Create a .NET Core web app in App Service on Linux

Note

This article deploys an app to App Service on Linux. To deploy to App Service on *Windows*, see [Create an ASP.NET Core web app in Azure](https://docs.microsoft.com/en-in/azure/app-service/app-service-web-get-started-dotnet).

[App Service on Linux](https://docs.microsoft.com/en-in/azure/app-service/containers/app-service-linux-intro) provides a highly scalable, self-patching web hosting service using the Linux operating system. This quickstart shows how to create a [.NET Core](https://docs.microsoft.com/aspnet/core/) app on App Service on Linux. You create the web app using the [Azure CLI](https://docs.microsoft.com/cli/azure/get-started-with-azure-cli), and you use Git to deploy the .NET Core code to the web app.

You can follow the steps below using a Mac, Windows, or Linux machine.

Prerequisites

To complete this quickstart:

* [Install Git](https://git-scm.com/)
* [Install the .NET Core SDK](https://www.microsoft.com/net/download/core)

If you don't have an Azure subscription, create a [free account](https://azure.microsoft.com/free/?ref=microsoft.com&utm_source=microsoft.com&utm_medium=docs&utm_campaign=visualstudio) before you begin.

Create the app locally

In a terminal window on your machine, create a directory named hellodotnetcore and change the current directory to it.

bashCopy

md hellodotnetcore

cd hellodotnetcore

Create a new .NET Core web app.

bashCopy

dotnet new web

Run the app locally

Restore the NuGet packages and run the app.

bashCopy

dotnet restore

dotnet run

Open a web browser, and navigate to the app at http://localhost:5000.

You see the **Hello World** message from the sample app displayed in the page.

In your terminal window, press **Ctrl+C** to exit the web server. Initialize a Git repository for the .NET Core project.

bashCopy

git init

git add .

git commit -m "first commit"

Launch Azure Cloud Shell

The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are two ways to launch the Cloud Shell:

|  |  |
| --- | --- |
| Click **Try It** in the upper right corner of a code block. |  |
| Click the **Cloud Shell** button on the menu in the upper right of the [Azure portal](https://portal.azure.com/). |  |
|  |  |

Create a deployment user

In the Cloud Shell, create deployment credentials with the [az webapp deployment user set](https://docs.microsoft.com/en-us/cli/azure/webapp/deployment/user?view=azure-cli-latest" \l "az_webapp_deployment_user_set)command. A deployment user is required for FTP and local Git deployment to a web app. The user name and password are account level. *They are different from your Azure subscription credentials.*

In the following example, replace *<username>* and *<password>* (including brackets) with a new user name and password. The user name must be unique. The password must be at least eight characters long, with two of the following three elements: letters, numbers, symbols.

Azure CLICopyTry It

az webapp deployment user set --user-name <username> --password <password>

If you get a 'Conflict'. Details: 409 error, change the username. If you get a 'Bad Request'. Details: 400 error, use a stronger password.

You create this deployment user only once; you can use it for all your Azure deployments.

Note

Record the user name and password. You need them to deploy the web app later.

Create a resource group

In the Cloud Shell, create a resource group with the [az group create](https://docs.microsoft.com/en-us/cli/azure/group?view=azure-cli-latest" \l "az_group_create) command.

A [resource group](https://docs.microsoft.com/en-in/azure/azure-resource-manager/resource-group-overview#terminology) is a logical container into which Azure resources like web apps, databases, and storage accounts are deployed and managed.

The following example creates a resource group named *myResourceGroup* in the *West Europe*location. To see all supported locations for App Service, run the az appservice list-locationscommand.

Azure CLICopyTry It

az group create --name myResourceGroup --location "West Europe"

You generally create your resource group and the resources in a region near you.

Create an Azure App Service plan

In the Cloud Shell, create an App Service plan in the resource group with the [az appservice plan create](https://docs.microsoft.com/en-us/cli/azure/appservice/plan?view=azure-cli-latest" \l "az_appservice_plan_create) command.

The following example creates an App Service plan named myAppServicePlan in the **Standard**pricing tier (--sku S1) and in a Linux container (--is-linux).

Azure CLICopyTry It

az appservice plan create --name myAppServicePlan --resource-group myResourceGroup --sku S1 --is-linux

When the App Service plan has been created, the Azure CLI shows information similar to the following example:

JSONCopy

{

"adminSiteName": null,

"appServicePlanName": "myAppServicePlan",

"geoRegion": "West Europe",

"hostingEnvironmentProfile": null,

"id": "/subscriptions/0000-0000/resourceGroups/myResourceGroup/providers/Microsoft.Web/serverfarms/myAppServicePlan",

"kind": "linux",

"location": "West Europe",

"maximumNumberOfWorkers": 1,

"name": "myAppServicePlan",

< JSON data removed for brevity. >

"targetWorkerSizeId": 0,

"type": "Microsoft.Web/serverfarms",

"workerTierName": null

}

Create a web app

In the Cloud Shell, create a [web app](https://docs.microsoft.com/en-in/azure/app-service/containers/app-service-linux-intro) in the myAppServicePlan App Service plan with the [az webapp create](https://docs.microsoft.com/en-us/cli/azure/webapp?view=azure-cli-latest" \l "az_webapp_create) command.

In the following example, replace <app\_name> with a globally unique app name (valid characters are a-z, 0-9, and -). The runtime is set to dotnetcore|1.1. To see all supported runtimes, run [az webapp list-runtimes](https://docs.microsoft.com/en-us/cli/azure/webapp?view=azure-cli-latest" \l "az_webapp_list_runtimes).

Azure CLICopyTry It

az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name <app\_name> --runtime "dotnetcore|1.1" --deployment-local-git

When the web app has been created, the Azure CLI shows output similar to the following example:

JSONCopy

Local git is configured with url of 'https://<username>@<app\_name>.scm.azurewebsites.net/<app\_name>.git'

{

"availabilityState": "Normal",

"clientAffinityEnabled": true,

"clientCertEnabled": false,

"cloningInfo": null,

"containerSize": 0,

"dailyMemoryTimeQuota": 0,

"defaultHostName": "<app\_name>.azurewebsites.net",

"deploymentLocalGitUrl": "https://<username>@<app\_name>.scm.azurewebsites.net/<app\_name>.git",

"enabled": true,

< JSON data removed for brevity. >

}

You’ve created an empty web app in a Linux container, with git deployment enabled.

Note

The URL of the Git remote is shown in the deploymentLocalGitUrl property, with the format https://<username>@<app\_name>.scm.azurewebsites.net/<app\_name>.git. Save this URL as you'll need it later.

Browse to your newly created web app. Replace *<app name>* with your web app name.

bashCopy

http://<app name>.azurewebsites.net

Push to Azure from Git

In the local terminal window, add an Azure remote to your local Git repository. This Azure remote was created for you in [Create a web app](https://docs.microsoft.com/en-in/azure/app-service/containers/quickstart-dotnetcore#create-a-web-app).

bashCopy

git remote add azure <deploymentLocalGitUrl-from-create-step>

Push to the Azure remote to deploy your app with the following command. When prompted for a password, make sure that you enter the password you created in [Configure a deployment user](https://docs.microsoft.com/en-in/azure/app-service/containers/quickstart-dotnetcore#configure-a-deployment-user), not the password you use to log in to the Azure portal.

bashCopy

git push azure master

This command may take a few minutes to run. While running, it displays information similar to the following example:

bashCopy

Counting objects: 22, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (18/18), done.

Writing objects: 100% (22/22), 51.21 KiB | 3.94 MiB/s, done.

Total 22 (delta 1), reused 0 (delta 0)

remote: Updating branch 'master'.

remote: Updating submodules.

remote: Preparing deployment for commit id '741f16d1db'.

remote: Generating deployment script.

remote: Project file path: ./hellodotnetcore.csproj

remote: Generated deployment script files

remote: Running deployment command...

remote: Handling ASP.NET Core Web Application deployment.

remote: ...............................................................................................

remote: Restoring packages for /home/site/repository/hellodotnetcore.csproj...

remote: ....................................

remote: Installing System.Xml.XPath 4.0.1.

remote: Installing System.Diagnostics.Tracing 4.1.0.

remote: Installing System.Threading.Tasks.Extensions 4.0.0.

remote: Installing System.Reflection.Emit.ILGeneration 4.0.1.

remote: ...

remote: Finished successfully.

remote: Running post deployment command(s)...

remote: Deployment successful.

To https://cephalin-dotnetcore.scm.azurewebsites.net/cephalin-dotnetcore.git

\* [new branch] master -> master

Browse to the app

Browse to the deployed application using your web browser.

bashCopy

http://<app\_name>.azurewebsites.net

The Node.js sample code is running in an web app with built-in image.1

**Congratulations!** You've deployed your first Node.js app to App Service on Linux.1

Update and redeploy the code

In the local directory, open the *Startup.cs* file. Make a small change to the text in the method call context.Response.WriteAsync:

C#Copy

await context.Response.WriteAsync("Hello Azure!");

Commit your changes in Git, and then push the code changes to Azure.

bashCopy

git commit -am "updated output"

git push azure master

Once deployment has completed, switch back to the browser window that opened in the **Browse to the app** step, and hit refresh.

Manage your new Azure web app

Go to the [Azure portal](https://portal.azure.com/) to manage the web app you created.

From the left menu, click **App Services**, and then click the name of your Azure web app.

You see your web app's Overview page. Here, you can perform basic management tasks like browse, stop, start, restart, and delete.

The left menu provides different pages for configuring your app.

Clean up resources

In the preceding steps, you created Azure resources in a resource group. If you don't expect to need these resources in the future, delete the resource group by running the following command in the Cloud Shell:

Azure CLICopyTry It

az group delete --name myResourceGroup

This command may take a minute to run.

Next steps