Microsoft Dynamics 365 Customer Engagement

DevOps 1 Day with Lab

Lab 2: Create Your First Azure Build Pipeline

Student Lab Manual

Conditions and Terms of Use

Copyright and Trademarks

Table of Contents

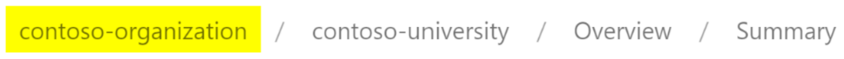
Introduction

In this lab, we will build an Azure Pipeline leveraging YAML, a superset of JSON, to pack the contents of our solution into a deployable artifact.

Install the Power Platform Build Tools for Azure DevOps

The pipeline we will be creating will depend on the PowerApps Build Tools for Azure DevOps. This is an extension available for free in the Visual Studio Marketplace and provides a suite of tasks related to configuring deployments for Dynamics 365 Customer Engagement applications.

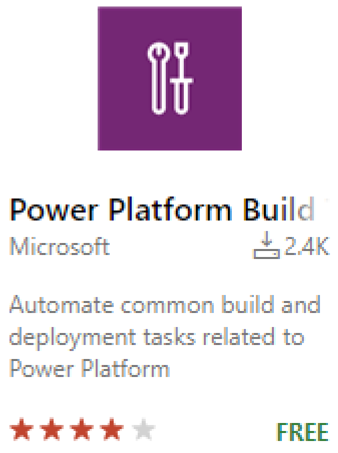
1. In Azure DevOps, click on your organization name in the breadcrumbs at the top of the browser window.



contoso-organization / contoso-university / Overview / Summary

2. At the bottom of the left-hand panel, click Organization Settings.
3. In the left-hand panel, click Extensions in the general section.
4. In the top-right corner, click Browse marketplace.
5. Search for "Power Platform Build Tools."

6. The first result should indicate Microsoft as the publisher. Click on it.



Power Platform Build
Microsoft 2.4K

Automate common build and deployment tasks related to Power Platform

★★★★☆ FREE

7. Click Get it free.
8. Your Azure DevOps organization should already be selected. Click Install.
9. After the extension finishes installing, click Proceed to organization.
10. Click on the contoso-university project to move onto the next part of the lab.

Create a Pipeline

You have two options to create your first pipeline, both of which will produce the same result:

- **Option 1:** Create a pipeline using Task Assistant.
  This option is recommended for non-developers.
- **Option 2:** Create a pipeline starting with a YAML pipeline definition file.
  This option is recommended for developers.

Option 1: Create a pipeline using Task Assistant

Azure DevOps includes an in-browser editor for designing YAML pipelines.

1. Click Pipelines.
2. Click New pipeline. If this is your first pipeline you may see Create Pipeline instead.
3. Click Azure Repos Git YAML Free private Git repositories, pull requests, and code search.
4. Click contoso-university.

5. Click Starter pipeline Start with a minimal pipeline that you can customize to build and deploy your code.. This will bring up the pipeline editor. Note that there were many pre-build pipelines that Microsoft has built, but at the time of writing, there are none dedicated to Dynamics 365 Customer Engagement.

6. Before we start editing the pipeline, give it a meaningful name. Click **azure-pipelines.yml ***, type *pipeline.yml*, and hit Enter.

7. Delete the comments and empty line on lines 1-5. The trigger keyword should be on line 1.

8. On line 5, start by changing the `vmImage` property from `'ubuntu-latest'` to `'windows-latest'`. Line 5 should now look like this:

   ```
   vmImage: 'windows-latest'
   ```

   Note: At this time, the Power Platform Build Tools only work on windows containers.

9. Delete the current steps. Select lines 8-15 and press Backspace or Delete.

   Your YAML file should look like this:

   ```
   trigger:
   - master

   pool:
     vmImage: 'windows-latest'

   steps:
   ```

   Note: the `trigger` section declares that this pipeline should be run once; any changes are committed to the `master` branch of the repository hosting this YAML file.

10. In the top-right corner, click Show assistant to open the task assistant.

11. Search for and select "Power Platform Tool Installer:"

    a. This will open a dialog to provide input parameters for the Power Platform Tool Installer task. Keep the default values, make sure cursor is on line 8, and click Add.

    b. A generated YAML task will be inserted at the bottom of the file. Deselect the YAML, add a new line, and delete any white space added to that line. The cursor should now be at the beginning of line 11.

    The Power Platform Tool Installer tasks installs several tools onto the container that the job is running on. The tool we are most interested in here is SolutionPackager.

12. In the task assistant, search for and select "Power Platform Pack Solution."

c. Enter the following values for the parameters:

   Source Folder of Solution to Pack: contents
   Solution Output File: packed-solution/contoso_university_core.zip
   Type of Solution: Both

d. Click Add.

e. Similarly, to the previous step, deselect the auto generated YAML, add a new line after line 15, and delete any whitespace. The cursor should now be on line 16 at the beginning of the line.

The Power Platform Pack Solution task does the opposite of the steps we completed in Lab 1. That is, it takes an unpacked contents folder and converts it to a pair of compressed zip files that are ready to be deployed into a Dynamics 365 Customer Engagement environment.

13. In the task assistant, search for and select "Publish pipeline artifacts."

f. Enter the following values for the parameters:

   File or directory path: packed-solution
   Artifact name: packed-solution
   Artifact publish location: Azure Pipelines

g. Click Add.

This is a generic Azure DevOps task that takes deployable files, such as the zip that we've generated, and makes them available to be consumed by downstream processes.

Your YAML should look like this:

```yaml
trigger:
- master

pool:
  vmImage: 'windows-latest'

steps:
- task: PowerPlatformToolInstaller@0
  inputs:
    DefaultVersion: true
- task: PowerPlatformPackSolution@0
  inputs:
    SolutionSourceFolder: 'contents'
    SolutionOutputFile: 'packed-solution/contoso_university_core.zip'
    SolutionType: 'Both'
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: 'packed-solution'
    artifact: 'packed-solution'
    publishLocation: 'pipeline'
```

14. Click Save and run, keep the default values, and click Save and run again.

You have now created and run your first pipeline. Proceed to the Conclusion to see how to examine the pipeline's output.

Option 2: Create a pipeline starting with a YAML pipeline definition file.

While the GUI presented in Azure DevOps is a great place to get started, eventually, developers will want to do the majority of YAML coding from within their favorite text editor.

The Azure Pipelines extension for VS Code, provides IntelliSense for YAML files. It's still under development, but it's worth installing and testing out.

1. Install the Azure Pipelines extension. You can click the link above, or from VS Code, hit Ctrl+Shift+X, search for "Azure Pipelines," and click "Install."

2. In your repository, create a folder called *.vscode*. Inside that folder, create a file called *settings.json*. Copy/paste the following contents into the file and save it.

```json
{
    "files.associations": {
        "*.yml": "azure-pipelines"
    }
}
```

   This will instruct VS Code to use the azure-pipelines extension for all YAML files in the repository.

3. Commit this addition to the repository. If it's not open already, press Ctrl+` to open the Terminal, and type the following commands:

```
git add .
git commit -m "added vs code settings"
```

4. Add a new file to the repository called *pipeline.yml*. and add the following blocks of code (alternatively, copy the contents of copy-to-repo folder into your repository):

   a.

```yaml
trigger:
- master
```

   This declares that the pipeline should run every time code is committed to the `master` branch.

   b.

```yaml
pool:
  vmImage: 'windows-latest'
```

   This tells Azure DevOps to run the pipeline on the most recent windows container version, which is currently Windows Server 2019.

   c.

```yaml
steps:
- task: PowerPlatformToolInstaller@0
  inputs:
    DefaultVersion: true
```

   This defines the `steps` property and adds the first task to the pipeline, which is to install tooling relevant to the PowerApps Build Tools. Specifically, we are going to need SolutionPackager.

   Note: If you read Option 1, you will notice that the YAML generated for this step includes inputs specifying the version of each tool to install. These inputs are optional and will default to the most recent versions.

d.

```
- task: PowerPlatformPackSolution@0
  inputs:
    SolutionSourceFolder: 'contents'
    SolutionOutputFile: 'packed-solution/contoso_university_core.zip'
    SolutionType: 'Both'
```

This runs SolutionPackager to take the `contents` folder and pack it into a pair of deployable `zip` files in a folder called `packed-solution`. It effectively performs the opposite steps that we did manually in Lab 1.

e.

```
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: 'packed-solution'
    artifact: 'packed-solution'
    publishLocation: 'pipeline'
```

This step takes the contents of the `packed-solution` folder and makes them available after the completion of this job for consumption in downstream processes.

5. Commit and push your changes.

```
git add .
git commit -m "created pipeline.yml"
git push
```

6. In the browser window, from your Azure DevOps project, click Pipelines.
7. Click New pipeline. If this is your first pipeline you may see Create Pipeline instead.
8. Click Azure Repos Git YAML Free private Git repositories, pull requests, and code search.
9. Click contoso-university.
10. Click Existing Azure Pipelines YAML file Select an Azure Pipelines YAML file in any branch of the repository. A panel will open on the right side of the window.
11. For Branch, keep the default value of master. Set Path to /pipeline.yml.
12. Click Continue.

13. Click Run.

You should now be able to watch the steps of your pipeline run.

Conclusion

The steps of your pipeline should run one at a time, being marked with a ✓ upon successful completion. Eventually, when the entire pipeline completes, click 1 published under the Artifacts header, and then click packed-solution to view the freshly generated artifacts.

Congratulations! You have now set up a pipeline that takes changes you commit to source control, your new source of truth, and converts them into deployable artifacts. Soon, you'll be creating processes to take those artifacts and deploy them directly into a Dynamics 365 Customer Engagement environment.