Table of Contents

[Demo Guide: Module 6 - DevOps with Containers 2](#_Toc525661554)

[Demo 1: Demo Sample Application in Azure Repository 2](#_Toc525661555)

[Demo 2: Azure Container Registry 6](#_Toc525661556)

[Demo 3: CI Build Pipeline on Azure DevOps 6](#_Toc525661557)

[Demo 4: CD Release Pipeline on Azure DevOps 8](#_Toc525661558)

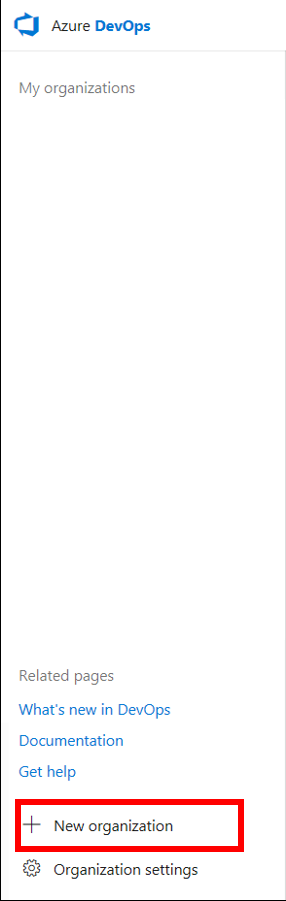
Demo Guide: Module 6 - DevOps with Containers

Demo 1: Demo Sample Application in Azure Repository

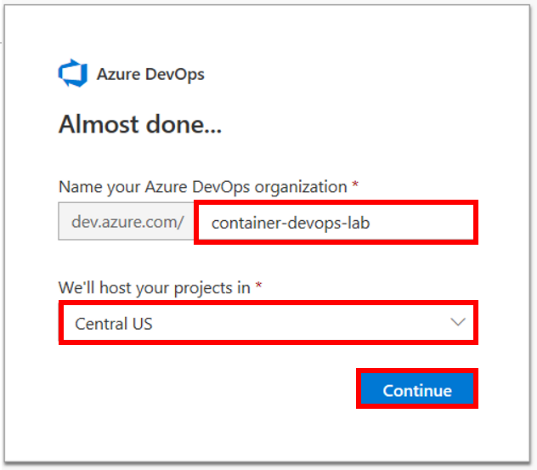
In this demo, you are going to show how to create an Azure DevOps account which contains toolset to create CI/CD pipeline. You should create your own Azure DevOps account **before** showing this demo (task 1 below). You may create an additional Azure DevOps account environment as part of the demo if you want to demonstrate how to import build/release pipelines in this new account.

Tasks

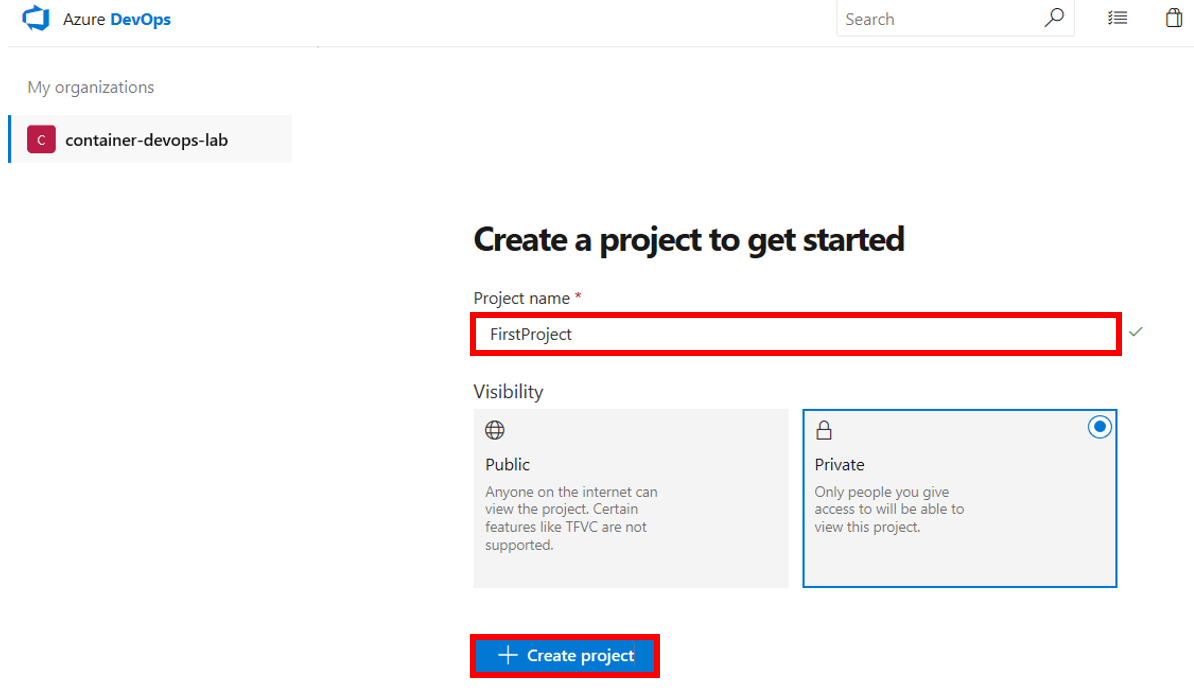
1. Create Azure DevOps account with a CI/CD pipeline
2. Go to the lab document for Lab 6 and complete the entire lab before the workshop starts, a completed CI/CD setup in Azure DevOps will be required to complete all demos. **Note: A copy of a working CI/CD JSON is included in the Demos\Module 6 for your reference**.
3. Demonstrate how to create an Azure DevOps organization and project. You can show how to
4. Navigate to dev.azure.com (or https://<Your\_Organization>s.visualstudio.com/ if you have an existing organization set up) and sign in with your Azure Pass account.
5. On Dev Portal, click New Organization button to add a new organization.



1. Give a unique name to your organization, select the closest datacenter and then click Continue to create a new Azure DevOps organization.



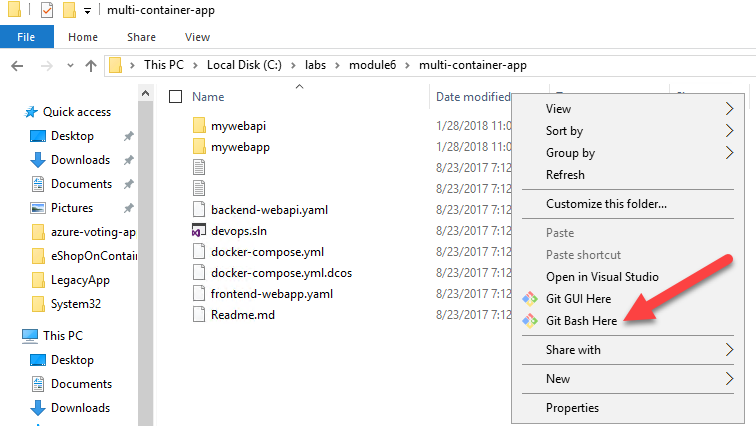
1. Choose a project name. Click on Create Project. This will take you to the url of your project.



1. Optional: Demonstrate how to push Git repository to the Azure Repository
2. Open File Explorer and navigate to C:\labs\module6.

**Make sure that the yaml files you are about to push reference your existing ACR repository instead of devopslabs.**

1. Right click choose Git Bash Here.



1. Run the following commands, one at a time (you can either fill in a name and email or leave it as the default). Git config (first two commands) is a one-time setup. Git init will initialize your folder as a git repository and add an empty hidden .git folder. Git add and commit are necessary steps anytime you make a change to your code and want it to be staged (tracked and ready to be committed) and committed (ready to be pushed to the server)

git config --global user.email [you@example.com](mailto:you@example.com)

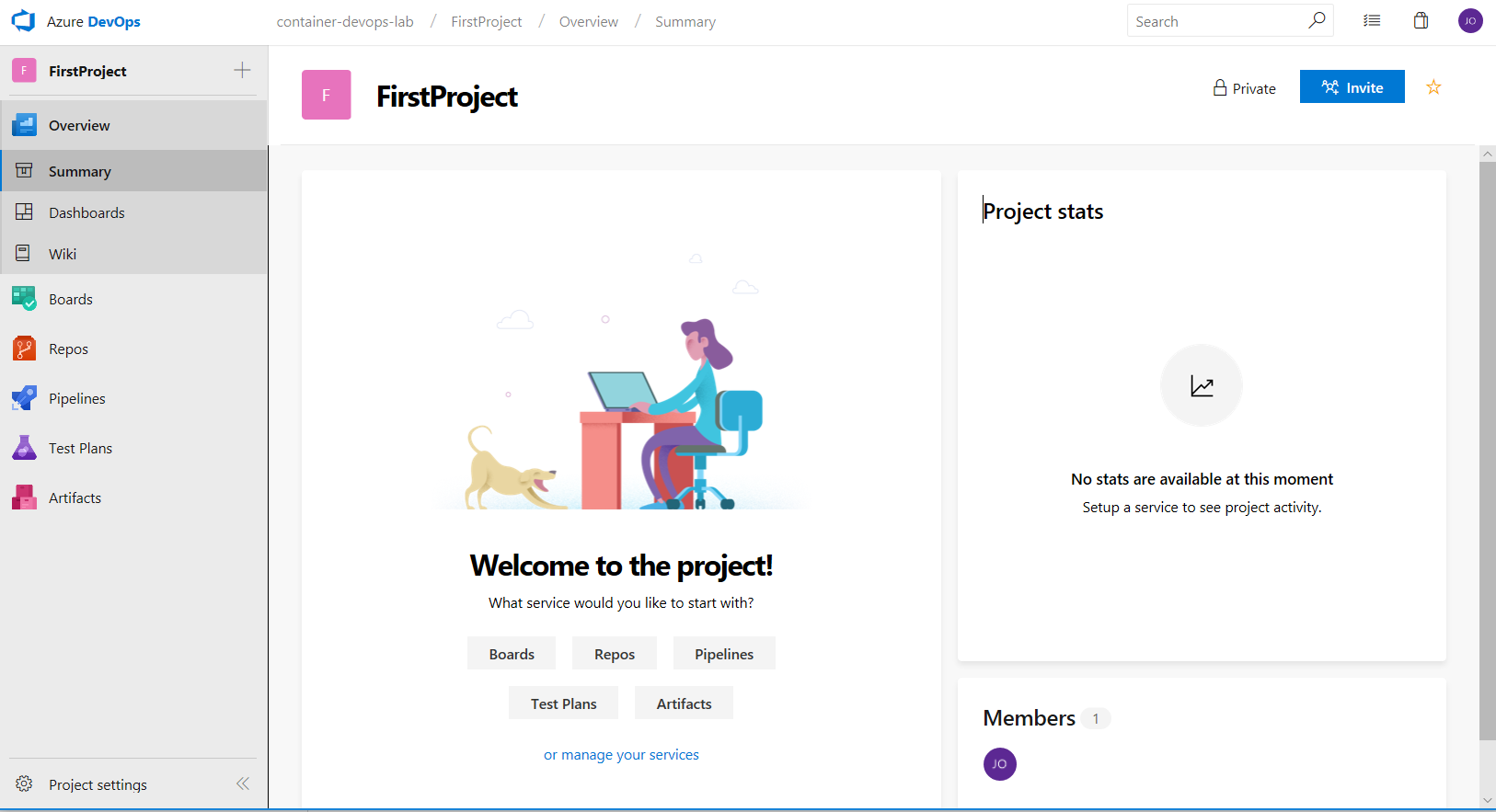
git config --global user.name "Your Name"

git init

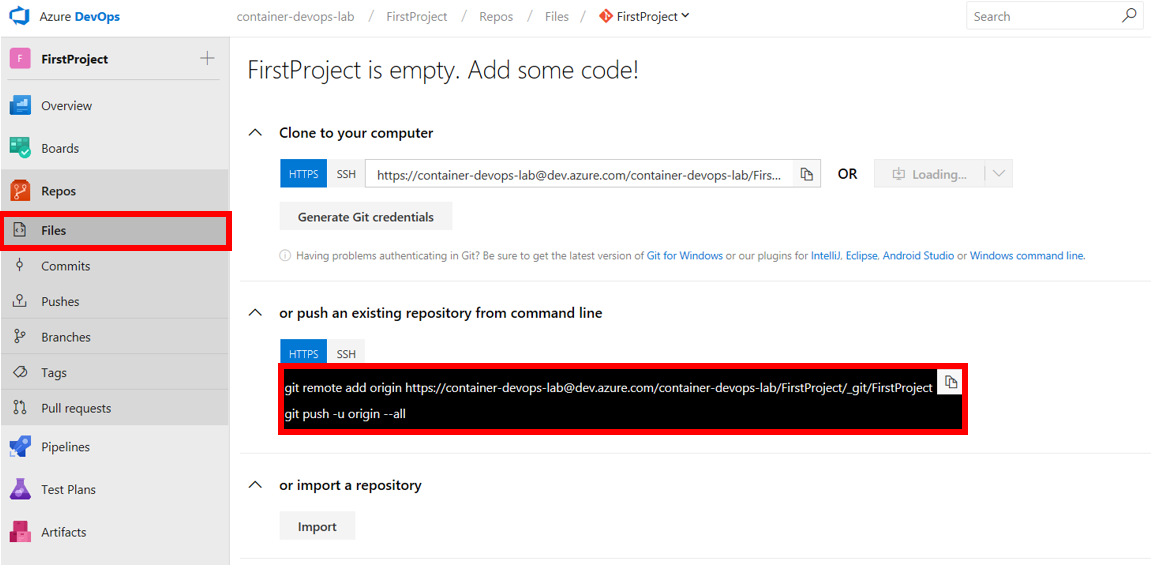
git add .

git commit -m 'adding my web project'

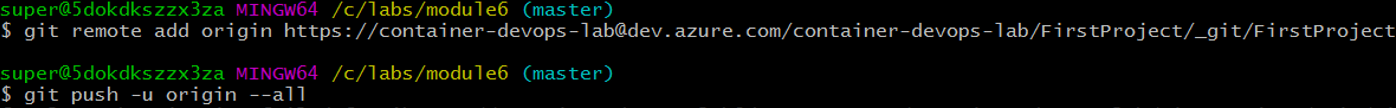
1. Navigate to <https://dev.azure.com/YourAccountName/YourProjectName> to see the environment. A sample link is <https://dev.azure.com/container-devops-lab/FirstProject>



1. Now you are ready to push your local Git repository to remote repository in Azure DevOps. Go to your Code repository by clicking on **Repos – Files**. You will see that your Git repository is empty, and you are ready to add your application into it. Copy the two commands to do this.

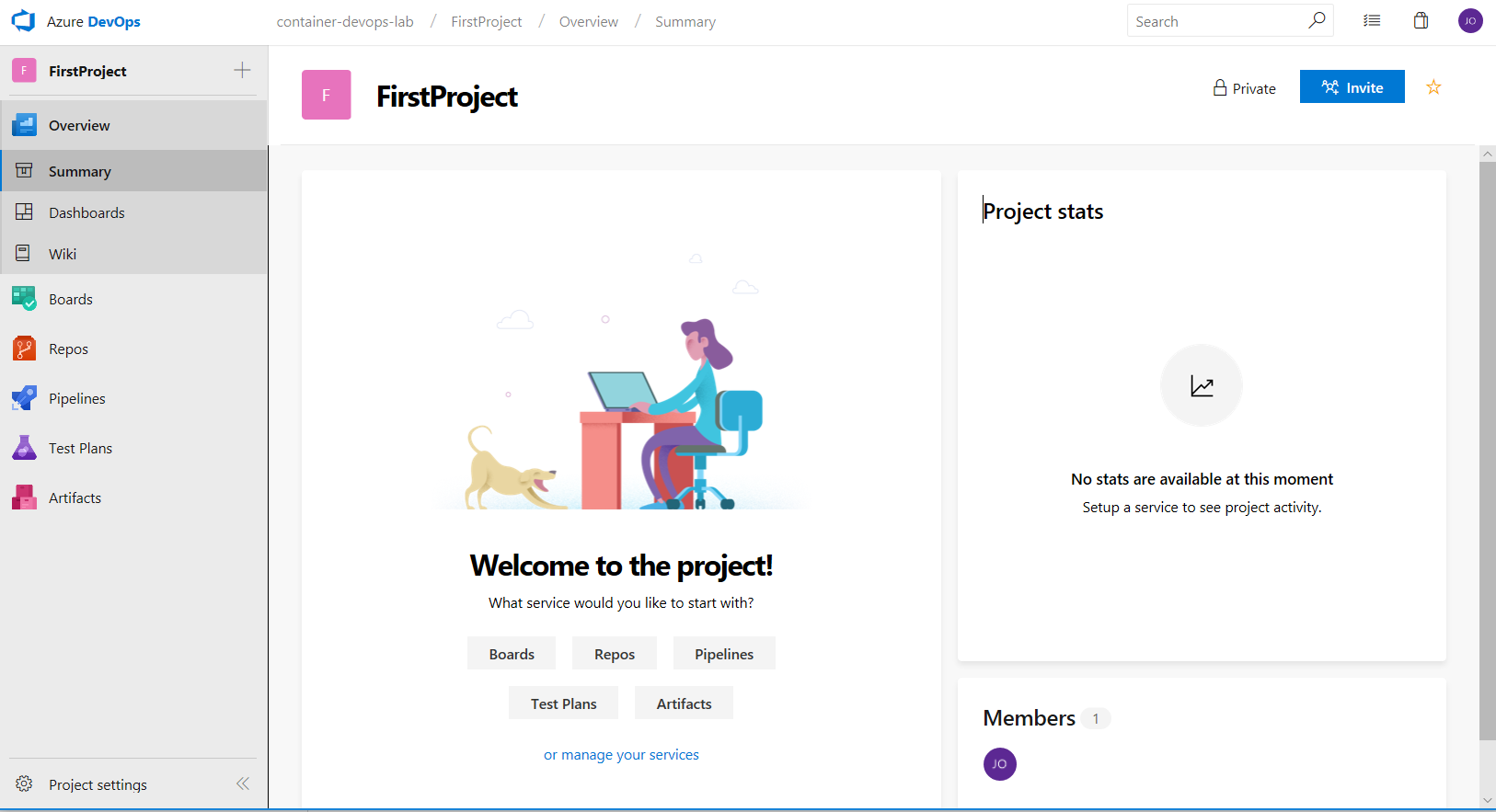


1. Paste them into Git Bash by pressing Shift + Insert (or right click and paste it into the Git Bash command line). It will automatically run the first line, hit enter to add the second line. “Git remote add origin” means to add a “remote URL such as your Azure repo URL” with the alias “origin” to your local git settings for this project folder. You could name it any alias you want, “origin” is just the default. “Git push” means to interact with the server and push code to that URL you provided as origin. The “-u” will “set-upstream” which means that it will default all pushes and pulls to the “origin URL” and you don’t have to specify “origin” in all your future commands.

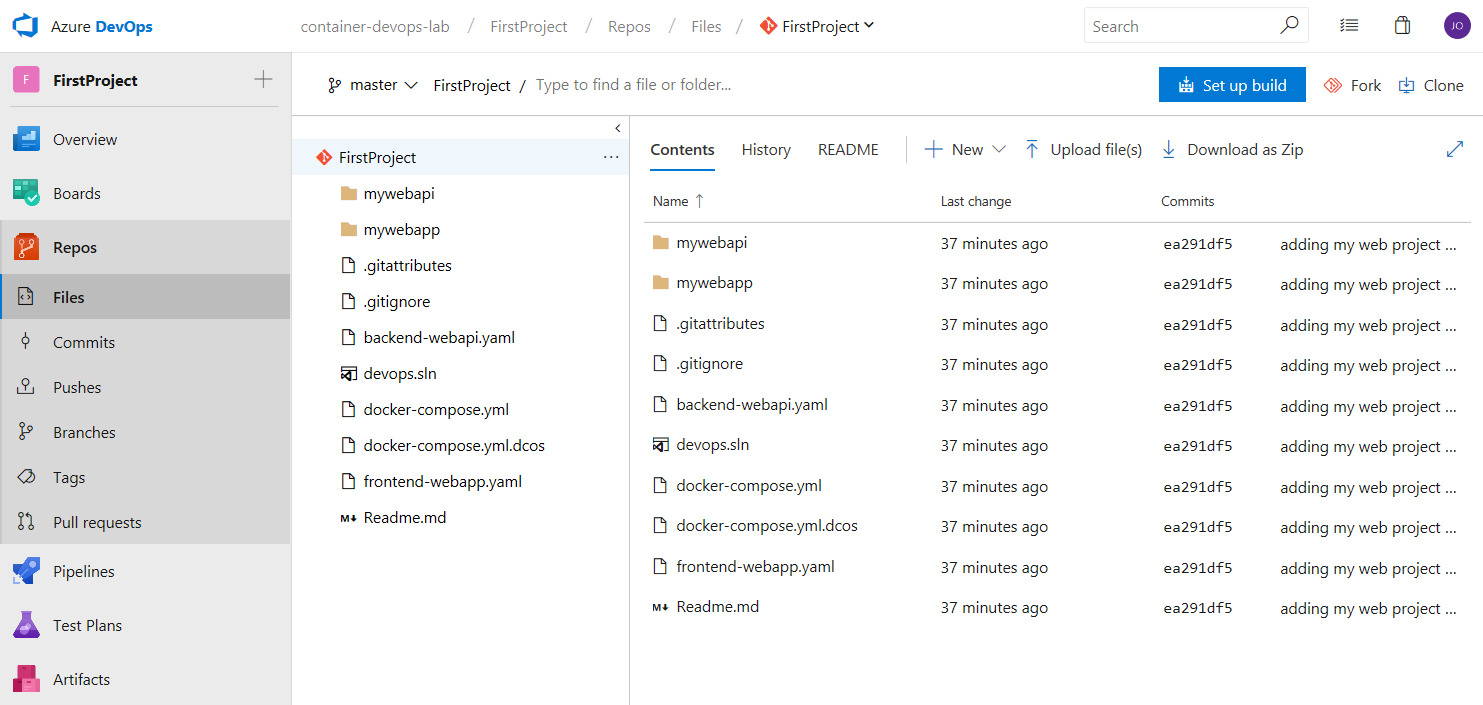


You will see a popup to authenticate to your Microsoft account. Once authenticated, your project will be pushed to remote Git repository in Azure DevOps if you refresh Azure DevOps repository in your browser.

1. Demonstrate the Azure repository file tab and how to see your project
2. Navigate to <https://dev.azure.com/YourAccountName/YourProjectName> to see the environment.



1. Go to your Code repository by clicking on **Repos** – **Files**. You will see your Git repository with the files previously uploaded



1. Demonstrate the webapp and webapi portions of the project, Dockerfiles, and docker-compose.yml files in the Azure Repository File tab.

Demo 2: Azure Container Registry

Tasks

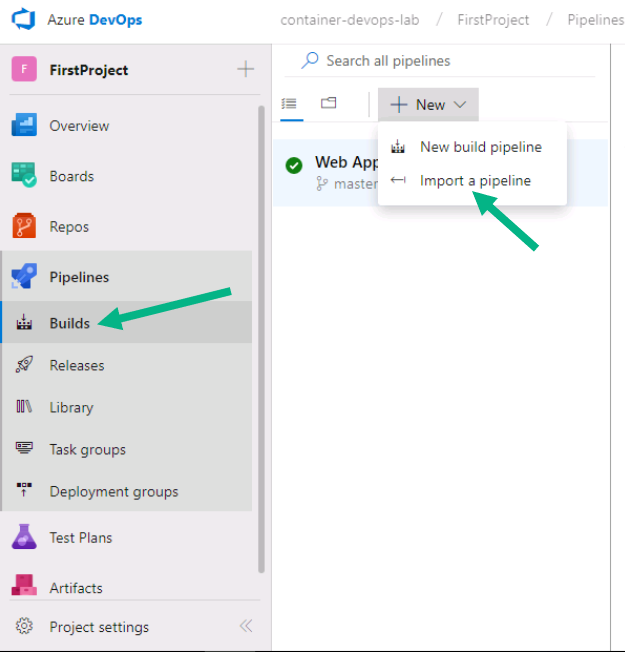
1. Demonstrate how you would deploy a new Azure Container Registry.
2. Demonstrate how to view the Azure Container Registry repo contents (repository section).
3. You can show the geo replication feature which is available in the Premium sku.
4. You can also talk about the benefit of ACR Tasks or Aqua and Twistlock integration.

Demo 3: CI Build Pipeline on Azure DevOps

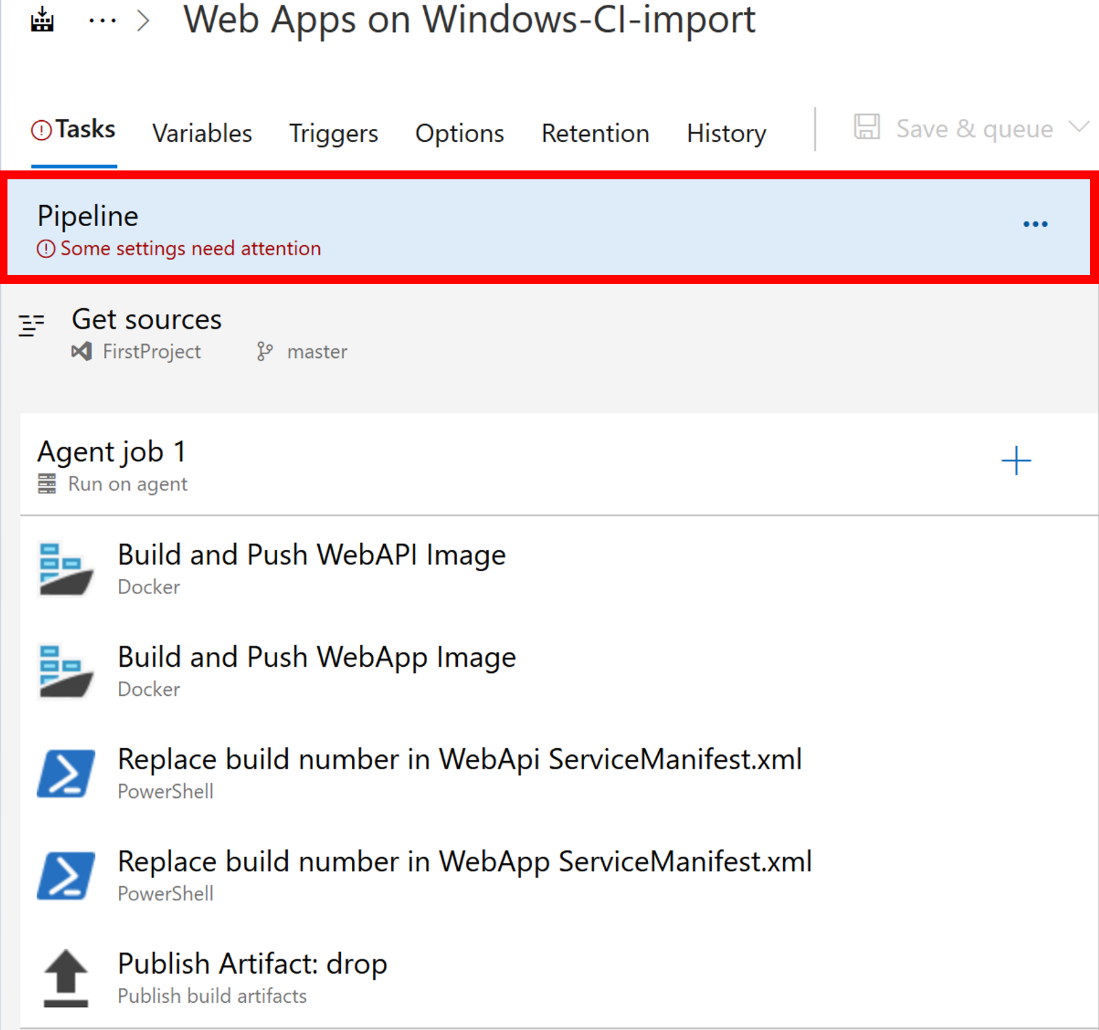
In this demo, you will demonstrate Build pipelines for both Linux and Windows Containers. These two parts have 2 options each: showing how to import the build pipeline from a JSON file or showing an existing pipeline.

Option 1 - Windows: Show import of a pipeline

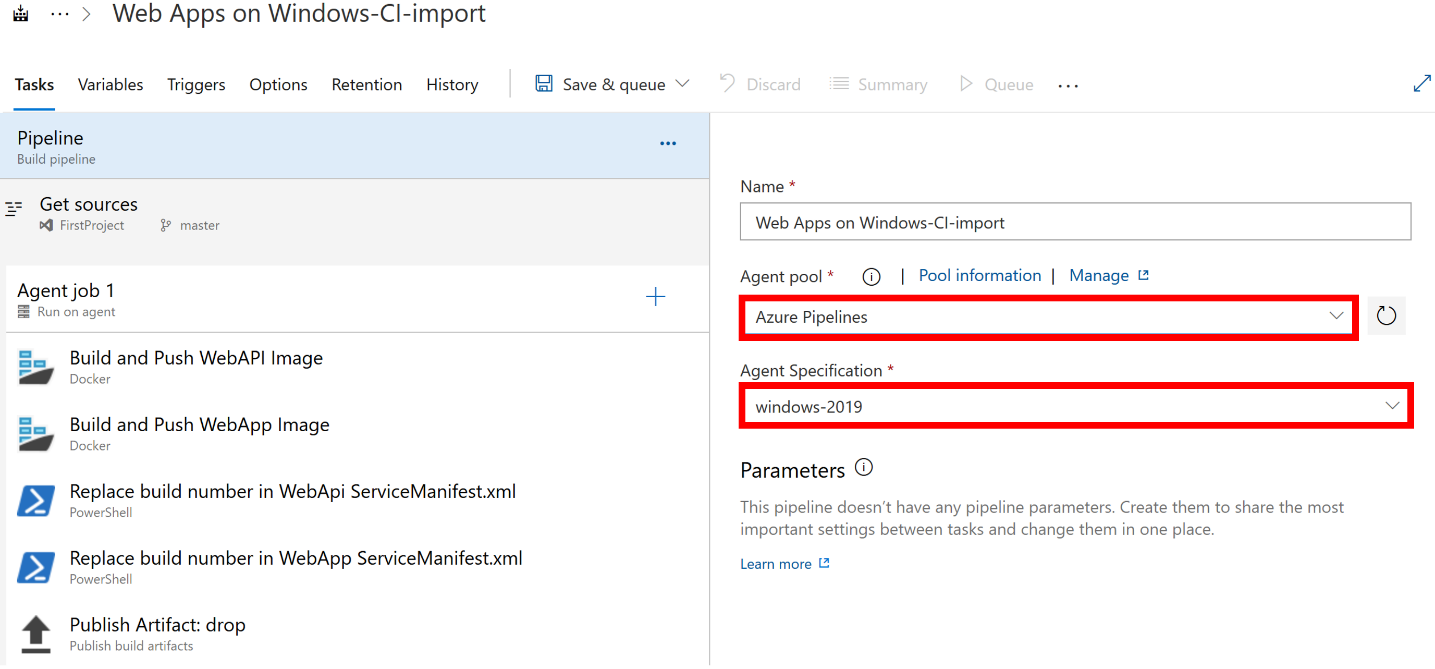
1. Navigate to **Pipelines** – **Builds** and click **New** – **Import a pipeline**.



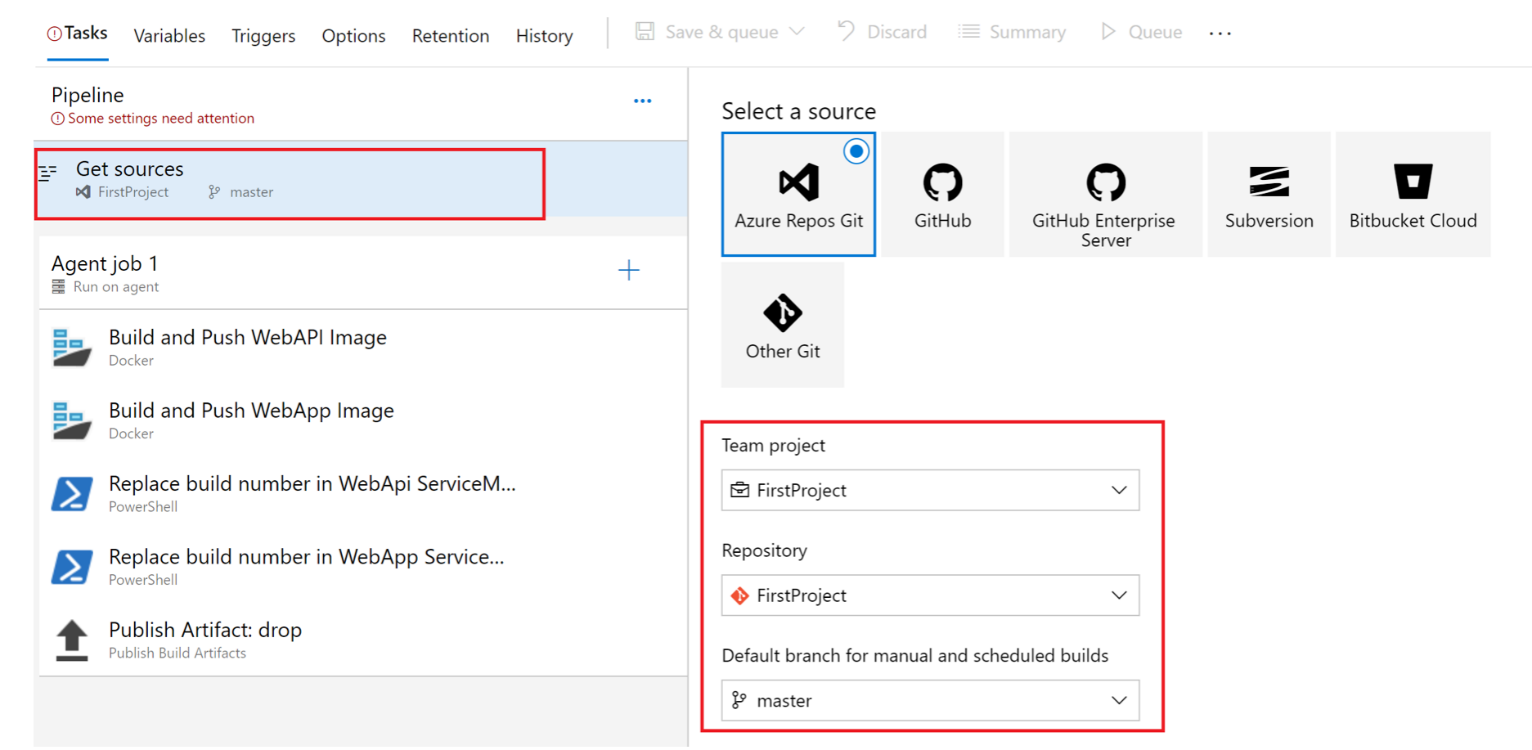
1. Select **C:\labs\module6-ext\Web Apps on Windows-CI.json** and click **Import**
2. You will see that some settings need to be updated. Click on **Pipeline**

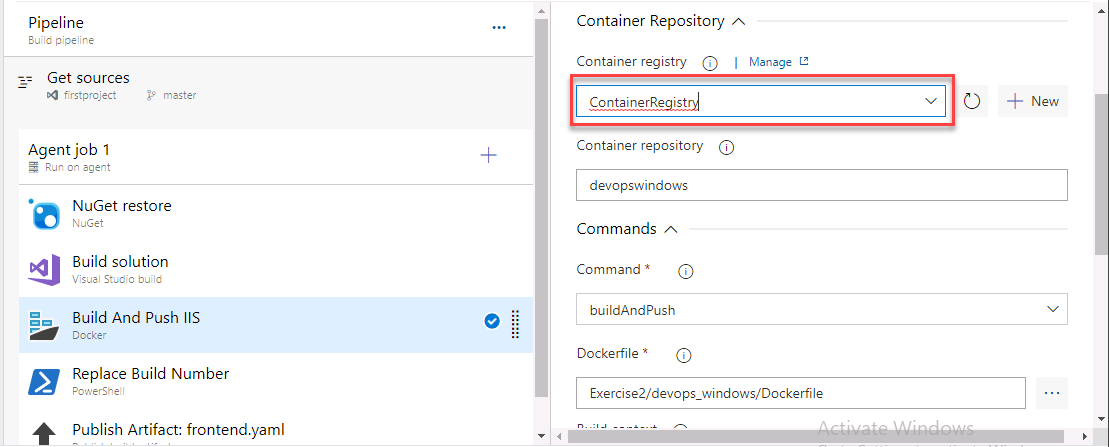


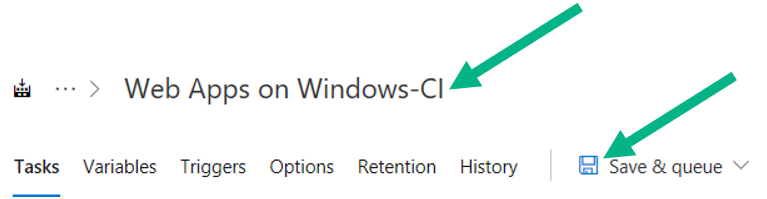
1. Select **Azure Pipelines** - **windows-2019** as an agent pool and specification.



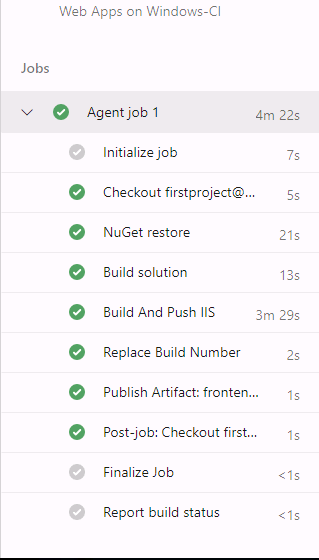
1. Click on **Get sources** and make sure that correct Git repository and branch is selected.



1. Select Build and Push IIS task and choose the container registry connection created in Exercise 1 (ContainerRegistry).
2. Make sure the Build pipeline is named **Web Apps on Windows-CI** to stay aligned with the following screenshots. Click **Save & queue**.

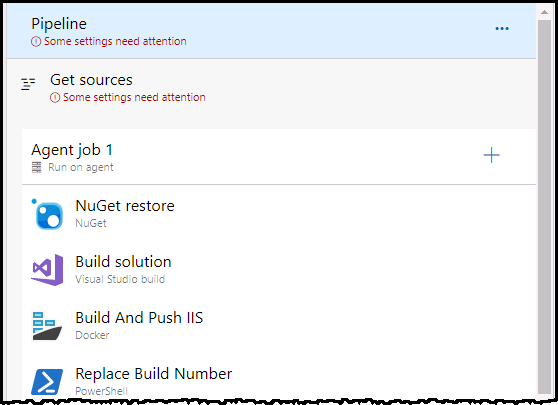


1. Your finished build should look like this (if you get an error please click on the step with the error and try debugging or the instructor will come by and help you):



Option 2 - Windows: Show your existing pipeline

1. Walk through each step of the CI Build process and explain how each task works. Explain what the Variables, Triggers, Options, and History tabs do in the Build pipeline. See below the screenshot for an explanation of each step and what you should demo.

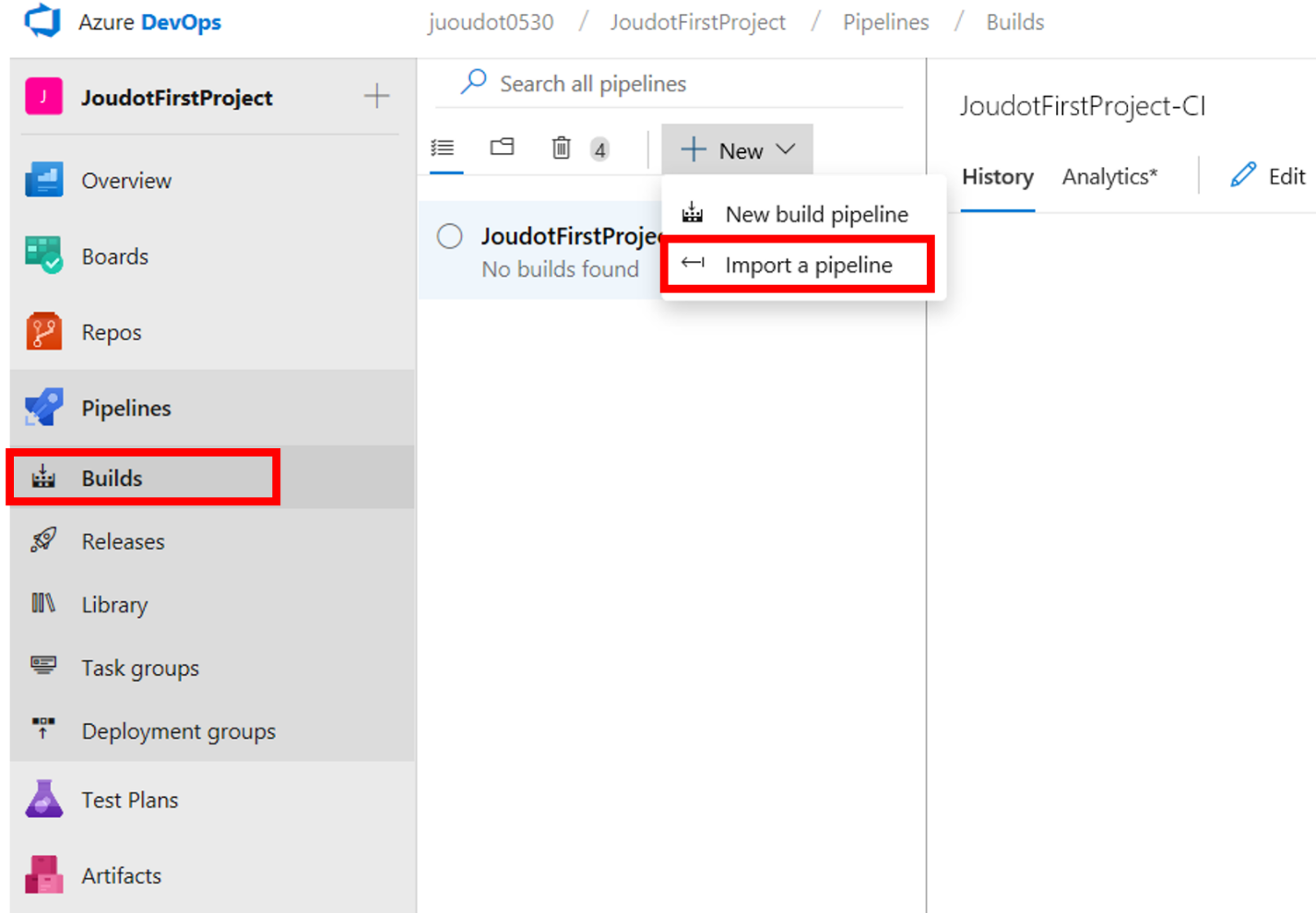


* Build Solution
  + This is a full framework ASP.NET Application using Visual Studio build task to build the web application
  + Call out the MS Build Arguments – The /p:**PublishProfile** is used to publish the web application in the location the docker file expects
* Build and push IIS
  + Builds the docker image based off of the **Dockerfile**. Names and tags the image appropriately so that it can be pushed to the correct Azure Container Registry
* Replace build number in Kubernetes YAML file **frontend.yaml**
  + Uses Powershell to modify the YAML file so that the correctly tagged image will be referenced.
* Publish Artifacts
  + Published the modified Kubernetes YAML file with the updated build number for the image tag

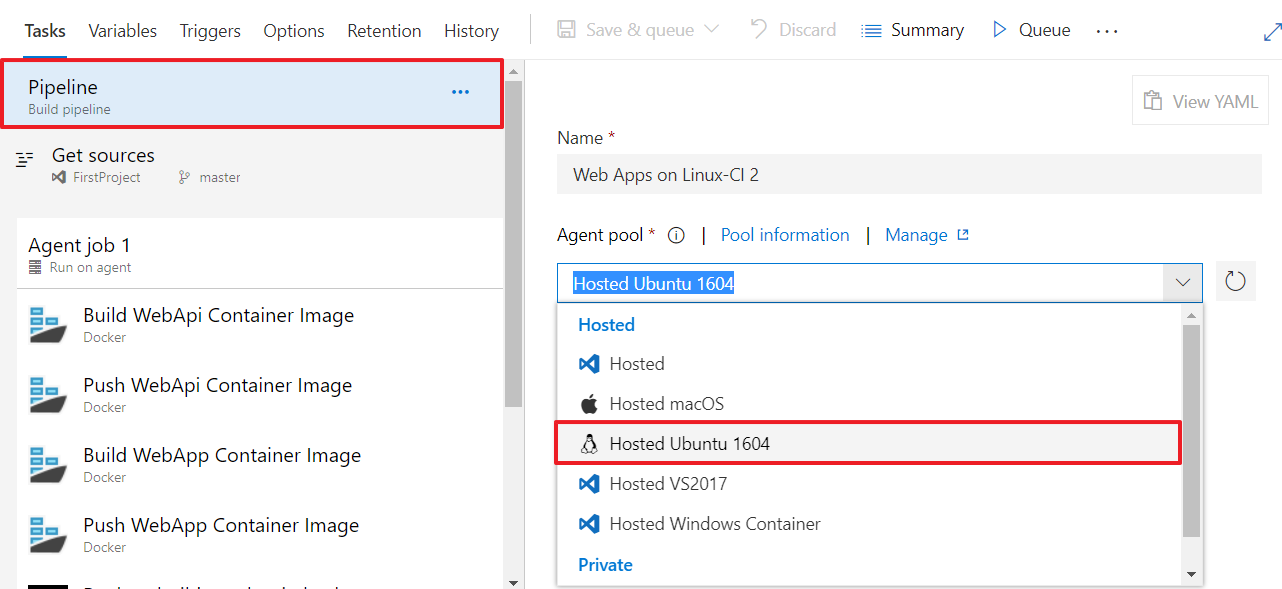
1. Show how to kick off a build pipeline.
2. Demo the results page of a completed successful build once it completes.

Option 1 – Linux: Show import of a pipeline

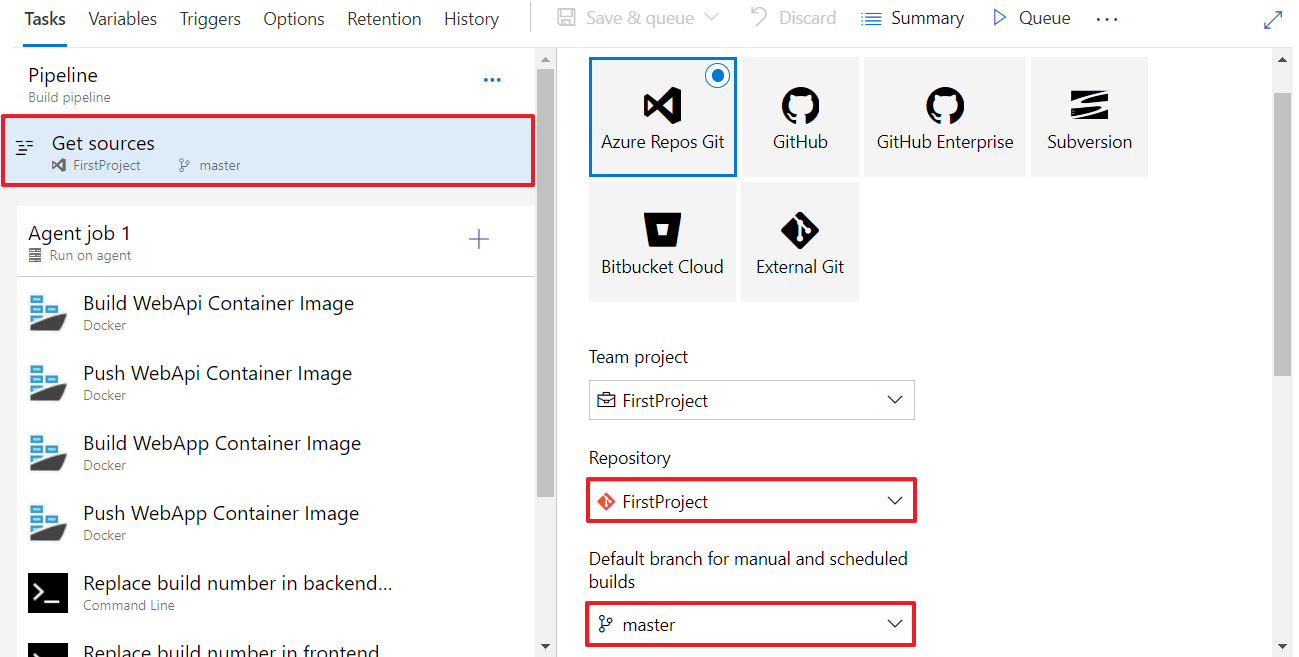
1. Navigate to **Pipelines** – **Builds** and click **New** – **Import a pipeline**.

**Note:** it seems that we can only do that when at least one pipeline exists. So you will need to create an empty pipeline first.

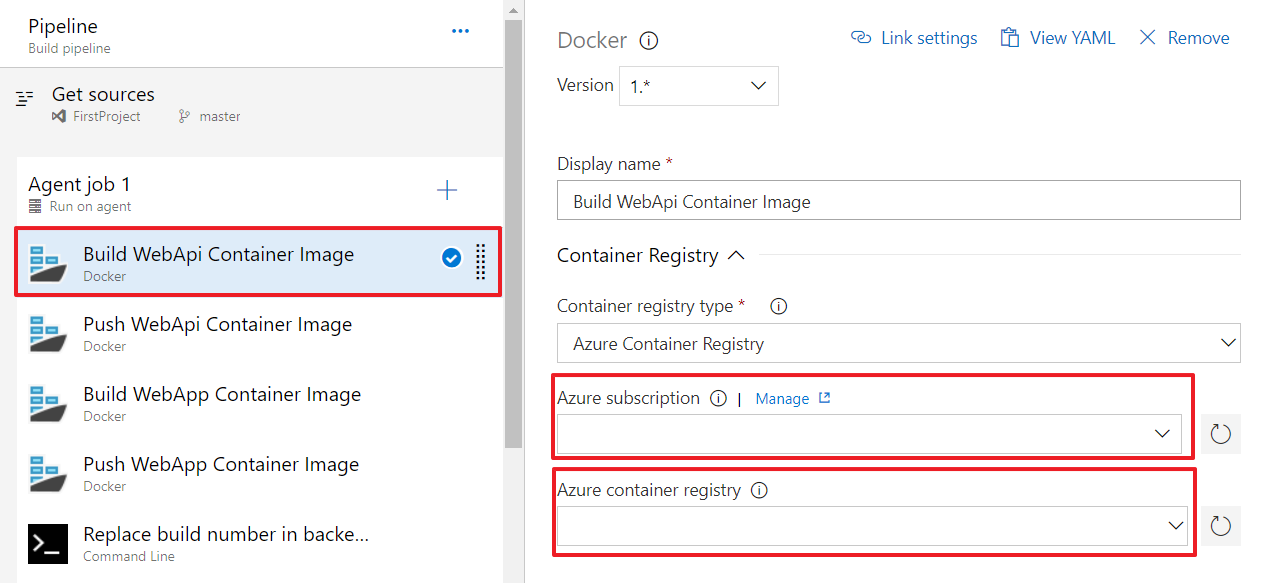
1. Select **Web Apps on Linux-CI.json** and click **Import**
2. Select **Hosted Ubuntu 1604** as an agent pool



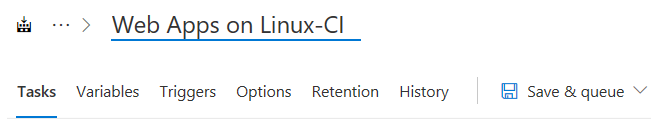
1. Click on **Get sources** and make sure that correct Git repository and branch is selected.



1. Select the docker tasks one by one and enter the subscription and ACR information.



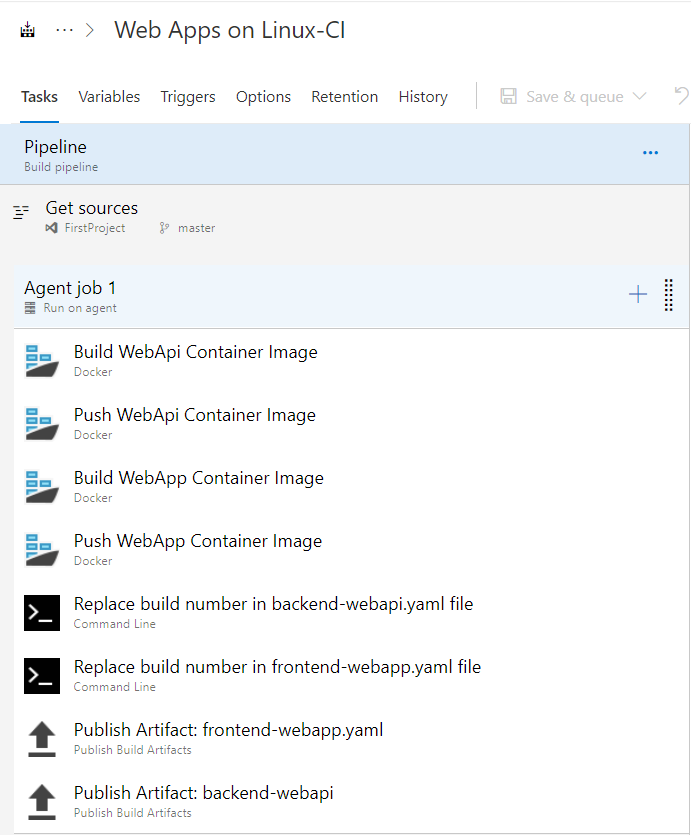
1. Make sure the Build pipeline is named Web Apps on Linux-CI to stay aligned with the following screenshots.



1. Once all the Docker tasks are updated, “Some settings need attention” should disappear and you can save the pipeline and show what is described in option 2.

Option 2 – Linux: Show an existing pipeline

1. Walk through each step of the CI Build process and explain how each task works. Explain what the Variables, Triggers, Options, and History tabs do in the Build pipeline. See below the screenshot for an explanation of each step and what you should demo.



* Build Web API
  + Build the image for the container based on the Dockerfile. Demo the contents of the Dockerfile in the Azure Repository File tab and explain how the Dockerfile is built. Names and tags the image appropriately so that it can be pushed to the correct Azure Container Registry in the next step.
* Push WebAPI
  + Push the built image from the previous step to the Azure Container Registry.
  + Call out the Container Registry authentication fields.
* Build Web App
  + Build the image for the container based on the Dockerfile. Demo the contents of the Dockerfile in the Azure Repository File tab and explain how the Dockerfile is built. Names and tags the image appropriately so that it can be pushed to the correct Azure Container Registry in the next step.
* Push Web App
  + Push the built image from the previous step to the Azure Container Registry.
* Replace build number in backend-webapi.yaml file
  + Edit YAML file so that the correctly tagged image will be referenced
* Replace build number in frontend-webapp.yaml file
  + Edit YAML file so that the correctly tagged image will be referenced
* Publish Artifact frontend-webapp.yaml
  + Drop a copy of the YAML file as the result of the build being completed so it can be utilized in the Release pipeline
* Publish Artifact backend-webapi.yaml
  + Drop a copy of the YAML file as the result of the build being completed so it can be utilized in the Release pipeline

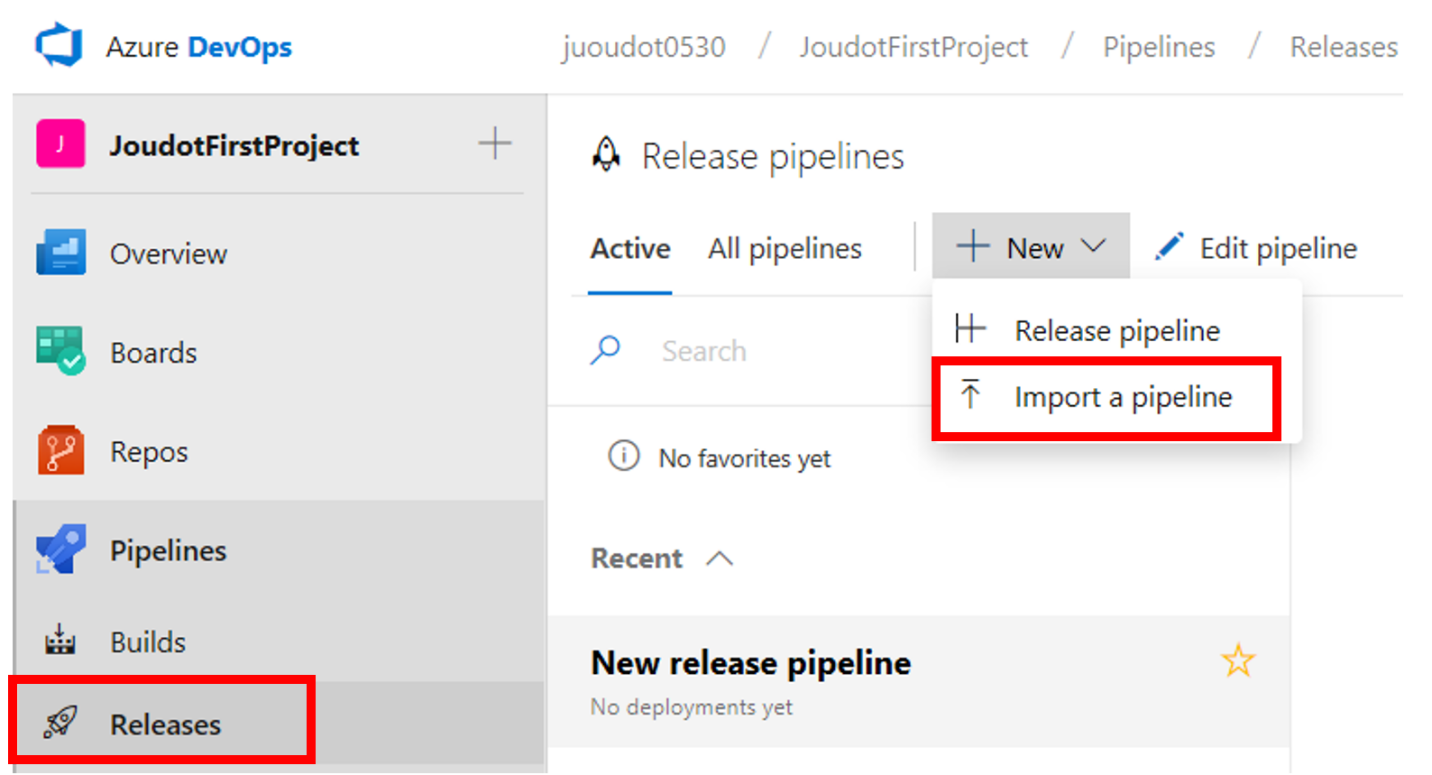
1. Show how to kick off a build pipeline.
2. Demo the results page of a completed successful build once it completes.

Demo 4: CD Release Pipeline on Azure DevOps

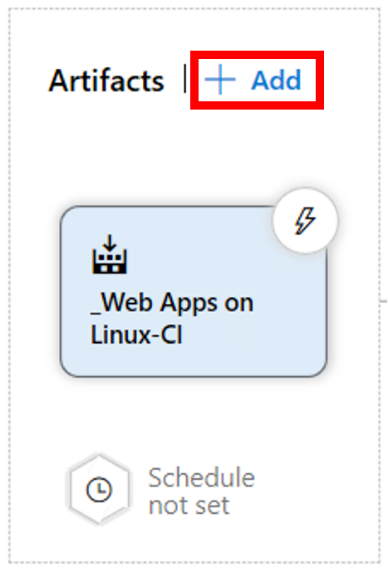
In this demo, you will demonstrate Release pipelines for both AKS and Service Fabric clusters. These two parts have 2 options each: showing how to import the build pipeline from a JSON file or showing an existing pipeline.

Option 1: AKS - Show import of a pipeline

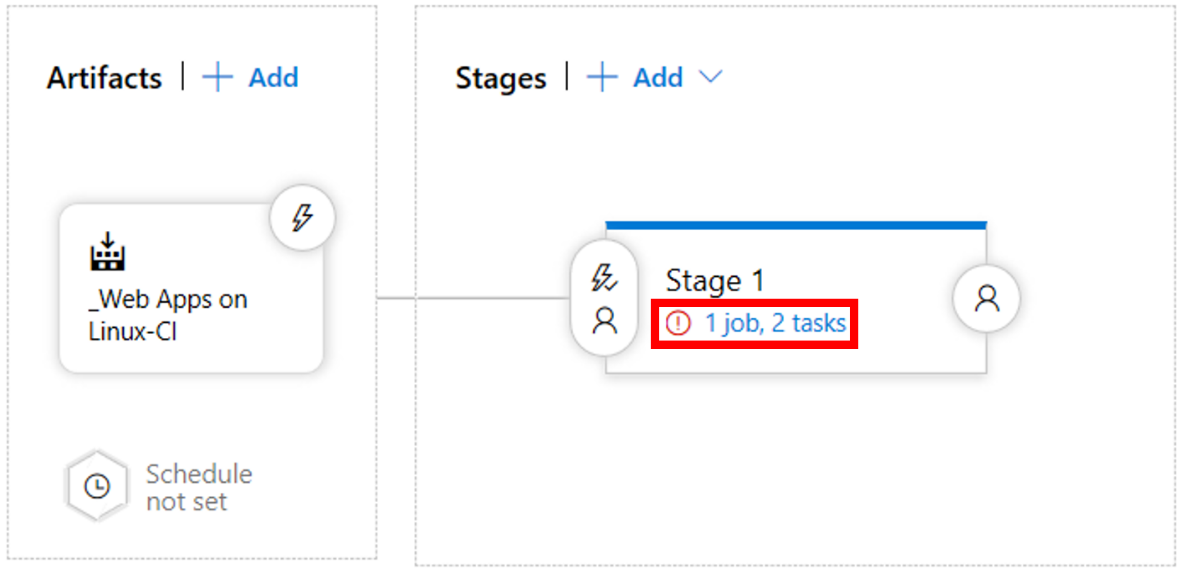
1. Navigate to **Pipelines** – **Releases** and click **New** – **Import a pipeline**.

**Note:** it seems that we can only do that when at least one pipeline exists. So you will need to create an empty pipeline firs

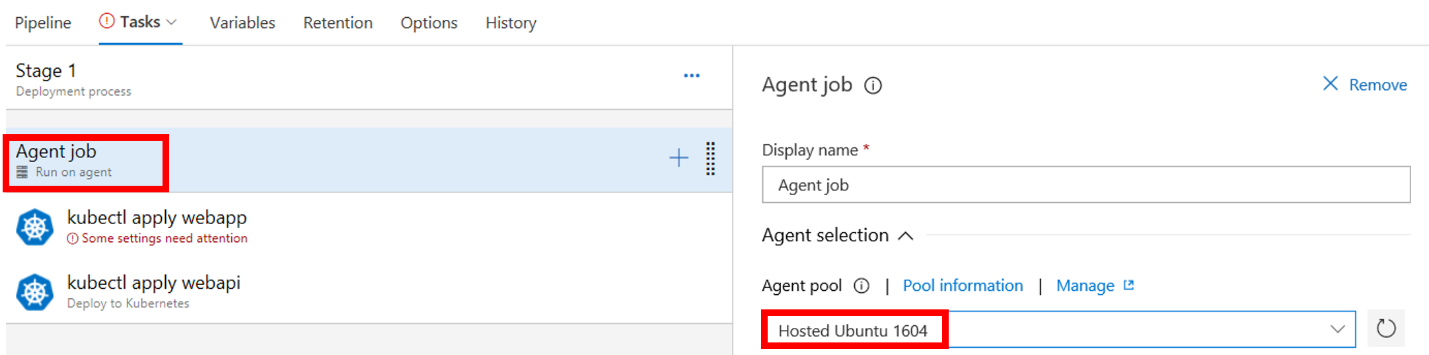
1. Select **AKS - Release-CD.json** and click **Import**
2. Make sure that the input Artifact is valid by deleting the default one: click on the Artifact Name and then **Delete**. Finally, click on **Add** **an Artifact** to add the artifact coming from your new build.



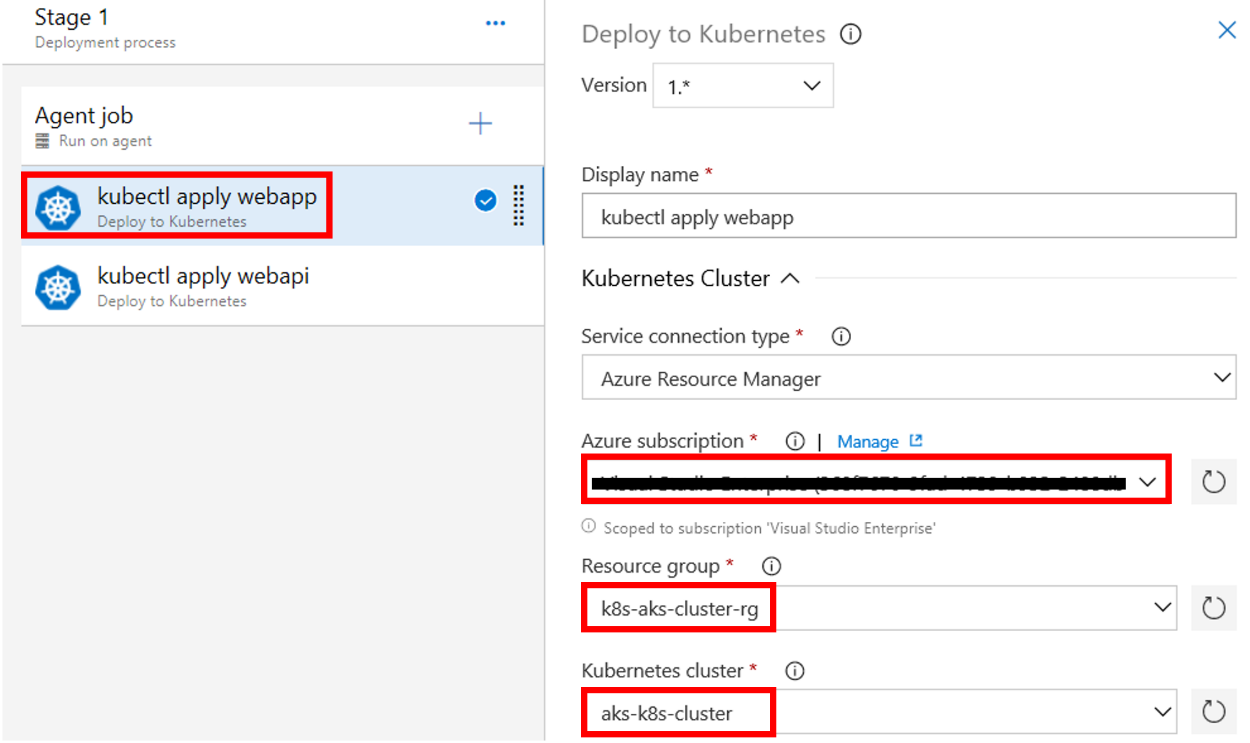
1. Click on the list of tasks under **Stage 1**

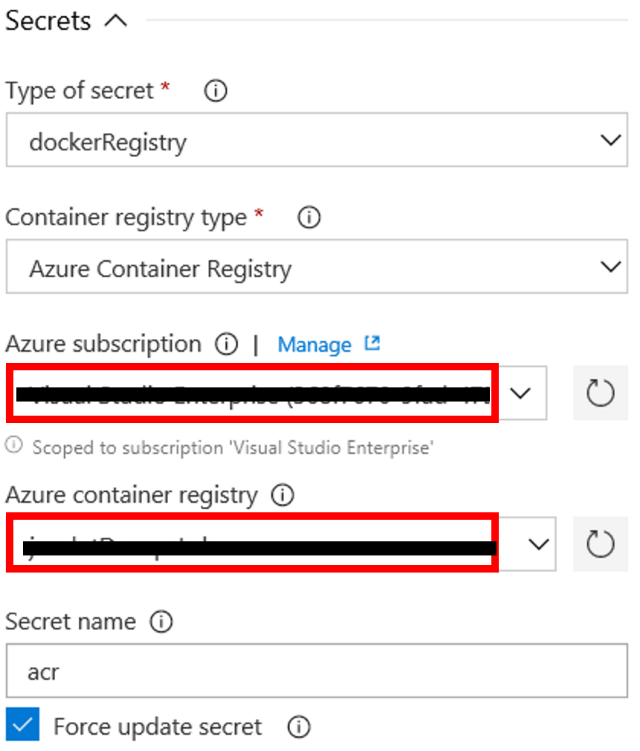


1. Under agent job, select **Hosted Ubuntu 1604** as an agent pool



1. Select the Kubernetes tasks one by one and enter the subscription, resource group and cluster information. Also enter the ACR information under **Secrets**



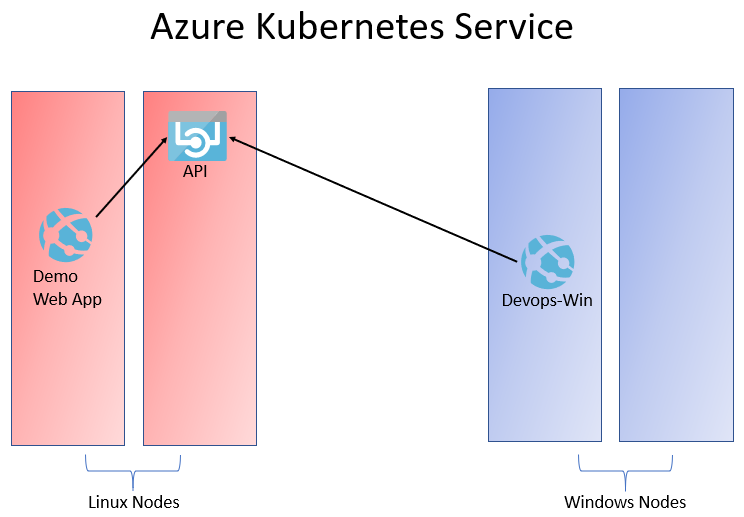


1. Once all the Kubernetes tasks are updated, “Some settings need attention” should disappear and you can save the pipeline and show what is described in option 2.

Option 2: AKS – Import Pipeline

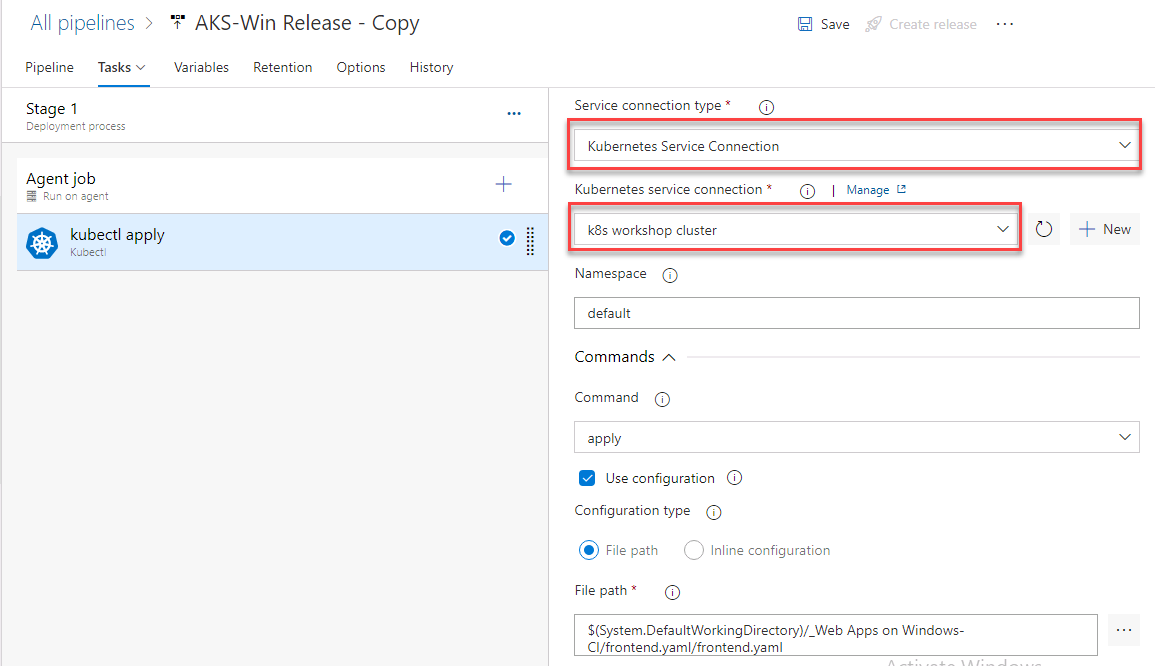
Do lab for Module 6 Exercise 6

In this Lab you are deploying a windows container that will contact the Linux Quotes microservice deployed in the previous exercises.



Option 2: Service Fabric - Show an existing pipeline

The Release pipeline consists of one task, deploy the YAML file to Kubernetes.



Once deployed and the container is running the Windows IIS Container should be able to get the quotes from the previously deployed quotes microservice.

