Contents

Overview of DevOps Starter

6/28/2022 • 3 minutes to read • Edit Online

DevOps Starter makes it easy to get started on Azure using either GitHub actions or Azure DevOps. It helps you launch your favorite app on the Azure service of your choice in just a few quick steps from the Azure portal.

DevOps Starter sets up everything you need for developing, deploying, and monitoring your application. You can use the DevOps Starter dashboard to monitor code commits, builds, and deployments, all from a single view in the Azure portal.

Advantages of using DevOps Starter

DevOps starter the following supports two CI/CD providers, to automate your deployments

- GitHub Actions
- Azure DevOps

DevOps Starter automates the setup of an entire continuous integration (CI) and continuous delivery (CD) for your application to Azure. You can start with existing code or use one of the provided sample applications. Then you can quickly deploy that application to various Azure services such as Virtual Machines, App Service, Azure Kubernetes Services (AKS), Azure SQL Database, and Azure Service Fabric.

DevOps Starter does all the work for the initial configuration of a DevOps pipeline including everything from setting up the initial Git repository, configuring the CI/CD pipeline, creating an Application Insights resource for monitoring, and providing a single view of the entire solution with the creation of a DevOps Starter dashboard in the Azure portal.

You can use DevOps Starter to:

- Quickly deploy your application to Azure
- Automate the setup of a CI/CD workflow or pipeline
- View and understand how to properly set up a CI/CD workflow or pipeline
- Further customize the release pipelines based on your specific scenarios

How to use DevOps Starter?

DevOps Starter is available from the Azure portal. You create a DevOps Starter resource just like you create any other Azure resource from the portal. DevOps Projects provides a step-by-step wizard-like experience for the various configuration options.

You choose several configuration options as part of the initial setup. These options include:

- Selecting your favoring CI/CD provider
- Using the provided sample app, or bringing your own code(only for Azure DevOps)
- Selecting an app language
- Choosing an app framework based on language
- Selecting an Azure service (deployment target)
- Select your GitHub or Azure DevOps organization
- Choosing your Azure subscription
- Picking the location of Azure services
- Choosing from various pricing tiers for Azure services

After creating your DevOps Starter, you can:

- Customize your GitHub workflow or Azure DevOps Pipeline
- Use pull requests to manage your code flow and keep your quality high
- Test and build each commit before you merge your code to raise the quality bar

After you use DevOps Starter, you can also delete all of the resources from a single place from the DevOps Starter dashboard on the Azure portal.

DevOps Starter and GitHub integration

DevOps Starter now supports GitHub actions as a CI/CD provider. It automates all of the work that's needed in GitHub to set up a CI/CD workflow using GitHub Actions. It creates a GitHub repository in an existing GitHub organization, and then commits a sample application to the new GitHub repository.

The automation also establishes a trigger for the workflow so that every new code commit initiates a build and deploy job within the workflow. The application is deployed to the Azure service of your choice. The GitHub workflow can be customized for additional scenarios.

DevOps Starter and Azure DevOps integration

DevOps Starter using Azure DevOps automates all of the work that's needed in Azure Pipelines to set up a CI/CD pipeline. It creates a Git repository in a new or existing Azure DevOps organization, and then commits a sample application or your existing code to a new Git repository.

The automation also establishes a CI trigger for the build so that every new code commit initiates a build. DevOps Starter creates a CD trigger and deploys every new successful build to the Azure service of your choice.

The build and release pipelines can be customized for additional scenarios. Additionally, you can clone the build and release pipelines for use in other projects.

Getting started with DevOps Starter

- Get started with DevOps Starter

DevOps Starter videos

- Create CI/CD with Azure DevOps Starter

Set up CI/CD for a Node.js app with DevOps Starter using GitHub Actions

6/28/2022 • 4 minutes to read • Edit Online

In this quickstart, you use the simplified DevOps Starter experience to set up a continuous integration (CI) and continuous delivery (CD) workflow for your Node.js app using GitHub Actions. You can use DevOps Starter to set up everything you need for developing, deploying, and monitoring your app.

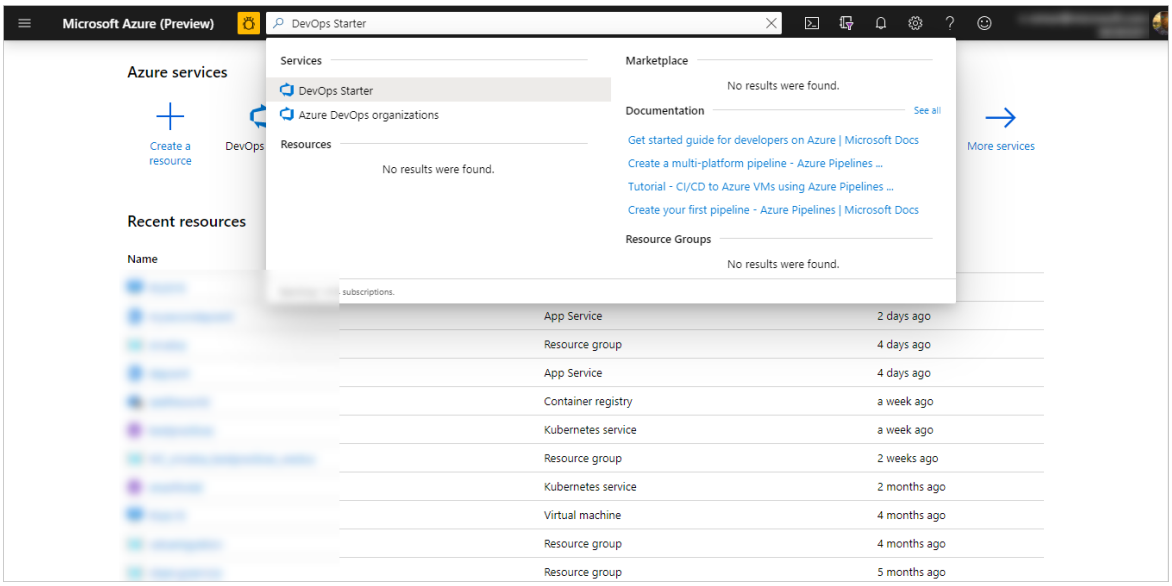
Prerequisites

- An Azure account with an active subscription. Create an account for free.
- A GitHub account.

Sign in to the Azure portal

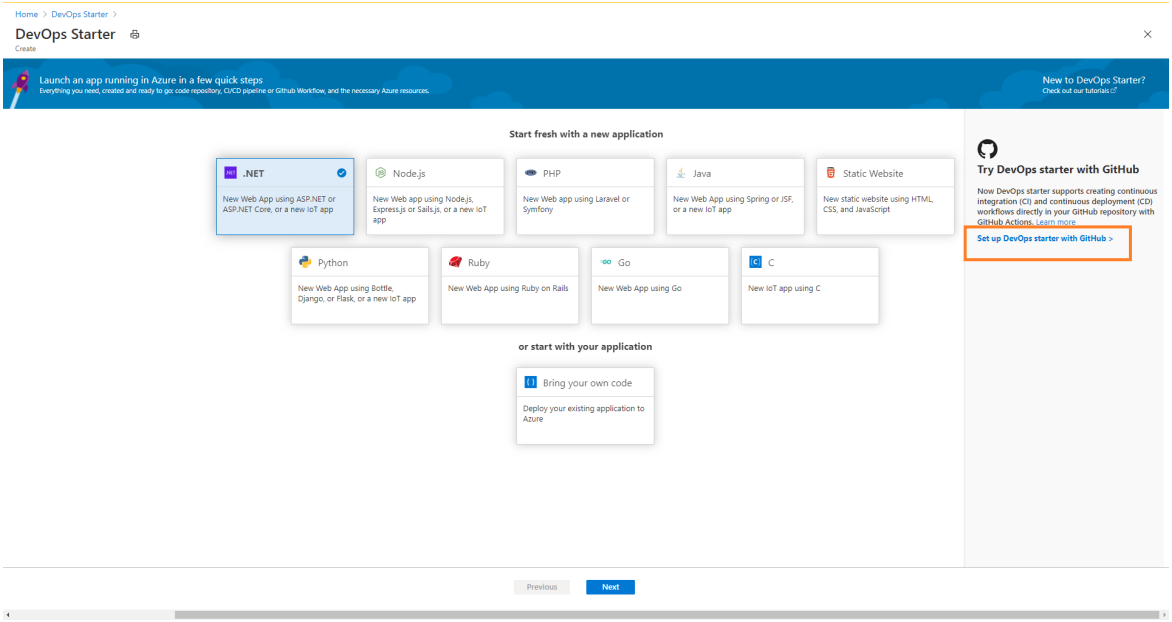
DevOps Starter creates a CI/CD workflow using GitHub actions. DevOps Starter also creates Azure resources in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.

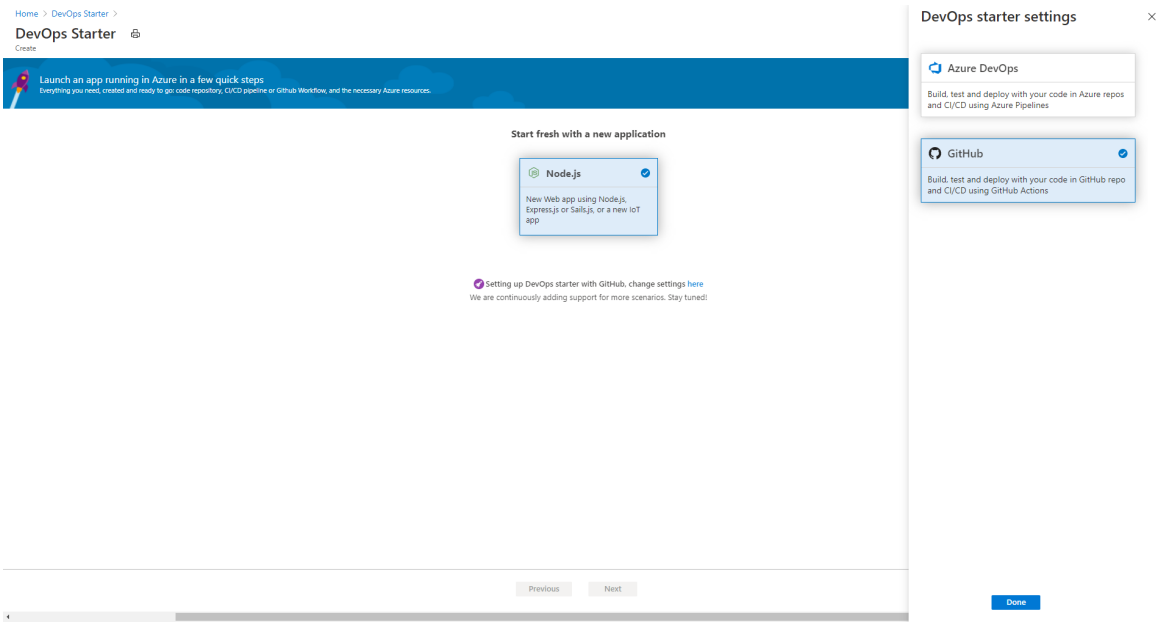


Select a sample application and Azure service

1. Click on **Set up DevOps starter with GitHub** on the right side banner.



2. Ensure that the CI/CD provider is selected as **GitHub Actions**.

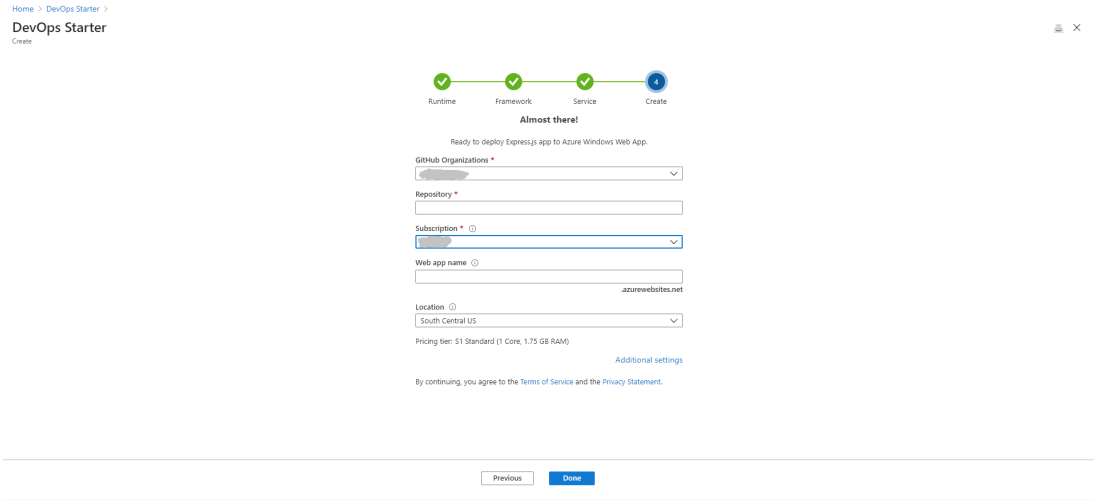


3. Select the **Node.js** sample application. The Node.js samples include a choice of several application frameworks.
4. The default sample framework is **Express.js**. Leave the default setting, and then select **Next**.
5. Windows Web App is the default deployment target. The application framework, which you chose previously, dictates the type of Azure service deployment target available here. Leave the default service, and then select **Next**.

Configure GitHub account and an Azure subscription

1. Authenticate with GitHub.
   a. Click on the **Authorize** button.
   b. Sign in to GitHub. If you don't have a GitHub account, you can sign up here as well.
2. Choose an existing **GitHub organization**.
   a. Choose a name for your GitHub repository.

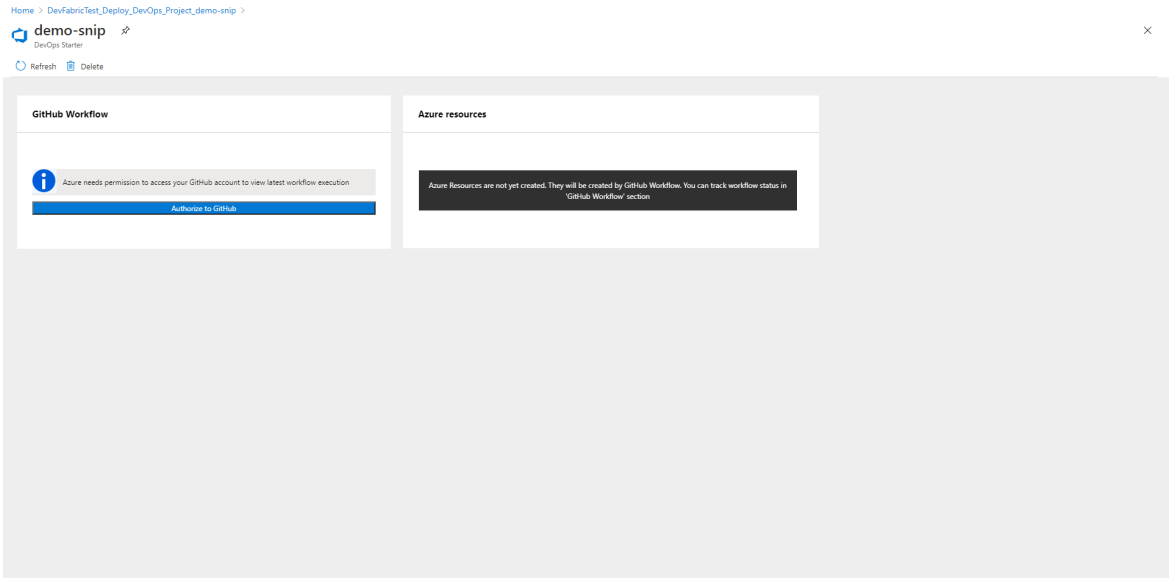
b. Select your Azure subscription and location, choose a name for your application, and then select **Done**.



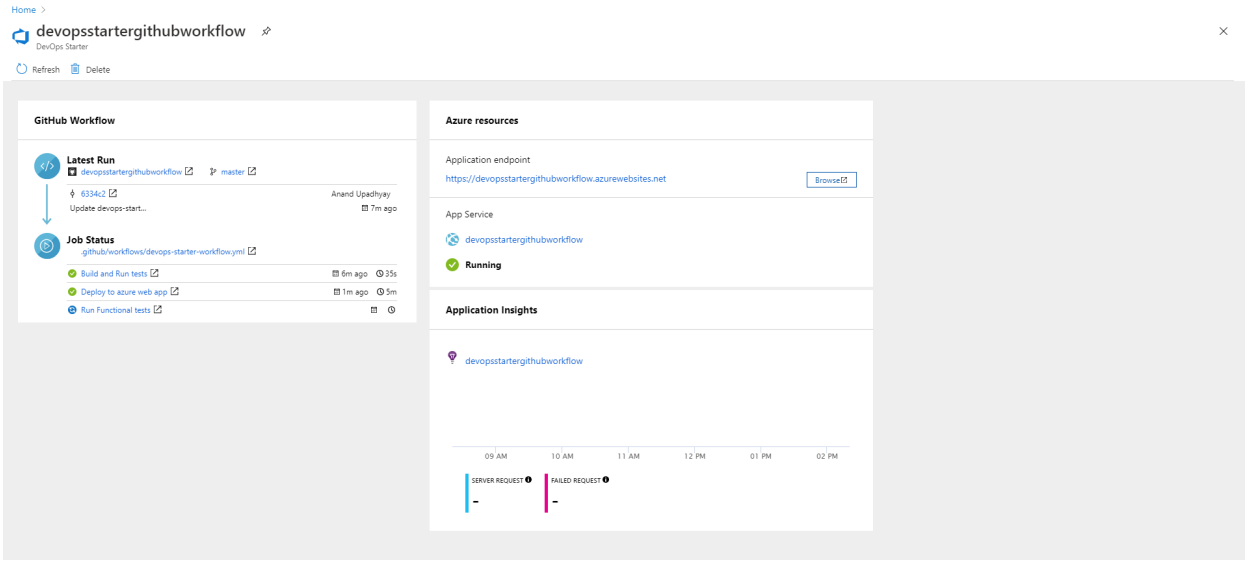
After a few minutes, the DevOps Starter dashboard is displayed in the Azure portal. A sample application is set up in a repository in your Azure DevOps organization, a GitHub workflow is triggered, and your application is deployed to Azure. This dashboard provides visibility into your code repository, the GitHub workflow, and your application in Azure.

3. Select **Browse** to view your running application.

   The dashboard contains details of GitHub workflow and Azure resources. To view the details of GitHub workflow like latest run, commits, and job status you will need to **Authorize to GitHub**.



DevOps Starter automatically configured a GitHub workflow with build and deploy jobs using GitHub actions. You're now ready to collaborate with a team on a Node.js app with a CI/CD process that automatically deploys your latest work to your web site.



Commit code changes and execute CI/CD

DevOps Starter creates a repository in GitHub. To view the repository and make code changes to your application, do the following:

1. On the left of the DevOps Starter dashboard, select the link for your main branch. This link opens a view to the newly created GitHub repository.
2. To view the repository clone URL, select **Clone** on the top right of the browser. You can clone your Git repository in your favorite IDE. In the next few steps, you can use the web browser to make and commit code changes directly to the main branch.
3. On the left side of the browser, go to the **/Application/views/index.pug** file.
4. Select **Edit**, and then make a change to some of the text. For example, change some of the text for one of the tags.
5. Select **Commit**, and then save your changes.
6. In your browser, go to the DevOps Starter dashboard.
   You should now see a GitHub workflow build job in progress. The changes you made are automatically built and deployed via a GitHub workflow.

View the GitHub workflow

In the previous step, DevOps Starter automatically configured a full GitHub workflow. Explore and customize the workflow as needed. Take the following steps to familiarize yourself with the workflow.

1. On the left of the DevOps Starter dashboard, select **GitHub workflow**. This link opens a browser tab and the GitHub workflow for your new project.

   > **NOTE**
   > Do not rename the workflow file. The name of the workflow file should be **devops-starter-workflow.yml** for the dashboard to reflect the changes

2. The workflow yaml file contains all the GitHub Actions required to build and deploy the application. Click on **edit file** option to customize your workflow file.
3. Under the **Code** tab of the repo click on **commits**. This view shows code commits that are associated

with the specific deployment.

4. Under the **Actions** tab of the repo, you can view the history of all the workflow runs of your repository.
5. Select the **latest run** to view all the jobs that ran in the workflow.
6. Click on the **jobs** to view the detailed logs of the workflow run. The logs contain useful information about the deployment process. They can be viewed both during and after deployments.
7. Click on the **Pull request** tab to view all the pull requests on your repository

Clean up resources

You can delete Azure App Service and other related resources when you don't need them anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

When you configured your CI/CD process, GitHub workflow was automatically created. You can modify this workflow to meet the needs of your team. To learn more about the GitHub Actions and workflow, see:

Customize GitHub workflow

Create a CI/CD pipeline for .NET with Azure DevOps Starter

6/28/2022 • 5 minutes to read • Edit Online

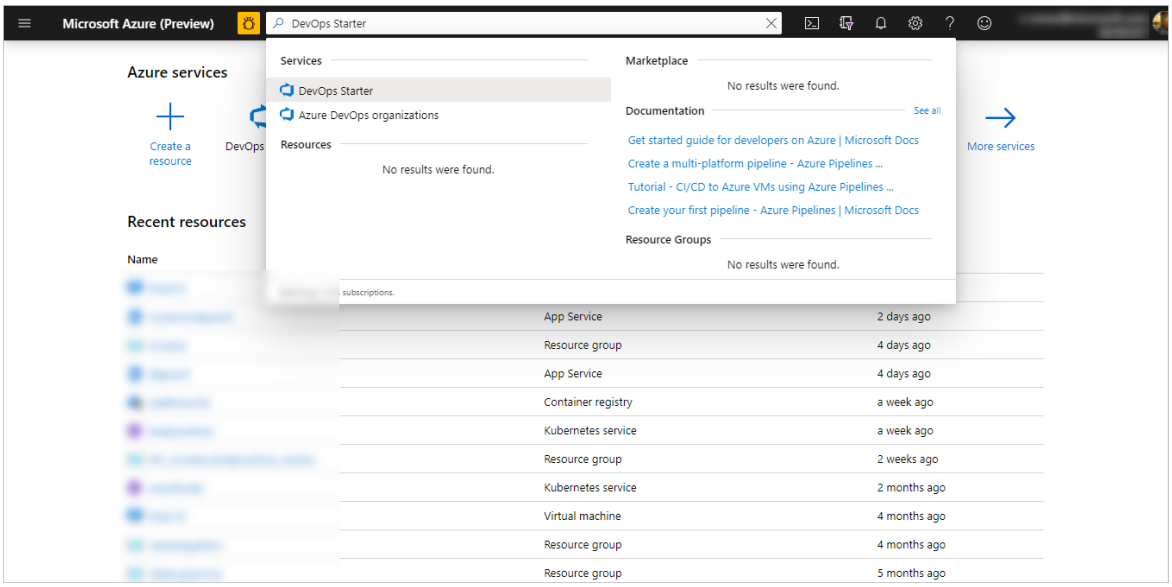
Configure continuous integration (CI) and continuous delivery (CD) for your .NET core or ASP.NET application with DevOps Starter. DevOps Starter simplifies the initial configuration of a build and release pipeline in Azure Pipelines.

If you don't have an Azure subscription, you can get one free through Visual Studio Dev Essentials.

Sign in to the Azure portal

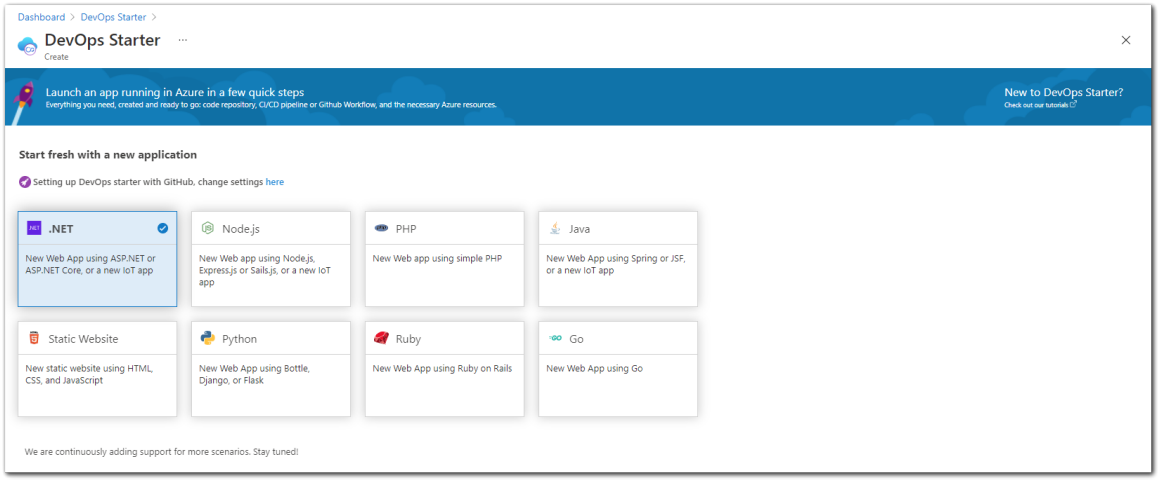
DevOps Starter creates a CI/CD pipeline in Azure DevOps. You can create a new Azure DevOps organization or use an existing organization. DevOps Starter also creates Azure resources in the Azure subscription of your choice.

1. Sign in to the Microsoft Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



Select a sample application and Azure service

1. Select the **.NET** sample application. The .NET samples include a choice of either the open-source ASP.NET framework or the cross-platform .NET Core framework.



> **NOTE**
>
> The default option for setting up DevOps starter is with **GitHub**, but this setting can be changed from the wizard.

2. This sample is an ASP.NET Core MVC application. Select the **.NET Core** application framework, then select **Next**.

3. Select **Windows Web App** as a deployment target, then select **Next**. Optionally, you can choose other Azure services for your deployment. The application framework, which you chose previously, dictates the type of Azure service deployment target's available here.

Configure Azure DevOps and an Azure subscription

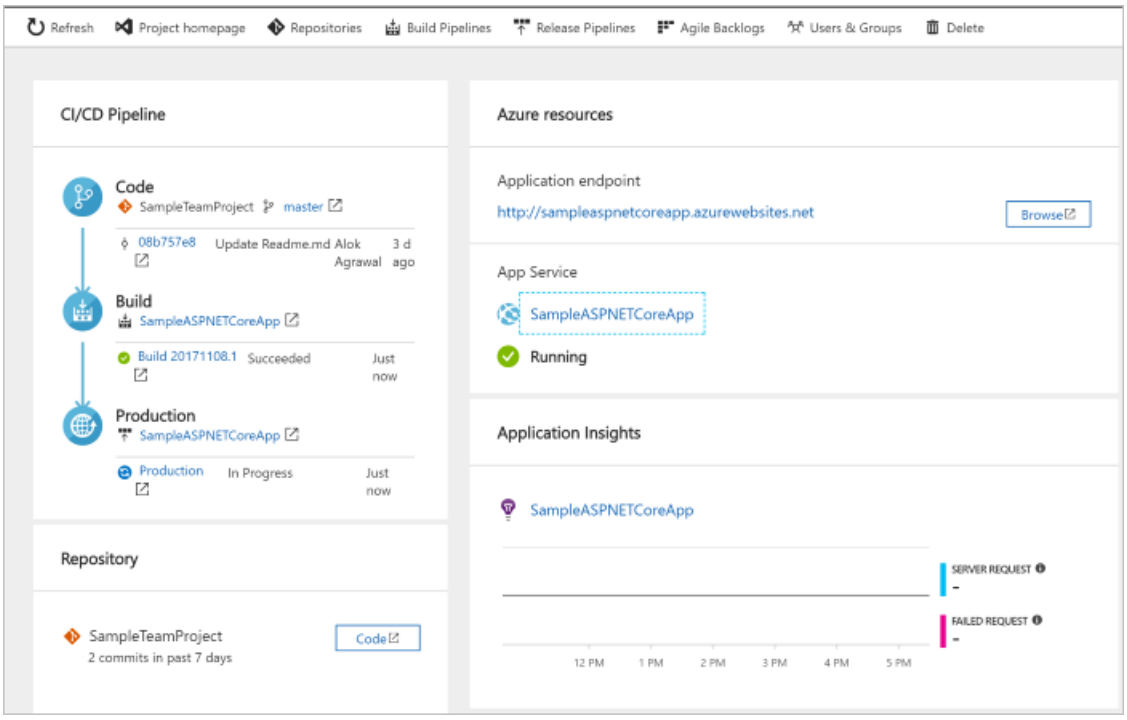
1. Enter a **Project name**.

2. Create a new free **Azure DevOps Organization** or choose an existing organization from the dropdown.

3. Select your **Azure Subscription**, enter a name for your **Web app** or take the default, then select **Done**. After a few minutes, the DevOps Starter Deployment Overview is displayed in the Azure portal.

4. Select **Go to resource** to view the DevOps Starter dashboard. In the upper right corner, pin the **Project** to your dashboard for quick access. A sample app is set up in a repo in your **Azure DevOps Organization**. A build is executed, and your app is deployed to Azure.

5. The dashboard provides visibility into your code repo, your CI/CD pipeline, and your app in Azure. At the right under Azure resources, select **Browse** to view your running app.



Commit code changes and execute CI/CD

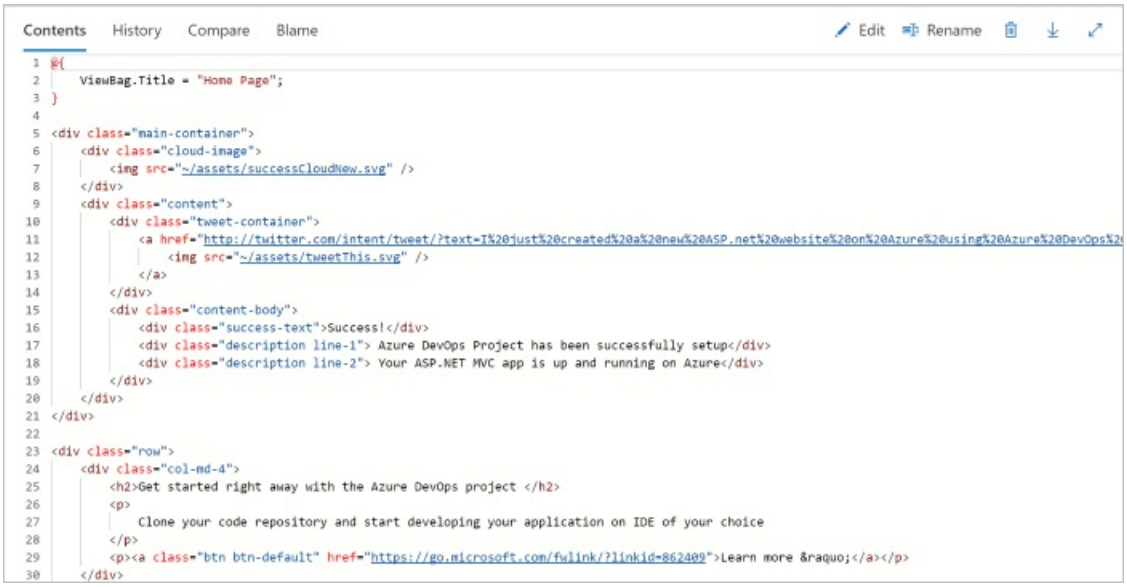
DevOps Starter created a Git repository in Azure Repos or GitHub. To view the repository and make code changes to your application, do the following:

1. On the left of the DevOps Starter dashboard, select the link for your **main** branch. This link opens a view to the newly created Git repository.

2. In the next few steps, you can use the web browser to make and commit code changes directly to the **main** branch. You can also clone your Git repository in your favorite IDE by selecting **Clone** from the top right of the repository page.

3. On the left, navigate the application file structure to **Application/aspnet-core-dotnet-core/Pages/Index.cshtml**.

4. Select **Edit**, and then make a change to the h2 heading. For example, type **Get started right away with the Azure DevOps Starter** or make some other change.



5. Select **Commit**, leave a comment and select **Commit** again.

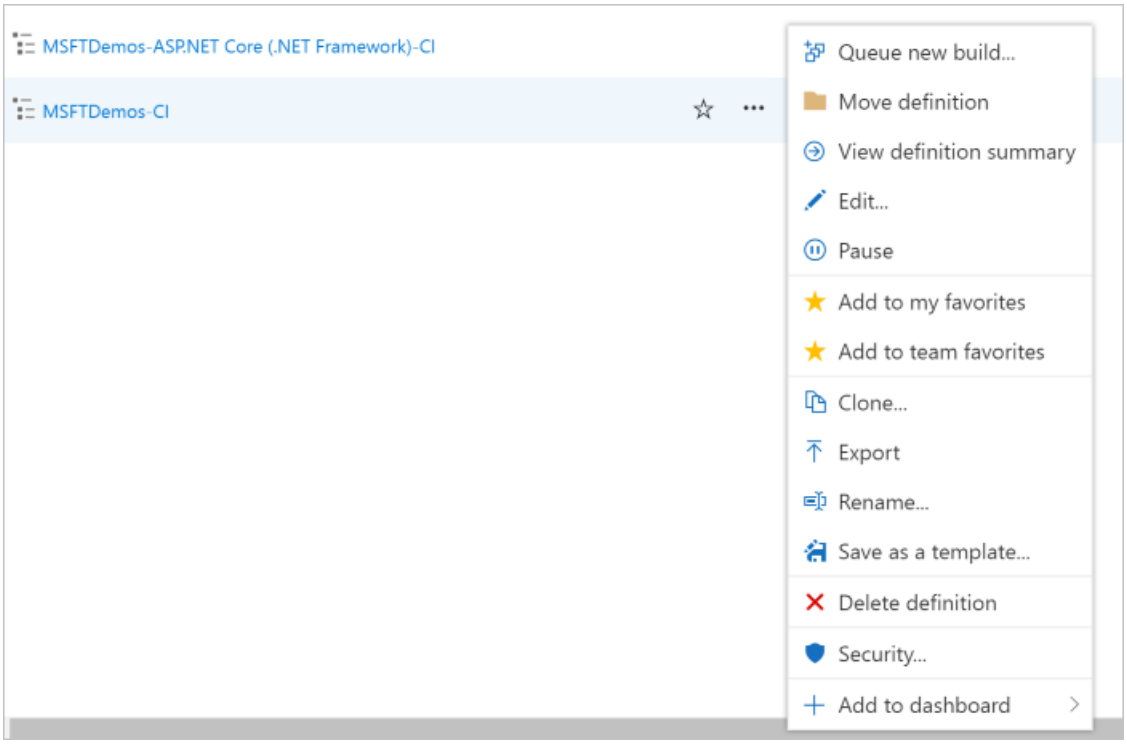
6. In your browser, go to the Azure DevOps Starter dashboard. You should now see a build is in progress.

The changes you made are automatically built and deployed via a CI/CD pipeline.

Examine the CI/CD pipeline

In the previous step, Azure DevOps Starter automatically configured a full CI/CD pipeline. Explore and customize the pipeline as needed. Take the following steps to familiarize yourself with the Azure DevOps build and release pipelines.

1. At the top of the DevOps Starter dashboard, select **Build Pipelines**. This link opens a browser tab and the Azure DevOps build pipeline for your new project.
2. Select the ellipsis (...). This action opens a menu where you can start several activities such as queuing a new build, pausing a build, and editing the build pipeline.
3. Select **Edit**.



4. 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 repository, restoring dependencies, and publishing outputs used that are used for deployments.
5. At the top of the build pipeline, select the build pipeline name.
6. Change the name of your build pipeline to something more descriptive, select **Save & queue**, and then select **Save**.
7. Under your build pipeline name, select **History**.
   In the **History** pane, you see an audit trail of your recent changes for the build. Azure Pipelines keeps track of any changes that are made to the build pipeline, and it allows you to compare versions.
8. Select **Triggers**. DevOps Starter automatically created a CI trigger, and every commit to the repository starts a new build. You can optionally choose to include or exclude branches from the CI process.
9. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.
10. Select **Build and Release**, then select **Releases**.
    DevOps Starter creates a release pipeline to manage deployments to Azure.

11. On the left, select the ellipsis (...) next to your release pipeline, and then select **Edit**. The release pipeline contains a pipeline, which defines the release process.
12. Under **Artifacts**, select **Drop**. The build pipeline you examined in the previous steps produces the output used for the artifact.
13. Next to the **Drop** icon, select the **Continuous deployment trigger**. This release pipeline has an enabled CD trigger, which runs a deployment every time there is a new build artifact available. Optionally, you can disable the trigger so that your deployments require manual execution.
14. On the left, select **Tasks**. The tasks are the activities that your deployment process performs. In this example, a task was created to deploy to Azure App Service.
15. On the right, select **View releases**. This view shows a history of releases.
16. Select the ellipsis (...) next to one of your releases, and then select **Open**. There are several menus to explore, such as a release summary, associated work items, and tests.
17. Select **Commits**. This view shows code commits that are associated with the specific deployment.
18. Select **Logs**. The logs contain useful information about the deployment process. They can be viewed both during and after deployments.

Clean up resources

You can delete Azure App Service and other related resources that you created when you don't need them anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

To learn more about modifying the build and release pipelines to meet the needs of your team, see this tutorial:

Customize CD process

Videos

Create a CI/CD pipeline in Azure Pipelines for Node.js with Azure DevOps Starter

6/28/2022 • 6 minutes to read • Edit Online

In this quickstart, you create a NodeJS progressive web app (PWA) using GatsbyJS and the simplified Azure DevOps Starter creation experience. When finished, you have a continuous integration (CI) and continuous delivery (CD) pipeline for your PWA in Azure Pipelines. Azure DevOps Starter sets up what you need for developing, deploying, and monitoring.

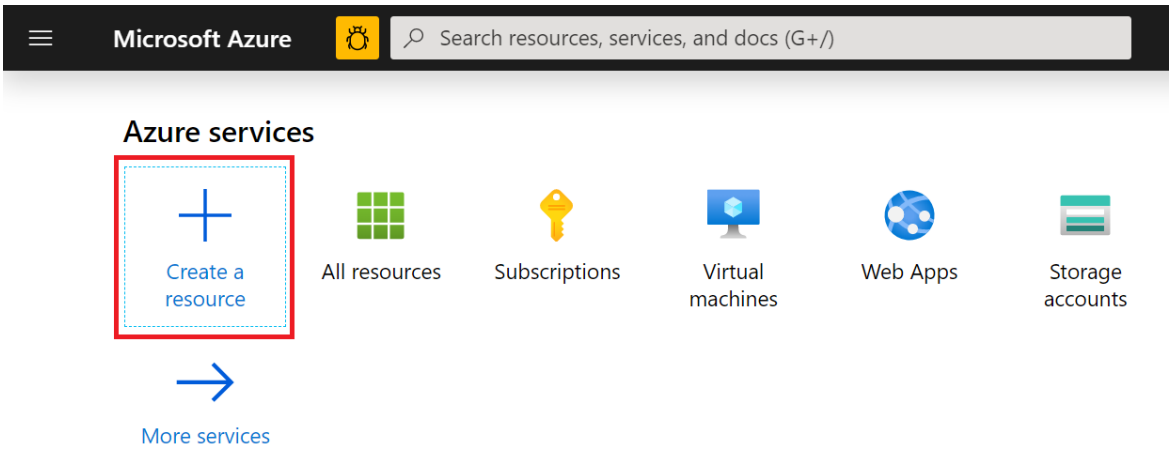
Prerequisites

- An Azure account with an active subscription. Create an account for free.
- An Azure DevOps organization.

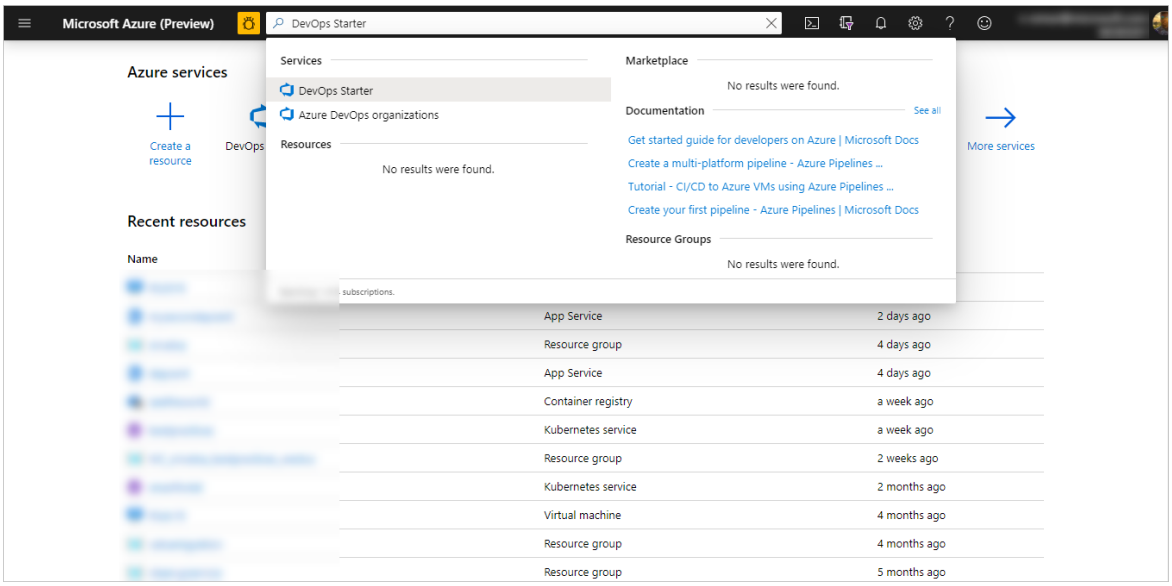
Sign in to the Azure portal

DevOps Starter creates a CI/CD pipeline in Azure Pipelines. You can create a new Azure DevOps organization or use an existing organization. DevOps Starter also creates Azure resources in the Azure subscription of your choice.

1. Sign in to the Azure portal, and in the left pane, select **Create a resource**.

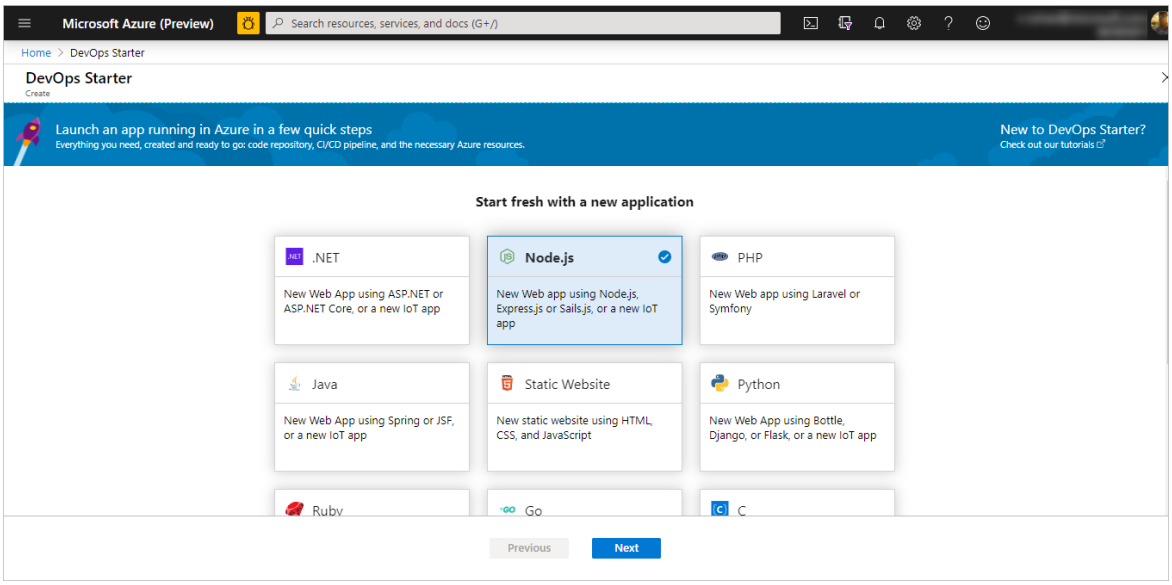


2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.

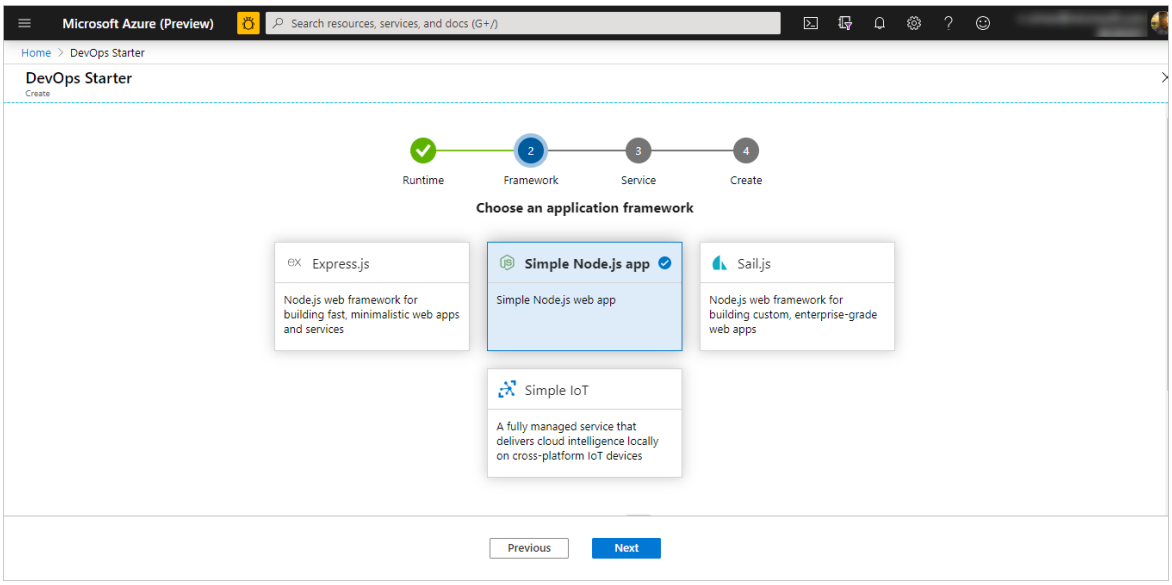


Select a sample application and Azure service

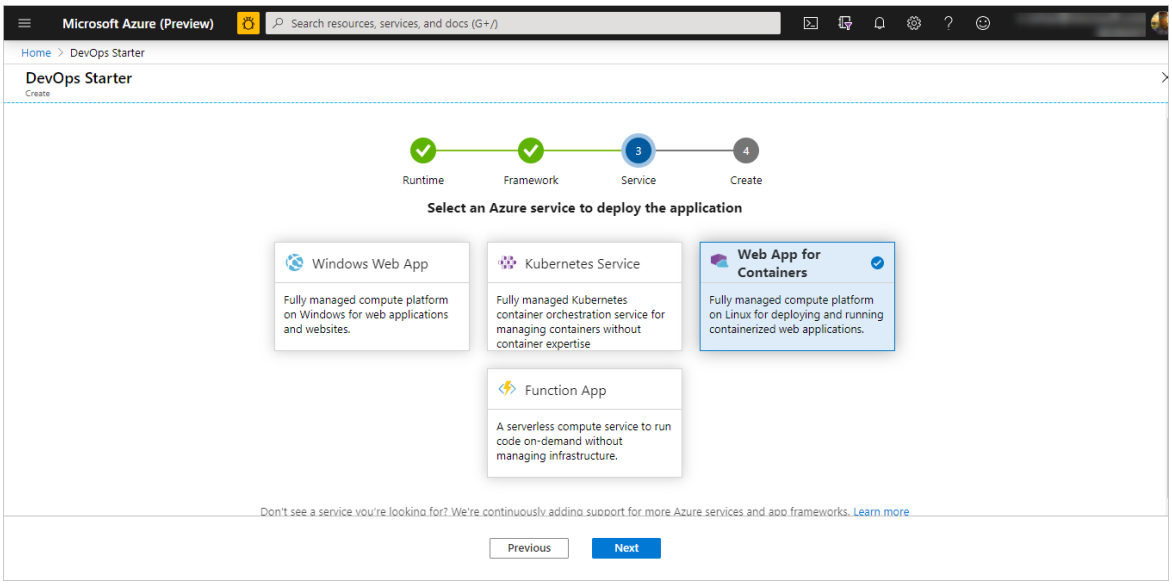
1. Select the Node.js sample application.



2. The default sample framework is **Express.js**. Change the selection to **Simple Node.js App** and then select **Next**.

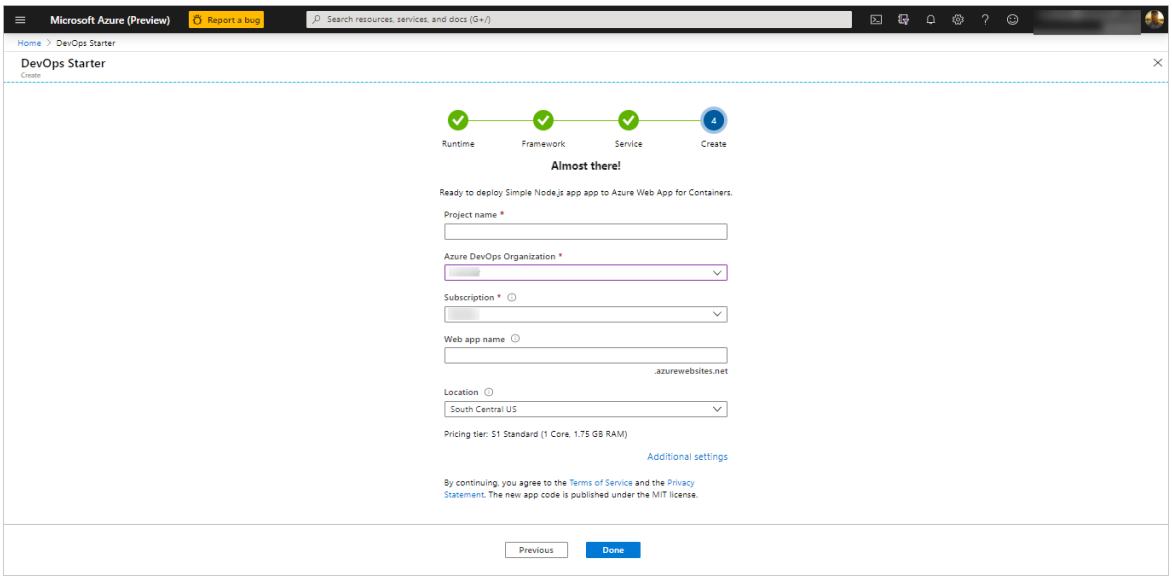


3. The deployment targets available in this step are dictated by the application framework selected in step 2. In this example, **Windows Web App** is the default deployment target. Leave **Web App for Containers** set and select **Next**.



Configure a project name and an Azure subscription

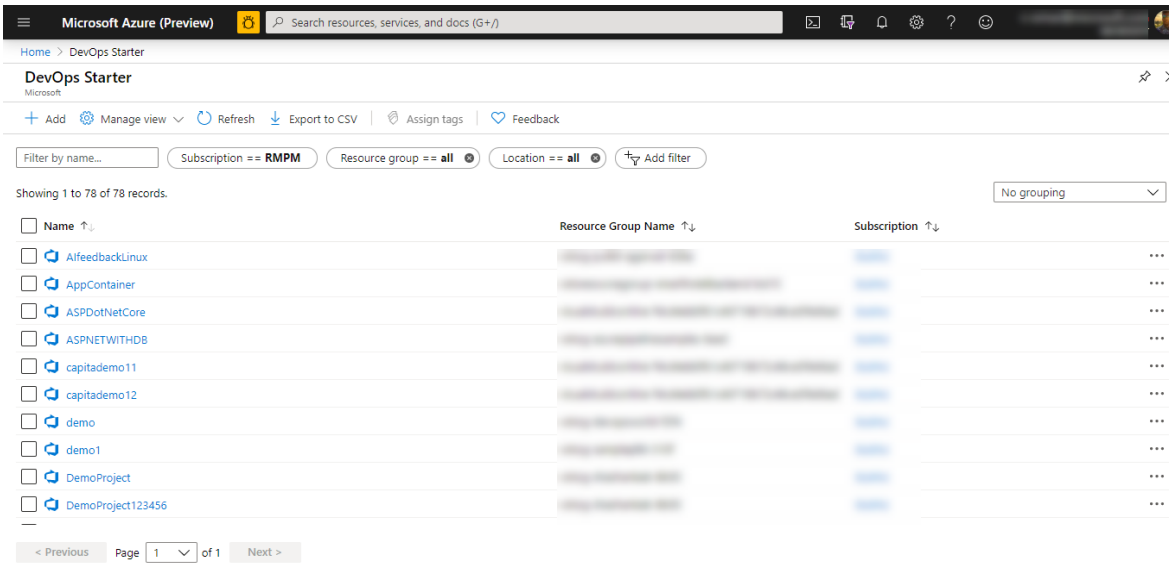
1. In the final step of the DevOps Starter creation workflow, you assign a project name, select an Azure subscription, and select **Done**.



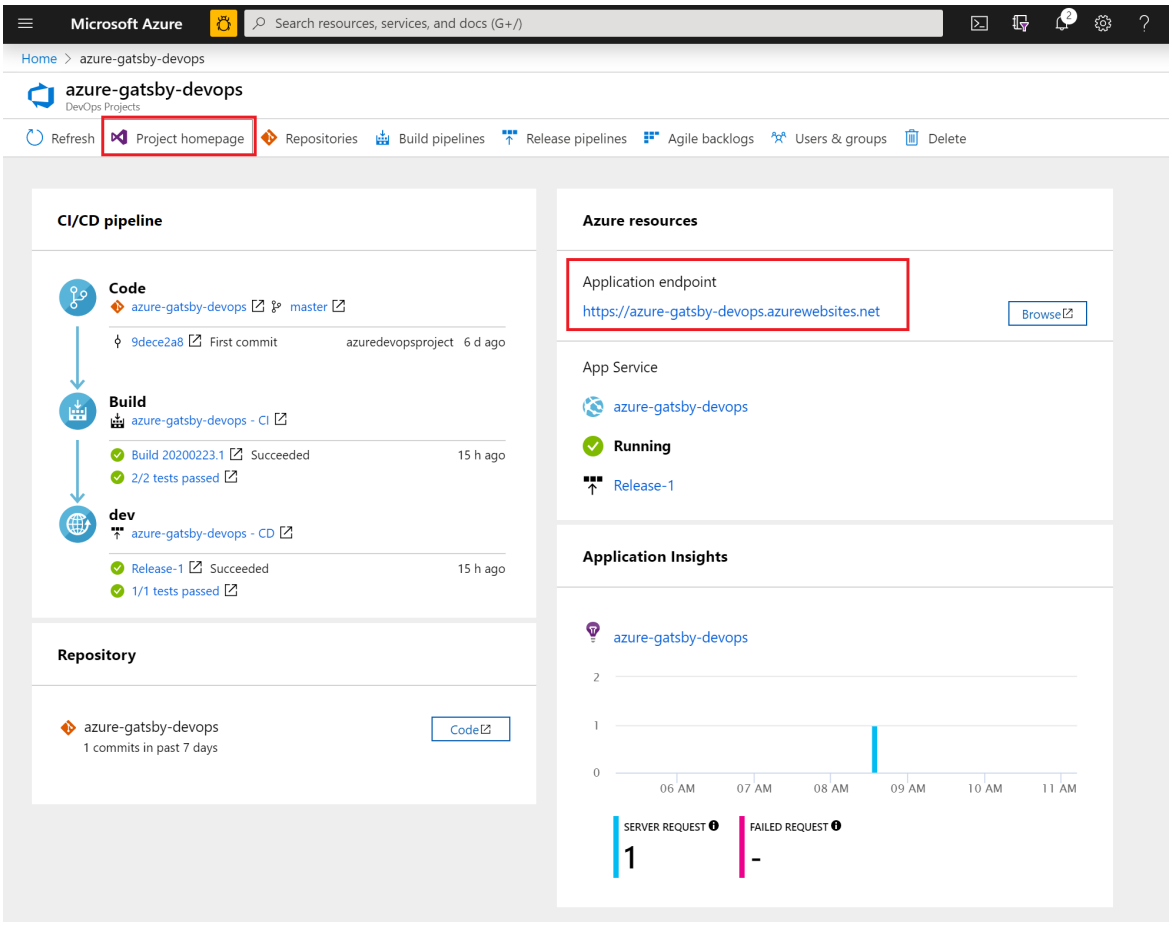
2. A summary page displays while your project is built and your application is deployed to Azure. After a brief period, a project is created in your Azure DevOps organization that includes a git repo, a Kanban board, a deployment pipeline, test plans, and the artifacts required by your app.

Managing your project

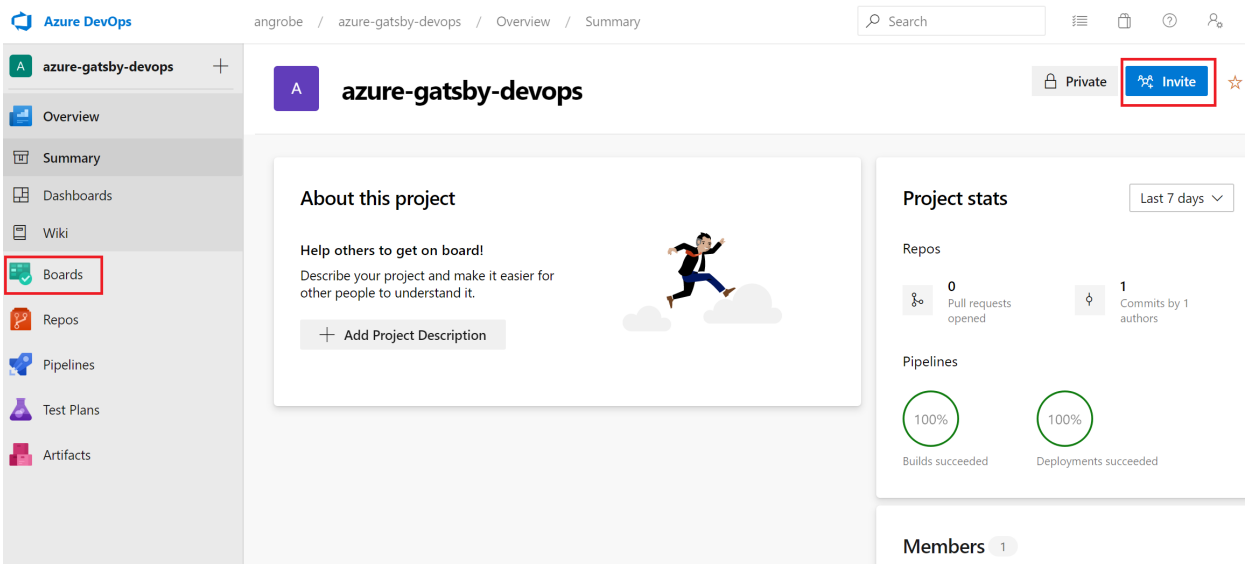
1. Navigate to **All Resources** and find your DevOps Starter. Select your **DevOps Starter**.



2. You are directed to a dashboard that provides visibility into your project homepage, code repository, the CI/CD pipeline, and a link to your running app. Select the **Project Homepage** to view your application in **Azure DevOps** and, in another browser tab, select the **Application Endpoint** to view the live sample app. We change this sample later to use GatsbyJS generated PWA.



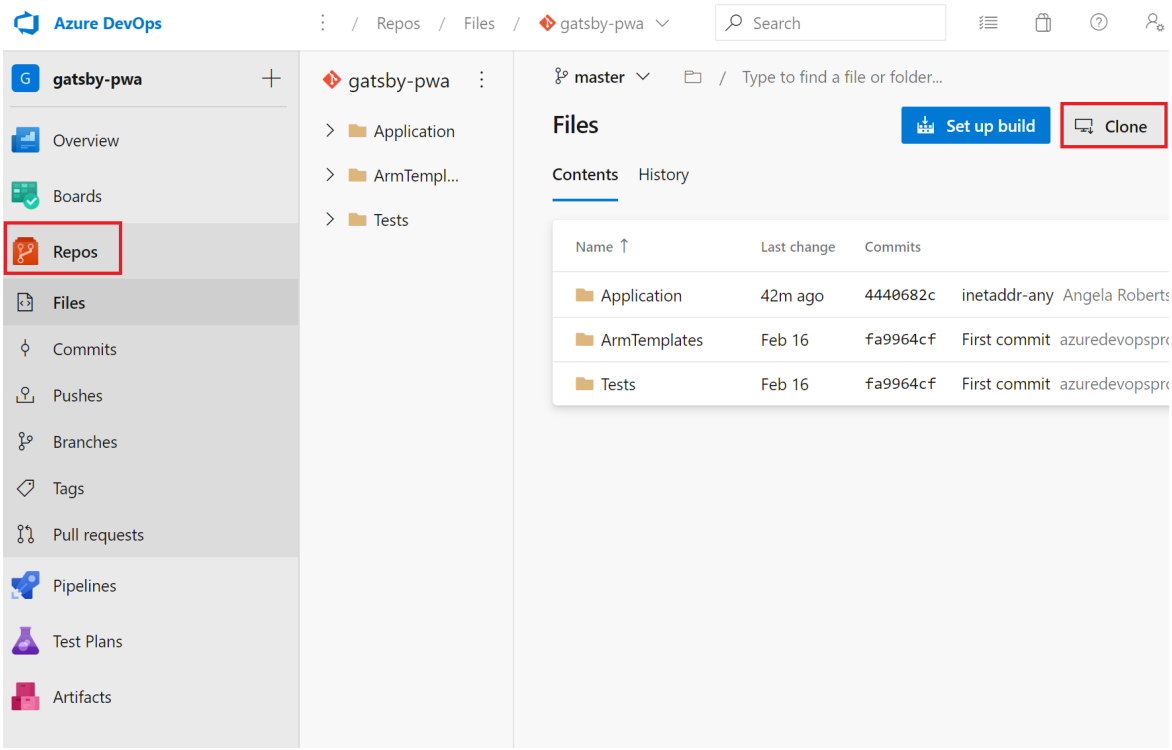
3. From your Azure DevOps project, you can invite team members to collaborate and establish a Kanban board to start tracking your work. For more information, see here.



Clone the repo and install your Gatsby PWA

DevOps Starter creates a git repository in Azure Repos or GitHub. This example has created an Azure Repo. The next step is to clone the repo and make changes.

1. Select **Repos** from your **DevOps Project** and then click **Clone**. There are various mechanisms to clone the git repo to your desktop. Choose the one that fits your development experience.



2. After the repo is cloned to your desktop, make some changes to the starter template. Start by installing the GatsbyJS CLI from your terminal.

```
npm install -g gatsby
```

3. From the terminal, navigate to the root of your repo. It should contain three folders that look like this:

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2/23/2020  10:42 PM                Application
d-----        2/23/2020   3:05 PM                ArmTemplates
d-----        2/23/2020   3:05 PM                Tests
```

4. We do not want all of the files in the Application folder because we are going to replace it with a Gatsby starter. Run the following commands, in sequence, to trim it down.

```
cp .\Application\Dockerfile .
rmdir Application
```

5. Use the Gatsby CLI to generate a sample PWA. Run `gatsby new` from the terminal to begin the PWA wizard and select `gatsby-starter-blog` for your starter template. It should resemble this sample:

```
c:\myproject> gatsby new
√ What is your project called? ... my-gatsby-project
? What starter would you like to use? » - Use arrow-keys. Return to submit.
    gatsby-starter-default
    gatsby-starter-hello-world
>   gatsby-starter-blog
    (Use a different starter)
```

6. You now have a folder named `my-gatsby-project`. Rename it to `Application` and copy the `Dockerfile` into it.

```
mv my-gatsby-project Application
mv Dockerfile Application
```

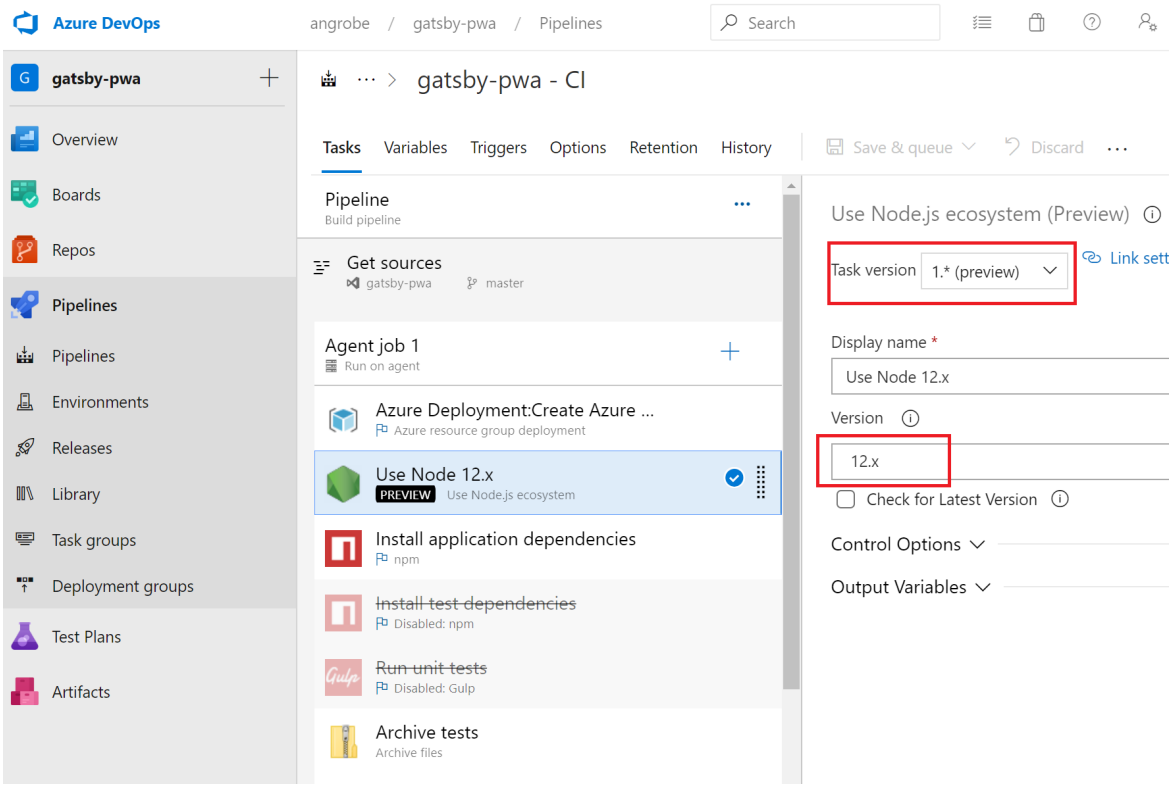
7. In your favorite editor, open the Dockerfile and change the first line from `FROM node:8` to `FROM node:12`. This change ensures that your container is using Node.js version 12.x instead of version 8.x. GatsbyJS requires more modern versions of Node.js.

8. Next, open the package.json file in the Application folder and edit the scripts field to ensure that your development and production servers listen on all available network interfaces (for example, 0.0.0.0) and port 80. Without these settings, the container app service is unable to route traffic to your Node.js app running inside your container. The `scripts` field should resemble what is below. Specifically, you want to change the `develop`, `serve`, and `start` targets from their defaults.

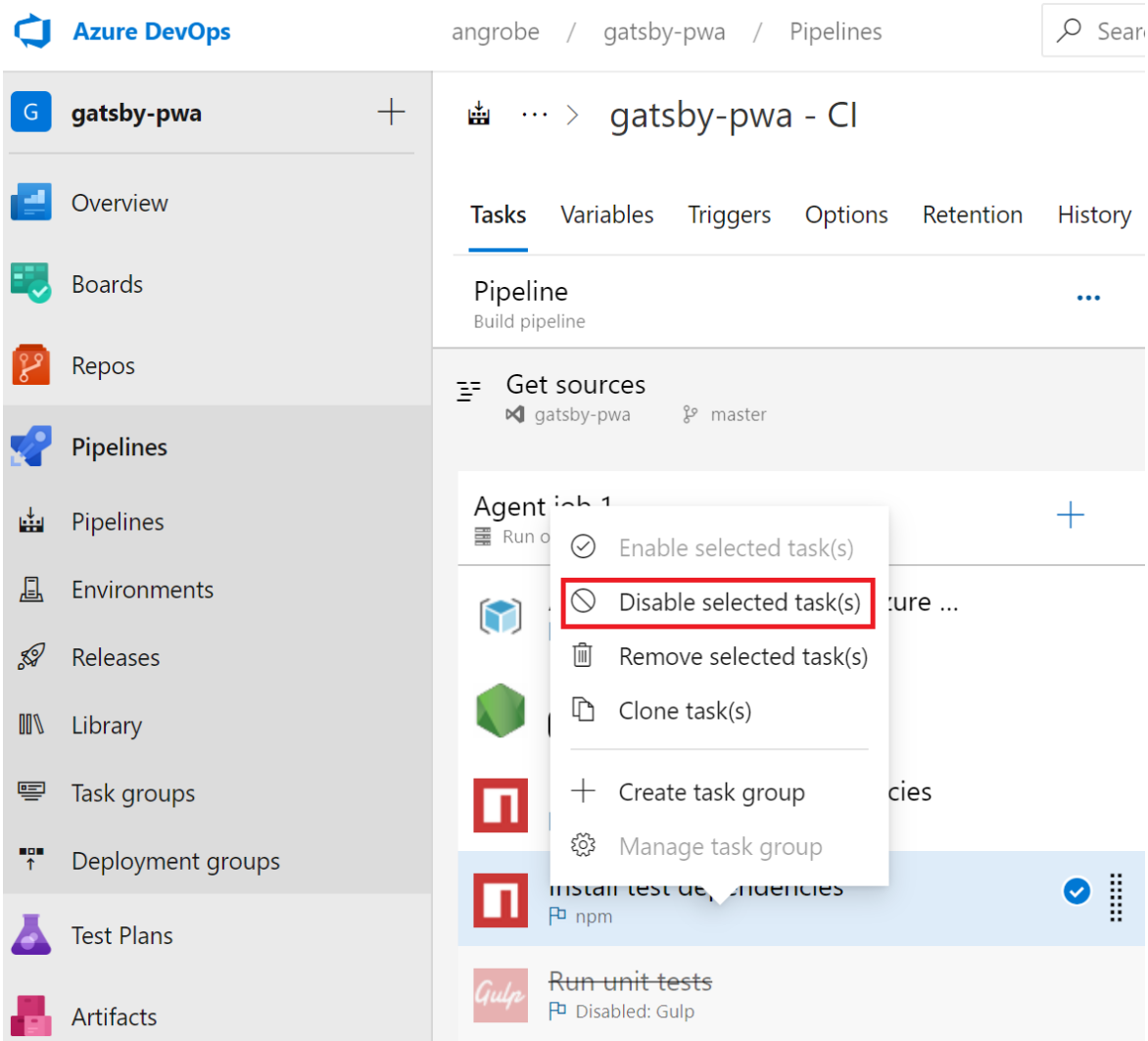
```
  "scripts": {
    "build": "gatsby build",
    "develop": "gatsby develop  -H 0.0.0.0 -p 80",
    "format": "prettier --write \"**/*.{js,jsx,json,md}\"",
    "start": "npm run serve",
    "serve": "npm run build && gatsby serve -H 0.0.0.0 -p 80",
    "clean": "gatsby clean",
    "test": "echo \"Write tests! -> https://gatsby.dev/unit-testing\" && exit 1"
  }
```

Edit Your CI/CD pipelines

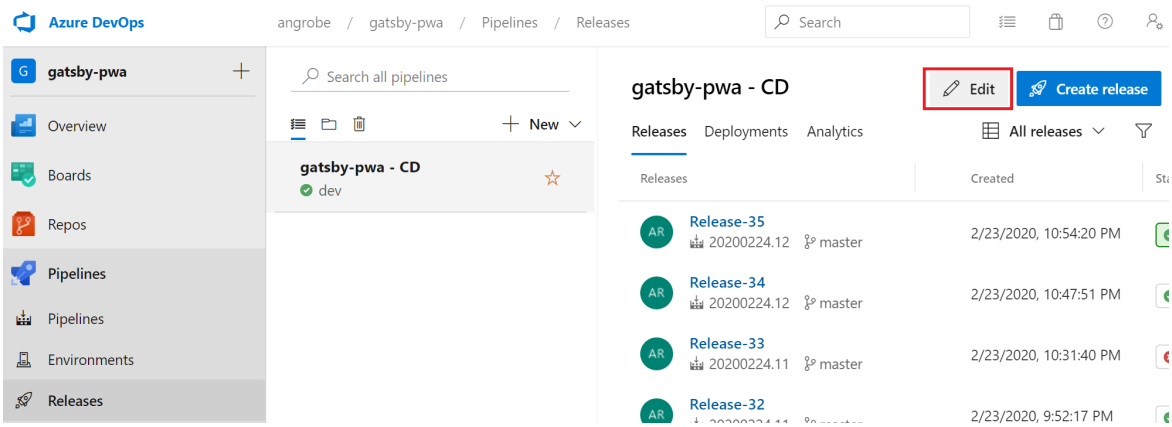
1. Before you commit the code in the previous section, make some changes to your build and release pipelines. Edit your 'Build Pipeline' and update the Node task to use Node.js version 12.x. Set the **Task version** field to 1.x and the **Version** field to 12.x.



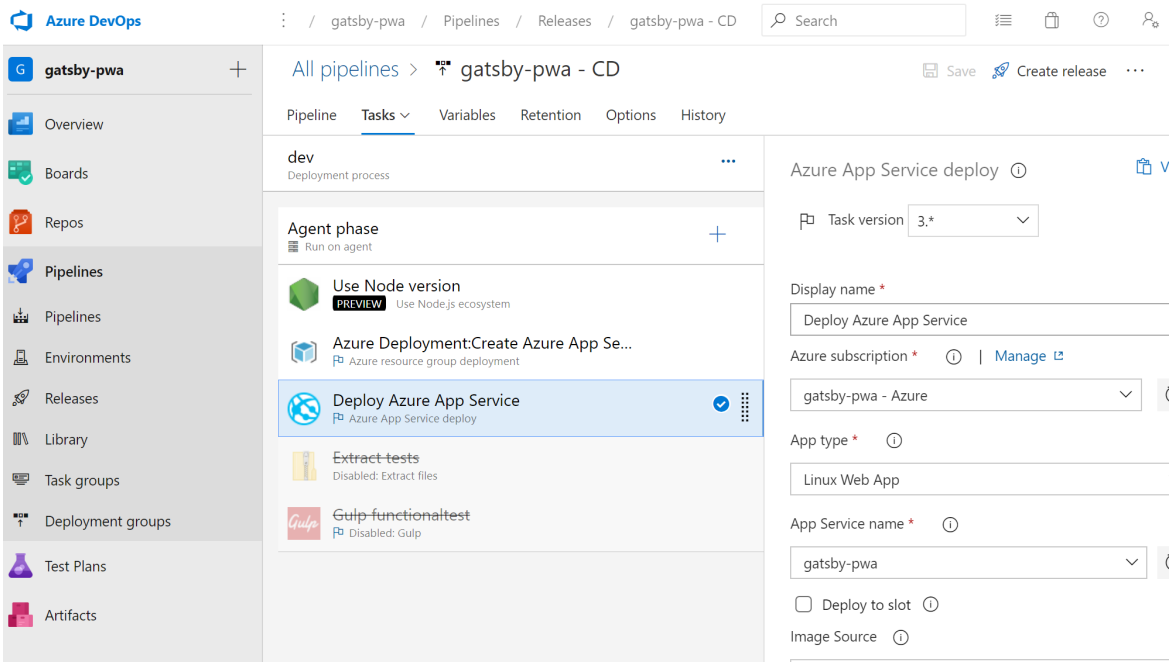
2. In this quickstart, we are not creating unit tests and we are disabling those steps in our build pipeline. When you write tests, you can re-enable these steps. Right-click to select the tasks labeled **Install test dependencies** and **Run unit tests** and disable them.



3. Edit your release pipeline.



4. As with the build pipeline, change the Node task to use 12.x and disable the two test tasks. Your release should resemble this screenshot.



5. On the left side of the browser, go to the **views/index.pug** file.

6. Select **Edit**, and then make a change to the h2 heading. For example, enter **Get started right away with Azure DevOps Starter** or make some other change.

7. Select **Commit**, and then save your changes.

8. In your browser, go to the DevOps Starter dashboard.
   You should now see a build in progress. The changes you made are automatically built and deployed through a CI/CD pipeline.

Commit your changes and examine the Azure CI/CD pipeline

In the previous two steps, you added a Gatsby generated PWA to your git repo and edited your pipelines to build and deploy the code. We can commit the code, and watch it progress through the build and release pipeline.

1. From the root of your project's git repo in a terminal, run the following commands to push your code to your Azure DevOps project:

```
git add .
git commit -m "My first Gatsby PWA"
git push
```

2. A build is started as soon as `git push` completes. You can follow the progress from the **Azure DevOps Dashboard**.

3. After a few minutes, your build and release pipelines should finish and your PWA should be deployed to a container. Click the **Application endpoint** link from the dashboard above and you should see a Gatsby starter project for blogs.

Clean up resources

You can delete Azure App Service and other related resources that you created when you don't need the resources anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

When you configure your CI/CD process, build and release pipelines are automatically created. You can change these build and release pipelines to meet the needs of your team. To learn more about the CI/CD pipeline, see:

Customize CD process

Set up a CI/CD pipeline for a Java app with Azure DevOps Starter

6/28/2022 • 5 minutes to read • Edit Online

In this quickstart, you use the simplified Azure DevOps Starter experience to set up a continuous integration (CI) and continuous delivery (CD) pipeline for your Java app in Azure Pipelines. You can use Azure DevOps Starter to set up everything you need for developing, deploying, and monitoring your app.

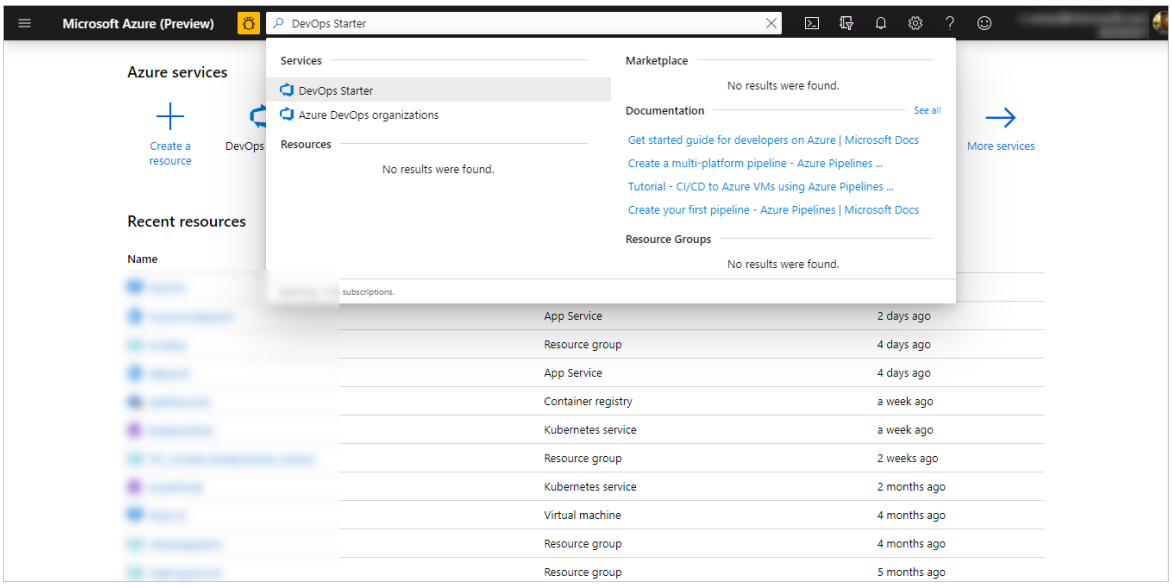
Prerequisites

- An Azure account with an active subscription. Create an account for free.
- An Azure DevOps account and organization.

Sign in to the Azure portal

DevOps Starter creates a CI/CD pipeline in Azure Pipelines. You can create a new Azure DevOps organization or use an existing organization. DevOps Starter also creates Azure resources in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



Select a sample application and Azure service

1. Select the Java sample application. The Java samples include a choice of several application frameworks.
2. The default sample framework is Spring. Leave the default setting, and then select **Next**. Web App For Containers is the default deployment target. The application framework, which you chose previously, dictates the type of Azure service deployment target available here.
3. Leave the default service, and then select **Next**.

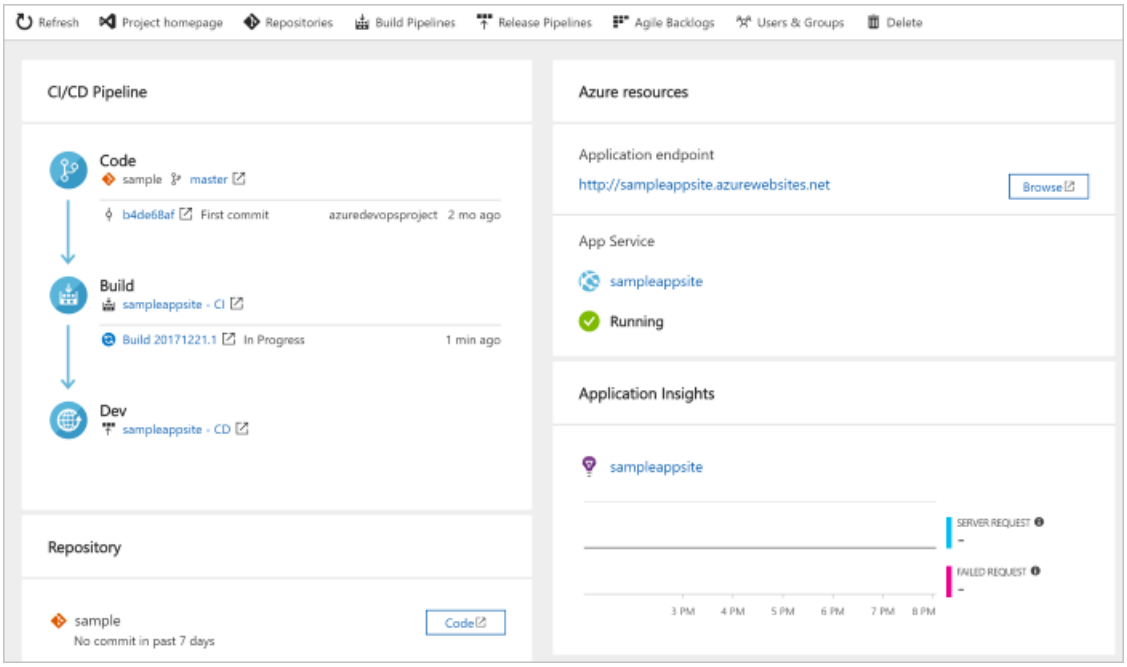
Configure Azure DevOps and an Azure subscription

1. Create a new Azure DevOps organization or choose an existing organization.

   a. Choose a name for your project.

   b. Select your Azure subscription and location, choose a name for your application, and then select **Done**.
   After a few minutes, the DevOps Starter dashboard is displayed in the Azure portal. A sample application is set up in a repository in your Azure DevOps organization, a build is executed, and your application is deployed to Azure. This dashboard provides visibility into your code repository, the CI/CD pipeline, and your application in Azure.

2. Select **Browse** to view your running application.



DevOps Starter automatically configured a CI build and release trigger. You're now ready to collaborate with a team on a Java app with a CI/CD process that automatically deploys your latest work to your web site.

Commit code changes and execute CI/CD

DevOps Starter creates a Git repository in Azure Repos or GitHub. To view the repository and make code changes to your application, do the following:

1. On the left of the DevOps Starter dashboard, select the link for your main branch. This link opens a view to the newly created Git repository.

2. To view the repository clone URL, select **Clone** on the top right of the browser. You can clone your Git repository in your favorite IDE. In the next few steps, you can use the web browser to make and commit code changes directly to the main branch.

3. On the left side of the browser, go to the **src/main/webapp/index.html** file.

4. Select **Edit**, and then make a change to some of the text. For example, change some of the text for one of the div tags.

5. Select **Commit**, and then save your changes.

6. In your browser, go to the DevOps Starter dashboard.
   You should now see a build in progress. The changes you just made are automatically built and deployed via a CI/CD pipeline.

Examine the CI/CD pipeline

In the previous step, DevOps Starter automatically configured a full CI/CD pipeline. Explore and customize the pipeline as needed. Take the following steps to familiarize yourself with the build and release pipelines.

1. At the top of the DevOps Starter dashboard, select **Build Pipelines**. This link opens a browser tab and the build pipeline for your new project.
2. Point to the **Status** field, and then select the ellipsis (...). This action opens a menu where you can start several activities such as queuing a new build, pausing a build, and editing the build pipeline.
3. Select **Edit**.
4. In this pane, you can examine the various tasks for your build pipeline. The build performs a variety of tasks such as fetching sources from the Git repository, restoring dependencies, and publishing outputs that are used for deployments.
5. At the top of the build pipeline, select the build pipeline name.
6. Change the name of your build pipeline to something more descriptive, select **Save & queue**, and then select **Save**.
7. Under your build pipeline name, select **History**.
   In the **History** pane, you see an audit trail of your recent changes for the build. Azure Pipelines keeps track of any changes that are made to the build pipeline, and it allows you to compare versions.
8. Select **Triggers**. DevOps Starter automatically created a CI trigger, and every commit to the repository starts a new build. You can optionally choose to include or exclude branches from the CI process.
9. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.
10. Select **Build and Release**, and then select **Releases**.
    DevOps Starter creates a release pipeline to manage deployments to Azure.
11. On the left, select the ellipsis (...) next to your release pipeline, and then select **Edit**. The release pipeline contains a pipeline, which defines the release process.
12. Under **Artifacts**, select **Drop**. The build pipeline you examined in the previous steps produces the output that's used for the artifact.
13. Next to the **Drop** icon, select the **Continuous deployment trigger**. This release pipeline has an enabled CD trigger, which runs a deployment every time there is a new build artifact available. Optionally, you can disable the trigger so that your deployments require manual execution.
14. On the left, select **Tasks**. The tasks are the activities that your deployment process performs. In this example, a task was created to deploy to Azure App Service.
15. On the right, select **View releases**. This view shows a history of releases.
16. Select the ellipsis (...) next to one of your releases, and then select **Open**. There are several menus to explore, such as a release summary, associated work items, and tests.
17. Select **Commits**. This view shows code commits that are associated with the specific deployment.
18. Select **Logs**. The logs contain useful information about the deployment process. They can be viewed both during and after deployments.

Clean up resources

You can delete Azure App Service and other related resources when you don't need them anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

When you configured your CI/CD process, build and release pipelines were automatically created. You can modify these build and release pipelines to meet the needs of your team. To learn more about the CI/CD pipeline, see:

Customize CD process

Create a CI/CD pipeline for Python with Azure DevOps Starter

6/28/2022 • 5 minutes to read • Edit Online

In this quickstart, you use the simplified Azure DevOps Starter experience to set up a continuous integration (CI) and continuous delivery (CD) pipeline for your Python app in Azure Pipelines. You can use Azure DevOps Starter to set up everything you need for developing, deploying, and monitoring your app.

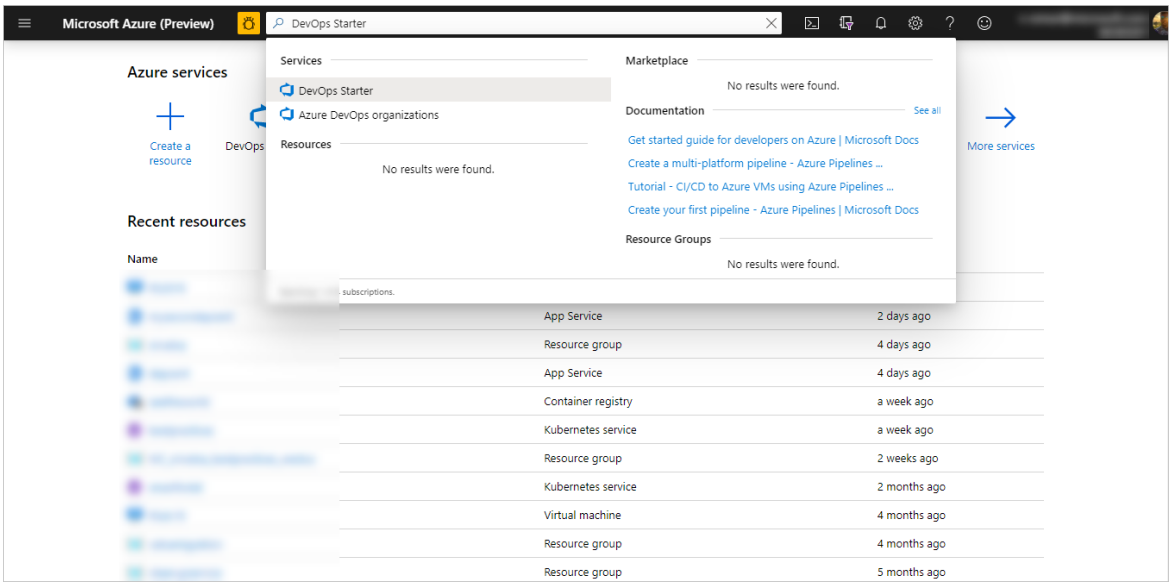
Prerequisites

- An Azure account with an active subscription. Create an account for free.
- An Azure DevOps account and organization.

Sign in to the Azure portal

DevOps Starter creates a CI/CD pipeline in Azure Pipelines. You can create a new Azure DevOps organization or use an existing organization. DevOps Starter also creates Azure resources in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



Select a sample application and Azure service

1. Select the Python sample application. The Python samples include a choice of several application frameworks.
2. The default sample framework is Django. Leave the default setting, and then select **Next**. Web App For Containers is the default deployment target. The application framework, which you chose previously, dictates the type of Azure service deployment target available here.
3. Leave the default service, and then select **Next**.

Configure Azure DevOps and an Azure subscription

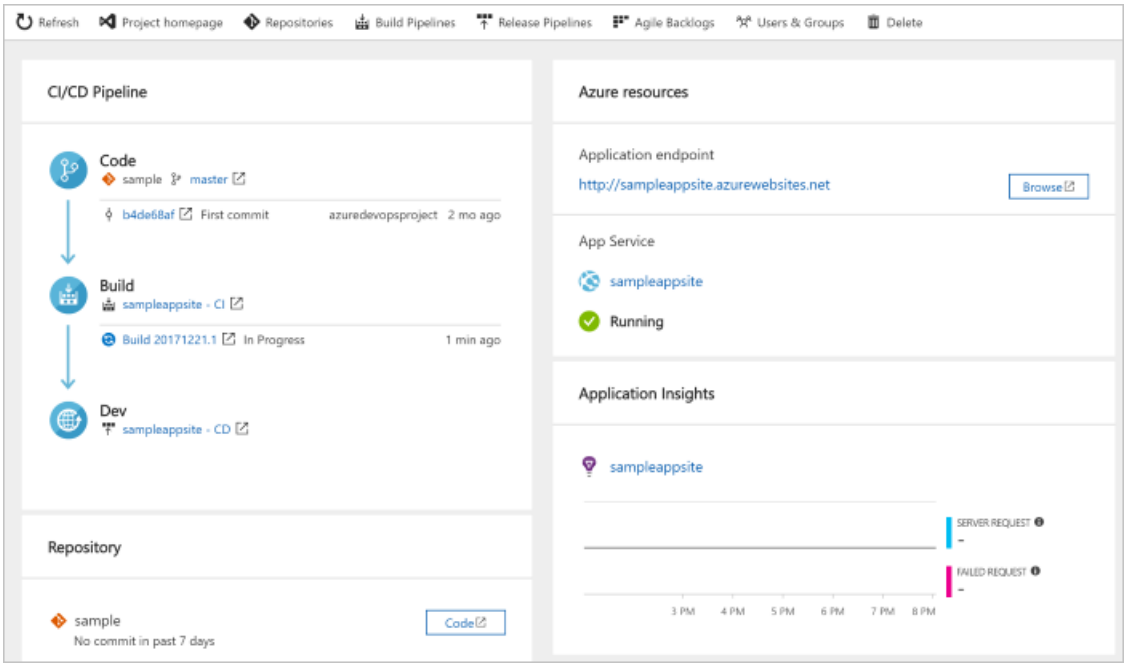
1. Create a new Azure DevOps organization or choose an existing organization.

   a. Enter a name for your project in Azure DevOps.

   b. Select your Azure subscription and location, enter a name for your application, and then select **Done**.

   After few minutes, the Starter dashboard is displayed in the Azure portal. A sample application is set up in a repository in your Azure DevOps organization, a build is executed, and your application is deployed to Azure. This dashboard provides visibility into your code repository, your CI/CD pipeline, and your application in Azure.

2. Select **Browse** to view your running application.



   DevOps Projects automatically configures a CI build and release trigger. You're now ready to collaborate with a team on a Python app by using a CI/CD process that automatically deploys your latest work to your website.

Commit code changes and execute CI/CD

DevOps Starter creates a Git repository in Azure Repos or GitHub. To view the repository and make code changes to your application, do the following:

1. On the left side of the DevOps Starter dashboard, select the link for your main branch. This link opens a view to the newly created Git repository.

2. To view the repository clone URL, select **Clone** from the top right of the browser. You can clone your Git repository in your favorite IDE. In the next few steps, you can use the web browser to make and commit code changes directly to the main branch.

3. On the left, go to the **app/templates/app/index.html** file.

4. Select **Edit**, and make a change to some of the text. For example, change some of the text for one of the div tags.

5. Select **Commit**, and then save your changes.

6. In your browser, go to the DevOps Starter dashboard. You should now see a build in progress. The

changes you just made are automatically built and deployed via a CI/CD pipeline.

Examine the CI/CD pipeline

In the previous step, DevOps Starter automatically configured a full CI/CD pipeline. Explore and customize the pipeline as needed. To familiarize yourself with the build and release pipelines, do the following:

1. At the top of the DevOps Starter dashboard, select **Build Pipelines**. A browser tab displays the build pipeline for your new project.
2. 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.
3. Select **Edit**.
4. 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 repository, restoring dependencies, and publishing outputs for deployments.
5. At the top of the build pipeline, select the build pipeline name.
6. Change the name of your build pipeline to something more descriptive, select **Save & queue**, and then select **Save**.
7. Under your build pipeline name, select **History**. You see 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.
8. Select **Triggers**. DevOps Starter automatically creates a CI trigger, and every commit to the repository starts a new build. You can optionally choose to include or exclude branches from the CI process.
9. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.
10. Select **Build and Release**, and then choose **Releases**.
    DevOps Projects creates a release pipeline to manage deployments to Azure.
11. Select the ellipsis next to your release pipeline, and then select **Edit**. The release pipeline defines the release process.
12. Under **Artifacts**, select **Drop**. The build pipeline you examined in the previous steps produces the output that's used for the artifact.
13. Next to the **Drop** icon, select the **Continuous deployment trigger**. The release pipeline has an enabled CD trigger, which runs a deployment every time there's a new build artifact available. Optionally, you can disable the trigger so that your deployments require manual execution.
14. On the left, select **Tasks**. The tasks are the activities that your deployment process performs. In this example, a task was created to deploy to Azure App Service.
15. On the right, select **View releases** to display a history of releases.
16. Select the ellipsis (...) next to one of your releases, and then select **Open**. There are several menus to explore from this view, such as a release summary, associated work items, and tests.
17. Select **Commits**. This view shows the code commits that are associated with the specific deployment.
18. Select **Logs**. The logs contain useful information about the deployment process. You can view them both during and after deployments.

Clean up resources

You can delete Azure App Service and related resources when you don't need them anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

When you configured your CI/CD process, build and release pipelines were automatically created. You can modify these build and release pipelines to meet the needs of your team. To learn more about the CI/CD pipeline, see:

Customize CD process

Create a CI/CD pipeline for PHP with Azure DevOps Starter

6/28/2022 • 5 minutes to read • Edit Online

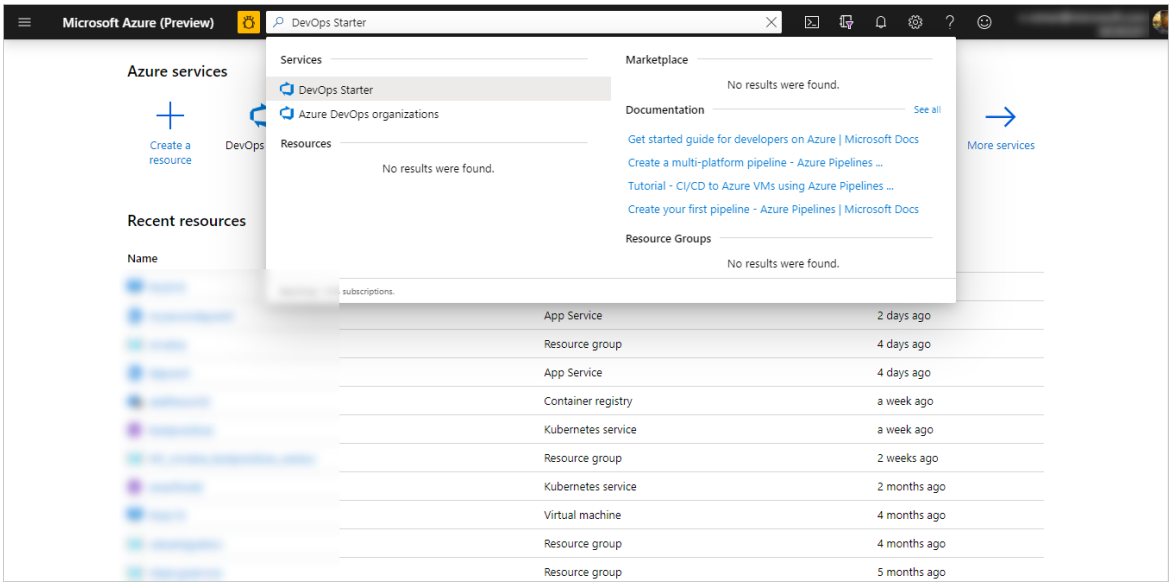
Azure DevOps Starter presents a simplified experience that creates Azure resources and sets up a continuous integration (CI) and continuous delivery (CD) pipeline for your PHP app in Azure Pipelines.

If you don't have an Azure subscription, you can get one for free through Visual Studio Dev Essentials.

Sign in to the Azure portal

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

1. Sign in to the Microsoft Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



Select a sample application and Azure service

1. Select the PHP sample application. The PHP samples include a choice of several application frameworks. The default sample framework is Laravel.
2. Leave the default setting, and then select **Next**.
3. Web App For Containers is the default deployment target. The application framework, which you chose previously, dictates the type of Azure service deployment target that's available here. Leave the default service, and then select **Next**.

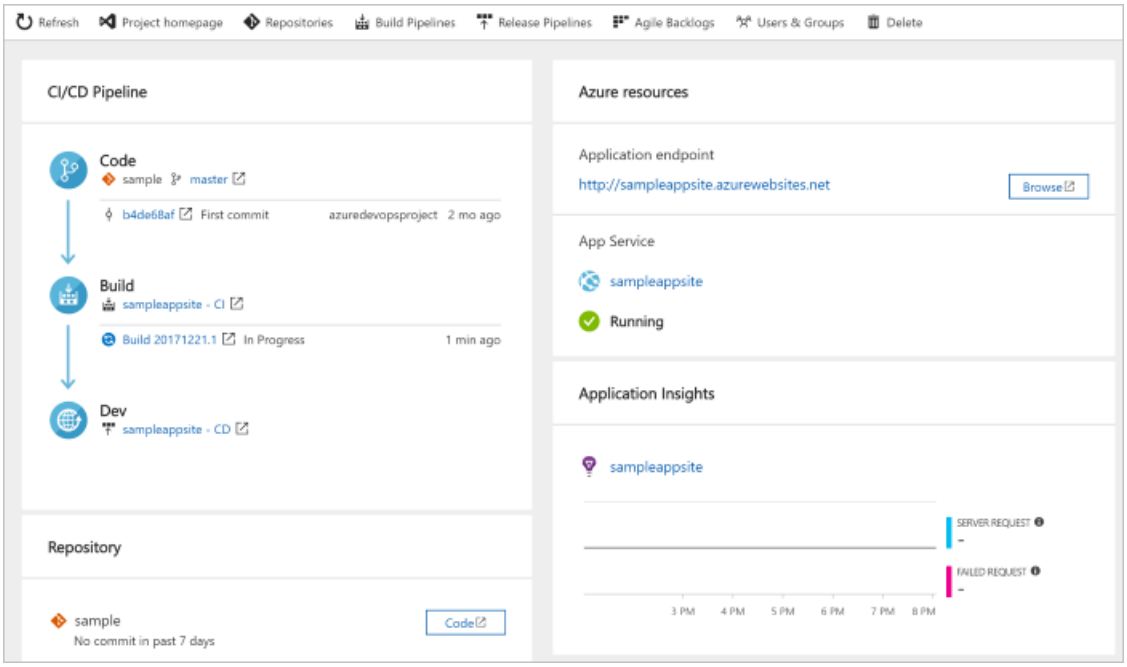
Configure Azure DevOps and an Azure subscription

1. Create a new Azure DevOps organization or select an existing organization.
    a. Choose a name for your project in Azure DevOps.

b. Select your Azure subscription and location, enter a name for your application, and then select **Done**.

After few minutes, the DevOps Starter dashboard is displayed in the Azure portal. A sample application is set up in a repository in your Azure DevOps organization, a build runs, and your application deploys to Azure. This dashboard provides visibility into your code repository, your CI/CD pipeline, and your application in Azure.

2. Select **Browse** to view your running application.



DevOps Starter automatically configured a CI build and release trigger. You're now ready to collaborate with a team on a PHP app with a CI/CD process that automatically deploys your latest work to your web site.

Commit code changes and execute CI/CD

DevOps Starter creates a Git repository in Azure Repos or GitHub. To view the repository and make code changes to your application, take the following steps:

1. On the left of the DevOps Starter dashboard, select the link for your main branch. This link opens a view to the newly created Git repository.
2. To view the repository clone URL, select **Clone** from the top right of the browser. You can clone your Git repository in your favorite IDE. In the next few steps, use the web browser to make and commit code changes directly to the main branch.
3. On the left, go to the **resources/views/welcome.blade.php** file.
4. Select **Edit**, and then make a change to some of the text. For example, change some of the text for one of the div tags.
5. Select **Commit**, and then save your changes.
6. In your browser, go to the DevOps Starter dashboard. You should now see a build in progress. The changes you just made are automatically built and deployed via a CI/CD pipeline.

Examine the CI/CD pipeline

DevOps Starter automatically configures a full CI/CD pipeline in Azure Pipelines. Explore and customize the

pipeline as needed. To familiarize yourself with the build and release pipelines, do the following:

1. At the top of the DevOps Starter dashboard, select **Build Pipelines**. This link opens a browser tab and the build pipeline for your new project.
2. Point to the **Status** field, and then select the **ellipsis** (...). A menu displays several options, such as queuing a new build, pausing a build, and editing the build pipeline.
3. Select **Edit**.
4. In this pane, you can examine the various tasks for your build pipeline. The build runs a variety of tasks, such as fetching sources from the Git repository, restoring dependencies, and publishing outputs that are used for deployments.
5. At the top of the build pipeline, select the build pipeline name.
6. Change the name of your build pipeline to something more descriptive, select, **Save & queue**, and then select **Save**.
7. Under your build pipeline name, select **History**. The **History** pane displays an audit trail of your recent changes for the build. Azure Pipelines keeps track of any changes that are made to the build pipeline, and it allows you to compare versions.
8. Select **Triggers**. DevOps Starter automatically created a CI trigger, and every commit to the repository starts a new build. You can optionally choose to include or exclude branches from the CI process.
9. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.
10. Select **Build and Release**, and then select **Releases**. DevOps Starter creates a release pipeline to manage deployments to Azure.
11. Select the ellipsis (...) next to your release pipeline, and then select **Edit**. The release pipeline contains a pipeline, which defines the release process.
12. Under **Artifacts**, select **Drop**. The build pipeline you examined in the previous steps produces the output that's used for the artifact.
13. Next to the **Drop** icon, select the **Continuous deployment trigger**. This release pipeline has an enabled CD trigger, which runs a deployment every time there's a new build artifact available. Optionally, you can disable the trigger so that your deployments require manual execution.
14. On the left, select **Tasks**. The tasks are the activities that your deployment process performs. In this example, a task was created to deploy to Azure App Service.
15. On the right, select **View releases** to display a history of releases.
16. Select the ellipsis (...) next to one of your releases, and then select **Open**. There are several menus to explore from this view such as a release summary, associated work items, and tests.
17. Select **Commits**. This view shows code commits that are associated with the specific deployment.
18. Select **Logs**. The logs contain useful information about the deployment process. They can be viewed both during and after deployments.

Clean up resources

You can delete Azure App Service and other related resources when you don't need them anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

When you configured your CI/CD process, build and release pipelines were automatically created. You can modify these build and release pipelines to meet the needs of your team. To learn more about the CI/CD pipeline, see this tutorial:

Customize CD process

Create a CI/CD pipeline for Ruby on Rails by using Azure DevOps Starter

6/28/2022 • 4 minutes to read • Edit Online

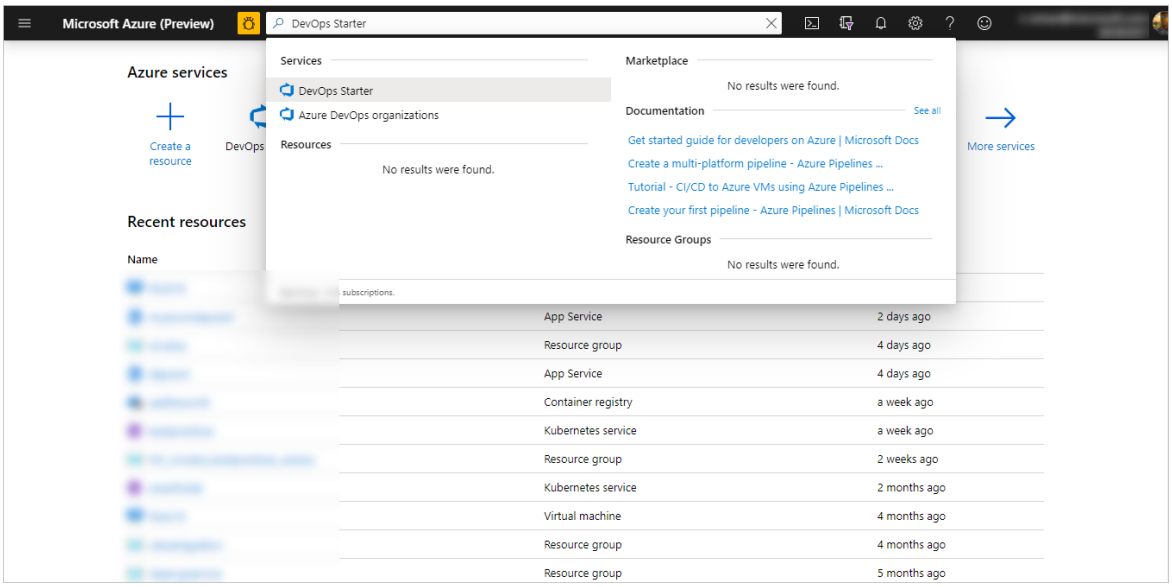
Configure continuous integration (CI) and continuous delivery (CD) for your Ruby on Rails app by using Azure DevOps Starter. DevOps Starter simplifies the initial configuration of an Azure DevOps build and release pipeline.

If you don't have an Azure subscription, you can get one free through Visual Studio Dev Essentials.

Sign in to the Azure portal

Azure DevOps Starter creates a CI/CD pipeline in Azure Repos. You can create a new Azure DevOps organization or use an existing organization. DevOps Starter also creates Azure resources in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



Select a sample app and Azure service

1. Select the **Ruby** sample app.
2. Select the **Ruby on Rails** application framework. When you're done, select **Next**.
3. **Web App on Linux** is the default deployment target. Optionally, you can select **Web App for Containers**. The application framework, which you chose previously, dictates the type of Azure service deployment target that's available here.
4. Select the target service of your choice, and then select **Next**.

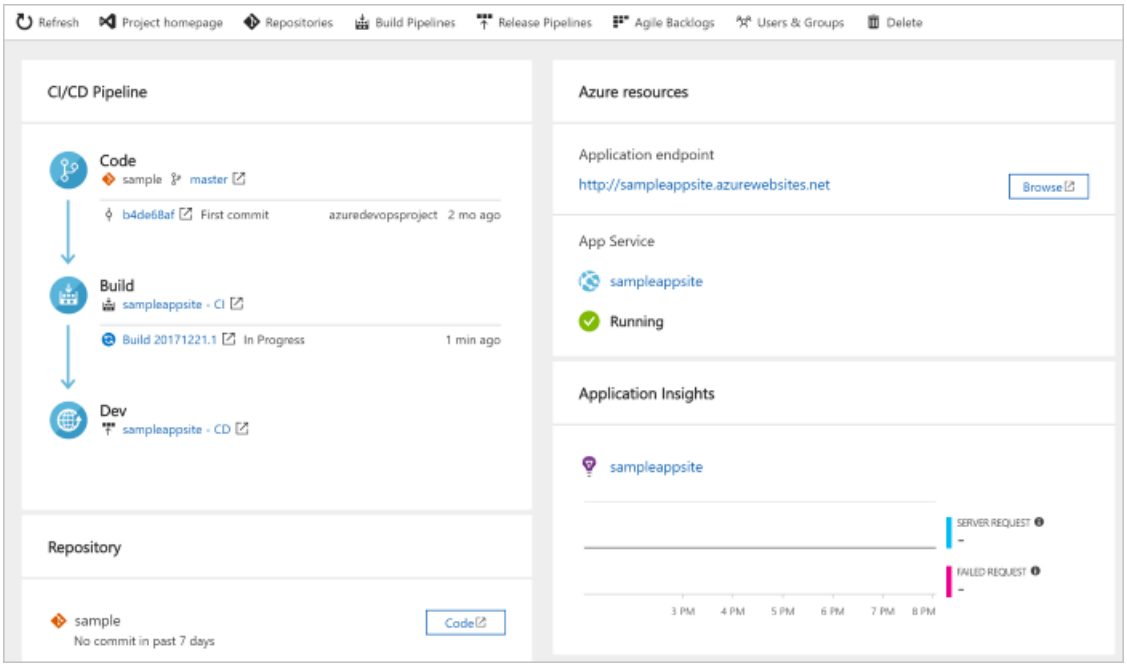
Configure Azure DevOps and an Azure subscription

1. Create a new free Azure DevOps organization or choose an existing organization.

2. Enter a name for your Azure DevOps project.

3. Select your Azure subscription and location, enter a name for your app, and then select **Done**.
   After a few minutes, the DevOps Starter dashboard is displayed in the Azure portal. A sample app is set up in a repo in your Azure DevOps organization, a build is executed, and your app is deployed to Azure.

   The dashboard provides visibility into your code repo, your CI/CD pipeline, and your app in Azure. At the right, select **Browse** to view your running app.



Commit your code changes and execute the CI/CD

Azure DevOps Starter creates a Git repo in Azure Pipelines or GitHub. To view the repo and make code changes to your app, do the following:

1. On the DevOps Starter dashboard, at the left, select the link for your main branch. The link opens a view to the newly created Git repo.

2. To view the repo clone URL, select **Clone** at the top right. You can clone your Git repo in your favorite IDE. In the next few steps, you can use the web browser to make and commit code changes directly to the main branch.

3. At the left, go to the *app/views/pages/home.html.erb* file, and then select **Edit**.

4. Make a change to the file. For example, modify some text within one of the div tags.

5. Select **Commit**, and then save your changes.

6. In your browser, go to the DevOps Starter dashboard. A build should be in progress. The changes you made are automatically built and deployed via a CI/CD pipeline.

Examine the Azure Pipelines CI/CD pipeline

Azure DevOps Starter automatically configures a full CI/CD pipeline in your Azure DevOps organization. Explore and customize the pipeline as needed. To familiarize yourself with the Azure DevOps build and release pipelines, do the following:

1. Go to the DevOps Starter dashboard.

2. At the top, 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 Starter 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.
11. Select **Build and Release**, and then select **Releases**. DevOps Starter creates a release pipeline to manage deployments to Azure.
12. Select the ellipsis (...) next to your release pipeline, and then select **Edit**. The release pipeline contains a *pipeline*, which defines the release process.
13. Under **Artifacts**, select **Drop**. The build pipeline you examined previously produces the output that's used for the artifact.
14. 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.
15. At the left, select **Tasks**. Tasks are the activities your deployment process performs. In this example, a task was created to deploy to Azure App Service.
16. At the right, select **View releases** to display a history of releases.
17. 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.
18. Select **Commits**. This view shows code commits that are associated with this deployment.
19. Select **Logs**. The logs contain useful information about the deployment process. You can view them both during and after deployments.

Clean up resources

When they are no longer needed, you can delete the Azure App Service instance and related resources that you created in this quickstart. To do so, use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

To learn more about modifying the build and release pipelines to meet the needs of your team, see:

Define your multi-stage continuous deployment (CD) pipeline

Create a CI/CD pipeline for Go using Azure DevOps Starter

6/28/2022 • 4 minutes to read • Edit Online

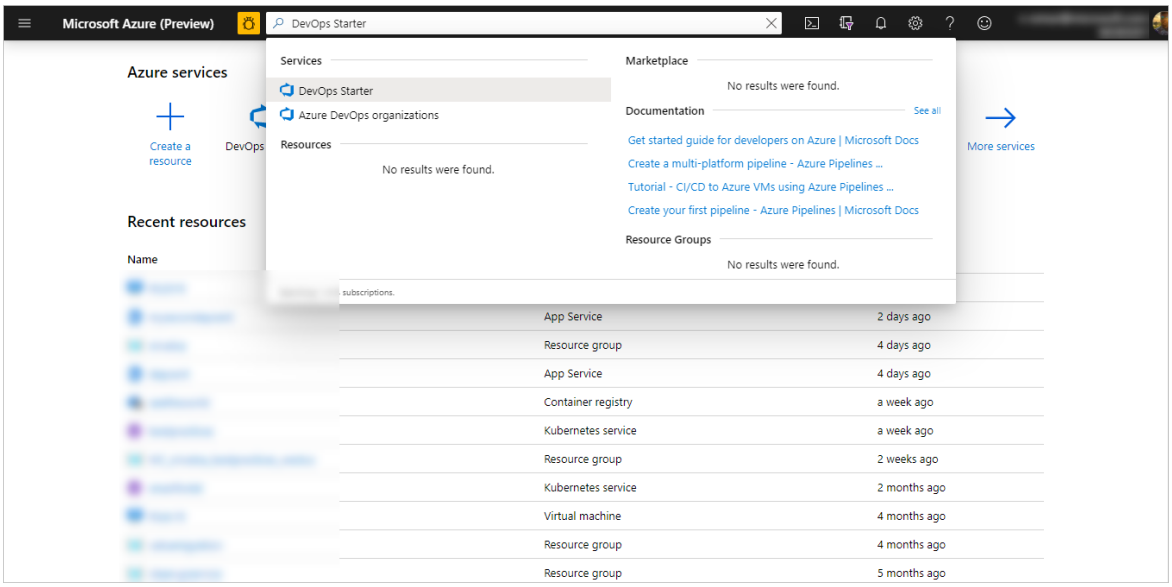
Configure continuous integration (CI) and continuous delivery (CD) for your Go app by using Azure DevOps Starter. DevOps Starter simplifies the initial configuration of an Azure DevOps build and release pipeline.

If you don't have an Azure subscription, you can get one free through Visual Studio Dev Essentials.

Sign in to the Azure portal

DevOps Starter creates a CI/CD pipeline in Azure Pipelines. You can create a new Azure DevOps organization or use an existing organization. DevOps Starter also creates Azure resources in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



Select a sample app and Azure service

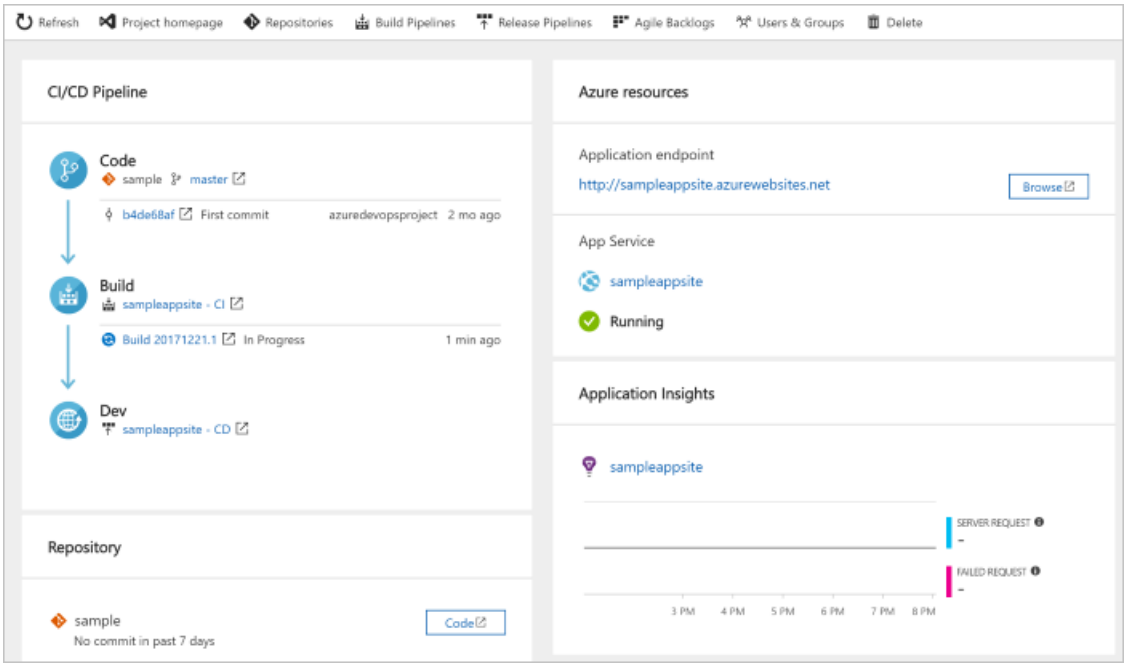
1. Select the **Go** sample app, and then select **Next**.
2. **Simple Go app** is the default framework. Select **Next**. The app framework, which you chose previously, dictates the type of Azure service deployment targets that are available for deployment.
3. Leave the default Azure service and select **Next**.

Configure Azure DevOps and an Azure subscription

1. Create a new free Azure DevOps organization or choose an existing organization.
2. Enter a name for your Azure DevOps project.
3. Select your Azure subscription and location, enter a name for your app, and then select **Done**. After a few minutes, the DevOps Starter dashboard is displayed in the Azure portal. A sample app is set up in a repo

in your Azure DevOps organization, a build is executed, and your app is deployed to Azure.

The dashboard provides visibility into your code repo, your CI/CD pipeline, and your app in Azure. At the right, select **Browse** to view your running app.



Commit your code changes and execute the CI/CD

DevOps Starter creates a Git repo in Azure Repos or GitHub. To view the repo and make code changes to your app, do the following:

1. On the DevOps Starter, at the left, select the link for your main branch. The link opens a view to the newly created Git repo.
2. To view the repo clone URL, select **Clone** at the top right. You can clone your Git repo in your favorite IDE. In the next few steps, you can use the web browser to make and commit code changes directly to the main branch.
3. At the left, go to the *views/index.html* file, and then select **Edit**.
4. Make a change to the file. For example, modify some text within one of the div tags.
5. Select **Commit**, and then save your changes.
6. In your browser, go to the DevOps Projects dashboard. A build should be in progress. The changes you made are automatically built and deployed via a CI/CD pipeline.

Examine the CI/CD pipeline

DevOps Starter automatically configures a full CI/CD pipeline in Azure Repos. Explore and customize the pipeline as needed. To familiarize yourself with the Azure DevOps build and release pipelines, do the following:

1. Go to the DevOps Starter dashboard.
2. At the top, 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 Starter 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.
11. Select **Build and Release**, and then select **Releases**. DevOps Starter creates a release pipeline to manage deployments to Azure.
12. Select the ellipsis (...) next to your release pipeline, and then select **Edit**. The release pipeline contains a *pipeline*, which defines the release process.
13. Under **Artifacts**, select **Drop**. The build pipeline you examined previously produces the output that's used for the artifact.
14. 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.
15. At the left, select **Tasks**. Tasks are the activities your deployment process performs. In this example, a task was created to deploy to Azure App Service.
16. At the right, select **View releases** to display a history of releases.
17. 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.
18. Select **Commits**. This view shows code commits that are associated with this deployment.
19. Select **Logs**. The logs contain useful information about the deployment process. You can view them both during and after deployments.

Clean up resources

When they are no longer needed, you can delete the Azure App Service instance and related resources that you created in this quickstart. To do so, use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

To learn more about modifying the build and release pipelines to meet the needs of your team, see:

Define your multi-stage continuous deployment (CD) pipeline

Tutorial: Deploy Node.js app to Azure Web App using DevOps Starter for GitHub Actions

6/28/2022 • 6 minutes to read • Edit Online

DevOps Starter for GitHub Actions presents a simplified experience where you can choose a sample application to create a continuous integration (CI) and continuous delivery (CD) workflow to deploy to Azure.

DevOps Starter also:

- Automatically creates Azure resources, such as a new Azure Web App.
- Creates and configures a workflow in GitHub that includes a build job for CI.
- The workflow also contains a deploy job for CD.
- Creates an Azure Application Insights resource for monitoring.

In this tutorial, you will:

- Use DevOps Starter to deploy a Node.js app
- Configure GitHub and an Azure subscription
- Examine the GitHub workflow
- Commit changes to GitHub and automatically deploy them to Azure
- Configure Azure Application Insights monitoring
- Clean up resources

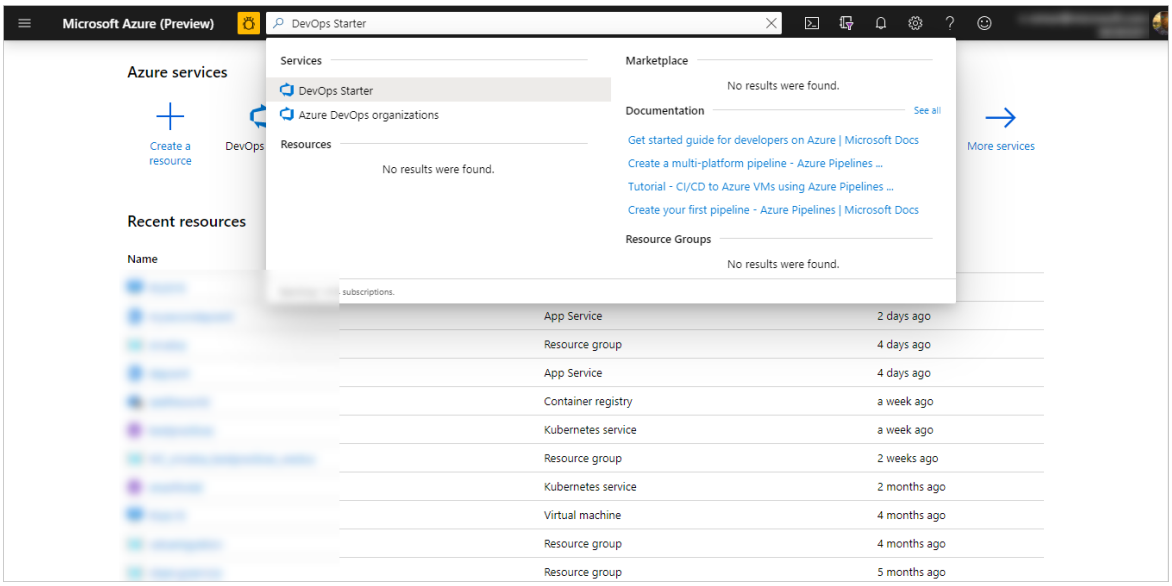
Prerequisites

- An Azure account with an active subscription. Create an account for free.

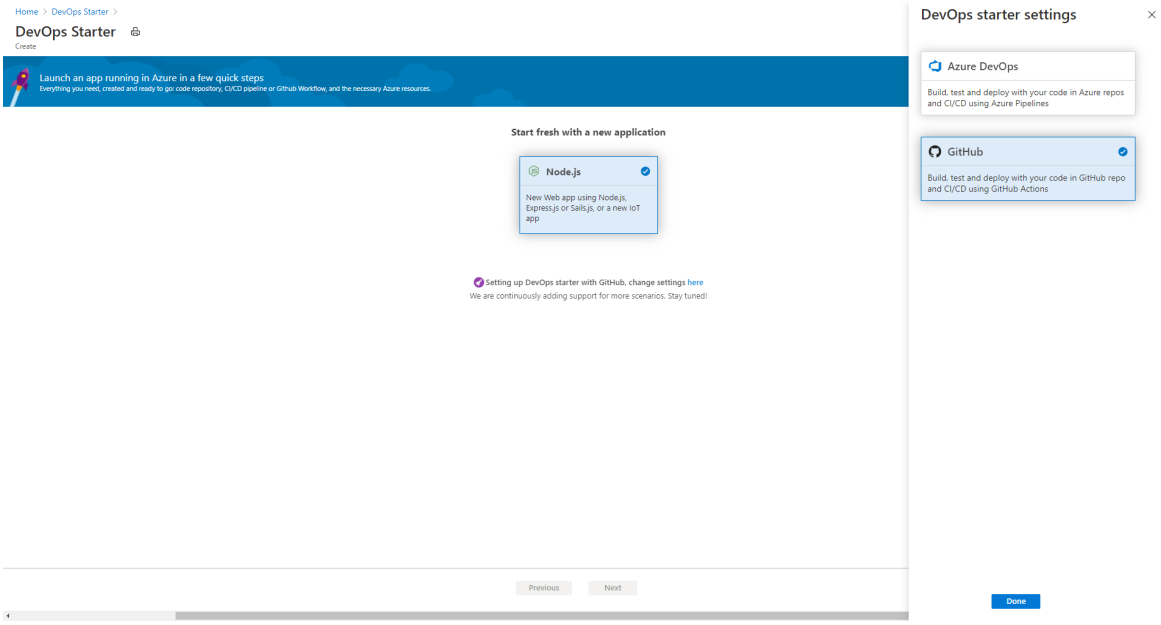
Use DevOps Starter to deploy a Node.js app

DevOps Starter creates a workflow in GitHub. You can use an existing GitHub organization. DevOps Starter also creates Azure resources such as Web App in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



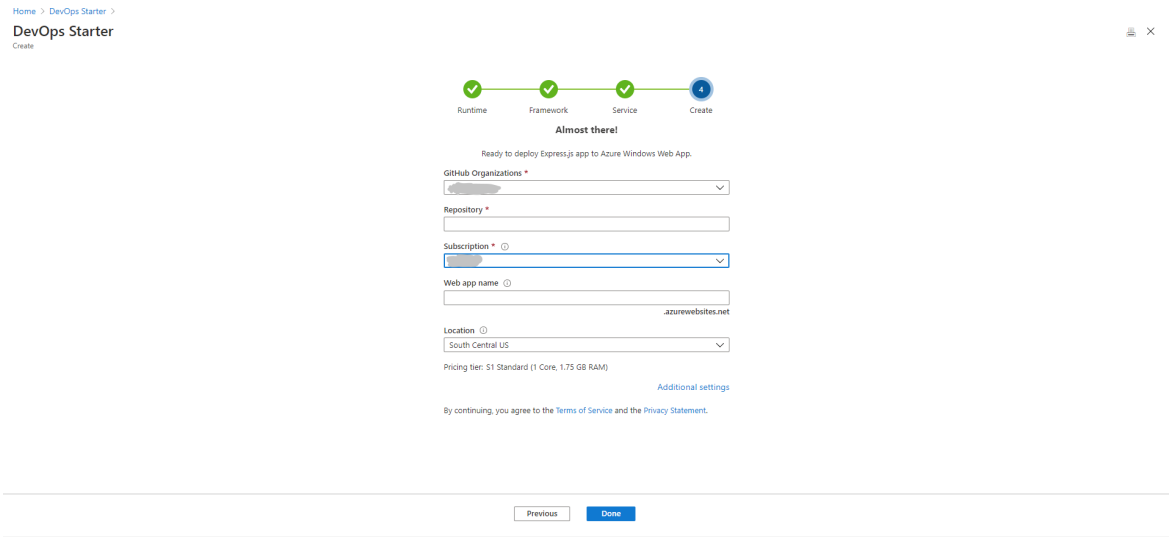
3. Ensure that the CI/CD provider is selected as **GitHub Actions**.



4. Select **Node.js**, and then select **Next**.
5. Under **Choose an application Framework**, select **Express.js**, and then select **Next**. The application framework, which you chose in a previous step, dictates the type of Azure service deployment target that's available here.
6. Select the **Windows Web App**, and then select **Next**.

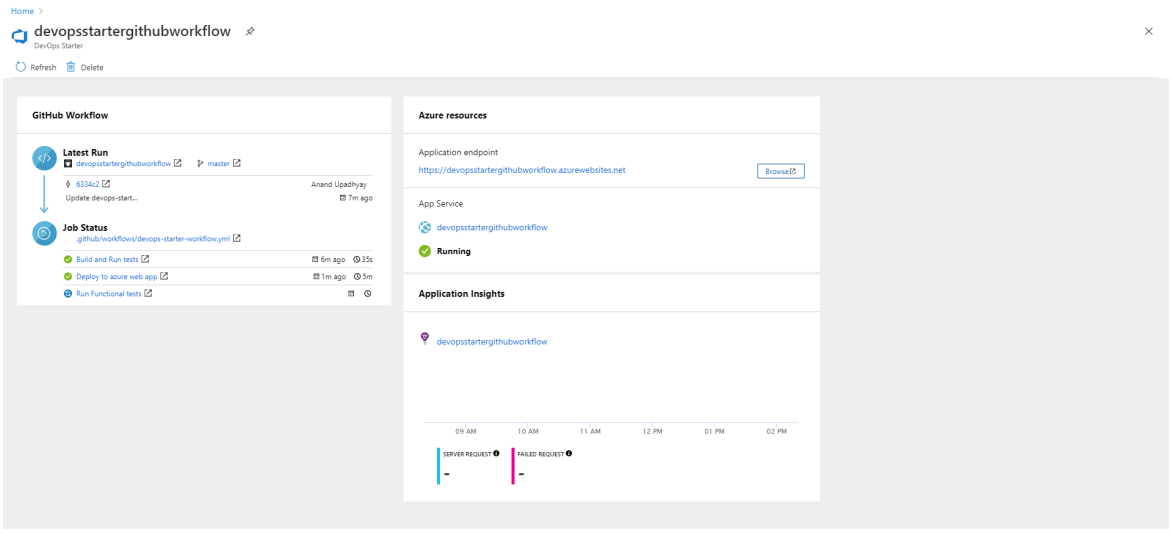
Configure GitHub and an Azure subscription

1. **Authorize** GitHub and select an existing GitHub organization.
2. Enter a name for your **GitHub Repository**.
3. Select your Azure subscription services. Optionally, you can select **Change** and then enter more configuration details, such as the location of the Azure resources.
4. Enter a Web App name, and then select **Done**. After a few minutes, the Azure Web App will be ready. A sample Node.js application is set up in a repo in your GitHub organization, a workflow is triggered, and your application is deployed to the newly created Azure Web App.



After it's completed, the DevOps Starter dashboard is displayed in the Azure portal. You can also navigate to the dashboard directly from **All resources** in the Azure portal.

The dashboard provides visibility into your GitHub code repo, your CI/CD workflow, and your running application in Azure.



DevOps Starter automatically configures a trigger that deploys code changes to your repo.

Examine the GitHub workflow

In the previous step, DevOps Starter automatically configured a full GitHub workflow. Explore and customize the workflow as needed. Take the following steps to familiarize yourself with the workflow.

1. On the left of the DevOps Starter dashboard, select **GitHub workflow**. This link opens a browser tab and the GitHub workflow for your new project.

   > **NOTE**
   > Do not rename the workflow file. The name of the workflow file should be **devops-starter-workflow.yml** for the dashboard to reflect the changes

2. The workflow yaml file contains all the GitHub Actions required to build and deploy the application. Click on **edit file** option to customize your workflow file.

3. Under the **Code** tab of the repo click on **commits**. This view shows code commits that are associated with the specific deployment.

4. Under the **Actions** tab of the repo you can view the history of all the workflow runs of your repository.
5. Select the **latest run** to view all the jobs that ran in the workflow.
6. Click on the **jobs** to view the detailed logs of the workflow run. The logs contain useful information about the deployment process. They can be viewed both during and after deployments.
7. Click on the **Pull request** tab to view all the pull requests on your repository

Commit code changes and execute CI/CD

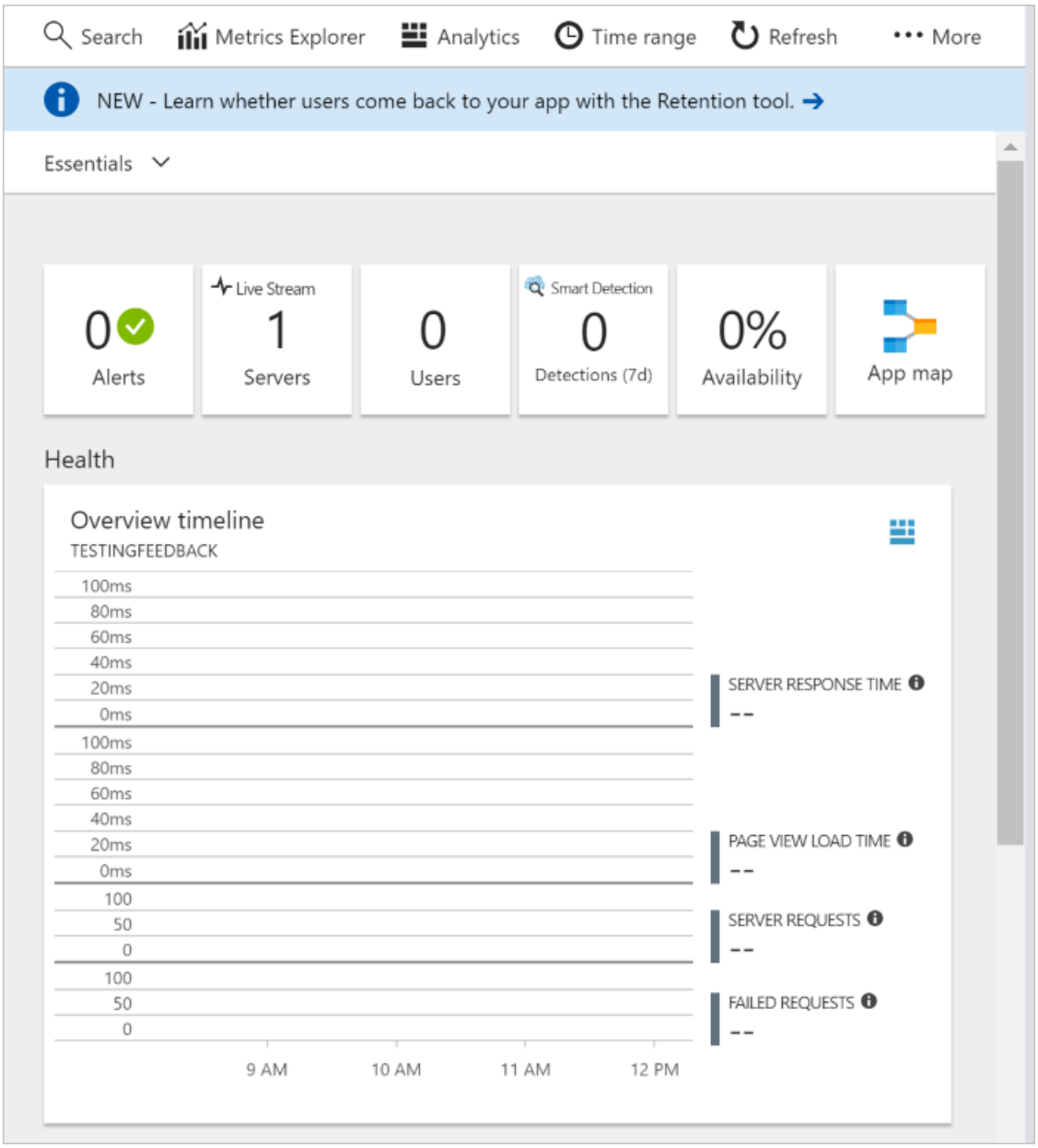
DevOps Starter creates a repository in GitHub. To view the repository and make code changes to your application, do the following:

1. On the left of the DevOps Starter dashboard, select the link for your main branch. This link opens a view to the newly created GitHub repository.
2. To view the repository clone URL, select **Clone** on the top right of the browser. You can clone your Git repository in your favorite IDE. In the next few steps, you can use the web browser to make and commit code changes directly to the main branch.
3. On the left side of the browser, go to the **/Application/views/index.pug** file.
4. Select **Edit**, and then make a change to some of the text. For example, change some of the text for one of the tags.
5. Select **Commit**, and then save your changes.
6. In your browser, go to the DevOps Starter dashboard.
   You should now see a GitHub workflow build job in progress. The changes you just made are automatically built and deployed via a GitHub workflow.
7. After the deploy is completed, refresh your application to verify your changes.

Configure Azure Application Insights monitoring

With Azure Application insights, you can easily monitor your application's performance and usage. DevOps Starter automatically configures an Application Insights resource for your application. You can further configure various alerts and monitoring capabilities as needed.

1. In the Azure portal, go to the DevOps Starter dashboard.
2. At the lower right, select the **Application Insights** link for your app. The **Application Insights** pane opens. This view contains usage, performance, and availability monitoring information for your app.



3. Select **Time range**, and then select **Last hour**. To filter the results, select **Update**. You can now view all activity from the last 60 minutes.
4. To exit the time range, select **x**.
5. Select **Alerts**, and then select **Add metric alert**.
6. Enter a name for the alert.
7. In the **Metric** drop-down list, examine the various alert metrics. The default alert is for a **server response time greater than 1 second**. You can easily configure a variety of alerts to improve the monitoring capabilities of your app.
8. Select the **Notify via Email owners, contributors, and readers** check box. Optionally, you can perform additional actions when an alert is displayed by executing an Azure logic app.
9. Select **OK** to create the alert. After a few moments, the alert appears as active on the dashboard.
10. Exit the **Alerts** area, and go back to the **Application Insights** pane.
11. Select **Availability**, and then select **Add test**.
12. Enter a test name, and then select **Create**. A simple ping test is created to verify the availability of your application. After a few minutes, test results are available, and the Application Insights dashboard displays

an availability status.

Clean up resources

If you are testing, you can avoid accruing billing charges by cleaning up your resources. When they are no longer needed, you can delete the Azure virtual machine and related resources that you created in this tutorial. To do so, use the **Delete** functionality on the DevOps Starter dashboard.

> **IMPORTANT**
> The following procedure permanently deletes resources. The *Delete* functionality destroys the data that's created by the project in DevOps Starter in both Azure, and you will be unable to retrieve it. Use this procedure only after you've carefully read the prompts.

1. In the Azure portal, go to the DevOps Starter dashboard.
2. At the top right, select **Delete**.
3. At the prompt, select **Yes** to *permanently delete* the resources.

You can optionally modify the workflow to meet the needs of your team. You can also use this CI/CD pattern as a template for your other repositories.

Next steps

In this tutorial, you learned how to:

- Use DevOps Starter to deploy a Node.js app
- Configure GitHub and an Azure subscription
- Examine the GitHub workflow
- Commit changes to GitHub and automatically deploy them to Azure
- Configure Azure Application Insights monitoring
- Clean up resources

To learn more about GitHub Actions and workflows, see:

Customize GitHub workflow

Create a CI/CD pipeline for GitHub repo using Azure DevOps Starter

6/28/2022 • 4 minutes to read • Edit Online

Azure DevOps Starter presents a simplified process for creating a continuous integration (CI) and continuous delivery (CD) pipeline to Azure. You can bring your existing code and Git repo, or you can select a sample application.

You will:

- Use DevOps Starter to create a CI/CD pipeline
- Configure access to your GitHub repo and choose a framework
- Configure Azure DevOps and an Azure subscription
- Commit changes to GitHub and automatically deploy them to Azure
- Examine the Azure Pipelines CI/CD pipeline
- Clean up resources

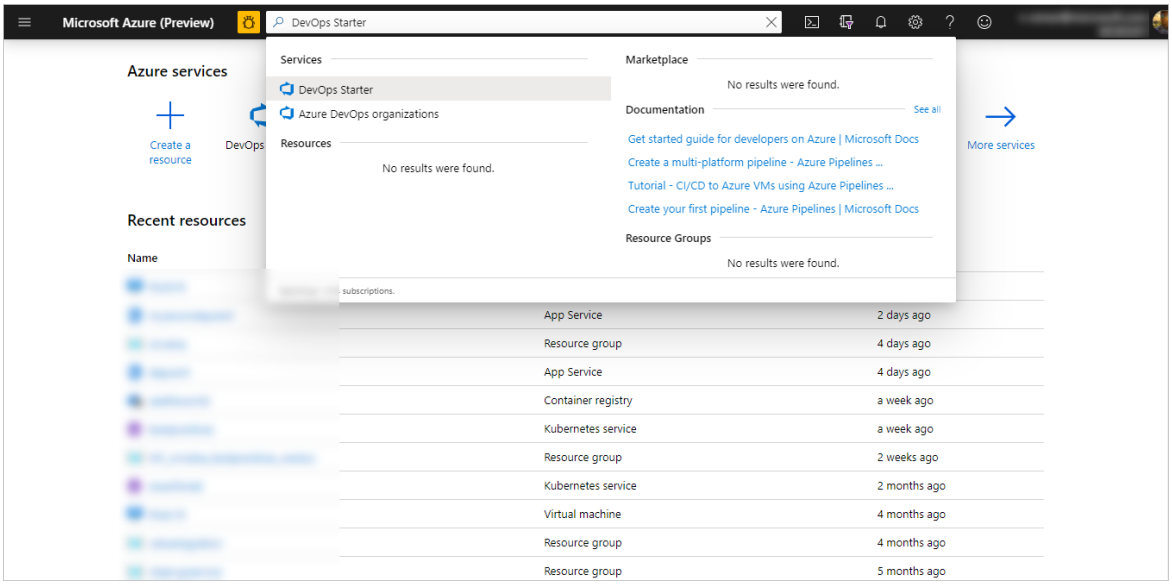
Prerequisites

- An Azure subscription. You can get one free through Visual Studio Dev Essentials.
- Access to a GitHub or external Git repo that contains .NET, Java, PHP, Node.js, Python, or static web code.

Sign in to the Azure portal

Azure DevOps Starter creates a CI/CD pipeline in Azure Pipelines. You can create a new Azure DevOps organization or use an existing organization. Azure DevOps Starter also creates Azure resources in the Azure subscription of your choice.

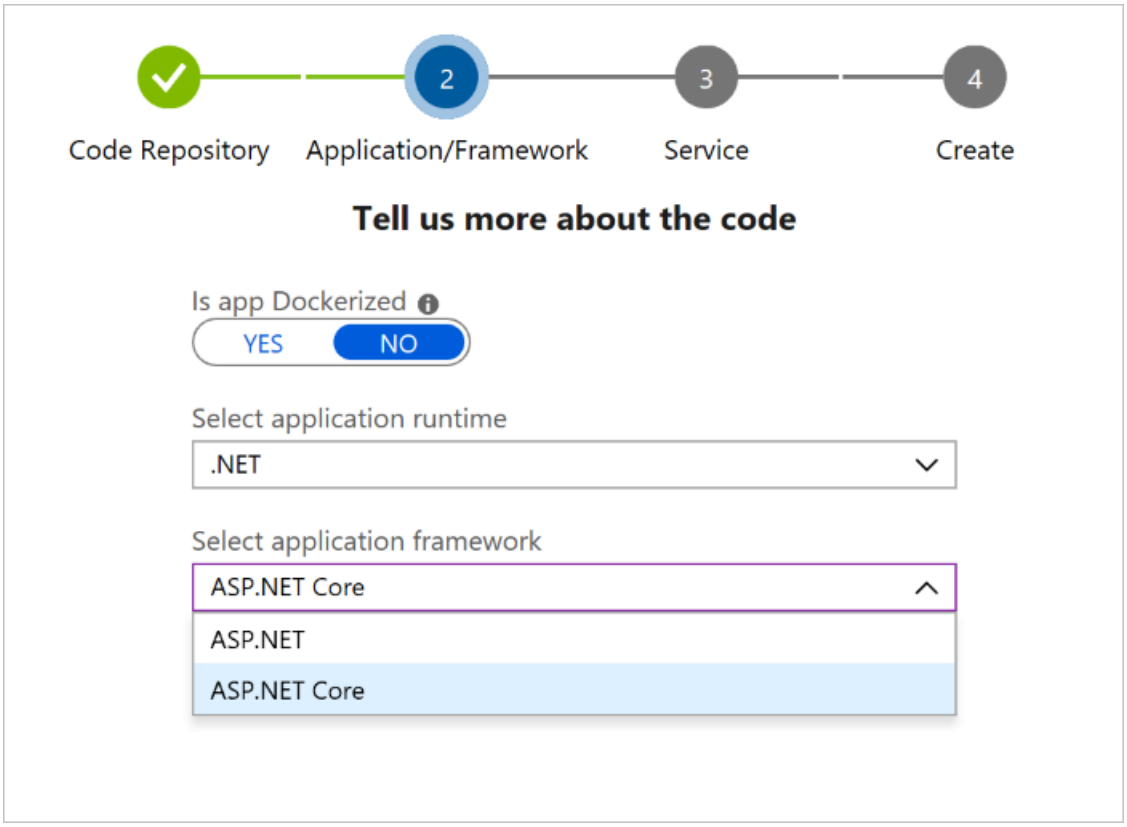
1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



3. Select **Bring your own code**, and then select **Next**.

Configure access to your GitHub repo and select a framework

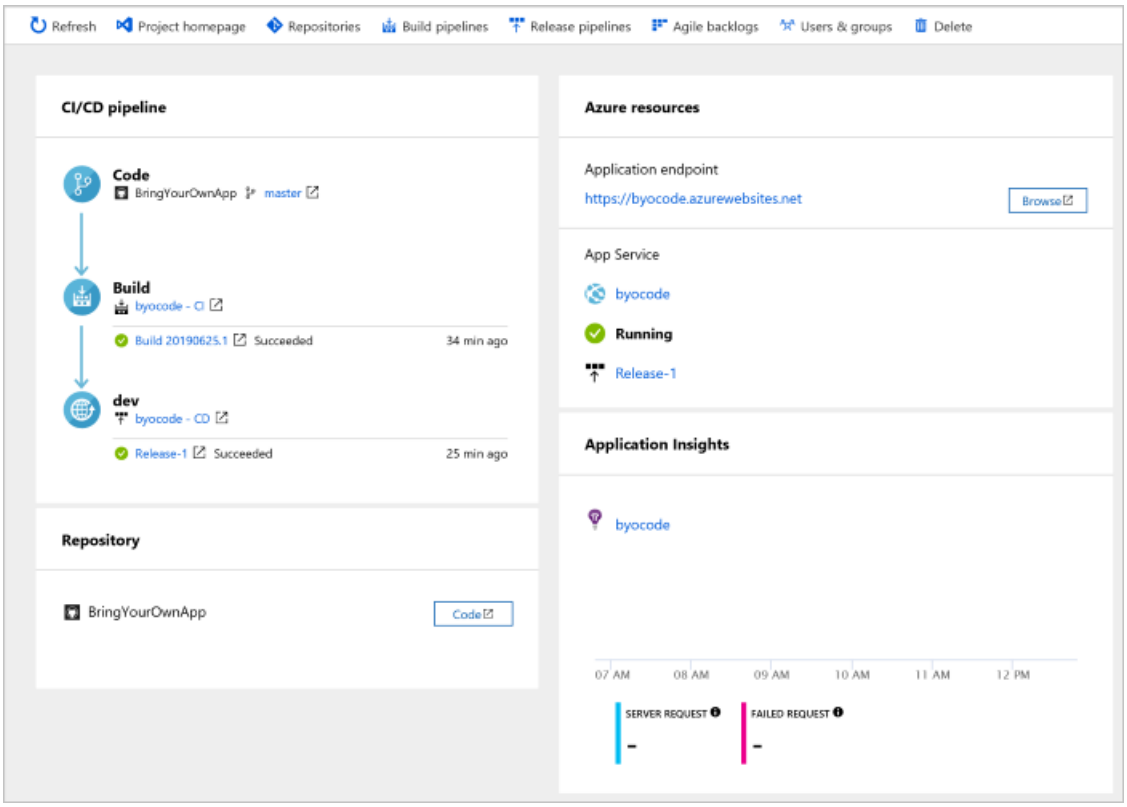
1. Select either **GitHub** or an external **Git** code repository. For this tutorial, select **GitHub**. You might be required to authenticate with GitHub the first time to allow Azure to access your GitHub repository.
2. Select a **Repository** and a **Branch**, and then select **Next**.
3. If you're using Docker containers, change **Is app Dockerized** to **YES**. For this tutorial, leave **NO** selected, and then select **Next**. For more information about using Docker containers, hover over the **i** icon.



4. From the drop-down menus, select an **application runtime** and an **application framework**, and then select **Next**. The application framework dictates the type of Azure service deployment target that's available.
5. Select an **Azure service** to deploy the application, and then select **Next**.

Configure Azure DevOps and an Azure subscription

1. Enter a name for **Project name**.
2. Create a new free organization in **Azure DevOps Organization** or select an existing organization from the drop-down menu.
3. Select your subscription in **Azure Subscription**, and either enter a name in **Web app** or use the default. Select a **Location**, and then select **Done**. After a few minutes, the DevOps Starter deployment overview is displayed in the Azure portal.
4. Select **Go to resource** to view the DevOps Starter dashboard. In the upper-right corner, pin the **Project** to your dashboard for quick access. Azure DevOps Starter automatically configures a CI build and release trigger. Your code remains in your GitHub repo or another external repo, and a sample app is set up in a repo in **Azure DevOps Organization**. Azure DevOps Starter runs the build and deploys the app to Azure.



5. The dashboard shows your code repo, your CI/CD pipeline, and your app in Azure. At the right, under Azure resources, select **Browse** to view your running app.

Commit changes to GitHub and automatically deploy them to Azure

You're now ready to collaborate on your app with a team. The CI/CD process automatically deploys your latest work to your website. Each change to the GitHub repo starts a build in Azure DevOps, and a CD pipeline runs a deployment to Azure.

1. From your DevOps Starter dashboard, select **Repositories**. Your GitHub repository opens in a new browser tab. Make a change to your application, and then select **Commit changes**.

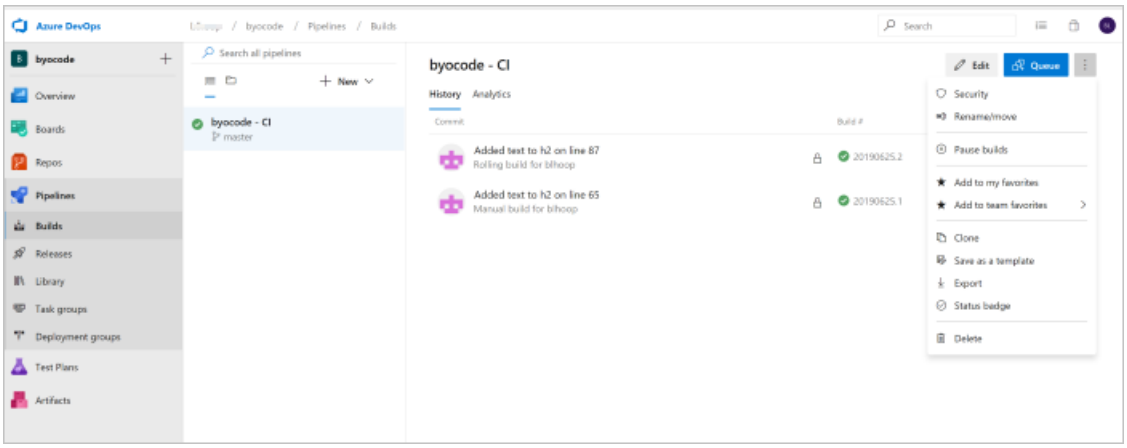
2. After a few moments, a build starts in Azure Pipelines. You can monitor the build status in the DevOps Starter dashboard. You can also monitor it in your Azure DevOps organization by selecting the **Build pipelines** tab from the DevOps Starter dashboard.

Examine the Azure Pipelines CI/CD pipeline

Azure DevOps Starter automatically configures a CI/CD pipeline in Azure Pipelines. Explore and customize the pipeline as needed. To familiarize yourself with the build and release pipelines, do the following:

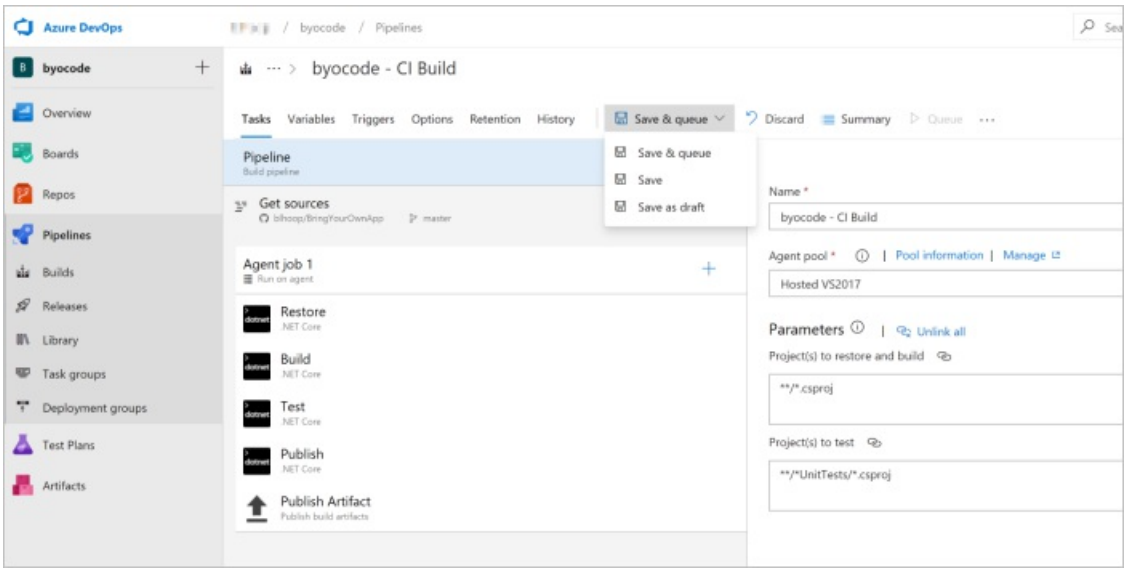
1. From the DevOps Starter dashboard, select **Build pipelines**.

2. After your **Azure Pipelines** page opens, you'll see a history of the most recent builds and the status for each build.



3. In the upper-right corner of the **Builds** page, you can select **Edit** to change the current build, **Queue** to add a new build, or the vertical ellipsis button (⋮) to open a menu with more options. Select **Edit**.

4. The build does various tasks, such as fetching sources from the repo, restoring dependencies, and publishing outputs for deployments. To the right, under **Name**, change the build pipeline name to something more descriptive. Select **Save & Queue**, and then select **Save**. Enter a comment, and then select **Save** again.



5. To see an audit trail of your recent changes for the build, select the **History** tab. Azure DevOps tracks any changes made to the build pipeline and allows you to compare versions.

6. Select the **Triggers** tab. Azure DevOps Projects automatically creates a CI trigger with some default settings. You can set triggers such as **Enable continuous integration** to run a build each time you commit a code change. You can also set triggers to schedule builds to run at specific times.

Clean up resources

When you no longer need Azure App Service and the related resources that you created in this tutorial, you can delete them. Use the **Delete** functionality on the DevOps Projects dashboard.

Next steps

When you configured your CI/CD process in this tutorial, you automatically created a build and release pipeline in Azure DevOps Projects. You can modify these build and release pipelines to meet the needs of your team.

To learn more about the CI/CD pipeline, see:

Define your multi-stage continuous deployment (CD) pipeline

To learn more about application monitoring, see:

What is Azure monitor?

Tutorial: Deploy your ASP.NET app to Azure virtual machines by using Azure DevOps Starter

6/28/2022 • 8 minutes to read • Edit Online

Azure DevOps Starter 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 Starter also:

- Automatically creates Azure resources, such as a new Azure virtual machine (VM).
- Creates and configures a release pipeline in Azure DevOps that includes a build pipeline for CI.
- Sets up a release pipeline for CD.
- Creates an Azure Application Insights resource for monitoring.

In this tutorial, you will:

- Use DevOps Starter to deploy your ASP.NET app
- Configure Azure DevOps and an Azure subscription
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Azure Repos and automatically deploy them to Azure
- Configure Azure Application Insights monitoring
- Clean up resources

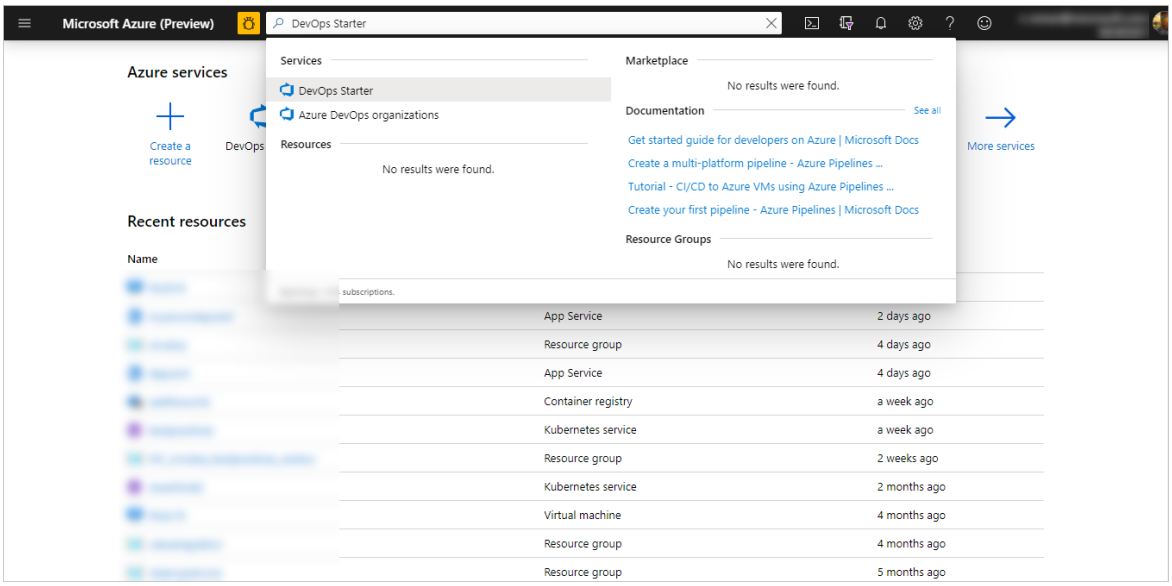
Prerequisites

- An Azure subscription. You can get one free through Visual Studio Dev Essentials.

Use DevOps Starter to deploy your ASP.NET app

DevOps Starter 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 virtual machines in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



3. Select **.NET**, and then select **Next**.

4. Under **Choose an application Framework**, select **ASP.NET**, and then select **Next**. The application framework, which you chose in a previous step, dictates the type of Azure service deployment target that's available here.

5. Select the virtual machine, and then select **Next**.

Configure Azure DevOps and an Azure subscription

1. Create a new Azure DevOps organization or select an existing organization.

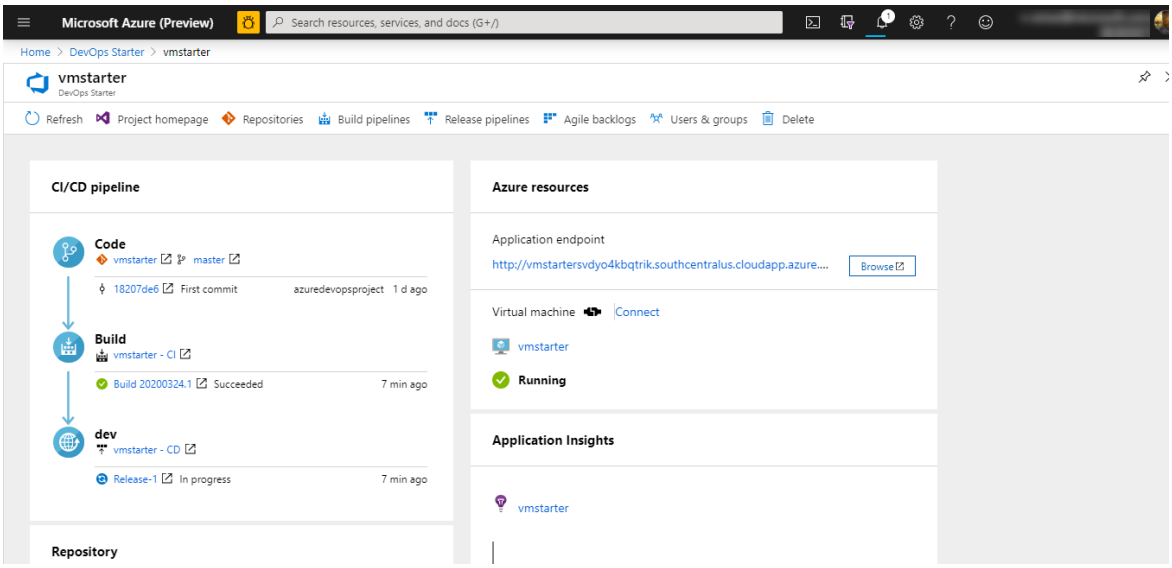
2. Enter a name for your Azure DevOps project.

3. Select your Azure subscription services. Optionally, you can select **Change** and then enter more configuration details, such as the location of the Azure resources.

4. Enter a virtual machine name, username, and password for your new Azure virtual machine resource, and then select **Done**. After a few minutes, the Azure virtual machine will be ready. A sample ASP.NET application is set up in a repo in your Azure DevOps organization, a build and release is executed, and your application is deployed to the newly created Azure VM.

   After it's completed, the DevOps Starter dashboard is displayed in the Azure portal. You can also navigate to the dashboard directly from **All resources** in the Azure portal.

   The dashboard provides visibility into your Azure DevOps code repo, your CI/CD pipeline, and your running application in Azure.



DevOps Starter automatically configures a CI build and release trigger that deploys code changes to your repo. You can further configure additional options in Azure DevOps. To view your running application, select **Browse**.

Examine the CI pipeline

DevOps Starter automatically configured a CI/CD pipeline in Azure Pipelines. You can explore and customize the pipeline. To familiarize yourself with the build pipeline, do the following:

1. At the top of the DevOps Starter dashboard, select **Build Pipelines**. A browser tab displays the build pipeline for your new project.
2. 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.
3. Select **Edit**.
4. 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.
5. At the top of the build pipeline, select the build pipeline name.
6. Change the name of your build pipeline to something more descriptive, select **Save & queue**, and then select **Save**.
7. 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.
8. Select **Triggers**. DevOps Starter 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.
9. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.

Examine the CD pipeline

DevOps Starter 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 CD pipeline, which provides the CD to the Azure virtual machine. To learn more about the Azure DevOps CD pipeline, do the

following:

1. Select **Build and Release**, and then select **Releases**. DevOps Starter 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 previous steps produces the output that's used for the artifact.
4. Next to the **Drop** icon, select **Continuous deployment trigger**. This release pipeline has an enabled CD trigger, which executes a deployment each time a new build artifact is available. Optionally, you can disable the trigger so that your deployments require manual execution.
5. At the left, select **Tasks**, and then select your environment. Tasks are the activities that your deployment process executes, and they're grouped in phases. This release pipeline happens in two phases:
   - The first phase contains an Azure Resource Group Deployment task that does two things:
     - Configures the VM for deployment
     - Adds the new VM to an Azure DevOps deployment group. The VM deployment group in Azure DevOps manages logical groups of deployment target machines
   - In the second phase, an IIS Web App Manage task creates an IIS website on the VM. A second IIS Web App Deploy task is created to deploy the site.
6. At the right, select **View releases** to display a history of releases.
7. 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.
8. Select **Commits**. This view shows code commits that are associated with this deployment. Compare releases to view the commit differences between deployments.
9. 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

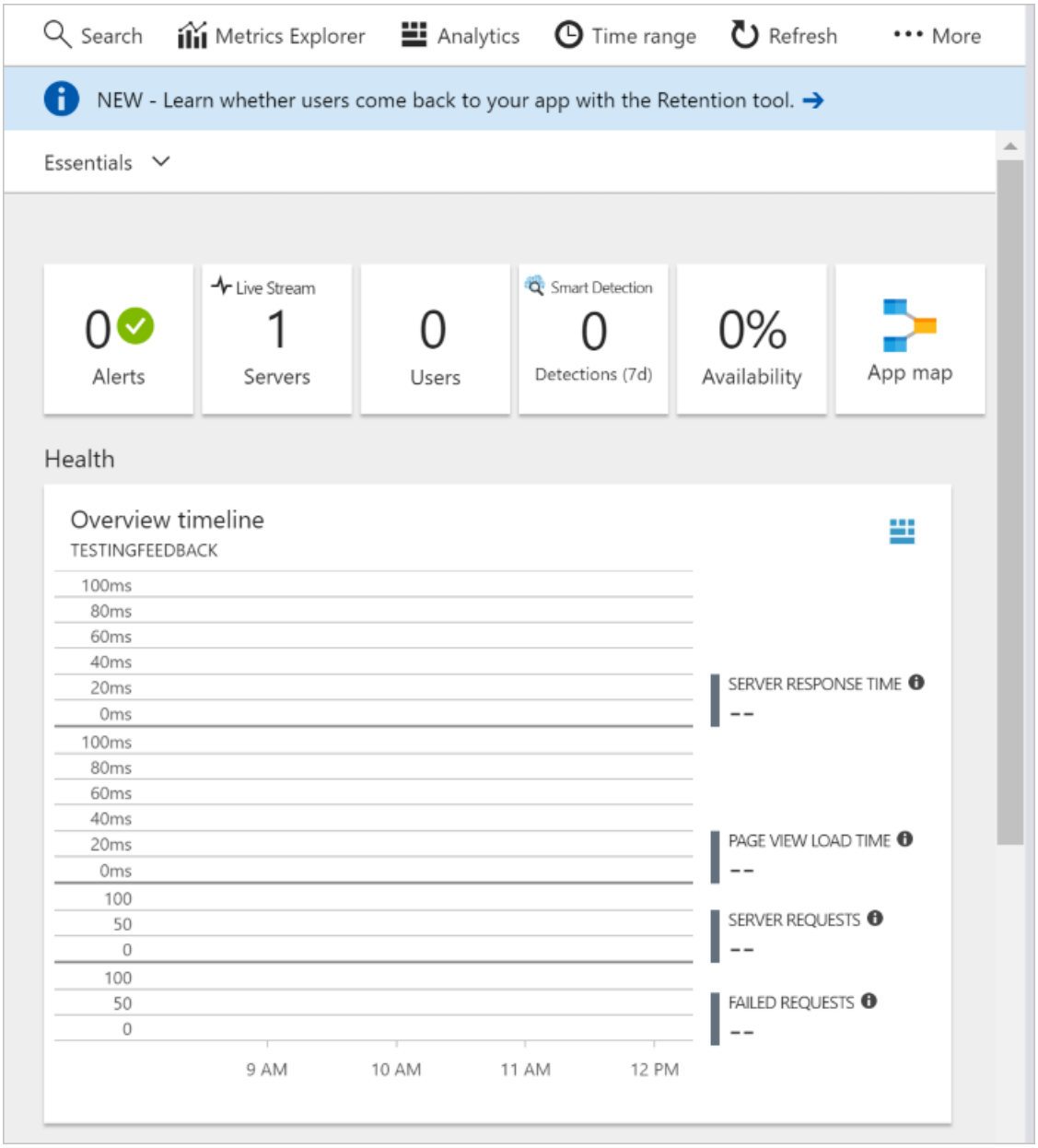
You're now ready to collaborate with a team on your app by using a CI/CD process that automatically deploys your latest work to your website. Each change to the Git repo starts a build in Azure DevOps, and a CD pipeline executes a deployment to Azure. Follow the procedure in this section, or use another technique to commit changes to your repo. The code changes initiate the CI/CD process and automatically deploy your changes to the IIS website on the Azure VM.

1. In the left pane, select **Code**, 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 in the DevOps Starter dashboard or in the browser with your Azure DevOps organization.
5. After the release is completed, refresh your application to verify your changes.

Configure Azure Application Insights monitoring

With Azure Application insights, you can easily monitor your application's performance and usage. DevOps Starter automatically configures an Application Insights resource for your application. You can further configure various alerts and monitoring capabilities as needed.

1. In the Azure portal, go to the DevOps Starter dashboard.
2. At the lower right, select the **Application Insights** link for your app. The **Application Insights** pane opens. This view contains usage, performance, and availability monitoring information for your app.



3. Select **Time range**, and then select **Last hour**. To filter the results, select **Update**. You can now view all activity from the last 60 minutes.
4. To exit the time range, select **x**.
5. Select **Alerts**, and then select **Add metric alert**.
6. Enter a name for the alert.
7. In the **Metric** drop-down list, examine the various alert metrics. The default alert is for a **server response time greater than 1 second**. You can easily configure a variety of alerts to improve the monitoring capabilities of your app.

8. Select the **Notify via Email owners, contributors, and readers** check box. Optionally, you can perform additional actions when an alert is displayed by executing an Azure logic app.
9. Select **OK** to create the alert. After a few moments, the alert appears as active on the dashboard.
10. Exit the **Alerts** area, and go back to the **Application Insights** pane.
11. Select **Availability**, and then select **Add test**.
12. Enter a test name, and then select **Create**. A simple ping test is created to verify the availability of your application. After a few minutes, test results are available, and the Application Insights dashboard displays an availability status.

Clean up resources

If you are testing, you can avoid accruing billing charges by cleaning up your resources. When they are no longer needed, you can delete the Azure virtual machine and related resources that you created in this tutorial. To do so, use the **Delete** functionality on the DevOps Starter dashboard.

> **IMPORTANT**
> The following procedure permanently deletes resources. The *Delete* functionality destroys the data that's created by the project in DevOps Starter in both Azure and Azure DevOps, and you will be unable to retrieve it. Use this procedure only after you've carefully read the prompts.

1. In the Azure portal, go to the DevOps Starter dashboard.
2. At the top right, select **Delete**.
3. At the prompt, select **Yes** to *permanently delete* the resources.

You can optionally modify these build and release pipelines to meet the needs of your team. You can also use this CI/CD pattern as a template for your other pipelines.

Next steps

In this tutorial, you learned how to:

- Use DevOps Starter to deploy your ASP.NET app
- Configure Azure DevOps and an Azure subscription
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Azure Repos and automatically deploy them to Azure
- Configure Azure Application Insights monitoring
- Clean up resources

To learn more about the CI/CD pipeline, see:

Define your multi-stage continuous deployment (CD) pipeline

Tutorial: Deploy your ASP.NET app and Azure SQL Database code by using Azure DevOps Starter

6/28/2022 • 8 minutes to read • Edit Online

Azure DevOps Starter 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 Starter also:

- Automatically creates Azure resources, such as a database in Azure SQL Database.
- Creates and configures a release pipeline in Azure Pipelines that includes a build pipeline for CI.
- Sets up a release pipeline for CD.
- Creates an Azure Application Insights resource for monitoring.

In this tutorial, you will:

- Use Azure DevOps Starter to deploy your ASP.NET app and Azure SQL Database code
- Configure Azure DevOps and an Azure subscription
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Azure Repos and automatically deploy them to Azure
- Connect to Azure SQL Database
- Clean up resources

Prerequisites

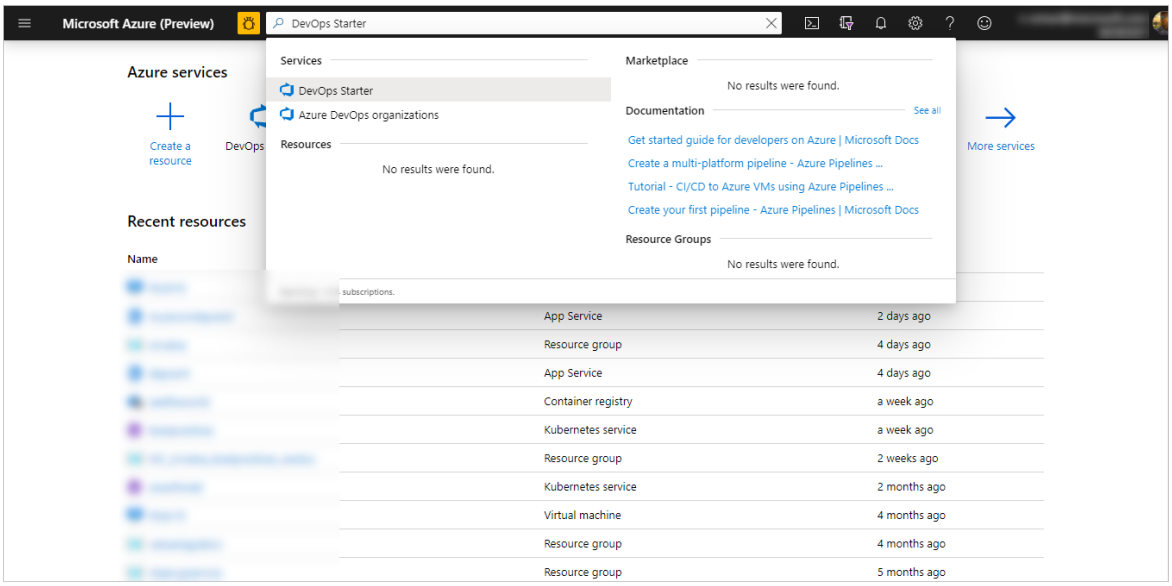
- An Azure subscription. You can get one free through Visual Studio Dev Essentials.

Create a project in DevOps Projects for an ASP.NET app and Azure SQL Database

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

1. Sign in to the Azure portal.

2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



3. Select **.NET**, and then select **Next**.
4. Under **Choose an application framework**, select **ASP.NET**.
5. Select **Add a database**, and then select **Next**. The application framework, which you chose in a previous step, dictates the type of Azure service deployment target that's available here.
6. 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 services. Optionally, to view additional Azure configuration settings and to identify the username in the **Database Server Login Details** section, you can select **Change**. Store the username for future steps in this tutorial. If you perform this optional step, exit the Azure configuration area before you select **Done**.
4. Select **Done**. After a few minutes, the process is completed and the DevOps Starter dashboard opens in the Azure portal. You can also navigate to the dashboard directly from **All resources** in the Azure portal. At the right, select **Browse** to view your running application.

Examine the CI pipeline

DevOps Starter automatically configures a full CI/CD pipeline in Azure Repos. You can explore and customize the pipeline. To familiarize yourself with the Azure DevOps build pipeline, do the following:

1. At the top of the DevOps Starter dashboard, select **Build pipelines**. A browser tab displays the build pipeline for your new project.
2. 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.
3. Select **Edit**.
4. 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 repository, restoring dependencies, and publishing outputs used for deployments.
5. At the top of the build pipeline, select the build pipeline name.

6. Change the name of your build pipeline to something more descriptive, select **Save & queue**, and then select **Save**.

7. Under your build pipeline name, select **History**. This pane displays an audit trail of your recent changes for the build. Azure Pipelines keeps track of any changes made to the build pipeline, and it allows you to compare versions.

8. Select **Triggers**. DevOps Starter automatically creates a CI trigger, and every commit to the repository starts a new build. Optionally, you can choose to include or exclude branches from the CI process.

9. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.

Examine the CD pipeline

DevOps Starter 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 CD pipeline, which provides the CD to the Azure virtual machine. To learn more about the Azure DevOps CD pipeline, do the following:

1. Select **Build and Release**, and then select **Releases**. DevOps Starter 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.

   DevOps Starter sets up a random SQL password and uses it for the release pipeline.

5. At the left, select **Variables**.

   > **NOTE**
   > Perform the following step only if you changed the SQL Server password. There is a single password variable.

6. Next to the **Value** box, select the padlock icon, enter the new password, and then select **Save**.

7. At the left, select **Tasks**, and then select your environment. Tasks are the activities that your deployment process executes, and they are grouped in phases. This release pipeline has a single phase, which contains an *Azure App Service Deploy* and *Azure SQL Database Deployment* task.

8. Select the *Execute Azure SQL* task, and examine the various properties that are used for the SQL deployment. Under **Deployment Package**, the task uses a *SQL DACPAC* file.

9. At the right, select **View releases** to display a history of releases.

10. 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.

11. Select **Commits**. This view shows code commits that are associated with this deployment. Compare releases to view the commit differences between deployments.

12. 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

> **NOTE**
> The following procedure tests the CI/CD pipeline with a simple text change. To test the SQL deployment process, you can optionally make a SQL Server schema change to the table.

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 repository. The code changes initiate the CI/CD process and automatically deploy your changes to Azure.

1. In the left pane, select **Code**, and then go to your repository.
2. Go to the *SampleWebApplication\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 in the DevOps Starter dashboard or in the browser with your Azure DevOps organization.
5. After the release is completed, refresh your application to verify your changes.

Connect to Azure SQL Database

You need appropriate permissions to connect to Azure SQL Database.

1. On the DevOps Starter dashboard, select **SQL Database** to go to the management page for SQL Database.
2. Select **Set server firewall**, and then select **Add client IP**.
3. Select **Save**. Your client IP now has access to the SQL Server Azure resource.
4. Go back to the **SQL Database** pane.
5. At the right, select the server name to navigate to the configuration page for **SQL Server**.
6. Select **Reset password**, enter a password for the SQL Server admin login, and then select **Save**. Be sure to keep this password to use later in this tutorial.

   You may now optionally use client tools such as SQL Server Management Studio or Visual Studio to connect to SQL Server and Azure SQL Database. Use the **Server name** property to connect.

   If you didn't change the database username when you initially configured the project in DevOps Projects, your username is the local part of your email address. For example, if your email address is *johndoe@microsoft.com*, your username is *johndoe*.

> **NOTE**
> If you change your password for the SQL login, you must change the password in the release pipeline variable, as described in the Examine the CD pipeline section.

Clean up resources

If you are testing, you can avoid accruing billing charges by cleaning up your resources. When they are no longer needed, you can delete Azure SQL Database and related resources that you created in this tutorial. To do so, use the **Delete** functionality on the DevOps Starter dashboard.

> **IMPORTANT**
> The following procedure permanently deletes resources. The *Delete* functionality destroys the data that's created by the project in DevOps Starter in both Azure and Azure DevOps, and you will be unable to retrieve it. Use this procedure only after you've carefully read the prompts.

1. In the Azure portal, go to the DevOps Starter dashboard.
2. At the top right, select **Delete**.
3. At the prompt, select **Yes** to *permanently delete* the resources.

Next steps

You can optionally modify these build and release pipelines to meet the needs of your team. You can also use this CI/CD pattern as a template for your other pipelines. In this tutorial, you learned how to:

- Use Azure DevOps Starter to deploy your ASP.NET app and Azure SQL Database code
- Configure Azure DevOps and an Azure subscription
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Azure Repos and automatically deploy them to Azure
- Connect to Azure SQL Database
- Clean up resources

To learn more about the CI/CD pipeline, see:

Define your multi-stage continuous deployment (CD) pipeline

Videos

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

6/28/2022 • 7 minutes to read • Edit Online

Azure DevOps Starter 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 Starter 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 to monitor performance for the container workloads on the AKS cluster

In this tutorial, you will:

- Use DevOps Starter 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

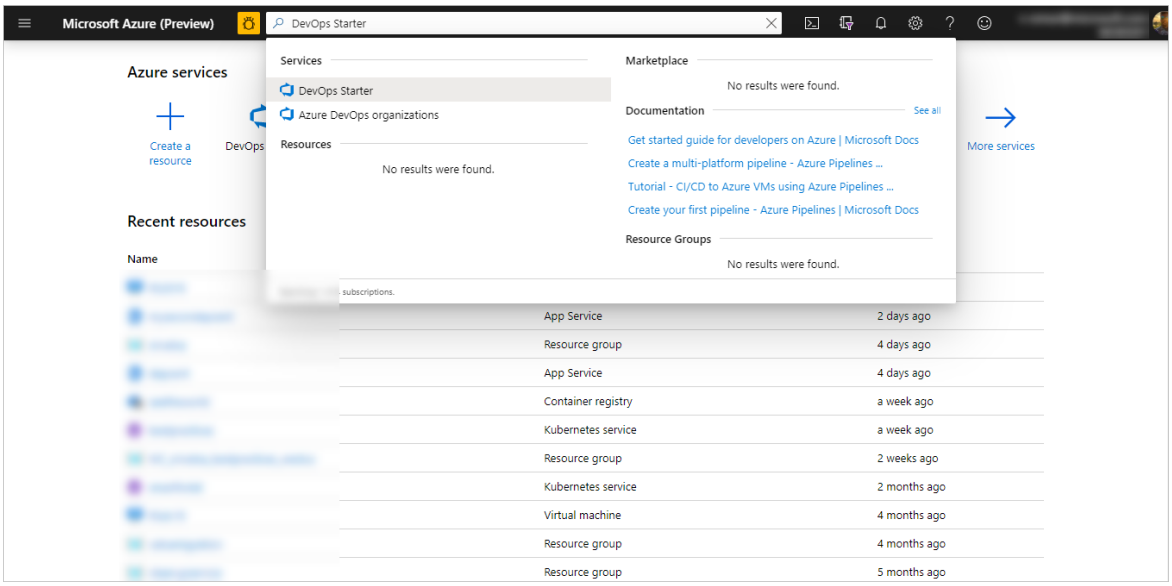
Prerequisites

- An Azure subscription. You can get one free through Visual Studio Dev Essentials.

Use DevOps Starter to deploy an ASP.NET Core app to AKS

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

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



3. Select **.NET**, and then select **Next**.
4. Under **Choose an application framework**, select **ASP.NET Core** and then select **Next**.
5. 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 Starter dashboard is displayed in the Azure portal. You can also go to the DevOps Starter 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 Starter 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 Starter 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 Starter 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 Starter dashboard.
2. At the top of the DevOps Starter 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 Starter 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 Starter 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 Starter 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

> **NOTE**
> 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 Starter 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

If you are testing, you can avoid accruing billing charges by cleaning up your resources. When they are no longer needed, you can delete the AKS cluster and related resources that you created in this tutorial. To do so, use the **Delete** functionality on the DevOps Starter dashboard.

> **IMPORTANT**
> The following procedure permanently deletes resources. The *Delete* functionality destroys the data that's created by the project in DevOps Starter in both Azure and Azure DevOps, and you will be unable to retrieve it. Use this procedure only after you've carefully read the prompts.

1. In the Azure portal, go to the DevOps Starter dashboard.
2. At the top right, select **Delete**.
3. At the prompt, select **Yes** to *permanently delete* the resources.

Next steps

You can optionally modify these build and release pipelines to meet the needs of your team. You can also use this CI/CD pattern as a template for your other pipelines. In this tutorial, you learned how to:

- Use DevOps Starter 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

To learn more about using the Kubernetes dashboard, see:

Use the Kubernetes dashboard

Tutorial: Deploy your ASP.NET Core app to Azure Service Fabric by using Azure DevOps Starter

6/28/2022 • 6 minutes to read • Edit Online

Azure DevOps Starter 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 Starter also:

- Automatically creates Azure resources, such as Azure Service Fabric.
- Creates and configures a release pipeline in Azure DevOps that sets up a CI/CD pipeline.
- Creates an Azure Application Insights resource for monitoring.

In this tutorial, you will:

- Use DevOps Starter to create an ASP.NET Core app and deploy it to Service Fabric
- Configure Azure DevOps and an Azure subscription
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Git and automatically deploy to Azure
- Clean up resources

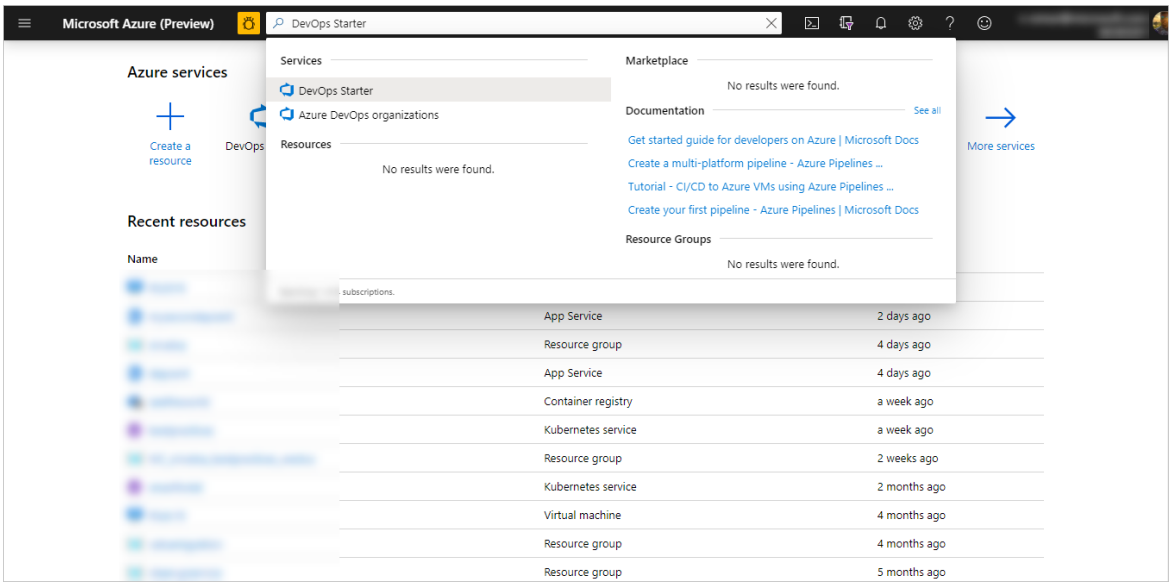
Prerequisites

- An Azure subscription. You can get one free through Visual Studio Dev Essentials.

Use DevOps Starter to create an ASP.NET Core app and deploy it to Service Fabric

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

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



3. Select **.NET**, and then select **Next**.
4. Under **Choose an application framework**, select **ASP.NET Core**, and then select **Next**.
5. Select **Service Fabric Cluster**, 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 node virtual machine size and operating system for the Service Fabric 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, a Service Fabric cluster is created, a CI/CD pipeline is executed, and your app is deployed to Azure.

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

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

Examine the CI pipeline

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

1. Go to the DevOps Starter dashboard.
2. At the top of the DevOps Starter 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. 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.
8. Select **Triggers**. DevOps Starter 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.
9. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.

Examine the CD pipeline

DevOps Starter 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 Starter 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 previously 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 Git and automatically deploy them to Azure

> **NOTE**
> 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, and a release deploys your changes

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, and then a release executes to deploy the changes. You can monitor the build status on the DevOps Starter dashboard or in the browser with Azure DevOps real-time logging.
5. After the release is completed, refresh your app to verify your changes.

Clean up resources

If you are testing, you can avoid accruing billing charges by cleaning up your resources. When they are no longer needed, you can delete the Azure Service Fabric cluster and related resources that you created in this tutorial. To do so, use the **Delete** functionality on the DevOps Starter dashboard.

> **IMPORTANT**
> The following procedure permanently deletes resources. The *Delete* functionality destroys the data that's created by the project in DevOps Starter in both Azure and Azure DevOps, and you will be unable to retrieve it. Use this procedure only after you've carefully read the prompts.

1. In the Azure portal, go to the DevOps Starter dashboard.
2. At the top right, select **Delete**.
3. At the prompt, select **Yes** to *permanently delete* the resources.

Next steps

You can optionally modify the Azure CI/CD pipeline to meet the needs of your team. You can also use this CI/CD pattern as a template for your other pipelines. In this tutorial, you learned how to:

- Use DevOps Starter to create an ASP.NET Core app and deploy it to Service Fabric
- Configure Azure DevOps and an Azure subscription
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Git and automatically deploy them to Azure
- Clean up resources

To learn more about Service Fabric and microservices, see:

Use a microservices approach for building applications

Deploy Node.js apps powered by Azure Cosmos DB with DevOps Starter

6/28/2022 • 6 minutes to read • Edit Online

Azure DevOps Starter offers a streamlined experience where you can create a continuous integration (CI) and continuous deployment (CD) pipeline to Azure. You do this by using your existing code and Git repository (repo) or by selecting a sample application.

DevOps Starter also:

- Automatically creates Azure resources, such as Azure Cosmos DB, Azure Application Insights, Azure App Service, and App Service plans
- Creates and configures a CI/CD release pipeline in Azure DevOps

In this tutorial, you will:

- Use DevOps Starter to deploy a Node.js app powered by Azure Cosmos DB
- Configure Azure DevOps and an Azure subscription
- Examine Azure Cosmos DB
- Examine the CI pipeline
- Examine the CD pipeline
- Commit the changes to Git and automatically deploy them to Azure
- Clean up the resources

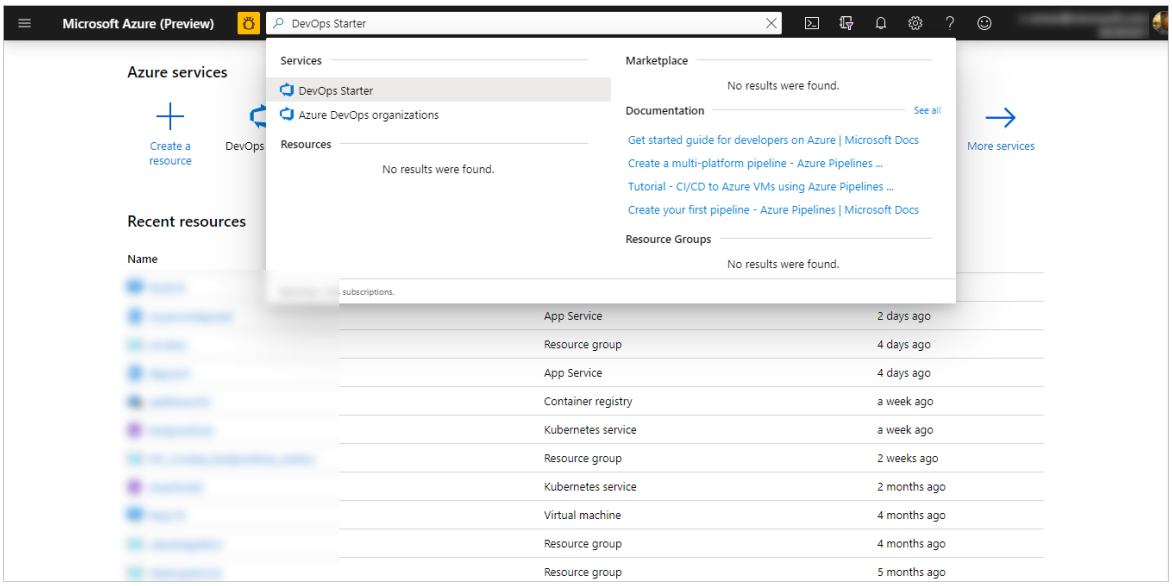
Prerequisites

You need an Azure subscription, which you can get through Visual Studio Dev Essentials for free.

Use DevOps Starter to deploy Node.js app

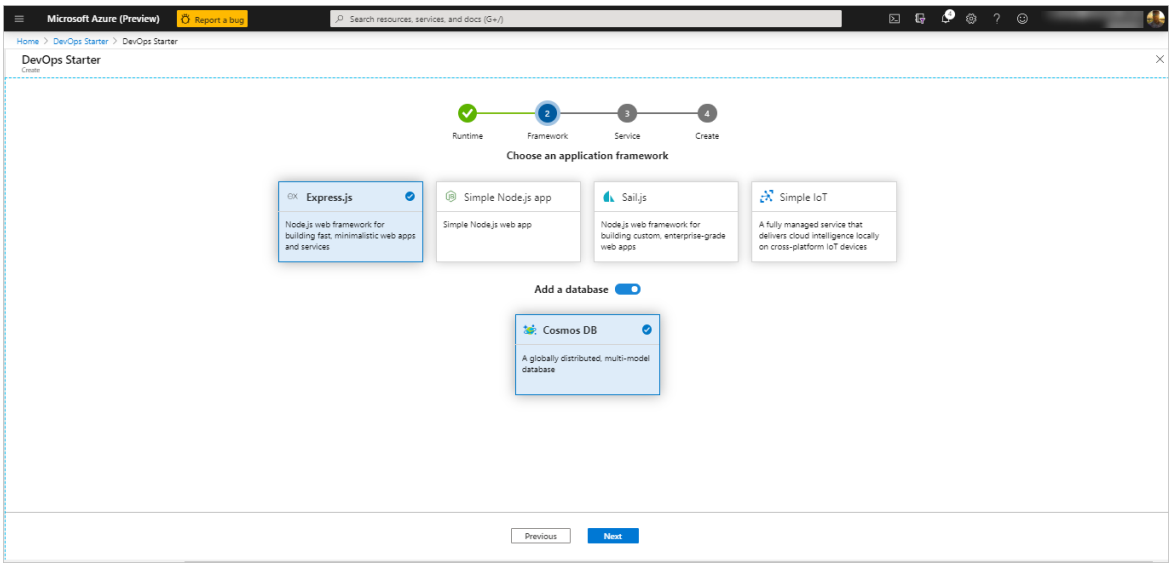
DevOps Starter creates a CI/CD pipeline in Azure Pipelines. You can create a new Azure DevOps organization or use an existing organization. DevOps Starter also creates Azure resources, such as Azure Cosmos DB, Application Insights, App Service, and App Service plans, in the Azure subscription of your choice.

1. Sign in to the Azure portal.
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



3. Select **Node.js** as the runtime, and then select **Next**. Under **Choose an application framework**, select **Express.js**.

4. Enable the section **Add a database** for **Cosmos DB**, and then select **Next**.



   Azure DevOps Starter supports various application frameworks, such as **Express.js**, **Sample Node.js app**, and **Sail.js**. In this tutorial, we use **Express.js**.

5. Select an Azure service to deploy the application, and then select **Next**. Your options include Windows Web App, Azure Kubernetes Service, and Azure Web App for Containers. In this tutorial, we use **Windows Web App**.

Configure Azure DevOps and Azure subscription

1. Enter a name for your Azure DevOps project.

2. Create a new Azure DevOps organization, or select an existing organization.

3. Select your Azure subscription.

4. To view additional Azure configuration settings or identify the pricing tier and location, select **Additional settings**. This pane shows various options for configuring the pricing tier and location of Azure services.

5. Exit the Azure configuration area, and then select **Done**.

6. The process finishes after a few minutes. A sample Node.js app is set up in a Git repo in your Azure

DevOps organization. Then, Azure Cosmos DB, App Service, App Service plan, and Application Insights resources are created, as well as a CI/CD pipeline. Your app is then deployed to Azure.

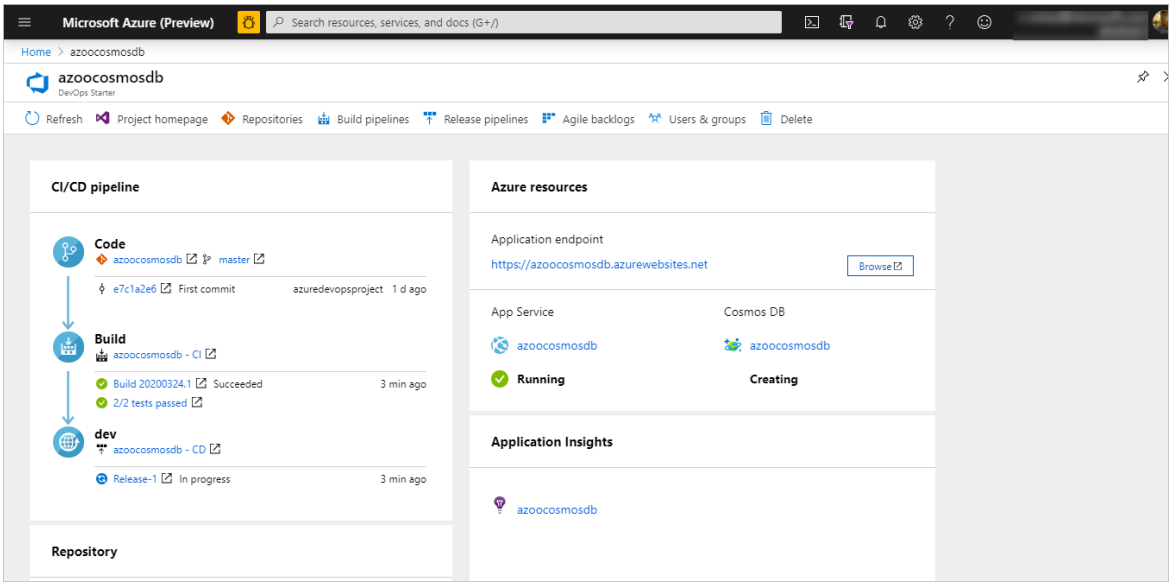
After all these processes finish, the Azure DevOps Starter dashboard displays in the Azure portal. You can also go to the DevOps Starter dashboard directly from **All resources** in the Azure portal.

This dashboard provides visibility into your Azure DevOps code repository, your CI/CD pipeline, and your Azure Cosmos DB database. You can configure additional CI/CD options in your Azure DevOps pipeline. On the right side of the dashboard, select **Azure Cosmos DB** to view these options.

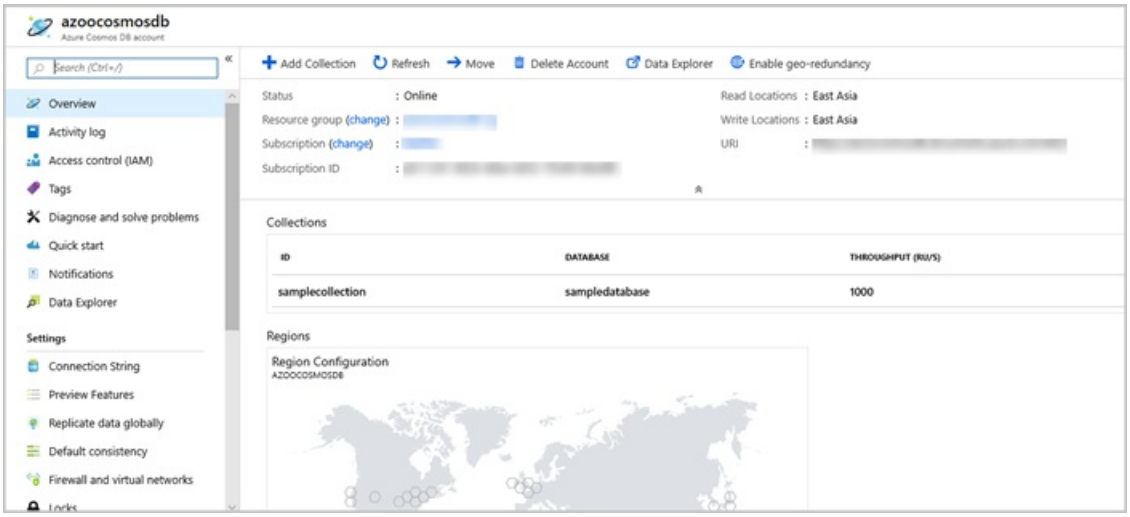
Examine Azure Cosmos DB

DevOps Starter automatically configures Azure Cosmos DB, which you can explore and customize. To familiarize yourself with Azure Cosmos DB, do the following:

1. Go to the DevOps Starter dashboard.



2. At the right, select Azure Cosmos DB. A pane opens for Azure Cosmos DB. From this view, you can perform various actions, such as monitoring operations and searching logs.

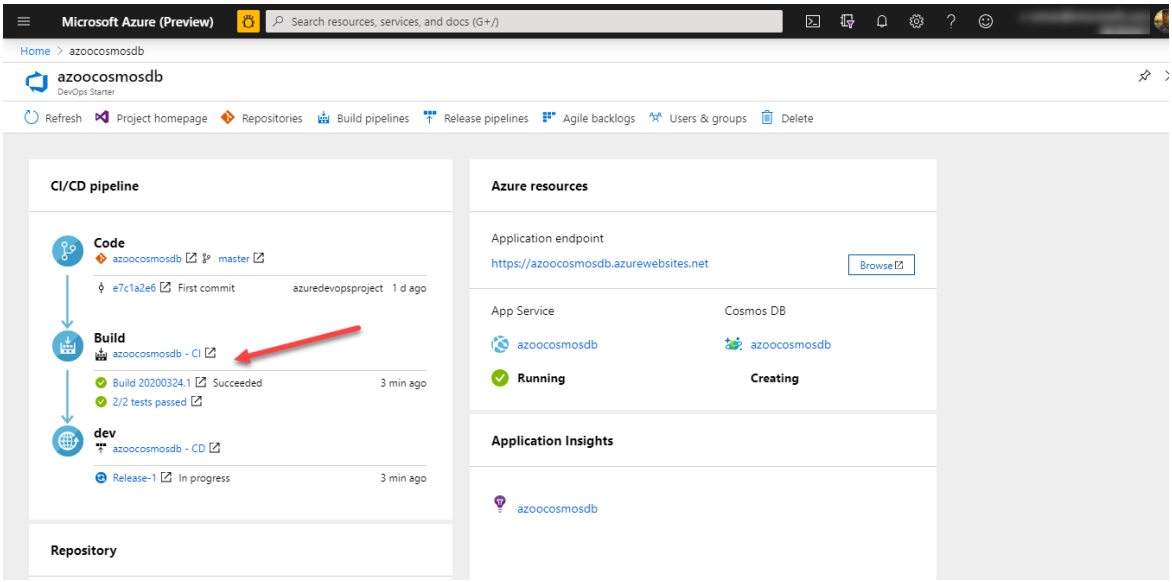


Examine the CI pipeline

DevOps Starter 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 Starter dashboard.

2. Select the hyperlink under **Build**. A browser tab displays the build pipeline for your new project.



3. Select **Edit**. In this pane, you can examine the various tasks for your build pipeline. The build performs various tasks, such as fetching source code from the Git repo, building the application, running unit tests, and publishing outputs that are used for deployments.

4. Select **Triggers**. DevOps Starter automatically creates a CI trigger, and every commit to the repo starts a new build. You can choose to include or exclude branches from the CI process.

5. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.

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, and then select **Save** from the **Save & queue** dropdown.

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.

Examine the CD release pipeline

DevOps Starter 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. Go to **Pipelines** and select **Releases**.

2. Select **Edit**.

3. Under **Artifacts**, select **Drop**. The build pipeline you examined in the previous steps produces the output that's used for the artifact.

4. To the right of the **Drop** icon, select **Continuous deployment trigger**. This release pipeline has enabled continuous deployment trigger, which executes a deployment every time a new build artifact is available. You can disable the trigger so that your deployments execute manually.

5. At the right, select the section **View releases** to display a history of releases.

6. Select the release, which will display the pipeline. Select any environment to check the release summary,

commits, or associated work items.

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 **View Logs**. The logs contain useful information about the deployment process. You can view them both during and after deployments.

Commit code changes and execute the CI/CD pipeline

> **NOTE**
> 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 deploys your latest work to your App Service. 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 **Repos** and then **Files**. Then go to your repo.

2. The repo already contains code based on the application language that you chose in the creation process. Open the **Application/views/index.pug** file.

3. Select **Edit**, and then make a change to **line number 15**. For example, you can change it to "My First deployment to Azure App Service powered by Azure Cosmos DB."

4. In the upper-right corner, select **Commit**, and then select **Commit** again to push your change.

   After a few seconds, a build starts in Azure DevOps and a release executes to deploy the changes. Monitor the build status on the DevOps Starter dashboard or in the browser with your Azure DevOps organization.

Clean up resources

Delete the related resources you've created when you don't need them anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

You can modify these build and release pipelines to meet the needs of your team. You can also use this CI/CD pattern as a template for your other pipelines. In this tutorial, you learned how to:

- Use DevOps Starter to deploy a Node.js app powered by Azure Cosmos DB
- Configure Azure DevOps and an Azure subscription
- Examine Azure Cosmos DB
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Git and automatically deploy them to Azure
- Clean up resources

See Define your multi-stage continuous deployment (CD) pipeline for more information and next steps.

Deploy to Azure Functions with DevOps Starter

6/28/2022 • 6 minutes to read • Edit Online

Azure DevOps Starter 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 Starter also:

- Automatically creates Azure resources, such as Azure Functions
- Creates and configures a release pipeline in Azure DevOps for CI/CD

In this tutorial, you will:

- Use DevOps Starter to deploy an ASP.NET app to Azure Function
- Configure Azure DevOps and an Azure subscription
- Examine the Azure Function
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Git and automatically deploy them to Azure
- Clean up resources

Currently the supported runtimes for functions are **.NET** and **Node.js**. We use. NET runtime for this tutorial to deploy to Azure Functions.

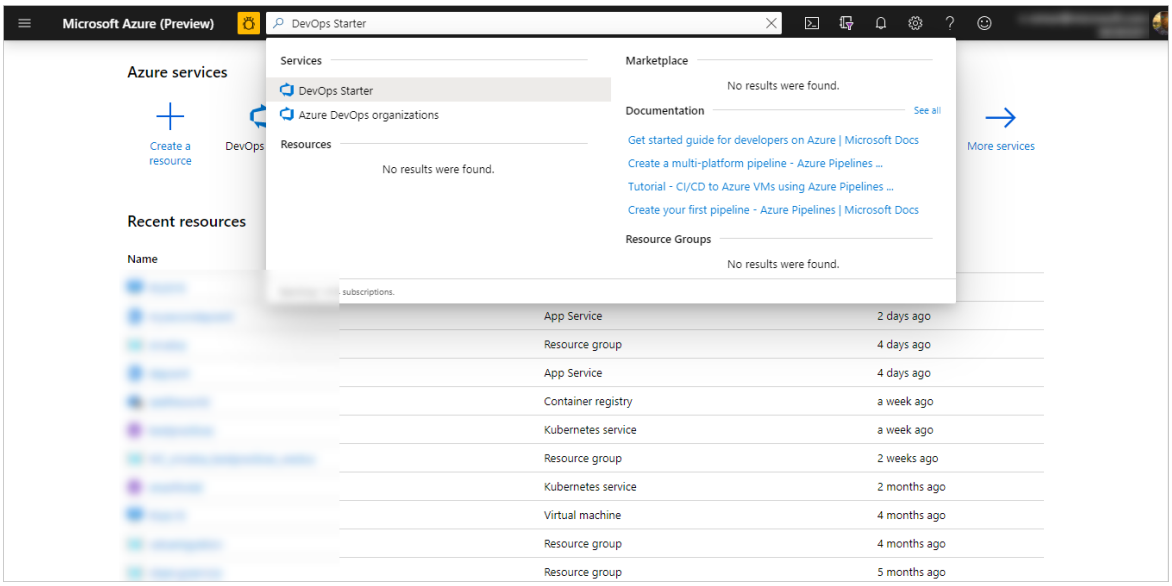
Prerequisites

- An Azure subscription. You can get one free through Visual Studio Dev Essentials

Use DevOps Starter to deploy an ASP.NET app to Azure Functions

DevOps Starter 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 IoTHub, in the Azure subscription of your choice.

1. Sign in to the Azure portal
2. In the search box, type **DevOps Starter**, and then select. Click on **Add** to create a new one.



3. Select **.NET**, and then select **Next**. Under **Choose an application framework**, select **ASP.NET** and click **Next**.

4. Select **Function App** and then select **Next**.

Configure Azure DevOps and Azure subscription

1. Enter a name for your Azure DevOps project.

2. Create a new Azure DevOps organization, or select an existing organization.

3. Select your Azure subscription.

4. To view additional Azure configuration settings and to identify the pricing tier and location, click on **Additional settings**. This pane displays various options for configuring the pricing tier and location of Azure services.

5. Exit the Azure configuration area, and then select **Done**.

6. After few minutes, the process is completed. A sample ASP.NET app is set up in a Git repo in your Azure DevOps organization, a Function App, and Application Insights is created, a CI/CD pipeline is executed, and your app is deployed to Azure.

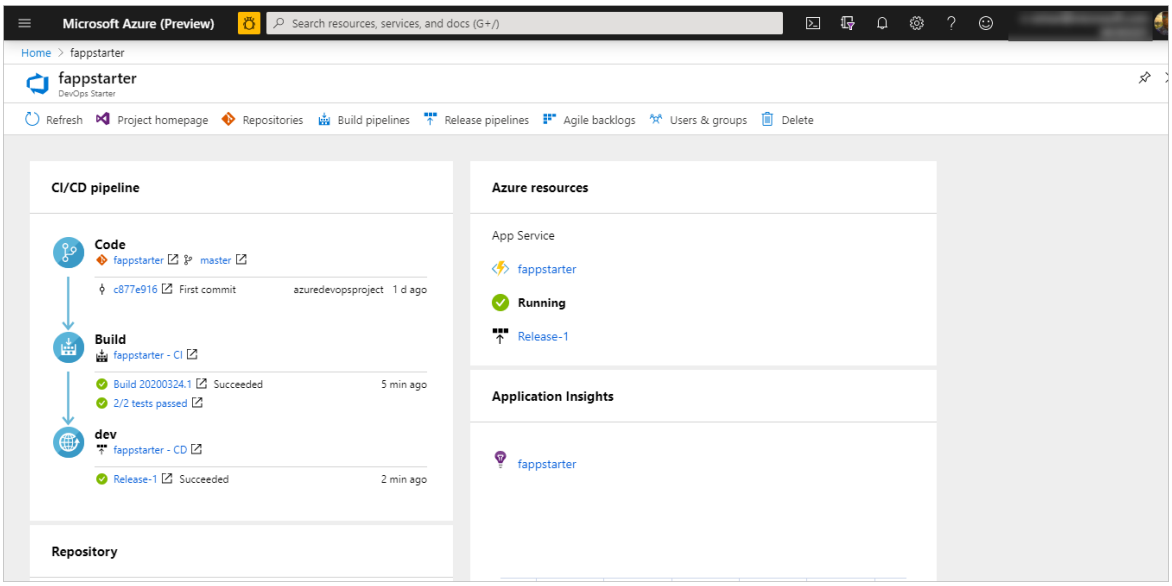
   After all this is completed, the Azure DevOps Starter dashboard is displayed in the Azure portal. You can also go to the DevOps Starter dashboard directly from **All resources** in the Azure portal.

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

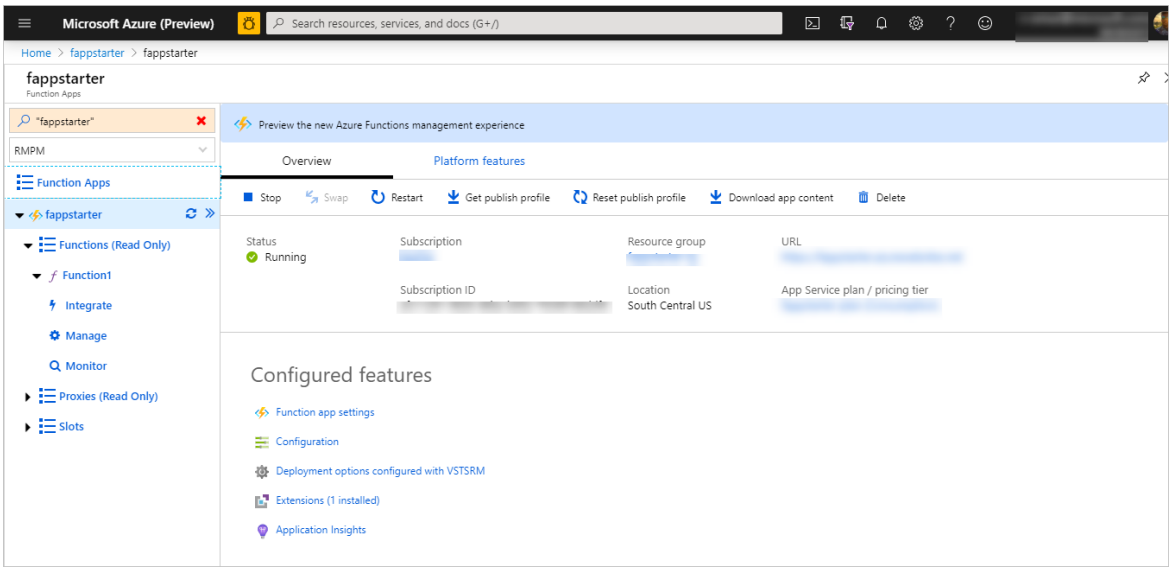
Examine the Function App

DevOps Starter automatically configures function app, which you can explore and customize. To get to know the function app, do the following:

1. Go to the DevOps Starter dashboard.



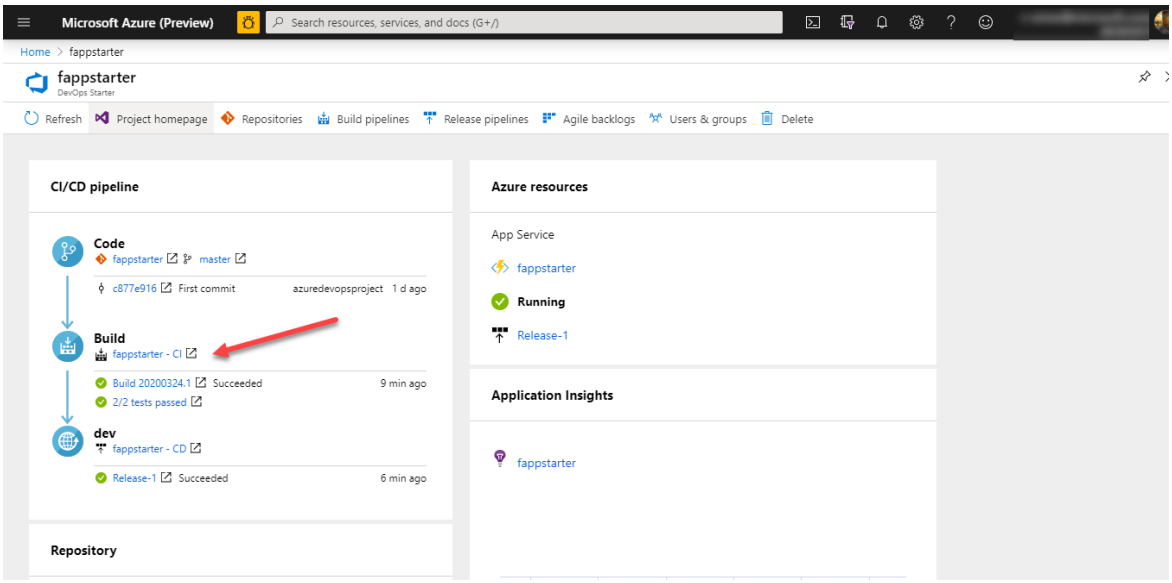
2. At the right, select the function app. A pane opens for the function app. From this view you can perform various actions such as operations monitoring, searching logs.



Examine the CI pipeline

DevOps Starter 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 Starter dashboard.
2. Click on the hyperlink under **Build**. A browser tab displays the build pipeline for your new project.



3. Select **Edit**. In this pane, you can examine the various tasks for your build pipeline. The build performs various tasks, such as fetching source code from the Git repo, building the application, running unit tests, and publishing outputs that are used for deployments.

4. Select **Triggers**. DevOps Starter 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.

5. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.

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, and then select **Save** from the **Save & queue** dropdown.

8. Under your build pipeline name, select **History**. This pane displays an audit trail of your recent changes for the build. Azure DevOps keep track of any changes made to the build pipeline, and it allows you to compare versions.

Examine the CD release pipeline

DevOps Starter 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. Navigate to the **Pipelines | Releases**.

2. Click on **Edit**.

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. Click on the release, which will display the pipeline. Click on any environment to check the release **Summary, Commits**, associated **Work Items**.

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 **View Logs**. The logs contain useful information about the deployment process. You can view them both during and after deployments.

Commit code changes and execute CI/CD

> **NOTE**
> 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 Azure Function. 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 **Repos | Files**, and then go to your repo.
2. The repository already contains code called **SampleFunctionApp** based on the application language that you chose in the creation process. Open the **Application/SampleFunctionApp/Function1.cs** file.
3. Select **Edit**, and then make a change to **line number 31** . For example, you can update it to **Hello there! Welcome to Azure Functions using DevOps Starter**
4. At the top right, select **Commit**, and then select **Commit** again to push your change.
5. Open the **Application/SampleFunctionApp.Test/Function1TestRunner.cs** file.
6. Select **Edit**, and then make a change to **line number 21**. For example, you can update it to **Hello there! Welcome to Azure Functions using Azure DevOps Starter**.

   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 Starter dashboard or in the browser with your Azure DevOps organization.

Clean up resources

You can delete the related resources that you created when you don't need them anymore. Use the **Delete** functionality on the DevOps Starter dashboard.

Next steps

You can optionally modify these build and release pipelines to meet the needs of your team. You can also use this CI/CD pattern as a template for your other pipelines. In this tutorial, you learned how to:

- Use DevOps Starter to deploy an ASP.NET Core app to Azure Function
- Configure Azure DevOps and an Azure subscription
- Examine the Azure Function
- Examine the CI pipeline
- Examine the CD pipeline
- Commit changes to Git and automatically deploy them to Azure
- Clean up resources