100 XP

# How to create a CI pipeline with Azure DevOps

4 minutes

Azure Pipelines supports continuous integration (CI) and continuous delivery (CD). CI/CD pipelines allow you to regularly and consistently test, build, and ship code.

- You define a *build pipeline* to build and test your code, and then to publish artifacts.
- You also define a *release pipeline* to consume and deploy those artifacts to deployment targets.

You can define pipelines using YAML syntax or through the user interface (classic). YAML is a human-readable data-serialization language. YAML is often used for configuration files and applications where data is stored or transmitted. A pipeline is one or more stages that describe a CI/CD process. Pipelines have a hierarchy of stages and jobs. Stages are the primary divisions in a pipeline. A stage has one or more jobs, which are units of work assignable to the same machine. You can arrange both stages and jobs into dependency graphs. A job is a linear series of steps.

In the CI stage, you define your pipeline in a YAML file called azure-pipelines.yml with the rest of your app. The pipeline is versioned with your code. It follows the same branching structure. You get validation of your changes through code reviews in pull requests and branch build policies. Every branch you use can modify the build policy by modifying the azure-pipelines.yml file.

## Steps in a pipeline

The overall steps are:

1. Configure Azure Pipelines to use your Git repo.
2. Edit your azure-pipelines.yml file to define your build.
3. Push your code to your version control repository. This action kicks off the default trigger to build and deploy and then monitor the results.
4. Your code is now updated, built, tested, and packaged. It can be deployed to any target.

## Module complete:

Unlock achievement