Azure DevOps provides several tools you can use for better team collaboration. It also has tools for automated build processes, testing, version control, and package management.

## [Azure Boards](https://azure.microsoft.com/products/devops/boards/" \t "https://learn.microsoft.com/en-us/training/modules/get-started-with-devops/az-portal): These are agile tools that help us plan, track, and discuss our work, even with other teams.

## [Azure Pipelines](https://azure.microsoft.com/products/devops/pipelines/" \t "https://learn.microsoft.com/en-us/training/modules/get-started-with-devops/az-portal):

Azure Pipelines is a cloud service that automatically builds and tests your code project and makes it available to other users. It works with just about any language or project type.

Azure Pipelines combines continuous integration (CI) and continuous delivery (CD) to test and build your code and ship it to any target constantly and consistently.

## These will let us build, test, and deploy with CI/CD that works with any language, platform, and cloud.

[Azure Test Plans](https://azure.microsoft.com/products/devops/test-plans/):

These are manual and exploratory testing tools.

## Explore best practices for source control

Completed100 XP

* 2 minutes
* ****Make small changes****. In other words, commit early and commit often. Be careful not to commit any unfinished work that could break the build.
* ****Do not commit personal files****. It could include application settings or SSH keys. Often personal files are committed accidentally but cause problems later when other team members work on the same code.
* ****Update often and right before pushing to avoid merge conflicts****.
* ****Verify your code change before pushing it to a repository****; ensure it compiles and tests are passing.
* ****Pay close attention to commit messages, as it will tell you why a change was made****. Consider committing messages as a mini form of documentation for the change.
* ****Link code changes to work items****. It will concretely link what was created to why it was created—or modified by providing traceability across requirements and code changes.
* ****No matter your background or preferences, be a team player and follow agreed conventions and workflows****. Consistency is essential and helps ensure quality, making it easier for team members to pick up where you left off, review your code, debug, and so on.

Using version control of some kind is necessary for any organization, and following the guidelines can help developers avoid needless time spent fixing errors and mistakes.

## **Use CI and CD for your project**

Continuous integration is used to automate tests and builds for your project. CI helps to catch bugs or issues early in the development cycle when they're easier and faster to fix. Items known as artifacts are produced from CI systems. The continuous delivery release pipelines use them to drive automatic deployments.

Continuous delivery is used to automatically deploy and test code in multiple stages to help drive quality. Continuous integration systems produce deployable artifacts, which include infrastructure and apps. Automated release pipelines consume these artifacts to release new versions and fixes to the target of your choice.

****Continuous integration (CI)****

Increase code coverage.

Build faster by splitting test and build runs.

Automatically deploy code to production.

Run tests continually.

****Continuous delivery (CD)****

Ensure deployment targets have the latest code.

Automatically ensure you don't ship broken code.

Use tested code from the CI process.

## **Use Azure Pipelines for CI and CD**

There are several reasons to use Azure Pipelines for your CI and CD solution. You can use it to:

* Work with any language or platform.
* Deploy to different types of targets at the same time.
* Integrate with Azure deployments.
* Build on Windows, Linux, or macOS machines.
* Integrate with GitHub.
* Work with open-source projects.

# Azure Pipelines:

## **Build automation and continuous integration**

The pipeline starts by building the binaries to create the deliverables passed to the following stages. New features implemented by the developers are integrated into the central code base, built, and unit tested. It's the most direct feedback cycle that informs the development team about the health of their application code.

## **Test automation**

The new version of an application is rigorously tested throughout this stage to ensure that it meets all wished system qualities. It's crucial that all relevant aspects—whether functionality, security, performance, or compliance—are verified by the pipeline. The stage may involve different types of automated or (initially, at least) manual activities.

## **Deployment automation**

A deployment is required every time the application is installed in an environment for testing, but the most critical moment for deployment automation is rollout time.

Since the preceding stages have verified the overall quality of the system, It's a low-risk step.

The deployment can be staged, with the new version being initially released to a subset of the production environment and monitored before being rolled out.

The deployment is automated, allowing for the reliable delivery of new functionality to users within minutes if needed.

## **Your pipeline needs platform provisioning and configuration management**

The deployment pipeline is supported by platform provisioning and system configuration management. It allows teams to create, maintain, and tear down complete environments automatically or at the push of a button.

Automated platform provisioning ensures that your candidate applications are deployed to, and tests carried out against correctly configured and reproducible environments.

It also helps horizontal scalability and allows the business to try out new products in a sandbox environment at any time.

## **Orchestrating it all: release and pipeline orchestration**

The multiple stages in a deployment pipeline involve different groups of people collaborating and supervising the release of the new version of your application.

Release and pipeline orchestration provide a top-level view of the entire pipeline, allowing you to define and control the stages and gain insight into the overall software delivery process.

By carrying out value stream mappings on your releases, you can highlight any remaining inefficiencies and hot spots and pinpoint opportunities to improve your pipeline.

These automated pipelines need infrastructure to run on. The efficiency of this infrastructure will have a direct impact on the effectiveness of the pipeline.

Introduction DevSecOps

# Azure DevOps - Udemy









