100 XP

Exercise - Create a CI pipeline for IoT Edge with Azure DevOps

15 minutes

Create Azure resources

Azure DevOps Projects creates a CI/CD pipeline in Azure DevOps. First, you'll need to create cloud services that will be used for the module.

1. Sign in to the Microsoft Azure portal.
2. Select the following **Deploy to Azure** button. The **Custom deployment** panel appears.

   Deploy to Azure

3. On the **Basics** tab, fill in the following values for each setting.

Expand table

Setting	Value
Deployment scope	
Subscription	Select your subscription
Resource group	Select a name from the dropdown, or select the **Create New** link, and in the **Name** text box, enter a name.
Parameters	
Region	Select the same region as your resource group

Setting	Value
Resource Name Suffix	Enter a globally unique value

4. Select **Review + create**, and then select **Create** to deploy your resources to Azure.

> ⓘ **Note**
>
> If you encounter any issues in the deployment, we advise deleting the created resource group (if any), and retrying with a new value for the **Resource Name Suffix** parameter.

5. After deployment successfully completes, select **Go to resource group** to review your resources.

Create an Azure DevOps project

1. If you don't have an Azure DevOps organization, you should follow the steps to create one for free.
   a. Open Azure Pipelines and choose **Start free**.
   b. Sign in/up with Microsoft.
   c. Give a name to your organization, and create it.

2. From the **Azure DevOps** page, on the upper right corner, select **New project**. The **Create new project** pane appears.

3. In the **Project name** text box, enter a project name.

4. In the **Description** text box, enter descriptive text about your new project.

5. Under **Visibility**, select either private or public.

6. Select **Create**. The project's welcome page appears.

7. In the left menu pane, select **Repos**, and in the **Import a repository** box, select **Import**. The **Import a Git repository** pane appears.

8. In the **Clone URL** field, enter this URL, and select **Import**.

```
https://github.com/MicrosoftDocs/mslearn-oxford-implement-cicd-iot-edge.git
```

Create a CI pipeline

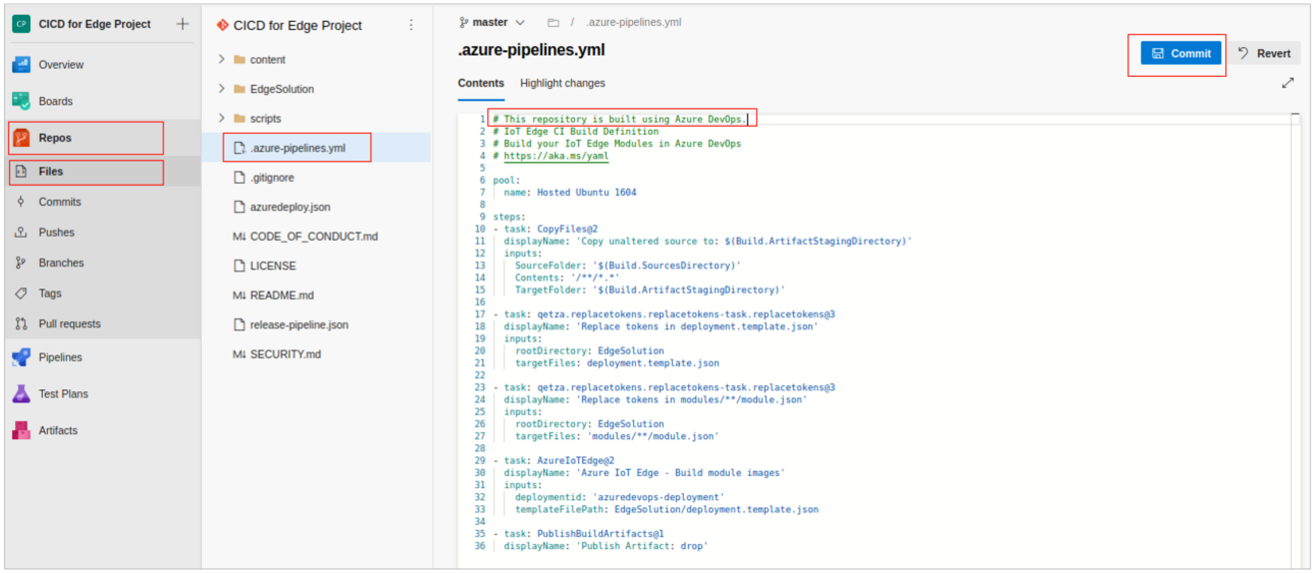
This repository contains an Azure DevOps build definition, which is preconfigured to build the included EdgeSolution in azure-pipelines.yml. This build definition relies on an external plugin called Replace Tokens.

1. Begin by installing the **Replace Tokens** task from the Visual Studio Marketplace by visiting this link and selecting **Get it free**. Then, select **Install** to install the token into the organization that contains your newly created Azure DevOps project.

2. After this task is successfully installed, return to the Azure DevOps project, and select **Repos** > **Files**. Select the edit icon to edit the `.azure-pipelines.yml` file.

3. Add the following comment to the top of the file, as shown below.

```
# This repository is built using Azure DevOps.
```



```
# This repository is built using Azure DevOps.
# IoT Edge CI Build Definition
# Build your IoT Edge Modules in Azure DevOps
# https://aka.ms/yaml

pool:
  name: Hosted Ubuntu 1604

steps:
- task: CopyFiles@2
  displayName: 'Copy unaltered source to: $(Build.ArtifactStagingDirectory)'
  inputs:
    SourceFolder: '$(Build.SourcesDirectory)'
    Contents: '/**/*.*'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'

- task: qetza.replacetokens.replacetokens-task.replacetokens@3
  displayName: 'Replace tokens in deployment.template.json'
  inputs:
    rootDirectory: EdgeSolution
    targetFiles: deployment.template.json

- task: qetza.replacetokens.replacetokens-task.replacetokens@3
  displayName: 'Replace tokens in modules/**/module.json'
  inputs:
    rootDirectory: EdgeSolution
    targetFiles: 'modules/**/module.json'

- task: AzureIoTEdge@2
  displayName: 'Azure IoT Edge - Build module images'
  inputs:
    deploymentid: 'azuredevops-deployment'
    templateFilePath: EdgeSolution/deployment.template.json

- task: PublishBuildArtifacts@1
  displayName: 'Publish Artifact: drop'
```

4. Select **Commit** to commit your change. The **Commit** pane appears. Select **Commit**.

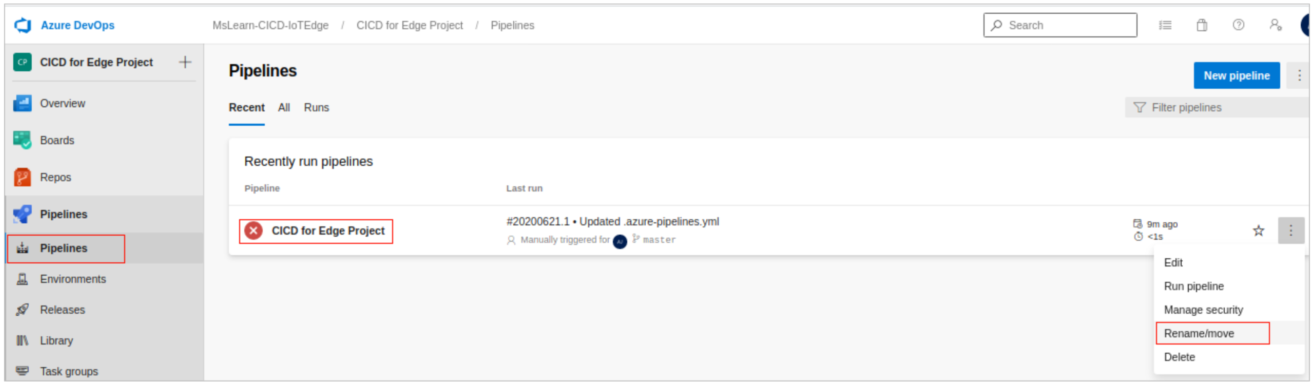
5. Return to the **Files** panel.

6. In the upper right corner, select **Set up build**, and then select **Run**. You should see that a build has kicked off upon editing the build definition.

7. In the left menu pane, select **Pipelines**. You will see the build will fail. This is to be expected, as Azure DevOps will create the build definition with a name that contains spaces, which cause a conflict in the "Azure IoT Edge - Build module images" task.

8. To fix this, select **Pipelines**. The **Pipelines** panel appears.

9. From the *Recently run pipelines*, at the far right, select the vertical ellipsis for your pipeline, and select **Rename/move**. The **Rename/move pipeline** dialog appears. In the **Name** text box, and rename the newly created build definition so that it does not contain spaces. Select **Save**.



Create build definition variables

1. Now you need to add build variables in order to run the build successfully. You'll need to obtain:

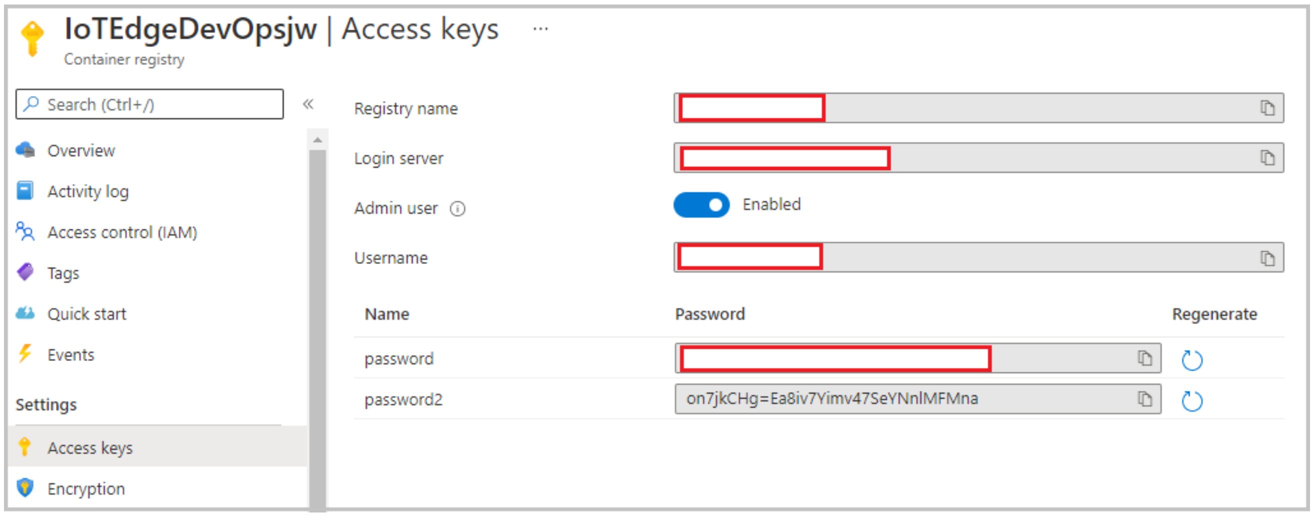
   - Azure Container Registry host name as `acr.host`
   - Azure Container Registry username as `acr.user`
   - Azure Container Registry password as `acr.password`

2. Go to the Azure portal, and navigate to the resource group you created for this module.

3. Select the **Container Registry** resource.

4. From the left menu pane, under **Settings**, select **Access keys**.

5. Copy the **registry name**, **Login server**, **Username**, and **password**.



Registry name

Login server

Admin user: Enabled

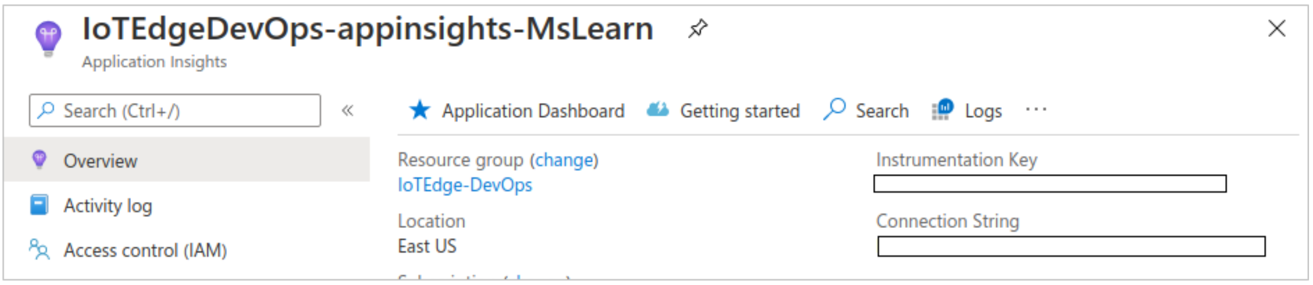
Username

Name	Password
password	
password2	on7jkCHg=Ea8iv7Yimv47SeYNnlMFMna

6. Obtain the Application Insights instrumentation key, which will be represented by `appinsights.instrumentationkey`. Go to the Azure portal, and navigate to the resource group you created for this module.

7. In the left menu pane, select **Overview**. Under the **Resources** tab, select the **Application Insights** resource.

8. In the **Essentials** section, copy the **Instrumentation Key**.



Resource group (change)
IoTEdge-DevOps

Location
East US

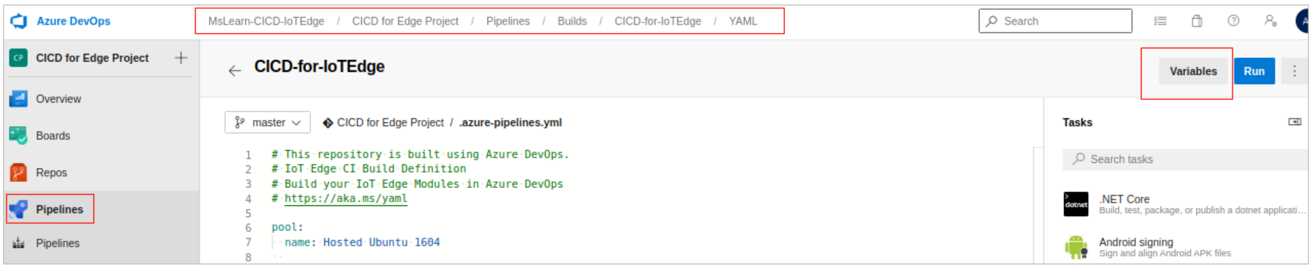
Instrumentation Key

Connection String

9. Go back to Azure DevOps project, and navigate to **Pipelines**.

10. In the pipeline you ran earlier, select the far left vertical ellipsis, and then select **Edit**.

11. In the upper left, select **Variables**. The **New variable** pane appears.


Azure DevOps
MsLearn-CICD-IoTEdge / CICD for Edge Project / Pipelines / Builds / CICD-for-IoTEdge / YAML
Search
CICD for Edge Project
Overview
Boards
Repos
Pipelines
Pipelines
CICD-for-IoTEdge
Variables
Run
master
CICD for Edge Project / .azure-pipelines.yml
Tasks
Search tasks
.NET Core
Build, test, package, or publish a dotnet applicati…
Android signing
Sign and align Android APK files

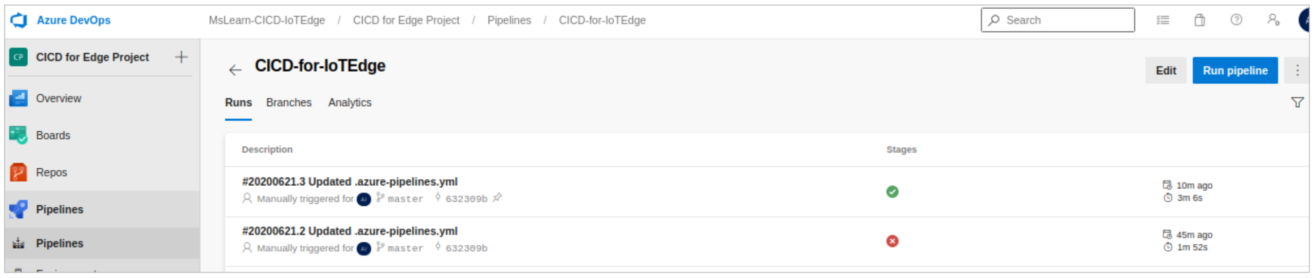
```
# This repository is built using Azure DevOps.
# IoT Edge CI Build Definition
# Build your IoT Edge Modules in Azure DevOps
# https://aka.ms/yaml

pool:
  name: Hosted Ubuntu 1604
```

12. Add four variables with names and values by slecting **OK** after each entry. When all four variables have been entered, select **Save**, and then select **Run**. The **Run pipeline** pane

appears. Select **Run**.

13. Review the status of the pipeline by going back to **Pipelines**. The build should finish successfully as follows.

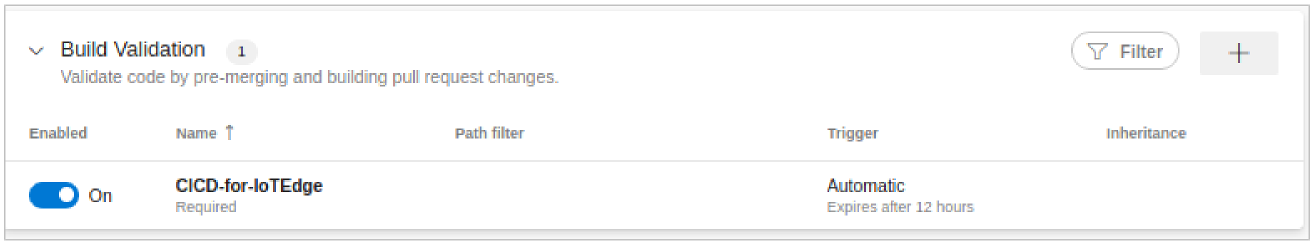


Description	Stages	
#20200621.3 Updated .azure-pipelines.yml Manually triggered for master 632309b	✓	10m ago 3m 6s
#20200621.2 Updated .azure-pipelines.yml Manually triggered for master 632309b	✗	45m ago 1m 52s

Apply a branch policy

1. With a successful build definition in place, we can now enforce continuous integration by applying a branch policy to the master branch. In the left menu pane, select **Repos**, and again, in the left menu pane, select **Branches**. Select the vertical ellipsis at the far end of the row for the master branch, and from the dropdown, select **Branch policies**.

2. In the middle of the **master** panel, select **Build Validation**, and then select the + icon (Add new build policy), and select the newly created build pipeline. Keep everything with their default values, and then select **Save**.



Validate code by pre-merging and building pull request changes.

Enabled	Name ↑	Path filter	Trigger	Inheritance
On	CICD-for-IoTEdge Required		Automatic Expires after 12 hours	

While this policy is enabled, all commits to feature branches will kick off an execution of the newly created build pipeline. It must succeed in order for a pull request of those changes to be made to the master branch.

Module complete:

Unlock achievement