

Table of Contents

Exercise Guide: Module 6 - DevOps with Containers..... 2

Exercise 1: Create an Azure DevOps Project ..... 2

Exercise 2: CI Build Pipeline on Azure DevOps ..... 5

Exercise 3: CD Release Pipeline on Azure DevOps..... 10

## Exercise Guide: Module 6 - DevOps with Containers

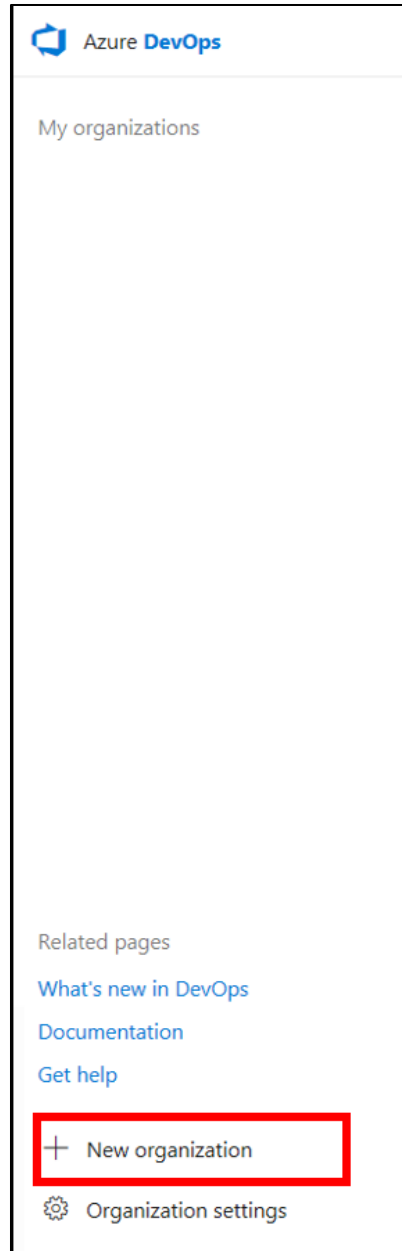
---

### Exercise 1: Create an Azure DevOps Project

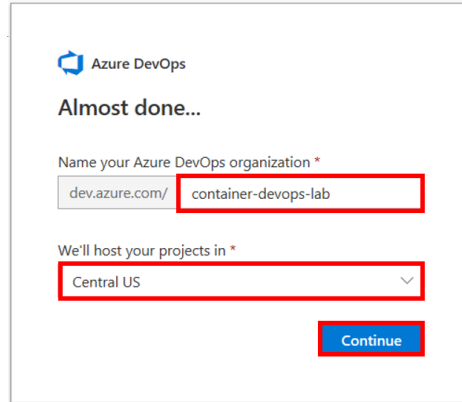
In this exercise, you are going to create an Azure DevOps account which contains toolset to create CI/CD pipeline.

#### Tasks

1. **Create an Azure DevOps organization and project.**
  1. Navigate to dev.azure.com (or <https://<Your Organization>s.visualstudio.com/> if you have an existing organization set up).
  2. On Dev Portal, click New Organization button to add a new organization.



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



Azure DevOps

**Almost done...**

Name your Azure DevOps organization \*

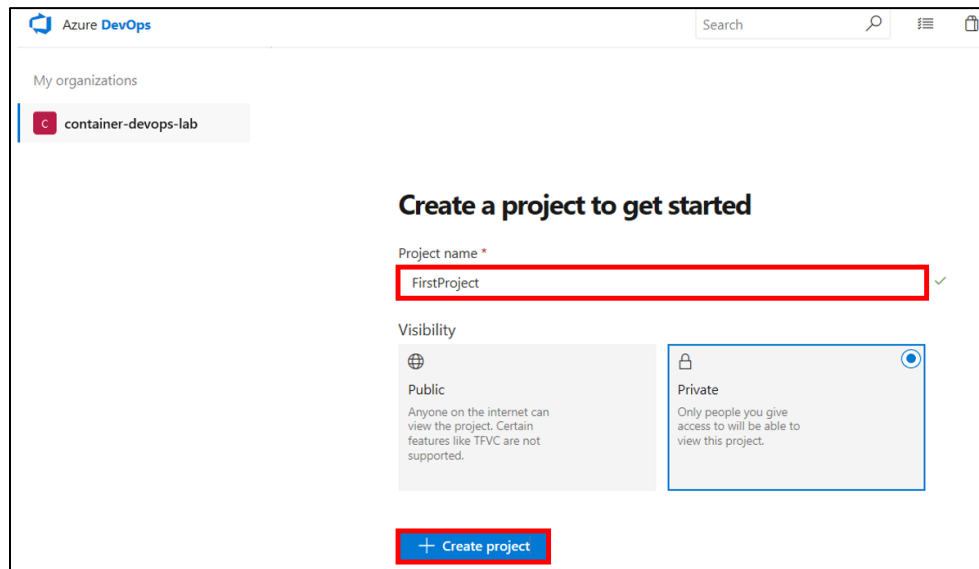
dev.azure.com/ **container-devops-lab**

We'll host your projects in \*

**Central US**

**Continue**

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



Azure DevOps

Search

My organizations

**container-devops-lab**

**Create a project to get started**

Project name \*

**FirstProject**

Visibility

**Public**  
Anyone on the internet can view the project. Certain features like TFVC are not supported.

**Private**  
Only people you give access to will be able to view this project.

**+ Create project**

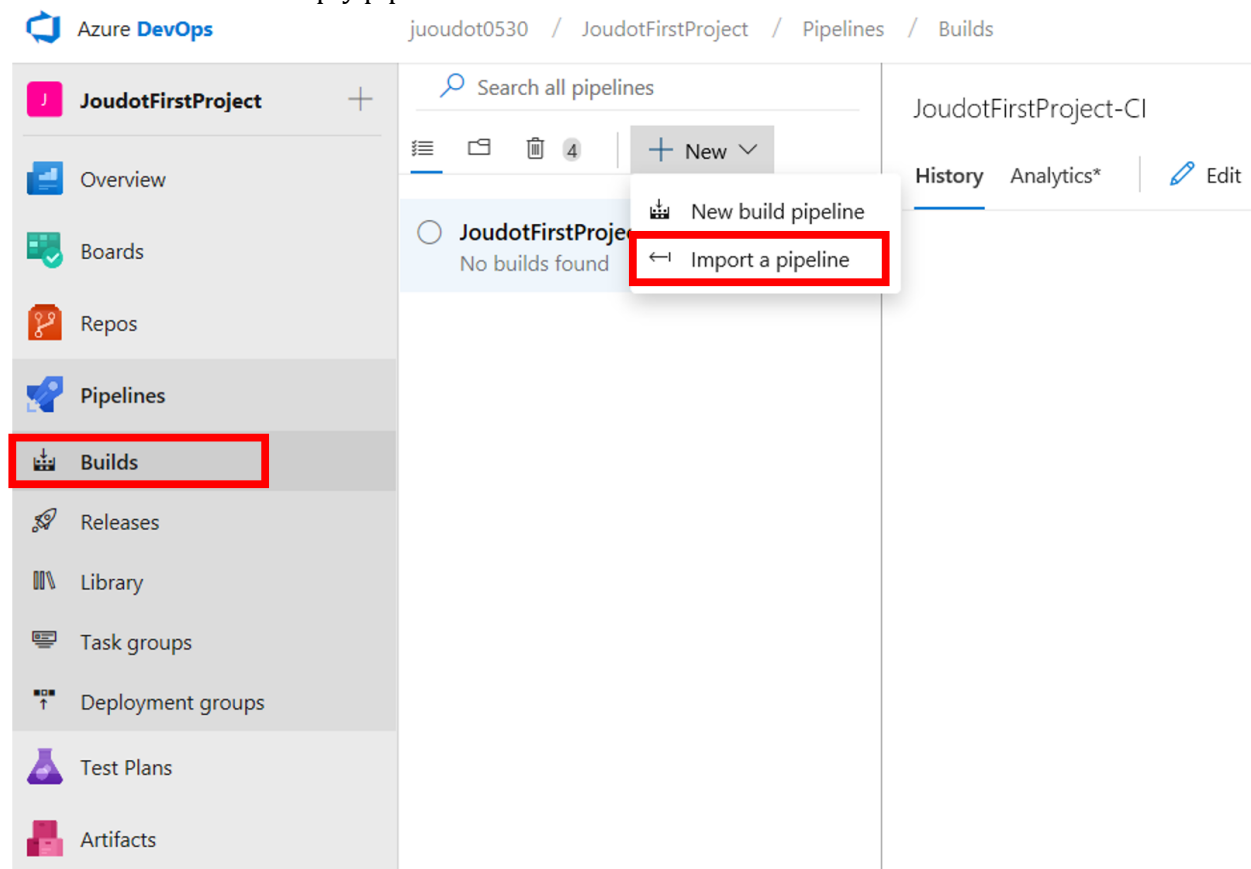
## Exercise 2: CI Build Pipeline on Azure DevOps

In this exercise, you will create Build pipelines for Linux containers. You will notice 2 options: showing how to import the build pipeline from a JSON file or showing an existing pipeline.

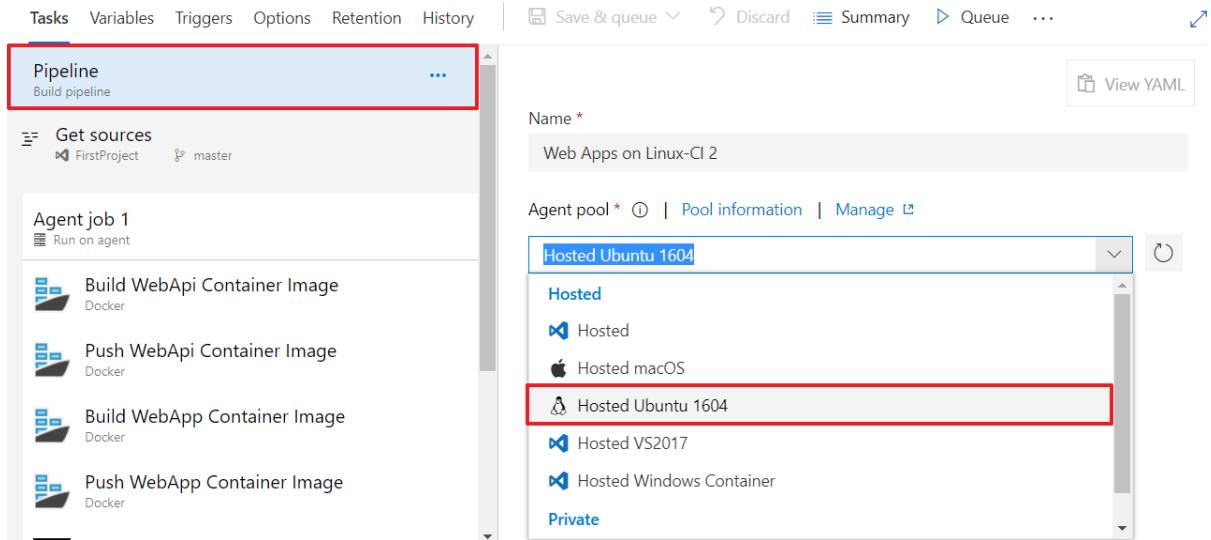
### Option 1 – Linux: Show import of a pipeline

1. Modify both the backend-webapi.yaml and frontend-webapp.yaml to point to your own instance of Azure Container Registry. Right now they point to devopslabacr.
2. Navigate to **Pipelines – Builds** and click **New – Import a pipeline**.

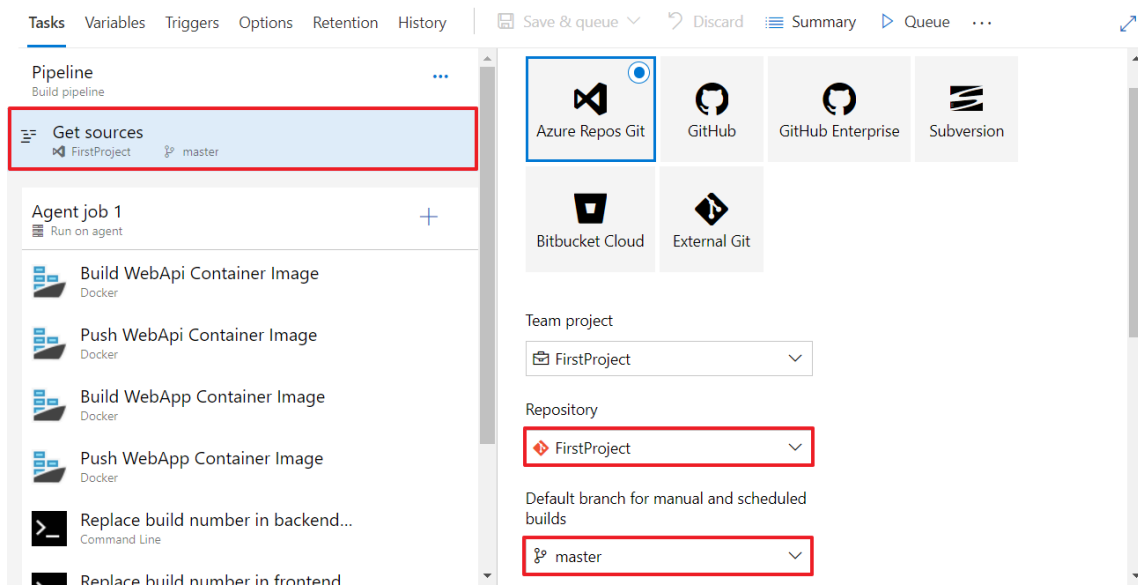
**Note:** it seems that we can only do that when at least one pipeline exists. So you will need to create an empty pipeline first.



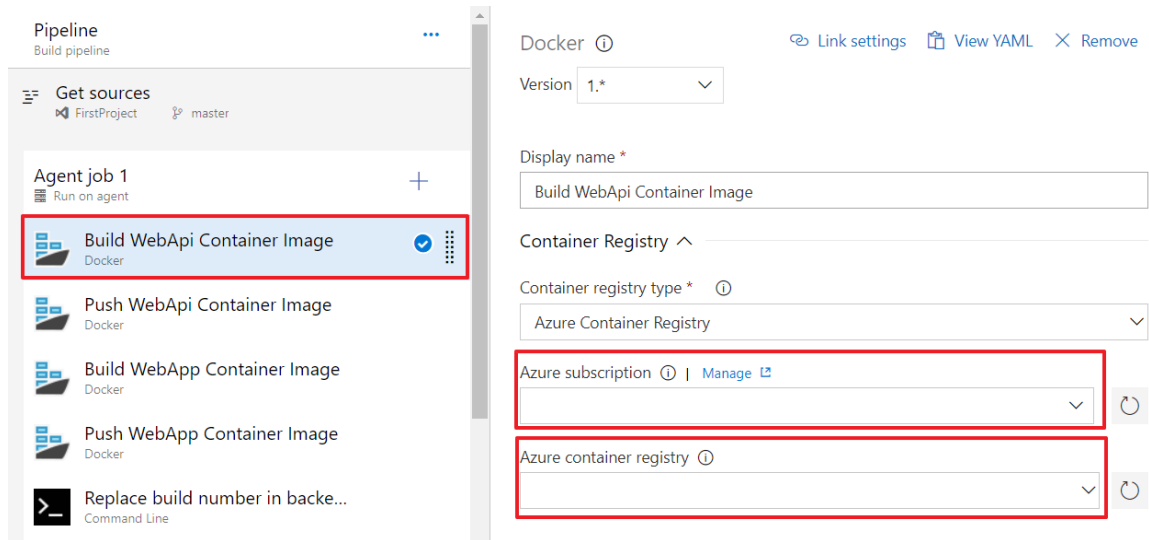
3. Select **Web Apps on Linux-Cl.json** and click **Import**
4. Select **Hosted Ubuntu 1604** as an agent pool



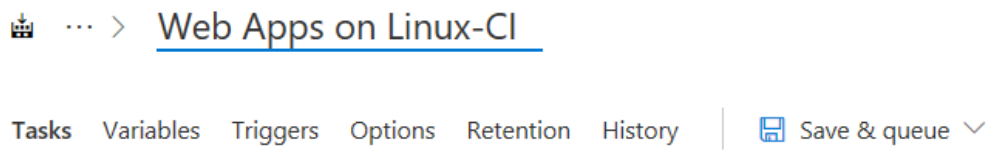
5. Click on **Get sources** and make sure that correct Git repository and branch is selected.



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



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






8. 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: Look at an existing pipeline




1. Walk through each step of the CI Build process and understand how each task works. Understand what the Variables, Triggers, Options, and History tabs do in the Build pipeline.









🏠 ... > Web Apps on Linux-CI

Tasks Variables Triggers Options Retention History |  Save & queue 

**Pipeline**   
Build pipeline

 **Get sources**  
 FirstProject  master

**Agent job 1**    
 Run on agent

-  **Build WebApi Container Image**  
Docker
-  **Push WebApi Container Image**  
Docker
-  **Build WebApp Container Image**  
Docker
-  **Push WebApp Container Image**  
Docker
-  **Replace build number in backend-webapi.yaml file**  
Command Line
-  **Replace build number in frontend-webapp.yaml file**  
Command Line
-  **Publish Artifact: frontend-webapp.yaml**  
Publish Build Artifacts
-  **Publish Artifact: backend-webapi**  
Publish Build Artifacts

- Build Web API
  - Build the image for the container based on the Dockerfile. Take a look at 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.
  - Check out the Container Registry authentication fields.
- Build Web App



- Build the image for the container based on the Dockerfile. Take a look at the contents of the Dockerfile in the Azure Repository File tab and understand 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

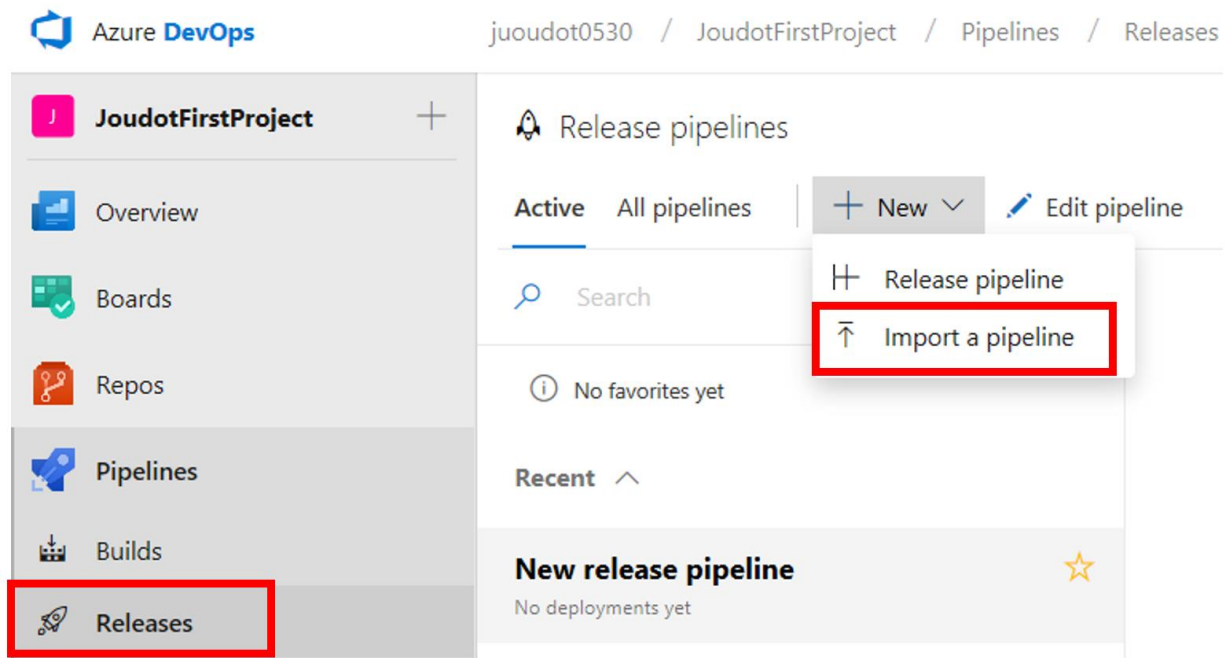
## Exercise 3: CD Release Pipeline on Azure DevOps

In this exercise, you will build a Release pipelines for an AKS cluster. 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**.

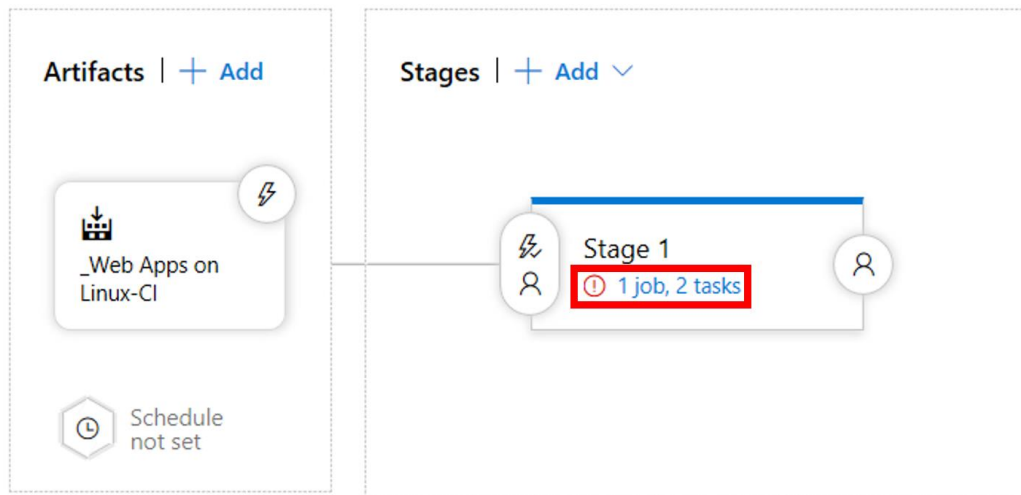
**Note:** it seems that we can only do that when at least one pipeline exists. So you will need to create an empty pipeline first



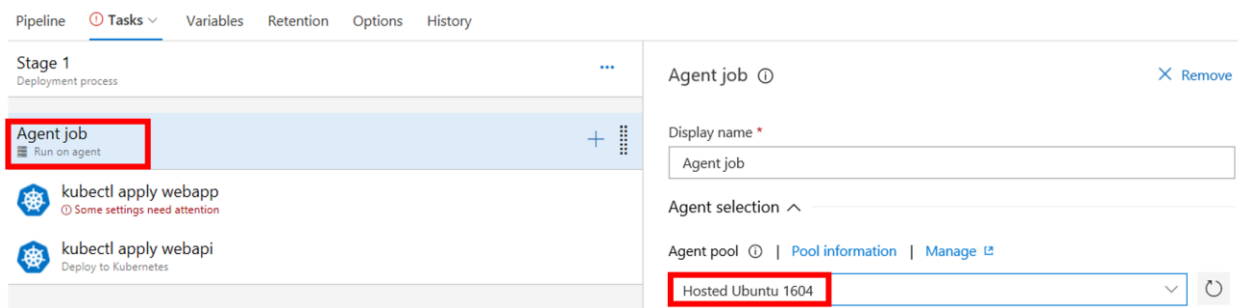
1. Select **AKS - Release-CD.json** and click **Import**
2. Make sure that the input Artifact is valid by deleting the default one: click on the Artifact Name and then **Delete**. Finally, click on **Add an Artifact** to add the artifact coming from your new build.



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



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



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

Stage 1  
Deployment process

Agent job  
Run on agent

**kubect! apply webapp**  
Deploy to Kubernetes

kubect! apply webapi  
Deploy to Kubernetes

Deploy to Kubernetes ⓘ

Version 1.\*

Display name \*  
kubect! apply webapp

Kubernetes Cluster ^

Service connection type \* ⓘ  
Azure Resource Manager

Azure subscription \* ⓘ | Manage [Manage](#)  
[Redacted] ⓘ

ⓘ Scoped to subscription 'Visual Studio Enterprise'

Resource group \* ⓘ  
k8s-aks-cluster-rg ⓘ

Kubernetes cluster \* ⓘ  
aks-k8s-cluster ⓘ

Secrets ^

Type of secret \* ⓘ  
dockerRegistry

Container registry type \* ⓘ  
Azure Container Registry

Azure subscription ⓘ | Manage [Manage](#)  
[Redacted] ⓘ

ⓘ Scoped to subscription 'Visual Studio Enterprise'

Azure container registry ⓘ  
[Redacted] ⓘ

Secret name ⓘ  
acr

☒ Force update secret ⓘ

- 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 – Look at an existing pipeline

- Walk through each step of the CD Release and understand how each task works.

## All pipelines > AKS - Release

Pipeline **Tasks** ▾ Variables Retention Options History

### Stage 1

Deployment process

#### Agent job

Run on agent



kubectl apply webapp

Deploy to Kubernetes



kubectl apply webapi

Deploy to Kubernetes

- Kubectrl apply webapp
    - Connects to Kubernetes and hands it the YAML file for the web app with the correct configuration to setup the cluster and the correct tags for the images that should be pulled from your Azure Container Registry.
  - Kubectrl apply webapi
    - Connects to Kubernetes and hands it the YAML file for the web api with the correct configuration to setup the cluster and the correct tags for the images that should be pulled from your Azure Container Registry.
2. Understand how to kick off a release pipeline: manually, on schedule, after build, after pre-condition is met.
  3. Check the results page of a completed release.
  4. Check the full deployed site.