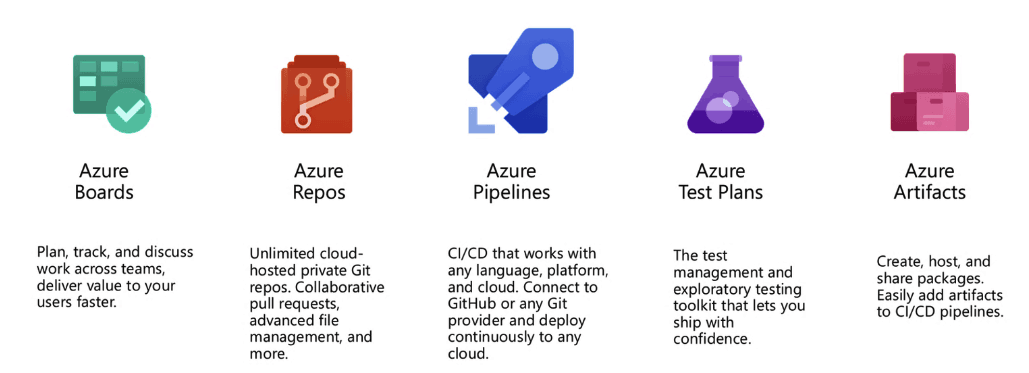
**Azure DevOps**

Azure DevOps is a comprehensive development platform provided by Microsoft that supports the entire software development lifecycle. It offers a suite of tools that help teams plan, develop, test, deliver, and monitor software projects efficiently and collaboratively. The Azure DevOps portal provides a unified interface to access these tools, making it a powerful solution for DevOps practices.



**Why Azure DevOps?**

* **All-in-One Platform:** Combines Boards, Repos, Pipelines, Artifacts, and Test Plans in one place.
* **Language & Platform Agnostic:** Supports .NET, Java, Python, Node.js, and more on Windows, Linux, and macOS.
* **CI/CD Automation**: Enables efficient Continuous Integration and Continuous Deployment.
* **Cloud-Hosted and Scalable**: Managed by Microsoft on Azure; also available on-premises as Azure DevOps Server.
* **Agile Project Management**: Built-in support for Scrum, Kanban, and custom workflows.
* **Traceability**: Full traceability from work items to commits, builds, tests, and deployments.
* **Collaboration Features:** Pull requests, code reviews, dashboards, and work item linking.
* **Extensible Ecosystem:** Integrates with tools like GitHub, Docker, Kubernetes, Jenkins, Slack, and more.
* **Cost-Effective:** Generous free tier; pay-as-you-go pricing that scales with team size.
* **Customizable Workflows:** Supports YAML pipelines, reusable templates, and REST APIs for automation.

**1. Azure Boards**

Azure Boards is an agile project management tool that helps teams plan, track, and discuss work across the development lifecycle.

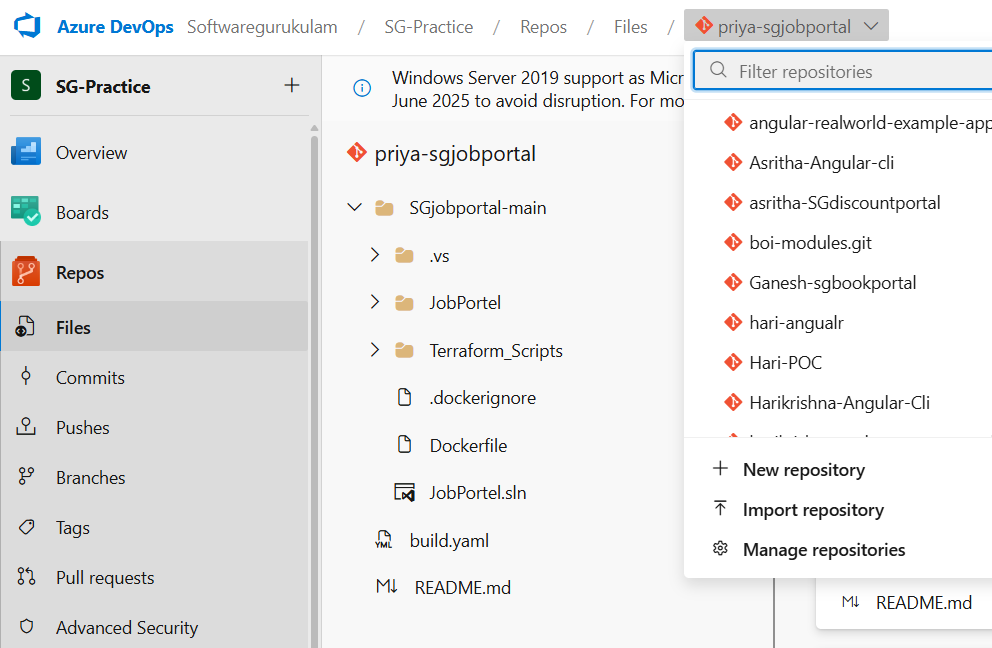
* **Work Items**: Capture tasks, bugs, user stories, and features.
* **Boards, Backlogs, and Sprints**: Visual tools for managing workflows, prioritizing work, and tracking progress.
* Fully customizable workflows and work item types.
* Seamlessly integrates with Azure Repos, Pipelines, and GitHub.

**2. Azure Repos**

**Azure Repos** is a set of version control tools that enables software development teams to manage their source code efficiently. It provides cloud-hosted repositories that support collaborative development, code review, and integration with CI/CD pipelines.

**Key Features of Azure Repos**

* **Unlimited Private Repositories**  
  Offers unlimited, secure, cloud-hosted Git repositories for your organization.
* **Branching and Merging**  
  Developers can work on feature branches independently and merge changes through pull requests.
* **Pull Requests (PRs)**
  + Facilitate code review, discussion, and approval before changes are merged.
  + Include built-in support for comments, voting, and required policies.
* **Code Search and History**  
  Enables fast search across repositories and provides a detailed history of changes for each file.
* **Branch Policies**  
  Enforce quality standards using branch protections like:
  + Required code reviews
  + Status checks (builds/tests must pass)
  + Limit who can push directly to protected branches
* **Security and Permissions**  
  Granular permission settings allow teams to control access at the repository, branch, and file level.
* **Integration with Azure Pipelines**  
  Automatically trigger builds and deployments when code is pushed or pull requests are created.
* **IDE Integration**  
  Seamless integration with tools like Visual Studio, VS Code, IntelliJ, and other Git clients.



**Typical Workflow**

1. Clone the repository locally.
2. Create a feature branch.
3. Make and commit code changes.
4. Push changes to Azure Repos.
5. Open a pull request for code review.
6. Review, approve, and merge the changes.
7. Azure Pipelines deploys the updated code.

**3. Azure Pipelines**

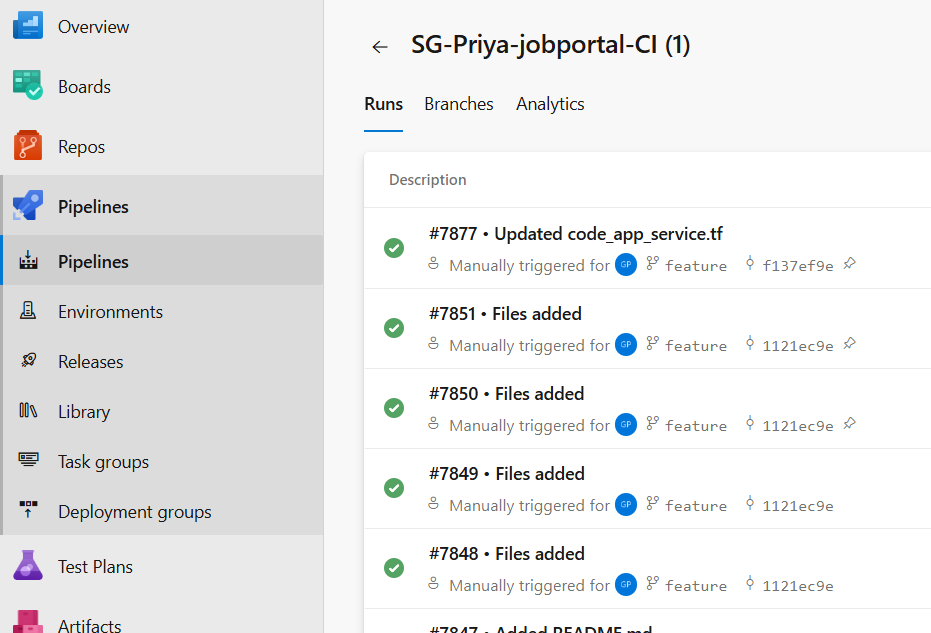
**Azure Pipelines** is a key component of Azure DevOps that enables teams to implement **Continuous Integration (CI)** and **Continuous Deployment (CD)** for any application, in any language, targeting any platform. It automates the processes of building, testing, and deploying code, ensuring faster and more reliable software delivery.

**Key Concepts**

1. **Continuous Integration (CI)**
   * Developers frequently push code to a shared repository.
   * Each code commit triggers an automated build and test pipeline.
   * Early detection of bugs and integration issues.
2. **Continuous Deployment (CD)**
   * Extends CI by automatically deploying code to staging or production after successful builds.
   * Enables rapid delivery of features and fixes to end users.

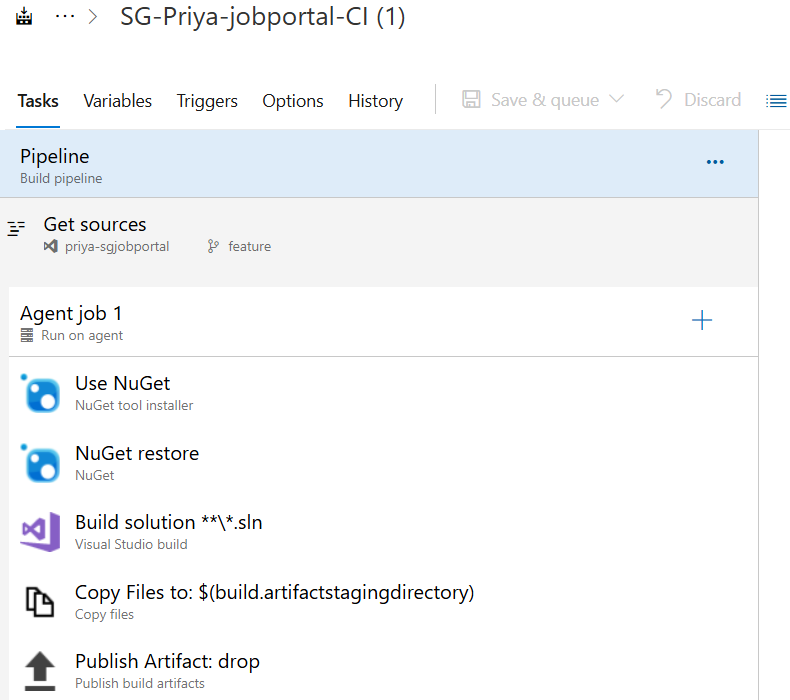
**Pipeline Types**

1. **Build Pipelines**
   * Compile code, run tests, and produce build artifacts (e.g., DLLs, binaries, deployment packages).
   * Example: Compile a .NET project and publish the output.
2. **Release Pipelines**
   * Deploy applications across multiple environments (e.g., Dev, QA, Production).
   * Includes stages, tasks, approval gates, and rollback options.



**Pipeline Definition Formats**

1. **Classic Editor (GUI-Based)**
   * Visual designer for building pipelines using drag-and-drop tasks.
   * Suitable for beginners or simple workflows.



1. **YAML Pipelines (Pipeline as Code)**
   * Written in YAML files stored in the repository.
   * Provides version control, reusability, and easier collaboration.

*YAML script for building and publishing .NET application artifacts:*

###############

trigger:

- feature

pool:

 vmImage: windows-latest

stages:

- stage: 'Build\_Stage'

  jobs:

  - job: 'BuildJob'

    steps:

    - task: NuGetToolInstaller@1

      inputs:

        versionSpec:

      displayName: 'NugetToolInstaller'

    - task: NuGetCommand@2

      inputs:

        command: 'restore'

        restoreSolution: '\*\*/\*.sln'

        feedsToUse: 'select'

    - task: MSBuild@1

      inputs:

        solution: '\*\*/\*.sln'

        platform: 'any cpu'

        configuration: 'release'

        clean: true

    - task: CopyFiles@2

      inputs:

        SourceFolder: '$(agent.builddirectory)'

        Contents: '\*\*/\*.tf'

        TargetFolder: '$(build.artifactstagingdirectory)'

    - task: PublishBuildArtifacts@1

      inputs:

        PathtoPublish: '$(Build.ArtifactStagingDirectory)'

        ArtifactName: 'drop'

        publishLocation: 'Container'

**Pipeline Components**

* **Triggers**: Define when a pipeline runs (e.g., on commit, schedule).
* **Agents**: Run the tasks in the pipeline; can be Microsoft-hosted or self-hosted.
* **Stages**: Logical divisions in the pipeline (e.g., Build, Test, Deploy).
* **Jobs**: Groups of steps run sequentially on an agent.
* **Steps/Tasks**: Individual actions like restore, build, test, publish, deploy.
* **Artifacts**: Output from build pipelines used in deployments.
* **Variables**: Used to manage dynamic values across stages and jobs.
* **Deployment groups:** Used to establish a connection between Azure DevOps and Virtual Machine.
* **Service Connection:** Used to establish a connection between Azure DevOps and other cloud services.

**Supported Technologies**

* Programming languages: .NET, Java, Python, Node.js, Go, PHP, Ruby, etc.
* Platforms: Windows, macOS, Linux.
* Containers: Docker, Kubernetes, Helm.
* Cloud providers: Azure, AWS, GCP (via plugins/extensions).

**4. Azure Artifacts**

Azure Artifacts provides integrated package management for software dependencies.

* **Package Types**: Supports NuGet, npm, Maven, Python, and Universal Packages.
* **Feeds**: Create and share feeds to manage and distribute packages securely.
* **Retention Policies**: Automatically clean up old or unused packages.
* **Integration**: Works seamlessly with Azure Pipelines to manage build artifacts.