Deploy an ASP.NET Core app to App Engine

About this codelab

subjectLast updated Feb 16, 2022

account\_circleWritten by Mete Atamel

[1. Overview](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#0)

[ASP.NET Core](https://docs.microsoft.com/en-us/aspnet/core/) is a new open-source and cross-platform framework for building modern cloud-based and internet-connected applications using the C# programming language.

In this lab, you will deploy a simple ASP.NET Core app to App Engine flexible environment. This codelab builds on the [Build and launch ASP.NET Core app from Google Cloud Shell](https://codelabs.developers.google.com/codelabs/cloud-aspnetcore-cloudshell) codelab. You might want to do that lab first before attempting this lab.

Google App Engine applications are easy to create, maintain, and scale as your traffic and data storage needs change. With App Engine, there are no servers to manage. You simply upload your application and it's ready to go.

App Engine applications automatically scale based on incoming traffic. App Engine natively supports load balancing, microservices, authorization, SQL and NoSQL databases, Memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning, all of which are highly customizable.

App Engine's environments, the [standard environment](https://cloud.google.com/appengine/docs/about-the-standard-environment) and the [flexible environment](https://cloud.google.com/appengine/docs/flexible/), support a host of programming languages, including C#, Java, Python, PHP, Node.js, Go, and more. The two environments give users maximum flexibility in how their application behaves, since each environment has certain strengths. For more information, read [Choosing an App Engine Environment](https://cloud.google.com/appengine/docs/the-appengine-environments).

What you'll learn

* How to package a simple ASP.NET Core app as a Docker container.
* How to deploy a simple ASP.NET Core app to App Engine.

What you'll need

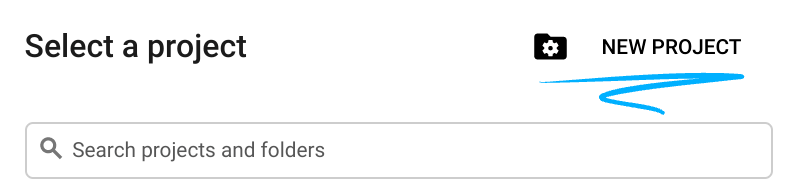
* A Google Cloud Platform project
* A browser, such [Chrome](https://www.google.com/chrome/browser/desktop/) or [Firefox](https://www.mozilla.org/firefox/)

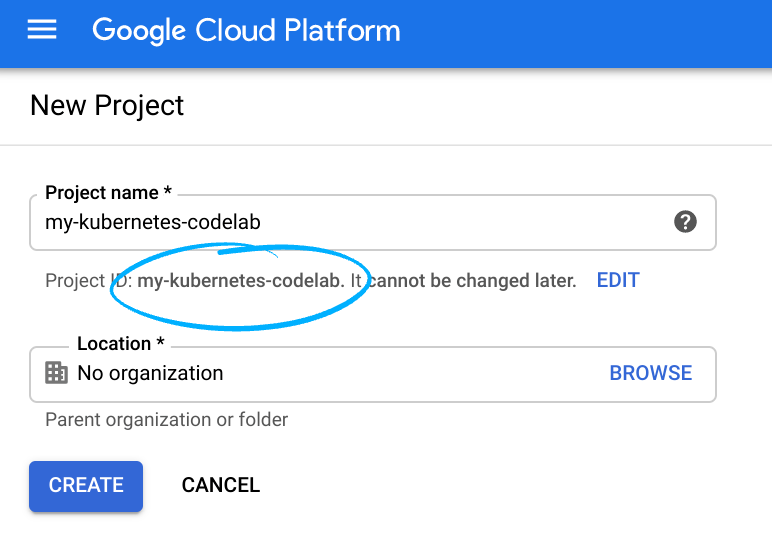
[2. Setup and Requirements](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#1)

**Self-paced environment setup**

1. Sign-in to the [Google Cloud Console](http://console.cloud.google.com/) and create a new project or reuse an existing one. If you don't already have a Gmail or Google Workspace account, you must [create one](https://accounts.google.com/SignUp).







* The **Project name** is the display name for this project's participants. It is a character string not used by Google APIs, and you can update it at any time.
* The **Project ID** must be unique across all Google Cloud projects and is immutable (cannot be changed after it has been set). The Cloud Console auto-generates a unique string; usually you don't care what it is. In most codelabs, you'll need to reference the Project ID (and it is typically identified as PROJECT\_ID), so if you don't like it, generate another random one, or, you can try your own and see if it's available. Then it's "frozen" after the project is created.
* There is a third value, a **Project Number** which some APIs use. Learn more about all three of these values in the [documentation](https://cloud.google.com/resource-manager/docs/creating-managing-projects#before_you_begin).

**Caution:** A project ID must be globally unique and cannot be used by anyone else after you've selected it. You are the only user of that ID. If a project is deleted, that ID can never be used again.

**Note:** If you're using a Gmail account, you can leave the default location set to **No organization**. If you're using a Google Workspace account, then choose a location that makes sense for your organization.

1. Next, you'll need to [enable billing](https://console.cloud.google.com/billing) in the Cloud Console in order to use Cloud resources/APIs. Running through this codelab shouldn't cost much, if anything at all. To shut down resources so you don't incur billing beyond this tutorial, follow any "clean-up" instructions found at the end of the codelab. New users of Google Cloud are eligible for the [$300 USD Free Trial](http://cloud.google.com/free) program.

**Start Cloud Shell**

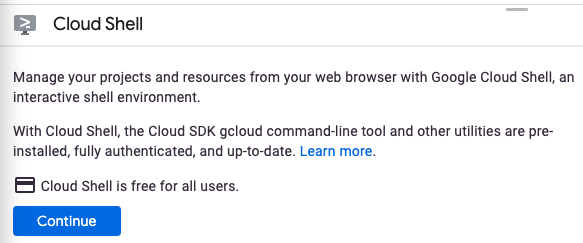
While Google Cloud can be operated remotely from your laptop, in this codelab you will be using [Google Cloud Shell](https://cloud.google.com/cloud-shell/), a command line environment running in the Cloud.

Activate Cloud Shell

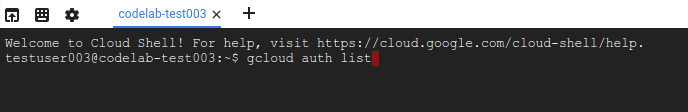
1. From the Cloud Console, click **Activate Cloud Shell**.



If you've never started Cloud Shell before, you're presented with an intermediate screen (below the fold) describing what it is. If that's the case, click **Continue** (and you won't ever see it again). Here's what that one-time screen looks like:



It should only take a few moments to provision and connect to Cloud Shell.



This virtual machine is loaded with all the development tools you need. It offers a persistent 5GB home directory and runs in Google Cloud, greatly enhancing network performance and authentication. Much, if not all, of your work in this codelab can be done with simply a browser or your Chromebook.

Once connected to Cloud Shell, you should see that you are already authenticated and that the project is already set to your project ID.

1. Run the following command in Cloud Shell to confirm that you are authenticated:

gcloud auth list

**Command output**

Credentialed Accounts

ACTIVE ACCOUNT

\* <my\_account>@<my\_domain.com>

To set the active account, run:

$ gcloud config set account `ACCOUNT`

**Note:** The gcloud command-line tool is the powerful and unified command-line tool in Google Cloud. It comes preinstalled in Cloud Shell. You will notice its support for tab completion. For more information, see [gcloud command-line tool overview](https://cloud.google.com/sdk/gcloud/" \t "_blank).

1. Run the following command in Cloud Shell to confirm that the gcloud command knows about your project:

gcloud config list project

**Command output**

[core]

project = <PROJECT\_ID>

If it is not, you can set it with this command:

gcloud config set project <PROJECT\_ID>

**Command output**

Updated property [core/project].

## [3. Create an ASP.NET Core app in Cloud Shell](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#2)

In Cloud Shell prompt, you can verify that the dotnet command line tool is already installed by listing the installed .NET SDKs:

dotnet --list-sdks

Next, create a new skeleton ASP.NET Core web app with a target framework of netcoreapp3.1:

dotnet new mvc -o HelloWorldAspNetCore -f netcoreapp3.1

This should create a project and restore its dependencies. You should see a message similar to below.

Restore completed in 11.44 sec for HelloWorldAspNetCore.csproj.  
  
Restore succeeded.

## [4. Run the ASP.NET Core app](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#3)

We're almost ready to run our app. Navigate to the app folder.

cd HelloWorldAspNetCore

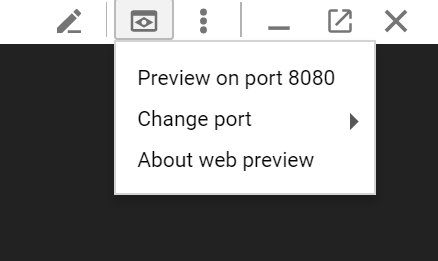
Finally, run the app.

dotnet run --urls=http://localhost:8080

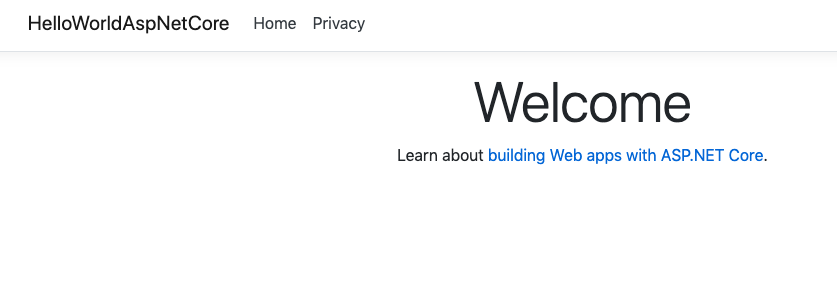
Application starts listening on port 8080.

Hosting environment: Production  
Content root path: /home/atameldev/HelloWorldAspNetCore  
Now listening on: http://[::]:8080  
Application started. Press Ctrl+C to shut down.

To verify that the app is running, click on the web preview button on the top right and select ‘Preview on port 8080'.



You'll see the default ASP.NET Core webpage:



Once you verify that the app is running, press Ctrl+C to shut down the app.

## [5. Publish the ASP.NET Core app](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#4)

Now, publish the app to get a self-contained DLL using the dotnet publish command.

dotnet publish -c Release

Running publish displays some messages with a successfully published DLL at the end of the process.

...  
  HelloWorldAspNetCore -> /home/atameldev/HelloWorldAspNetCore/bin/Release/netcoreapp3.1/HelloWorldAspNetCore.dll  
  HelloWorldAspNetCore -> /home/atameldev/HelloWorldAspNetCore/bin/Release/netcoreapp3.1/HelloWorldAspNetCore.Views.dll  
  HelloWorldAspNetCore -> /home/atameldev/HelloWorldAspNetCore/bin/Release/netcoreapp3.1/publish/

## [6. Create app.yaml for App Engine flexible](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#5)

The app.yaml file describes how to deploy the app to App Engine, in this case, the App Engine flexible environment.

First, navigate to the publish folder. This should be under bin/Release folder but the exact path depends on the version of .NET:

cd bin/Release/netcoreapp3.1/publish/

Create an app.yaml file inside the publish folder:

cat <<EOT >> app.yaml  
env: flex  
runtime: aspnetcore  
EOT

Notice how app.yaml file specifies the environment as flex and runtime as aspnetcore.

## [7. Deploy to App Engine flexible](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#6)

You're ready to deploy your app to App Engine flexible using gcloud. Inside the publish directory, run the following:

gcloud app deploy --version v0

During deployment, you may be asked to choose a region for your application. Choose a region where you want your app to run in.

Please choose a region for your application. After choosing a region,   
you cannot change it. Which region would you like to choose?  
 [1] europe-west   (supports standard and flexible)  
 [2] us-central    (supports standard and flexible)  
 [3] us-east1      (supports standard and flexible)  
 [4] asia-northeast1 (supports standard and flexible)  
 [5] cancel

**Note**

First-time deployment may take several minutes. This is because App Engine flexible environment automatically provisions a Google Compute Engine virtual machine for you behind the scenes, then installs the application and starts it. If your deployment times out, please let the instructor know.

This will create an image for your application in the cloud, save that image to Google Container Registry and deploy to App Engine. During deployment, you can actually see the container image being built:

Operation completed over 1 objects/571.8 KiB.  
BUILD  
Step #0: Pulling image: gcr.io/gcp-runtimes/aspnetcorebuild@sha256:d7b7975acb374fc3a9655a4e529993e6270cfa78023885684626528bc379f8eb  
Step #0: sha256:d7b7975acb374fc3a9655a4e529993e6270cfa78023885684626528bc379f8eb: Pulling from gcp-runtimes/aspnetcorebuild

In the end, you should see that the app is deployed.

...  
Deployed service [default] to [https://<project-id>.appspot.com]

After you've deployed the application,visit it by opening the URL http://<project-id>.appspot.com in your web browser.

**Note**

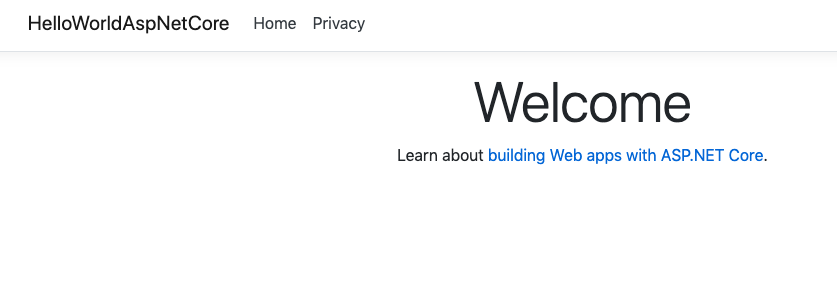
If you don't remember your project-id, please refer to the "Setup and Requirements" section or check the deployment logs for something similar to this:

Deployed service [default] to [https://<project-id>.appspot.com]

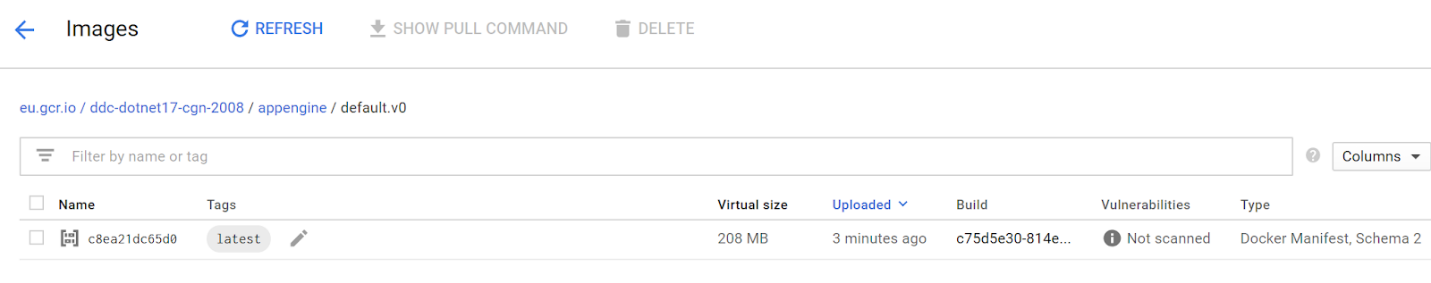
**Note**

Optionally, you can instead purchase and use a top-level domain name for your app, or use one that you have already [registered](https://cloud.google.com/appengine/docs/python/console/using-custom-domains-and-ssl).

You'll see the default ASP.NET Core webpage in a new tab.



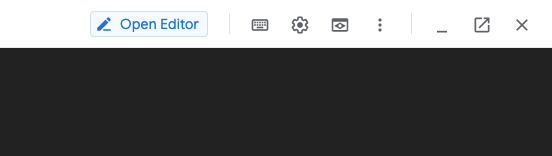
You can also take a look at the container image created for you in the cloud. In cloud console, go to Container Registry > Images and then in appengine folder, you should see the image for your application.



## [8. Deploy a new version of your service](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#7)

At some point, the application that you've deployed to production will require bug fixes or additional features. App Engine is here to help you deploy a new version to production without impacting your users.

First, let's modify the application. Open the code editor from Cloud Shell.



Navigate to Index.cshtml under Views/Home folder of HelloWorldAspNetCore and update the default message to this:

Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core on Google Cloud Platform!

Save the changes and then go back to Cloud Shell. Inside HelloWorldAspNetCore,publish the app to get a self-contained DLL.

dotnet publish -c Release

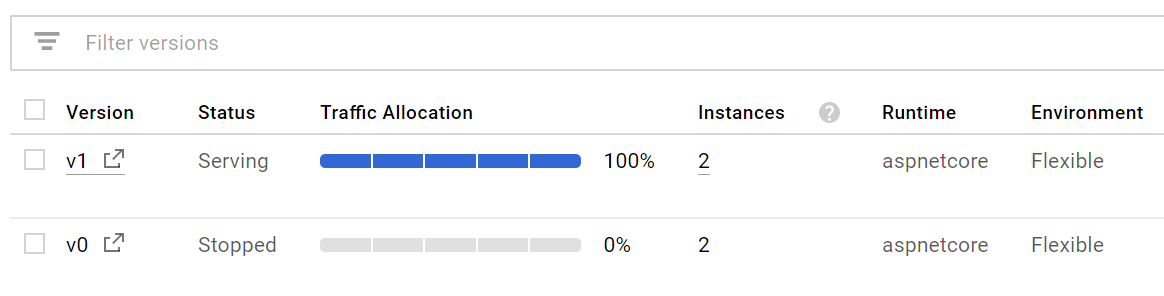
Navigate to the publish directory.

cd bin/Release/netcoreapp3.1/publish/

You can now deploy a new version of your app (v1 in this case).

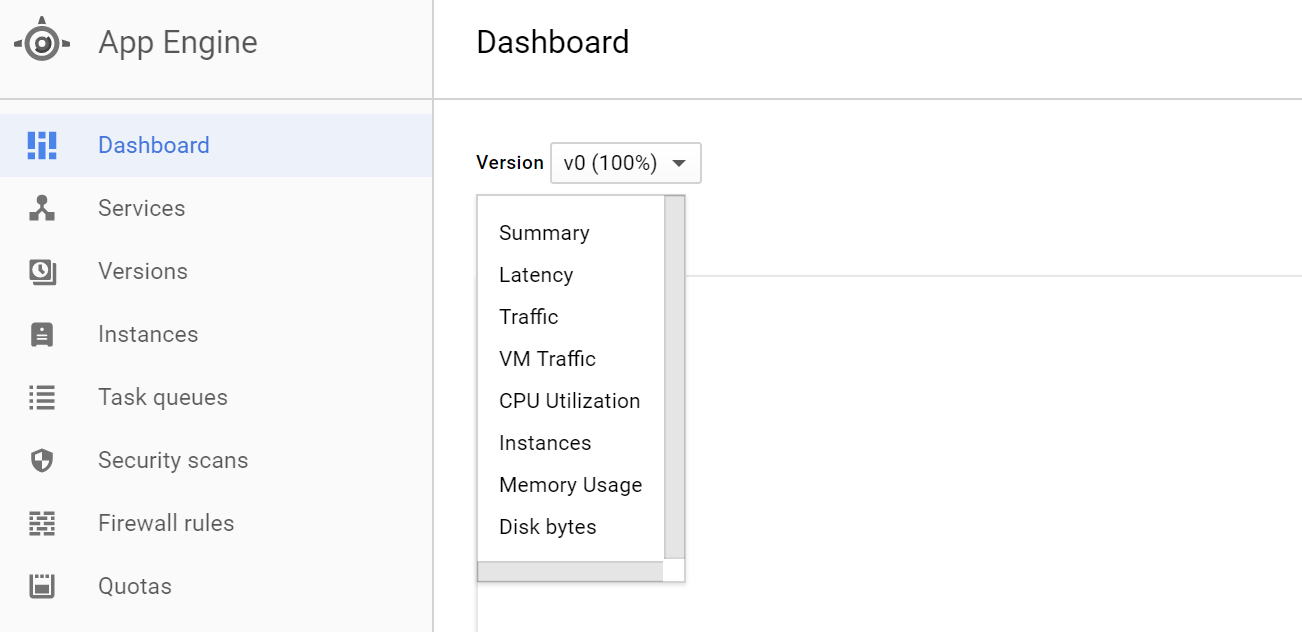
gcloud app deploy --version v1

Once it's deployed, you can go to App Engine versions section of Google Cloud Console to see the new version of your app serving all of the traffic with the new message.

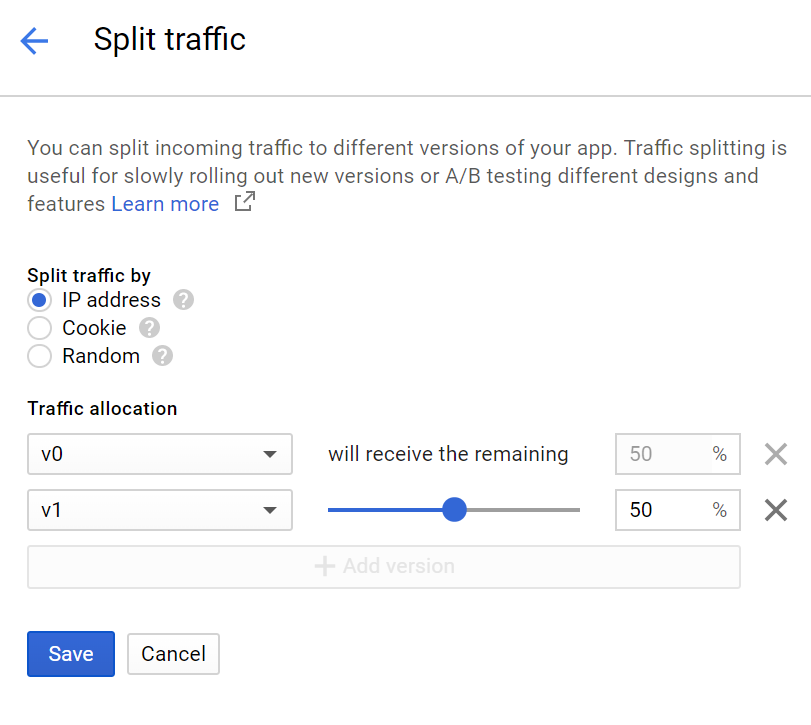


## [9. Dashboards and Traffic Splitting](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#8)

In App Engine, under Dashboard section, you can see a number of dashboards for your application for latency, CPU etc. Explore them on your own.



Under Versions section, you can see the deployed versions of your app and you can split traffic between different versions in Traffic Splitting section. Let's split the traffic between two versions:

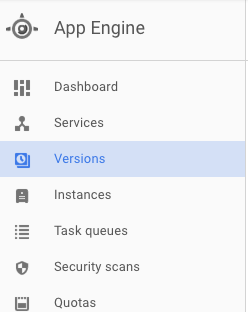


[10. Congratulations!](https://codelabs.developers.google.com/codelabs/cloud-app-engine-aspnetcore#9)

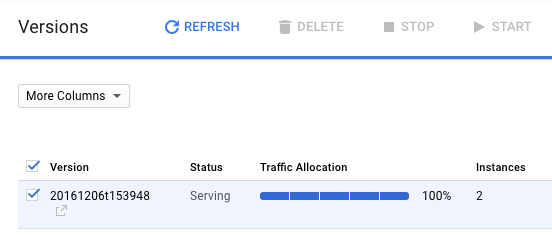
C**leanup**

It's time to shutdown the app to save on cost and to be an overall good cloud citizen.

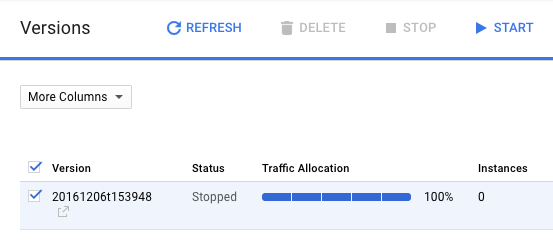
Go to the versions section of App Engine.



Select the version and stop it.



Once the version is stopped, the backing instances will be deleted and you should see instance count to drop down to zero.



**What we've covered**

There! You've created an ASP.NET Core app, packaged it as a Docker container, and deployed it to Google App Engine Flexible.

* How to package a simple ASP.NET Core app as a Docker container.
* How to deploy a simple ASP.NET Core app to App Engine.

Next Steps

* Learn more about [Windows on Google Cloud Platform](https://cloud.google.com/windows).
* Learn more about [.NET on Google Cloud Platform](https://cloud.google.com/dotnet/).
* Learn more about [SQL Server on Google Cloud Platform](https://cloud.google.com/sql-server/).
* Learn more about [Cloud Tools for Visual Studio](https://cloud.google.com/visual-studio).
* Learn more about [Cloud Tools for PowerShell](https://cloud.google.com/powershell/).

License

This work is licensed under a Creative Commons Attribution 2.0 Generic License.