Deploy an ASP.NET COre app TO Windows containers using ECS

# Introduction

## Overview

With the ever-growing complexity of modern workloads, the mutual challenges of application scaling and deployment have been greatly simplified by the introduction of [Docker](https://www.docker.com/), an open source platform that provides a standard approach to application containerization.

By allowing applications to be packaged and deployed alongside their dependencies, while keeping them isolated from other applications, containers have not only become a standard deployment approach for new applications—and, in particular, applications based on micro-services—but they’ve also become popular mechanisms for packaging and isolating legacy applications.

[Amazon Elastic Container Service](https://aws.amazon.com/ecs/) (ECS) is a container orchestration services that simplifies running and scaling containerized applications on AWS.

This article demonstrates how to use ECS to build and deploy a default ASP.NET Core website using Visual Studio on Windows.

## Modules

1. Setup Development Environment
2. Configure ECS cluster
3. Create ASP.NET Core web application
4. Deploy Application to ECS

## Side-bar

* AWS Experience—Beginner
* Time to Complete—35 mins
* Cost to Complete—Each service used in this architecture is eligible for the AWS Free Tier. If you are outside the usage limits of the Free Tier, completing this learning path will cost you less than $0.25\*.
* Tutorial Prereqs—To complete this learning path, you will need:  
  ✓ An AWS Account\*\*

✓ An IAM user with access key credentials\*\*\*  
✓ A Docker Account\*\*\*\*  
✓ Visual Studio 2017 for Windows

\*This estimate assumes you follow the recommended configurations throughout the tutorial and terminate all resources within 24 hours.

\*\*Accounts that have been created within the last 24 hours might not yet have access to the resources required for this learning. If you don’t have an account visit <https://aws.amazon.com> and click *Sign Up*.

\*\*\* You must have a set of valid AWS credentials, consisting of an access key and a secret key, which are used to sign programmatic requests to AWS. You can obtain a set of account credentials when you create your account, although we recommend you do not use these credentials and instead [create an IAM user](http://docs.aws.amazon.com/IAM/latest/UserGuide/Using_SettingUpUser.html) and use those credentials.  
\*\*\*\* If you don’t have a Docker account, visit <https://www.docker.com> and click *Create Docker ID*.

# Module 1: Setup development environment

In this module you'll configure your development environment for working with .NET Core and Docker.

These instructions provide options for working with Visual Studio 2017 for Windows.

## Implementation Instructions

Follow the step-by-step instructions below to setup your development environment. (To expand the section, click on each step number).

### Step 1: Setup Visual Studio 2017 for Windows

First of all, you will need to ensure the following components are installed:

1. The **.NET Core SDK 2.x** for Windows:   
   <https://www.microsoft.com/net/download/>
2. Visual Studio 2017 version 15.3 or later
3. The AWS Toolkit for Visual Studio:  
   <https://aws.amazon.com/visualstudio/>

### Step 2: Setup Docker on Windows

You will then need to ensure Docker for Windows is installed:

1. Download [Docker for Windows](https://store.docker.com/editions/community/docker-ce-desktop-windows) and install the latest stable version of Docker.
2. Run the installer and login with your Docker credentials.
3. Once installation has finished you can verify the installation by opening a command prompt and running the following command:  
   docker -v

### Side-bar

* Time to Complete—15 mins

# Module 2: Configure ECS Cluster

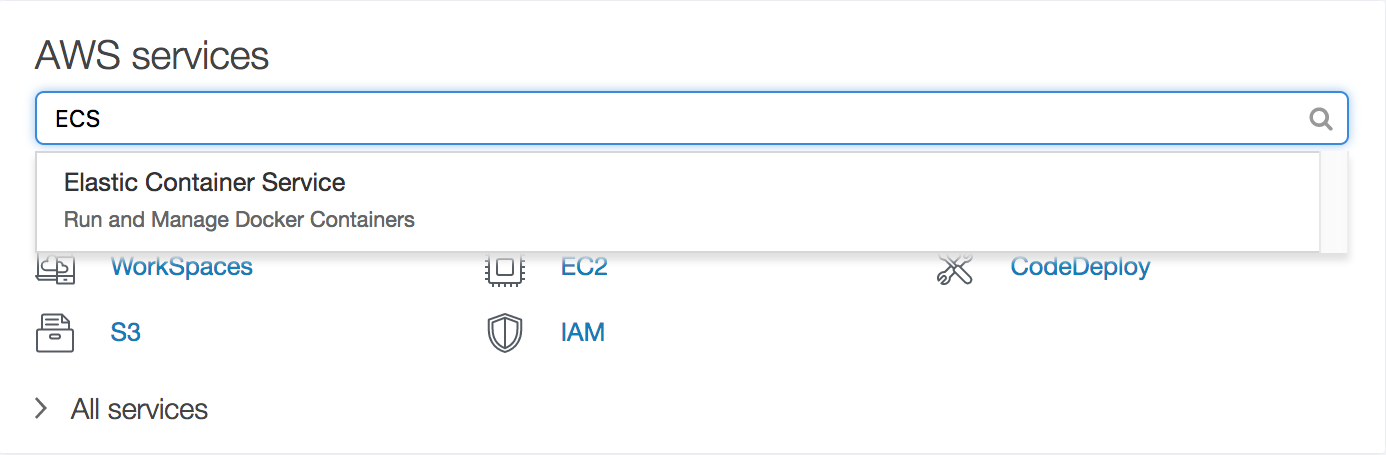
In this module we will configure a cluster for hosting our application using AWS Elastic Container Service (ECS) cluster.

## Implementation Instructions

Follow the step-by-step instructions below to setup the cluster. (To expand the section, click on each step number)

### Step 1: Create ECS Cluster

1. Open the *AWS Management Console* in a web browser and search for ECS.



1. On the ECS page, navigate to *Clusters* and then click *Create Cluster* to start the Cluster Creation Wizard.
2. Under *Select cluster template,* choose *EC2 Windows + Networking*, and then click *Next*.
3. Under *Configure cluster* enter the following field values:
   1. *Cluster name*: TestWindowsCluster
4. Under *Instance configuration* enter the following field values:
   1. *Provisioning model*: On-Demand Instance
   2. *EC2 instance type*: t2.medium
   3. *Number of instances:* 1
   4. *EBS storage (GiB)*: 50
   5. *Key pair:* None - unable to SSH
5. Leave the default settings under *Networking* and *Container instance IAM role*, and then click *Next*.

Once the services are ready, all the tasks in the AWS Management Console window will turn green.

Now you’ve successfully created an ECS cluster, it’s time to create a basic ASP.NET Core application to host in the cluster.

### Side-bar

* Time to Complete—5 mins

# Module 3: Create ASP.NET COre Application

In this module we will create a default ASP.NET Core MVC web application using the Microsoft-supplied project template.

## Implementation Instructions

Follow the step-by-step instructions below to create an ASP.NET Core application. (To expand the section, click on each step number).

### Step 1: Create Application in Visual Studio 2017 for Windows

If your development environment is Visual Studio 2017 on Windows, you can create an ASP.NET Core application as follows:

1. Open Visual Studio and then create a new project by navigating to the *File* > *New* > *Project* menu item.
2. In the *New Project* dialog, select *Visual C#* > *.NET Core* > *ASP.NET Core Web Application* template.
3. Enter *TestAspNetCoreApp* in the *Name* field, and then click *Ok*.
4. Chose the *Web Application (Model-View-Controller)* template, and then ensure *Enable Docker Support* is checked, and the *OS* drop-down is set to *Windows*. Click *Ok* to complete the wizard.

The wizard creates all the files and configuration necessary for hosting an ASP.NET Core Web application in a container, although before deploying the solution, you’ll need to edit the *Dockerfile* file to ensure it targets the same version of Windows Server used by the cluster.

The Cluster Creation Wizard currently uses Windows Server 2016, so you’ll need to locate and open the *Dockerfile* in Visual Studio and then update all references to *nanoserver-1709* to *nanoserver-sac2016*, as shown below:

FROM microsoft/aspnetcore:2.0-nanoserver-**sac2016** AS base  
WORKDIR /app  
EXPOSE 80

FROM microsoft/aspnetcore-build:2.0-nanoserver-**sac2016** AS build  
WORKDIR /src  
COPY TestAspNetCoreApp.sln ./  
COPY TestAspNetCoreApp/TestAspNetCoreApp.csproj TestAspNetCoreApp/  
RUN dotnet restore -nowarn:msb3202,nu1503  
COPY . .  
WORKDIR /src/TestAspNetCoreApp  
RUN dotnet build -c Release -o /app

FROM build AS publish  
RUN dotnet publish -c Release -o /app

FROM base AS final  
WORKDIR /app  
COPY --from=publish /app .  
ENTRYPOINT ["dotnet", "TestAspNetCoreApp.dll"]

You can now build the solution by going to the *Build* menu and clicking on *Build Solution*. Once the build has succeeded you are ready to deploy the solution to ECS.

### Side-bar

* Time to Complete—10 mins

# Module 4: Deploy application to ECS

In this module we will deploy the ASP.NET Core application to ECS.

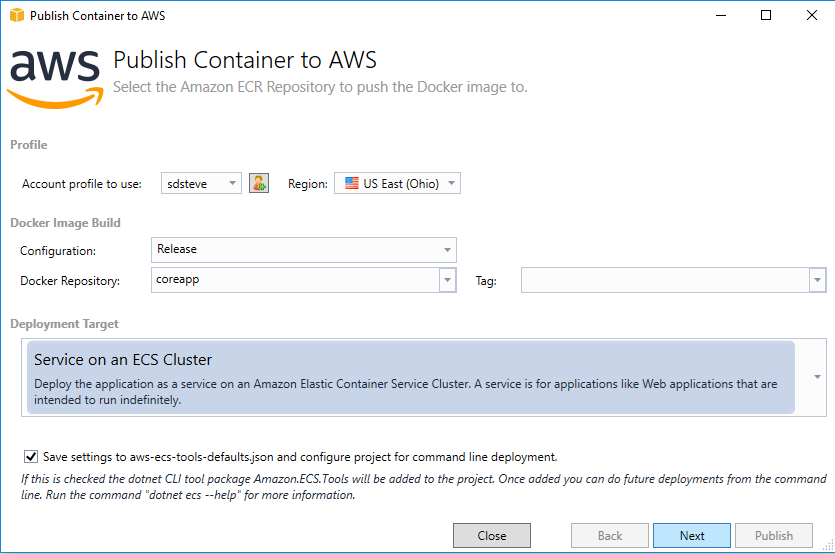
## Implementation Instructions

Follow the step-by-step instructions below to deploy the ASP.NET Core application. (To expand the section, click on each step number).

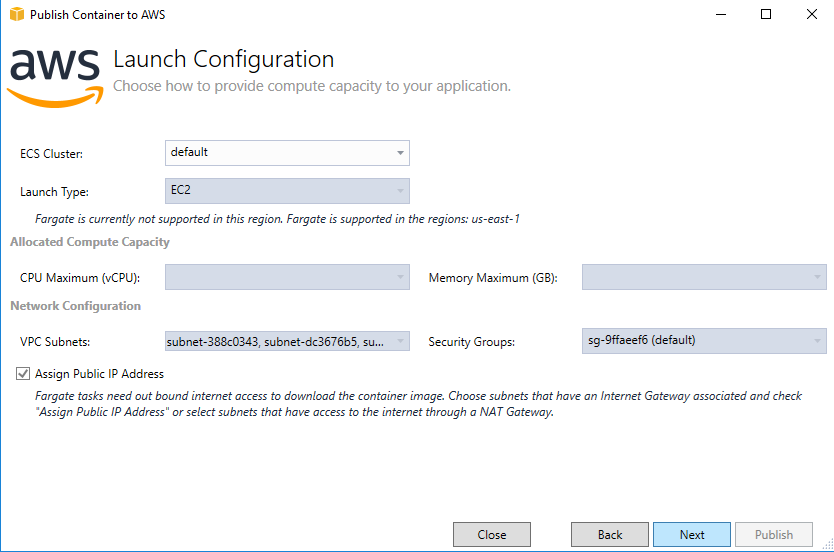
### Step 1: Deploy using Visual Studio 2017 for Windows

If your development environment is Visual Studio 2017 on Windows, you can deploy the ASP.NET Core application as follows:

1. Right click on the *TestAspNetCoreApp* project within the *Solution Explorer* window and then select *Publish Container to AWS…*
2. The *Publish Container* wizard will start. You can leave the settings on the default values, but ensure the *Deployment Target* is set to *Service on an ECS Cluster*, and then click *Next*.



1. On the *Launch Configuration* dialog change the *Launch Type* to EC2, and then click *Next*.



1. Under *Service Configuration* you can create a new ECS Service, by leaving the default settings and clicking *Next*.
2. Leave the default settings on the *Application Load Balancer Configuration* dialog and click *Next*.
3. On the *Task Definition* dialog, click on *Create New,* set the *Hard Limit* to 500, under *Port Mapping* update the *Host Port* to 8080, and then click *Publish*.

Publishing the ASP.NET Core application will take a couple of minutes to complete, and once it finishes the application will be almost be ready for testing.

However, before testing the application you will need to update the service’s task definition to use the *NAT* network mode, since Windows containers don’t currently support *bridge* networking.

To update the task defintion’s network mode, carry out the following steps:

1. In AWS Management Console, navigate to ECS *Task Definitions*, select the checkbox next to the *TestAspNetCoreApp* task and then click on *Create new revision*.
2. Under *Network Mode* update the value to *<default*>, which uses NAT on Windows, and then click *Create*.

Finally, update the service to use the new task definition.

1. Select *Clusters* in the AWS Management Console’s *ECS* screen.
2. Click on *TestWindowsCluster* to open the *Cluster definition* screen.
3. Under the *Services* tab click on *TestApNetCoreApp* to open the *Service definition* screen.
4. Click *Update* to edit the service definition.
5. Under *Task Definition*, update the *Revision* to *2 (latest).*
6. Click *Next step*
7. Click *Next step* on the *Configure network* screen
8. Click *Next step* on the *Auto Scaling* screen
9. Click *Update Service* on the *Review* screen

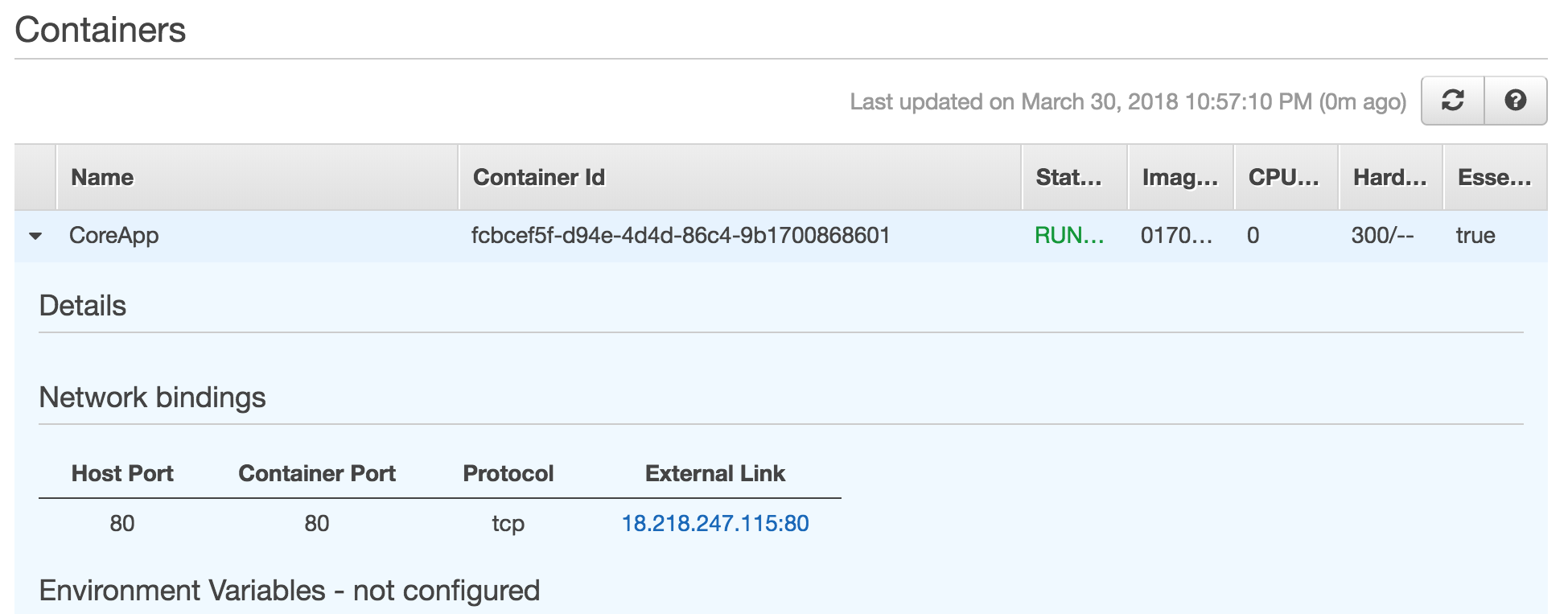
The ECS service will automatically create a task based on the new definition, and once the new task enters a *Running* state, the application is ready for testing.

### Step 2: test Deployment

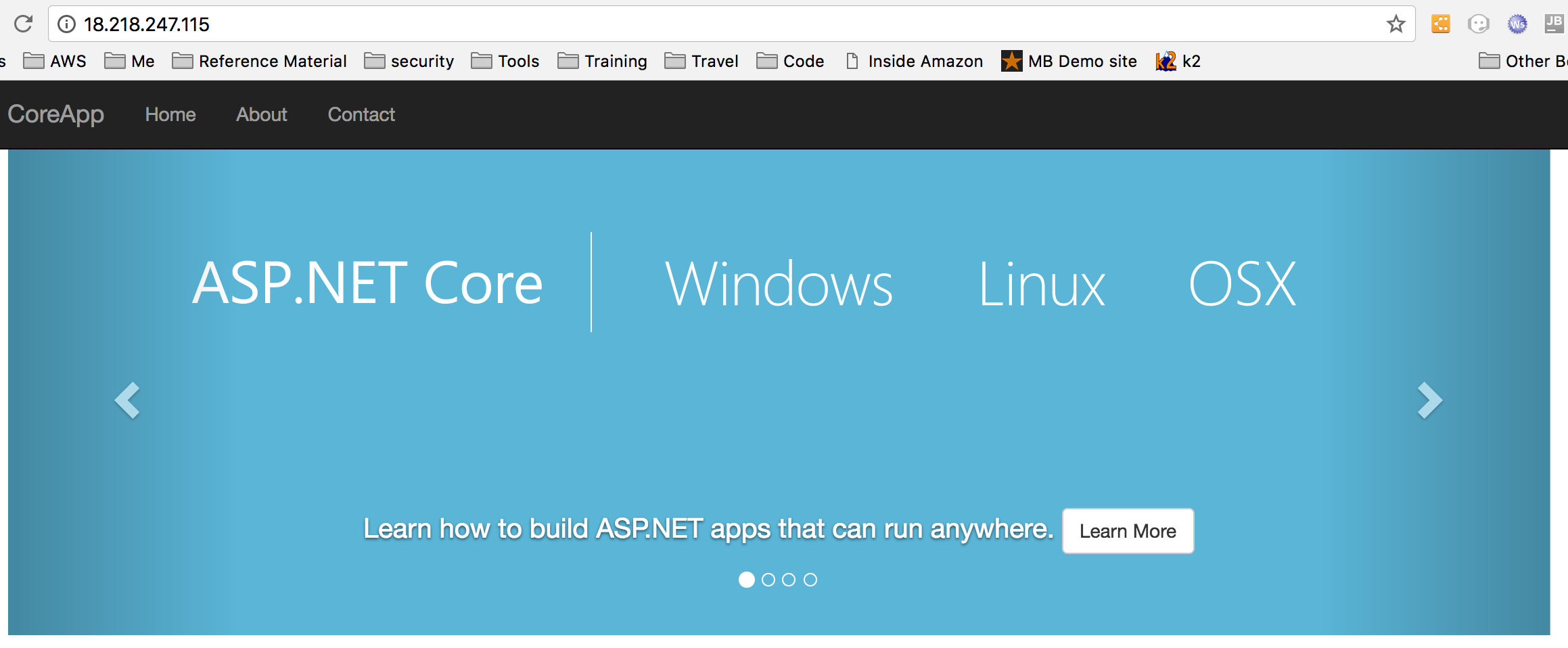
You are now ready to test the ASP.NET Core application in your browser.

To view the application, you’ll need to retrieve the ECS service’s public IP address from the AWS Management Console, which you can do by opening the *Task* screen which you’ll find by clicking the task name under the *Tasks* tab of the *Service definition* screen.

The container’s public IP address is then listed in the *Network bindings* section listed under *Containers*.



Click on the URL under the *External Link* section, and the ASP.NET Core site will load in your browser.



Congratulations, you have now deployed an ASP.NET Core application to ECS.

### Side-bar

* Time to Complete—5 mins