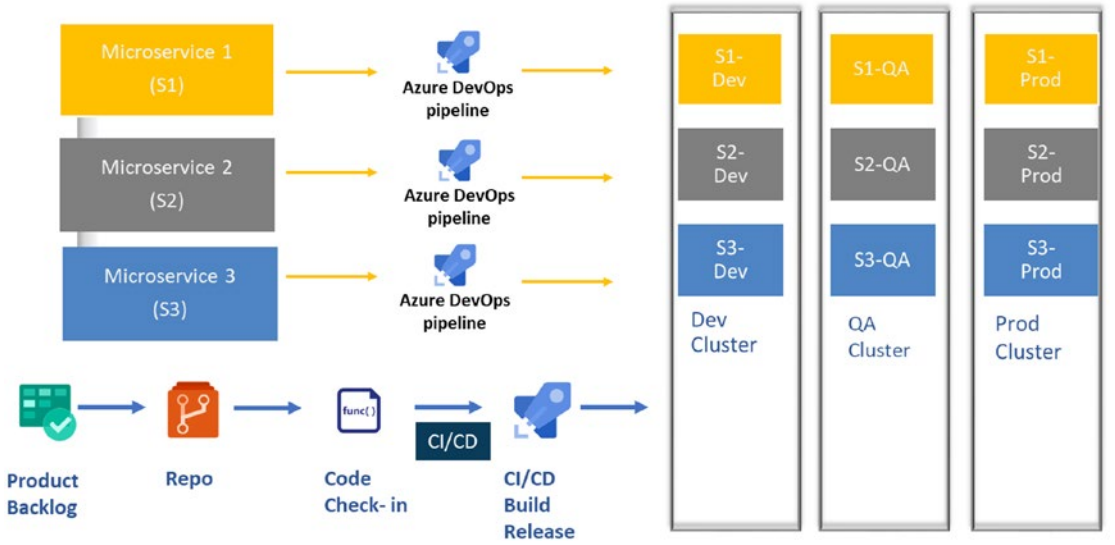


Figure 8-1. *Microservices architecture*

The DevOps Approach for Databases

There are a number of data platform tools and technologies, such as Oracle Database, Azure Data Lake Store, SSIS packages, Azure SQL, and Cosmos DB, dealing with various data requirements. To store data, there are multiple data stores, both SQL and NoSQL based. Azure DevOps supports various types of databases and associated platforms such as SSIS packages, data lake deployments, and so on.

SQL Server database

SQL Server database DevOps starts from the database project creation using Visual Studio. Visual Studio supports the development of schema files and the unit testing of the SQL schema. Moreover, proper version control using Azure Repos will be handled through Visual Studio. The database project supports the configuration of the pre-conditions and post-conditions required at deployment time. These conditions will be used to execute the cleanup activities, user creation, and other common environment-specific configurations.

DACPAC packaging is used by Azure DevOps to package the SQL builds and releases. DACPAC is a file format that contains the database objects and has an extension of `.dacpac`. Both the task-based build pipeline and YAML-based build support DACPAC-based database project handling. See Figure 8-2.

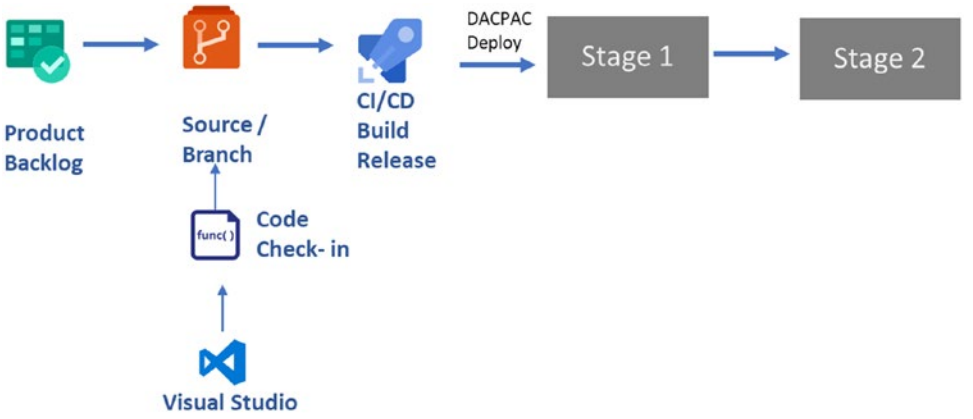


Figure 8-2. *DevOps, SQL database*

Moreover, MySQL deployment supports using MySQL tasks and YAML scripts in the Azure DevOps pipeline. SSIS packages are supported in Azure DevOps, and the package generated is similar to DACPAC with an extension of `.ispac`. Azure offers another pipeline called the Data Factory Pipeline for Azure Data Factory with all the required support for data cleaning, transformation, etc.

The DevOps Approach for Machine Learning Models

Machine learning models, are an integral part of modern application development. Machine learning models have different process steps compared to a normal application life cycle. In general, an ML life cycle will be defined using the steps shown in Figure 8-3.

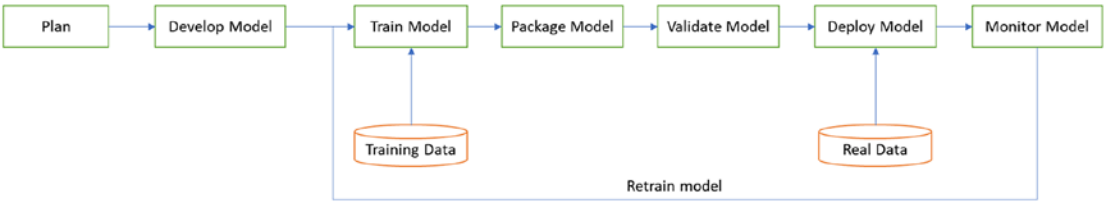


Figure 8-3. *ML pipeline*

1. *Plan*: Identify the requirements for a model, and decide on the algorithm and technology and implementation plan.
2. *Develop the model*: Develop the model using specific algorithms and languages such as Python or R.

3. *Train the model*: Feed the training data and train the model. Model training varies based on various factors such as the type of algorithm used: supervised, semisupervised or unsupervised, data volume, expected accuracy, and so on.
4. *Package the model*: Package the model for deployment. DevOps activities start from here.
5. *Validate the model*: Validate the model in an environment.
6. *Deploy the model*: Deploy the model to target environment and feed the real data on which the model needs to act on.
7. *Monitor the model*: Monitor the accuracy, response, performance, etc., of the deployed model. If required, retrain the model using additional data.

DevOps activities start from the packaging step, and sometimes before that. In general, ML models can be deployed to Azure using one of the following options:

- Manual deployment
- Azure DevOps and deployment as a container in app service
- Azure DevOps and deployment via Azure Container Instance (ACI)
- Azure DevOps and deployment via Azure Kubernetes Services (AKS)
- Azure Machine Learning Workspace Pipeline

Manual deployment requires developers to package and deploy the model to Azure services. This is not a recommended option. Azure DevOps and Azure ML Workspace Pipeline are the two options recommended based on the target users.

Table 8-1 shows the different approaches based on the user, according to https://docs.microsoft.com/en-us/azure/machine-learning/concept-ml-pipelines.

Table 8-1. *ML Approaches*

Scenario	User	Azure Offering	Strengths
Model orchestration	Data scientist	Azure Machine Learning Pipelines	Distribution, caching, code-first, reuse
Data orchestration	Data engineer	Azure Data Factory Pipelines	Strongly typed movement, data-centric activities
Code and app orchestration	App Developer/Ops	Azure DevOps Pipelines	Most open and flexible activity support, approval queues, phases with gating

If the user is a data scientist, then use the end-to-end ML pipeline via Azure Machine Learning Pipelines. This approach is called *machine learning operationalization* (MLOps).

Machine Learning Operationalization

Figure 8-4 shows the MLOps using Azure ML Pipelines.

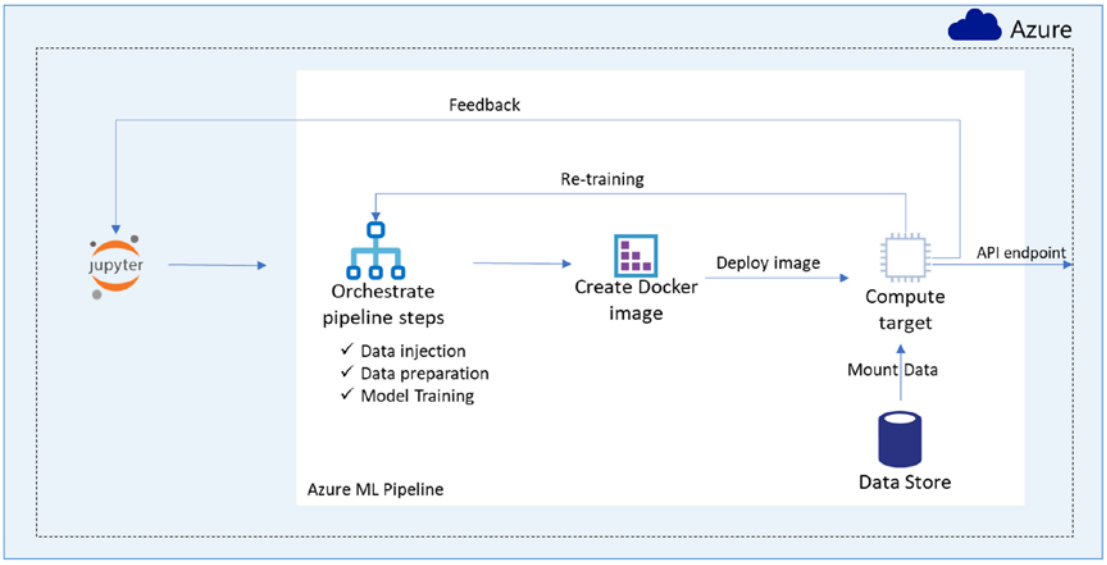


Figure 8-4. *MLOPs*

The orchestration step is important; it deals with the core activities of data injection, data cleaning, transformation, training, and so on. "Create Docker image" will create a Docker container that the user can deploy to a compute target. Under "Compute target," the

user can select either a CPU-based virtual machine or a GPU-enabled virtual machine. If the models are based on a neural network or require GPU capability, then select the GPU-based compute target. Mainly, this flow is used by data scientists, who deal with the ML life cycle.

Azure DevOps

Azure DevOps provides more control and a streamlined process to handle ML model development and deployment. Similar to any other programming language, ML models can follow the same DevOps process starting from the requirements in Azure Boards, code versioning using Azure Repos, and deployment using Azure Pipelines. Figure 8-5 depicts a high-level DevOps architecture using Azure DevOps and AKS.

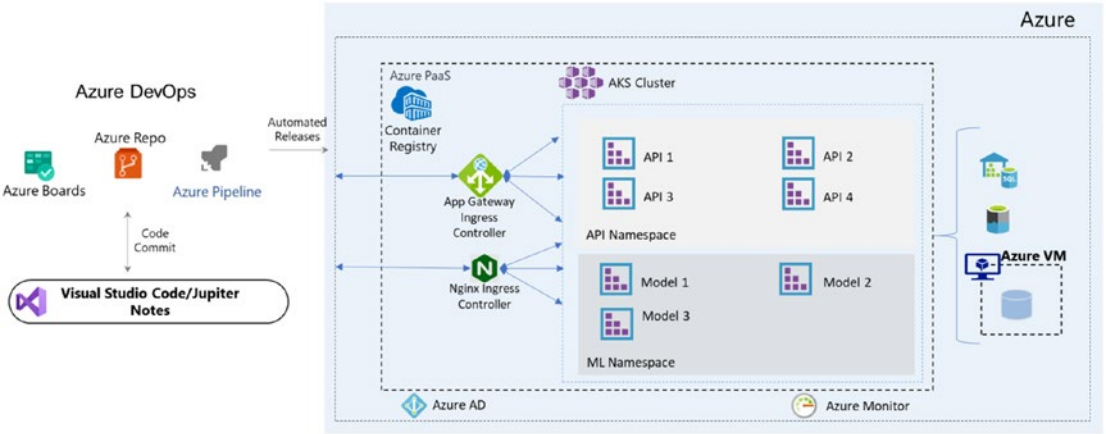


Figure 8-5. *Azure DevOps and AKS*

Azure DevOps Pipelines stores the container image in Azure Container Registry and deploys it to AKS using the release pipeline. To implement further security, we can configure ingress controllers to handle the security and routing for the models. In general, models will be protected using internal IPs and exposed for external consumption through APIs. In this case, the APIs will be configured in the ingress controller. Additional layers for the API management and app services will be added based on the system requirements. It is not recommended to expose the ingress controllers outside the Azure, instead configure API Management in front of Ingress controller to handle the incoming requests and apply security validation.

The AKS setup is used to train ML models; please refer to the reference architecture from Microsoft at `https://azure.microsoft.com/en-in/solutions/architecture/machine-learning-with-aks/`.

The DevOps Approach for COTS Applications

Commercial-off-the-shelf (COTS) applications are software products that are ready to use for a specific purpose. For example, SAP, Salesforce, and CRM are some COTS applications. The DevOps implementation in COTS applications is different from a traditional application, meaning an application specific to one customer's need.

Figure 8-6 shows the use of Azure DevOps along with Project Piper from SAP for implementing end-to-end DevOps for SAP platform applications. Project Piper is SAP's open source solution for continuous integration and delivery. For more details, please refer to https://sap.github.io/jenkins-library/.

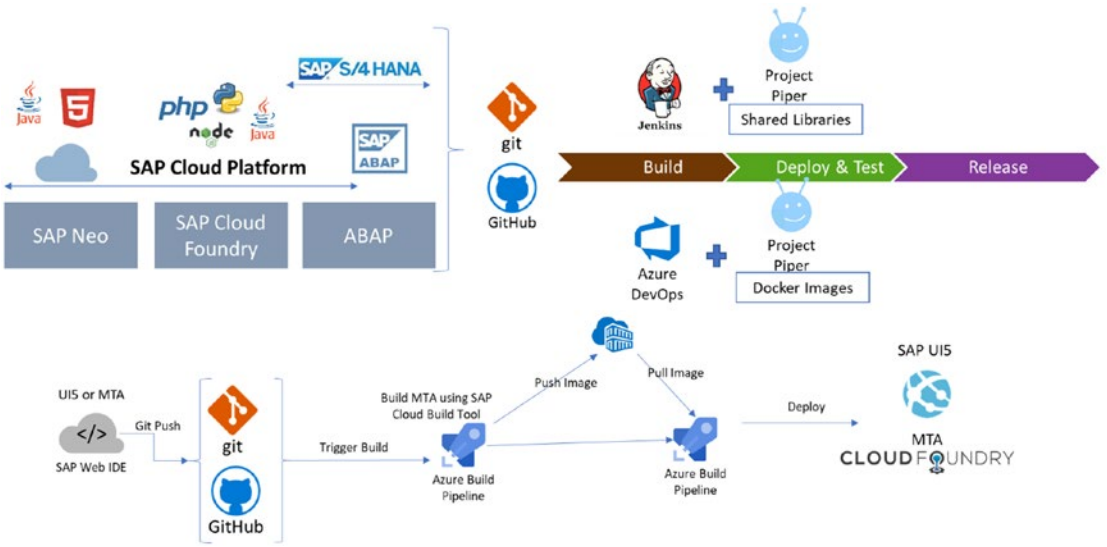


Figure 8-6. *SAP*

DevOps for COTS depends on the application support for implementing DevOps. In general, implementation of end to end DevOps for COTS is not possible completely. Currently, most of the COTS development and deployment involves few manual tasks, which will not be handled by the DevOps tools. Some of the major COTS players like SAP and Saleforce, changing the process for custom implementation and deployment to align with DevOps. We can implement some aspects of DevOps but still involve some manual elements to accomplish the complete release cycle.

The DevOps Approach for the Support Team

Frequently DevOps ends up neglecting the support team, which plays an important role in the complete application development cycle.

- How we can include the support team as part of the DevOps ecosystem?
- What are the advantages or limitations with this approach?
- Is it feasible to include a support team?
- How will be the L1, L2, and L3/L4 teams align to DevOps principles?

DevOps is defined as the collaboration of people to release value in a faster way using appropriate processes and technology adoption. The main concept of DevOps is people, especially the team culture. DevOps eliminates silos between dev and ops using practices such as smaller, much more frequent releases and blameless post-mortems, in combination with technologies and tools that empower transparent communication and cooperation across teams.

There are many variations of DevOps implementations in terms of team formations, tool usage, agile methodologies, and so on. One such variation of DevOps is BizDevOps, where the business is added to the context of DevOps and extends the boundaries. See Figure 8-7.

Figure 8-7. *BizDevOps*

The DevOps team without a support ecosystem may be able to deliver value faster to the client. But client satisfaction mainly depends on how quickly the issues or challenges are addressed. If you deliver hundreds of amazing features to the client but take more time to fix bugs and release, fail to respond to a production outage in a timely fashion, or make an effort to reduce application downtime for the business, this will impact the client satisfaction inversely.

Most of the time, the traditional support model fails to address client satisfaction in an effective manner. See Figure 8-8.

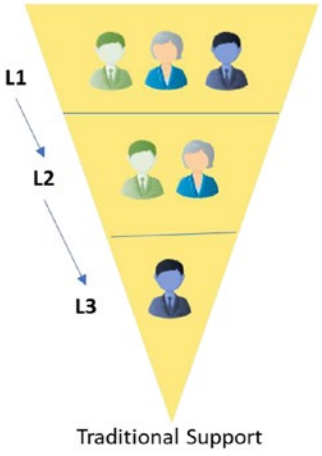


Figure 8-8. *Traditional support model*

A traditional support model involves a layered or hierarchical model, where the L1 team receives the tickets, which will then be escalated or passed to the L2 team based on analysis. The L3/L4 team will do major code changes. Some of the concerns with the current layered approach are as follows:

- Delay in ticket handling
- Tiered approach
- Less collaboration
- Ticket bouncing
- Triage by nonexperts
- Siloed teams
- Less involvement in core development activities
- Under-utilized skillset
- No change in operation
- No rotation
- Traditional team

Intelligent Swarming Support Model

The intelligent swarming support model is derived as a service innovation to address the traditional support model. The Consortium for Service Innovations (https://www.serviceinnovation.org/intelligent-swarming/) defines the intelligent swarming support model as a new support model that removes the tiered support model. It offers "improvement in operational efficiencies, employee engagement, and customer satisfaction and loyalty, and it brings with it a host of questions around practices and measurements." See Figure 8-9.

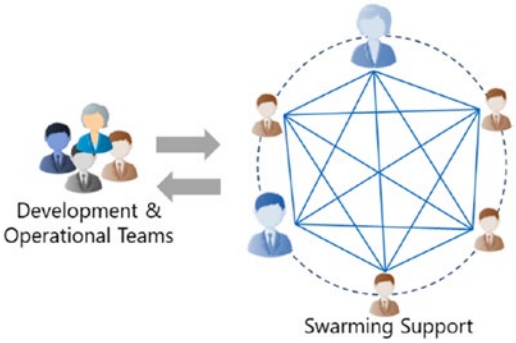


Figure 8-9. *Swarming model*

In the swarming support model, expert teams are formed as swarm teams. There may be one or more swarm teams to handle the support tickets. If the L1 support is not automated fully, then there may be one swarm team to support the L1 tickets and help in automation. The L2/L3/L4 areas will be handled by a single team of experts who understand the application functionality and technology.

Figure 8-10 shows a normal swarming support setup with a single swarm team in rotation with the development team. The automation of repeatable operations and ticket handling will enable the team to cross-skill and be part of the swarm team, which works with dev and support teams in rotation.

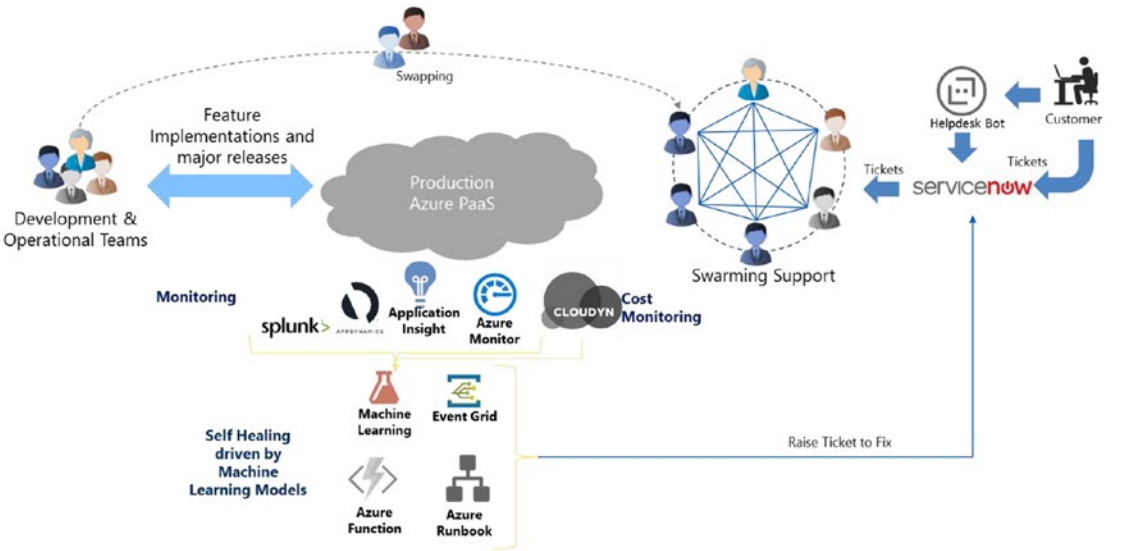


Figure 8-10. *Intelligent swarming support model*

The following are some of the benefits of the swarming support model:

- Swiftness in solving tickets
- Flat team
- Quick collaboration
- One owner
- Experts handle the tickets
- Single team; swapping with core team
- Higher team satisfaction
- Proper skill utilization
- Culture and process change
- Rotation between support and core team
- Proper team combination for successful implementation

Summary

The DevOps approach and architecture vary from application to application, but the basic components are the same. More and more COTS applications are starting to support external DevOps tools to integrate with the core deployment model. A proper DevOps architecture helps manage the release of value quickly in an agile manner.

Index

A. K K, *Azure DevOps for Web Developers*, https://doi.org/10.1007/978-1-4842-6412-6

D, E, F

G, H

I, J, K

L

M, N

O

R

S

T

U

V

W, X, Y, Z