**Azure Repos: The Keeper of Code**

**Introduction to Azure Repos:**

Azure Repos is a set of version control tools that you can use to manage your code. Whether your software project is large or small, using version control as soon as possible is a good idea.

Version control systems are software that helps you track changes you make in your code over time. As you edit your code, you tell the version control system to take a snapshot of your files. The version control system saves that snapshot permanently so you can recall it later if you need it. Use version control to save your work and coordinate code changes across your team.

Even if you're a single developer, version control helps you stay organized as you fix bugs and develop new features. Version control keeps a history of your development so that you can review and even roll back to any version of your code with ease.

Azure Repos provides two types of version control:

* [Git](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops#git): distributed version control
* [Team Foundation Version Control (TFVC)](https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops#tfvc): centralized version control

## **Git:**

Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning that your local copy of code is a complete version control repository. These fully functional local repositories make it easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server.

Git in Azure Repos is standard Git. You can use the clients and tools of your choice, such as Git for Windows, Mac, partners' Git services, and tools such as Visual Studio and Visual Studio Code.

## **Benefits:**

**Free private Git repositories, pull requests, and code search**

Get unlimited private Git repository hosting and support for TFVC that scales from a hobby project to the world’s largest repository.

Securely connect with and push code into your Git repos from any IDE, editor, or Git client.

**Collaborate to build better code**

Perform more effective Git code reviews with threaded discussion and continuous integration for each change. Use forks to promote collaboration with inner source workflows.

**Automate with built-in CI/CD**

Set up continuous integration/continuous delivery (CI/CD) to automatically trigger builds, tests, and deployments with every completed pull request using Azure Pipelines or your tools.

**Protect your code quality with branch policies**

Keep code quality high by requiring code reviewer signoff, successful builds, and passing tests before pull requests can be merged. Customize your branch policies to maintain your team’s high standards.

**Use with your favorite tools**

Use Git and TFVC repositories on Azure Repos with your favorite editor and IDE.

## **Azure Repos Concepts**

1. **Repository:** A repository is a location for our code, which is managed by version control. It supports Git and TFVC so we can create multiple repositories in a single project and various branches for each repository.
2. **Branch:** A branch is a lightweight reference that keeps a history of commits and provides a way to isolate changes for a feature or a bug fix from our master branch and other work.
3. **Branch policies:** It is an essential part of the Git workflow. We use them to help protect the critical branches is our development, as the master.
4. **Pull and Clone:** Create a complete local copy of an existing Git repo by cloning it. A pull command updates the code in our local repository with the code that is in the remote repository.
5. **Push and Commit:** A commit is a group of change saved to our local repository. We can share these changes to the remote repository by pushing.
6. **Fork:** A fork is a complete copy of a repository, including all file commits, and (optionally) branches.
7. **Git:** Git is a distributed version control system. Our local copy of code is a complete version control repository that makes it easy to work offline or remotely.
8. **Notification:** Using notification, we will receive an email whenever any changes occur to work items, code reviews, pull requests, source control files and builds.
9. **Projects:** A project provides a place where a group of people can plan, track progress, and collaborate on building software solutions.
10. **Teams:** A team corresponds to a selected set of project members. With teams, organizations can subcategorize work to better focus on all of the work they track within a project.

**Azure Repos Integration (Visual Studio Code):**

### Connect your favorite development environment

Connect your favorite development environment to Azure Repos to access your repos and manage your work. Share your code using any of the following IDE’s (Integrated Development Environment):

* [Command-line](https://learn.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-cmdline?view=azure-devops)
* [Visual Studio Code](https://marketplace.visualstudio.com/vscode)
* [Visual Studio](https://learn.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-vs?view=azure-devops)
* [Xcode](https://learn.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-xcode?view=azure-devops)
* [Eclipse](https://learn.microsoft.com/en-us/previous-versions/azure/devops/all/java/download-eclipse-plug-in)
* [IntelliJ](https://learn.microsoft.com/en-us/previous-versions/azure/devops/all/java/download-intellij-plug-in)

**Visual Studio Code** is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). Begin your journey with VS Code with these [introductory videos](https://code.visualstudio.com/docs/introvideos/overview).

<https://learn.microsoft.com/en-us/visualstudio/get-started/tutorial-open-project-from-repo?view=vs-2022>

* + - Files
    - Commits
    - Pushes
    - Branches
      * Merge Branch

**Branch Policy**

<https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies-overview?view=azure-devops>

Branch policies are an important part of the Git workflow and enable you to:

* Isolate work in progress from the completed work in your main branch
* Guarantee changes build before they get to main
* Limit who can contribute to specific branches
* Enforce who can create branches and the naming guidelines for the branches
* Automatically include the right reviewers for every code change
* Enforce best practices with required code reviewers

The following table summarizes the policies you can define to customize a branch. For an overview of all repository and branch policies and settings, see [Git repository settings and policies](https://learn.microsoft.com/en-us/azure/devops/repos/git/repository-settings?view=azure-devops).

**Policy**

**Default**

**Description**

[**Require a minimum number of reviewers**](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#require_reviewers)

Off

Require approval from a specified number of reviewers on pull requests.

[**Check for linked work items**](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#check-linked-wi)

Off

Encourage traceability by checking for linked work items on pull requests.

[**Check for comment resolution**](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#check-comment-resolution)

Off

Check to see that all comments have been resolved on pull requests.

[**Limit merge types**](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#limit-merge-types)

Off

Control branch history by limiting the available types of merge when pull requests are completed.

[**Add Build Validation policies**](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#build-validation)

Off

Add one or more policies to validate code by pre-merging and building pull request changes. Can also enable or disable policies.

[**Add Status Check policies**](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#require-approval-from-external-services)

Off

Add one or more policies to require other services to post successful status to complete pull requests. Can also enable or disable policies.

[**Automatically included reviewers**](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#include-code-reviewers)

Off

Add one or more policies to designate code reviewers to automatically include when pull requests change certain areas of code. Can also enable or disable policies.

**Branch Security**

<https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-permissions?view=azure-devops>

Set up permissions to control who can read and update the code in a branch on your Git repo. You can set permissions for individual users and groups, and inherit and override permissions as needed from your [repo permissions](https://learn.microsoft.com/en-us/azure/devops/repos/git/set-git-repository-permissions?view=azure-devops#git-repository).

**Pull Request**

Create pull requests (PRs) to change, review, and merge code in a [Git repository](https://learn.microsoft.com/en-us/azure/devops/repos/git/creatingrepo?view=azure-devops). You can create PRs from branches in the upstream repository or from branches in your [fork](https://learn.microsoft.com/en-us/azure/devops/repos/git/forks?view=azure-devops) of the repository. Your team can [review the PRs](https://learn.microsoft.com/en-us/azure/devops/repos/git/review-pull-requests?view=azure-devops) and give feedback on changes. Reviewers can step through the proposed changes, leave comments, and vote to approve or reject the PRs. Depending on [branch policies](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops) and other requirements, your PR might need to meet various criteria before you can [complete the PR](https://learn.microsoft.com/en-us/azure/devops/repos/git/complete-pull-requests?view=azure-devops) and merge the changes into the target branch.

**Clone Repos to GitHub**

Paste the Azure Repo URL into GitHub where it says “Your old repository's clone URL” and click “Begin Import”. You will be asked for your Azure DevOps credentials. Once you've entered them, the import will begin, and your code will be moved to GitHub with all the commit and revision history.

* + - Clone Repos From GitHub

**Repositories Level Security**

Discuss why and what?

**Set Git repository permissions:** [**https://learn.microsoft.com/en-us/azure/devops/repos/git/set-git-repository-permissions?view=azure-devops**](https://learn.microsoft.com/en-us/azure/devops/repos/git/set-git-repository-permissions?view=azure-devops)

You grant or restrict access to repositories to lock down who can contribute to your source code and manage other features. You can set permissions across all Git repositories by making changes to the top-level **Git repositories** entry. Individual repositories inherit permissions from the top-level **Git Repositories** entry.

<https://learn.microsoft.com/en-us/azure/devops/organizations/security/view-permissions?view=azure-devops&tabs=preview-page>

# 

# Azure Pipelines: Day 3

* + - Pipelines
    - Releases
      * Stages
      * Approval
    - Azure Pipeline with Azure App Services
    - YAML Pipeline
    - Azure Release Pipeline
    - Release pipeline
    - Tasks