**Deploy ASP.NET Core apps to Azure Kubernetes Service with Azure DevOps Projects**

Azure DevOps Projects presents a simplified experience where you can bring your existing code and Git repo or choose a sample application to create a continuous integration (CI) and continuous delivery (CD) pipeline to Azure.

**DevOps Projects also:**

* Automatically creates Azure resources, such as Azure Kubernetes Service (AKS).
* Creates and configures a release pipeline in Azure DevOps that sets up a build and release pipeline for CI/CD.
* Creates an Azure Application Insights resource for monitoring.
* Enables [Azure Monitor for containers](https://docs.microsoft.com/azure/azure-monitor/insights/container-insights-overview) to monitor performance for the container workloads on the AKS cluster

**In this tutorial, you will:**

* Use DevOps Projects to deploy an ASP.NET Core app to AKS
* Configure Azure DevOps and an Azure subscription
* Examine the AKS cluster
* Examine the CI pipeline
* Examine the CD pipeline
* Commit changes to Git and automatically deploy them to Azure
* Clean up resources

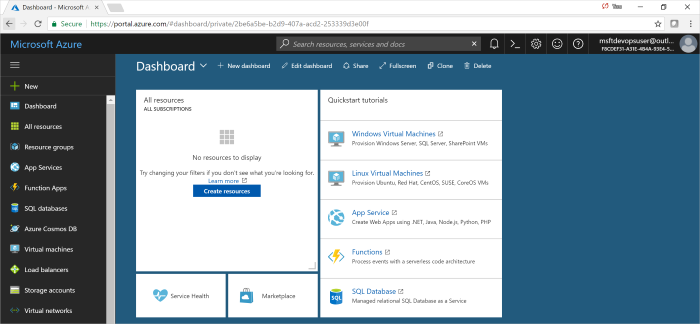
## **Prerequisites**

* An Azure subscription

## **Use DevOps Projects to deploy an ASP.NET Core app to AKS**

DevOps Projects creates a CI/CD pipeline in Azure Pipelines. You can create a new Azure DevOps organization or use an existing organization. DevOps Projects also creates Azure resources, such as an AKS cluster, in the Azure subscription of your choice.

1. Sign in to the [Azure portal](https://portal.azure.com/).
2. In the left pane, select **Create a resource**.
3. In the search box, type **DevOps Projects**, and then select **Create**.



1. Select **.NET**, and then select **Next**.
2. Under **Choose an application framework**, select **ASP.NET Core**.
3. Select **Kubernetes Service**, and then select **Next**.

## **Configure Azure DevOps and an Azure subscription**

1. Create a new Azure DevOps organization, or select an existing organization.
2. Enter a name for your Azure DevOps project.
3. Select your Azure subscription.
4. To view additional Azure configuration settings and to identify the number of nodes for the AKS cluster, select **Change**.  
   This pane displays various options for configuring the type and location of Azure services.
5. Exit the Azure configuration area, and then select **Done**.  
   After a few minutes, the process is completed. A sample ASP.NET Core app is set up in a Git repo in your Azure DevOps organization, an AKS cluster is created, a CI/CD pipeline is executed, and your app is deployed to Azure.

After all this is completed, the Azure DevOps Project dashboard is displayed in the Azure portal. You can also go to the DevOps Projects dashboard directly from **All resources** in the Azure portal.

This dashboard provides visibility into your Azure DevOps code repository, your CI/CD pipeline, and your AKS cluster. You can configure additional CI/CD options in your Azure DevOps pipeline. At the right, select **Browse** to view your running app.

## **Examine the AKS cluster**

DevOps Projects automatically configures an AKS cluster, which you can explore and customize. To familiarize yourself with the AKS cluster, do the following:

1. Go to the DevOps Projects dashboard.
2. At the right, select the AKS service.  
   A pane opens for the AKS cluster. From this view you can perform various actions, such as monitoring container health, searching logs, and opening the Kubernetes dashboard.
3. At the right, select **View Kubernetes dashboard**.  
   Optionally, follow the steps to open the Kubernetes dashboard.

## **Examine the CI pipeline**

DevOps Projects automatically configures a CI/CD pipeline in your Azure DevOps organization. You can explore and customize the pipeline. To familiarize yourself with it, do the following:

1. Go to the DevOps Projects dashboard.
2. At the top of the DevOps Projects dashboard, select **Build Pipelines**.  
   A browser tab displays the build pipeline for your new project.
3. Point to the **Status** field, and then select the ellipsis (...).  
   A menu displays several options, such as queueing a new build, pausing a build, and editing the build pipeline.
4. Select **Edit**.
5. In this pane, you can examine the various tasks for your build pipeline.  
   The build performs various tasks, such as fetching sources from the Git repo, restoring dependencies, and publishing outputs used for deployments.
6. At the top of the build pipeline, select the build pipeline name.
7. Change the name of your build pipeline to something more descriptive, select **Save & queue**, and then select **Save**.
8. Under your build pipeline name, select **History**.  
   This pane displays an audit trail of your recent changes for the build. Azure DevOps keeps track of any changes made to the build pipeline, and it allows you to compare versions.
9. Select **Triggers**.  
   DevOps Projects automatically creates a CI trigger, and every commit to the repo starts a new build. Optionally, you can choose to include or exclude branches from the CI process.
10. Select **Retention**.  
    Depending on your scenario, you can specify policies to keep or remove a certain number of builds.

## **Examine the CD release pipeline**

DevOps Projects automatically creates and configures the necessary steps to deploy from your Azure DevOps organization to your Azure subscription. These steps include configuring an Azure service connection to authenticate Azure DevOps to your Azure subscription. The automation also creates a release pipeline, which provides the CD to Azure. To learn more about the release pipeline, do the following:

1. Select **Build and Release**, and then select **Releases**.  
   DevOps Projects creates a release pipeline to manage deployments to Azure.
2. Select the ellipsis (...) next to your release pipeline, and then select **Edit**.  
   The release pipeline contains a pipeline, which defines the release process.
3. Under **Artifacts**, select **Drop**.  
   The build pipeline you examined in the previous steps produces the output that's used for the artifact.
4. At the right of the **Drop** icon, select **Continuous deployment trigger**.  
   This release pipeline has an enabled CD trigger, which executes a deployment every time a new build artifact is available. Optionally, you can disable the trigger so that your deployments require manual execution.
5. At the right, select **View releases** to display a history of releases.
6. Select the ellipsis (...) next to a release, and then select **Open**.  
   You can explore several menus, such as a release summary, associated work items, and tests.
7. Select **Commits**.  
   This view shows code commits that are associated with this deployment. Compare releases to view the commit differences between deployments.
8. Select **Logs**.  
   The logs contain useful information about the deployment process. You can view them both during and after deployments.

## Commit changes to Azure Repos and automatically deploy them to Azure

The following procedure tests the CI/CD pipeline by making a simple text change.

You're now ready to collaborate with a team on your app by using a CI/CD process that automatically deploys your latest work to your website. Each change to the Git repo starts a build in Azure DevOps, and a CD pipeline executes a deployment to Azure. Follow the procedure in this section, or use another technique to commit changes to your repo. For example, you can clone the Git repo in your favorite tool or IDE, and then push changes to this repo.

1. In the Azure DevOps menu, select **Code** > **Files**, and then go to your repo.
2. Go to the Views\Home directory, select the ellipsis (...) next to the Index.cshtml file, and then select **Edit**.
3. Make a change to the file, such as adding some text within one of the div tags.
4. At the top right, select **Commit**, and then select **Commit** again to push your change.  
   After a few moments, a build starts in Azure DevOps and a release executes to deploy the changes. Monitor the build status on the DevOps Projects dashboard or in the browser with your Azure DevOps organization.
5. After the release is completed, refresh your app to verify your changes.

## **Clean up resources**

Delete All Resources