Setting up a CI/CD pipeline for a project in Azure DevOps involves configuring both a Continuous Integration (CI) pipeline and a Continuous Deployment (CD) pipeline.

1. Create an Azure DevOps Project and Repository:

**[Create an Organization and Project:](https://www.google.com/search?cs=0&sca_esv=c7bde4720c8231c4&q=Create+an+Organization+and+Project&sa=X&ved=2ahUKEwjmvfv1rcmPAxU4L1kFHYDZMzQQxccNegQIBxAD&mstk=AUtExfDty8osPI55JfYyrqT6V1kh3NcubmKHrEjQfSepAzxZyBY_AUVCdsIfxtAoJ4VU3H_xcssmqsry0pUFmVRxSZ8OfXuFWO4lKu2c0_CU3uhsk_LArQqkZwVptf_bDi0um08&csui=3" \t "_blank)**

[.](https://www.google.com/search?cs=0&sca_esv=c7bde4720c8231c4&q=Create+an+Organization+and+Project&sa=X&ved=2ahUKEwjmvfv1rcmPAxU4L1kFHYDZMzQQxccNegQIBxAD&mstk=AUtExfDty8osPI55JfYyrqT6V1kh3NcubmKHrEjQfSepAzxZyBY_AUVCdsIfxtAoJ4VU3H_xcssmqsry0pUFmVRxSZ8OfXuFWO4lKu2c0_CU3uhsk_LArQqkZwVptf_bDi0um08&csui=3" \t "_blank)

If not already present, create an Azure DevOps organization and then a new project within it.

**[Set Up Code Repository:](https://www.google.com/search?cs=0&sca_esv=c7bde4720c8231c4&q=Set+Up+Code+Repository&sa=X&ved=2ahUKEwjmvfv1rcmPAxU4L1kFHYDZMzQQxccNegQICxAD&mstk=AUtExfDty8osPI55JfYyrqT6V1kh3NcubmKHrEjQfSepAzxZyBY_AUVCdsIfxtAoJ4VU3H_xcssmqsry0pUFmVRxSZ8OfXuFWO4lKu2c0_CU3uhsk_LArQqkZwVptf_bDi0um08&csui=3" \t "_blank)**

[.](https://www.google.com/search?cs=0&sca_esv=c7bde4720c8231c4&q=Set+Up+Code+Repository&sa=X&ved=2ahUKEwjmvfv1rcmPAxU4L1kFHYDZMzQQxccNegQICxAD&mstk=AUtExfDty8osPI55JfYyrqT6V1kh3NcubmKHrEjQfSepAzxZyBY_AUVCdsIfxtAoJ4VU3H_xcssmqsry0pUFmVRxSZ8OfXuFWO4lKu2c0_CU3uhsk_LArQqkZwVptf_bDi0um08&csui=3" \t "_blank)

Link your project to a code repository (Azure Repos, GitHub, etc.) and ensure your project's code is pushed to this repository.

2. Configure the Continuous Integration (CI) Pipeline:

* **Create a New Pipeline:**

In your Azure DevOps project, navigate to "Pipelines" > "Pipelines" and select "New pipeline."

* **Choose Repository and Template:**

Select your source code repository and choose a suitable template (e.g., [ASP.NET](https://dotnet.microsoft.com/en-us/apps/aspnet) Core, Node.js, Maven) or start with an empty job to define your build steps manually.

* **Define Build Tasks:**

Add tasks to compile your code, run unit tests, perform code analysis, and package artifacts (e.g., NuGet packages, Docker images, deployable archives).

* **Configure Triggers:**

Enable continuous integration by setting up triggers to automatically run the pipeline on code changes (e.g., on every push to a specific branch).

* **Save and Run:**

Save your build pipeline and run it to ensure it builds successfully and generates the necessary artifacts.

3. Configure the Continuous Deployment (CD) Pipeline (Release Pipeline):

* **Create a New Release Pipeline:**

Navigate to "Pipelines" > "Releases" and select "New release pipeline."

* **Select Artifacts:**

Link your build pipeline as an artifact source, ensuring the CD pipeline uses the outputs of your CI build.

* **Define Stages and Environments:**

Create stages representing your deployment environments (e.g., Development, Staging, Production).

* **Add Deployment Tasks:**

Within each stage, add tasks to deploy your application to the target environment (e.g., Azure App Service deployment, Kubernetes deployment, copying files to a VM).

* **Configure Pre-deployment and Post-deployment Approvals:**

Implement approval gates to control deployments to sensitive environments, requiring manual approval before proceeding.

* **Set Up Triggers:**

Configure continuous deployment triggers to automatically initiate releases upon new successful builds or on a scheduled basis.

* **Implement Rollback Mechanisms:**

Consider adding tasks or configurations for rollback in case of deployment failures.

* **Save and Deploy:**

Save your release pipeline and initiate a release to test the deployment process.

4. Monitor and Iterate:

* **Monitor Pipeline Runs:** Utilize Azure DevOps' monitoring tools to track the status of your builds and releases.
* **Set Up Alerts:** Configure alerts for pipeline failures or other critical events.
* **Refine and Optimize:** Continuously review and optimize your pipeline for efficiency, reliability, and security.

demo-video: <https://www.youtube.com/watch?v=lC5c7MyA_Fg&ab_channel=BestITCourses>