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 operating system level containerization.

Allowing applications to be packaged and deployed with their dependencies while keeping them isolated from other applications, containers have not only become the standard for new applications, but they’ve also become popular 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, or the .NET Core CLI for Windows, Mac OS X, or Linux.

## 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—30 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\*\*\*\*  
✓ (Optionally) 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 various development environments, including: Visual Studio 2017 for Windows, or the .NET Core CLI for Windows, Mac, or Linux.

## 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

If your development environment is Visual Studio 2017 on Windows, 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 1: Setup .NET Core CLI on Windows, Mac, or Linux

If you are using .NET Core CLI on Windows, Mac, or Linux, you will need to install a few components, as follows:

1. The **.NET Core SDK 2.x** for Windows, Mac, or Linux:   
   <https://www.microsoft.com/net/download/>
2. Install the AWS Lambda templates with the AWS Lambda NuGet package by running dotnet new -i Amazon.Lambda.Templates::\* in a terminal window.
3. Verify the new AWS Lambda templates have been installed by running dotnet new lambda.EmptyFunction -l in a terminal window. If the command returns details of a single Lambda Empty Function template then the templates have been installed correctly.

You will also need the AWS Command Line Interface (AWS CLI) installed:

The AWS CLI for Windows, Mac, or Linux: <https://aws.amazon.com/cli/>

1. Once installed, you can configure the CLI by running the aws configure command in a terminal or command-line window.
2. When prompted, enter your AWS Access Key ID and press Return.
3. You will then be prompted for your AWS Secret Access Key, which you should enter and then press Return.
4. For the default region name you should enter your chosen region code (e.g. eu-west-1)
5. Finally, for the default output format you can just press Return.

Finally, you will also need a text editor or an IDE for modifying, such as vi, emacs, nano, [Visual Studio for Mac](https://www.microsoft.com/net/download/), or [Visual Studio Code](https://code.visualstudio.com/) for Windows, Mac, or Linux.

### Step 2: Setup Docker on Windows

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 to docker.
3. Once installation has finished you can verify the installation by opening a command prompt and running the following command:  
   docker -v

### Step 2: Setup Docker on Mac Os X

1. Download [Docker for Mac OS X](https://store.docker.com/editions/community/docker-ce-desktop-mac) and install the latest stable version of Docker.
2. Run the installer and login to docker.
3. Once installation has finished you can verify the installation by opening a terminal window and running the following command:  
   docker -v

### Step 2: Setup Docker on Linux

1. Download [Docker](https://www.docker.com/community-edition) for your Linux distribution and install the latest stable version of Docker.
2. Run the installer and login to docker.
3. Once installation has finished you can verify the installation by opening a terminal window and running the following command:  
   docker -v

### Side-bar

* Time to Complete—15 mins

# Module 2: Configure ECS Cluster

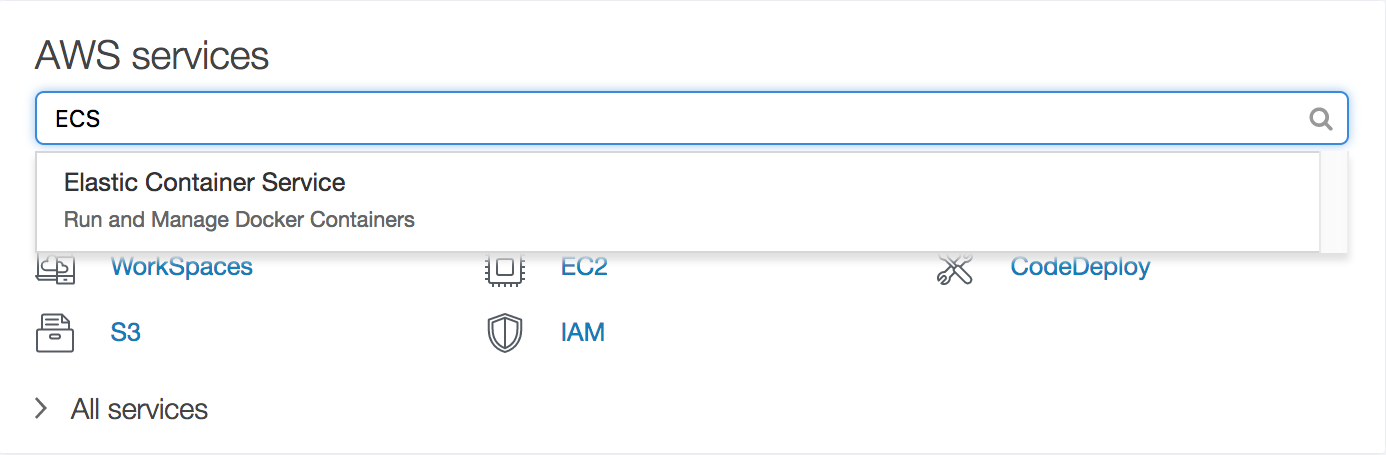
In this module we will configure a cluster for hosting our application using AWS Elastic Container Service (ECS) cluster.

## 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

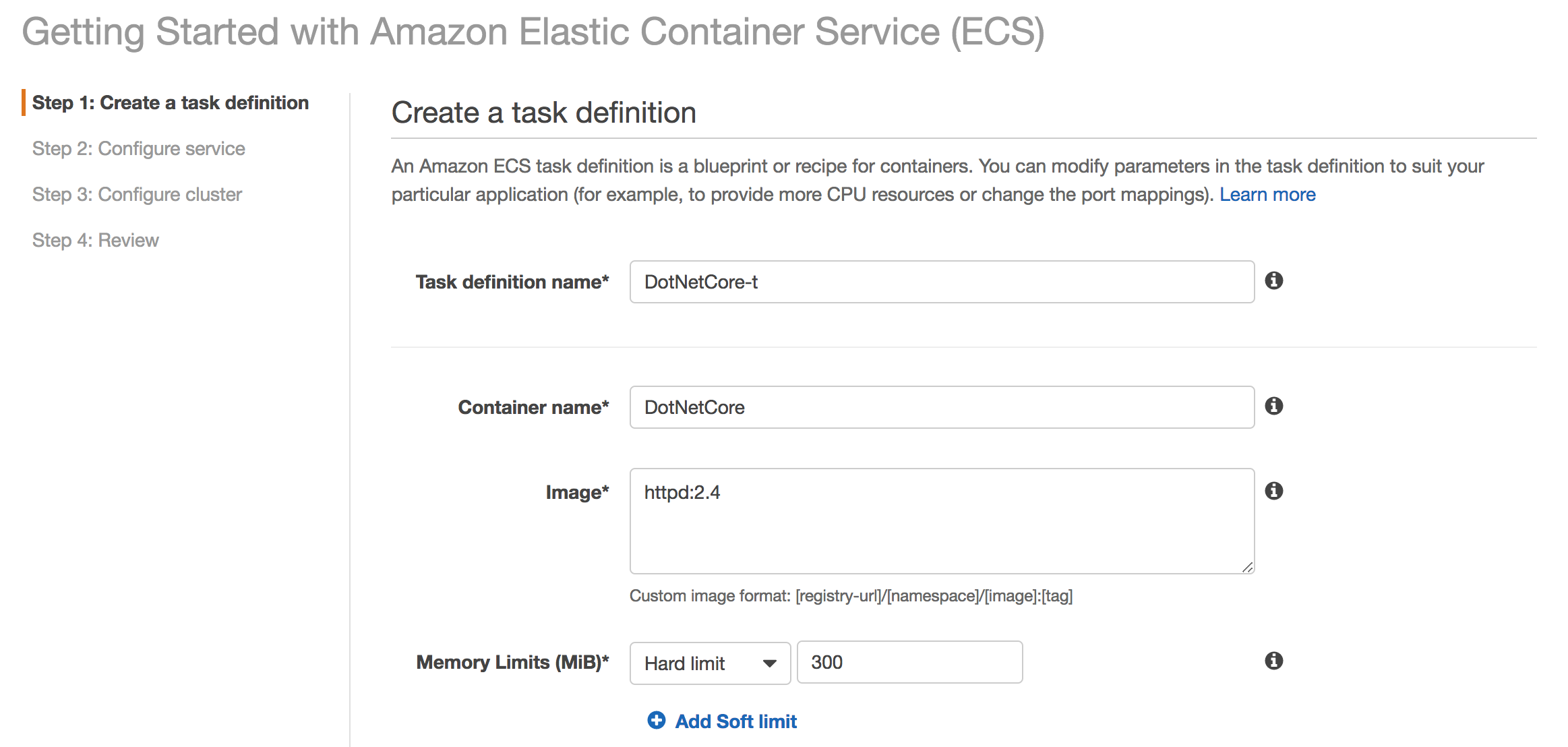
1. Open the AWS console and search for ECS.



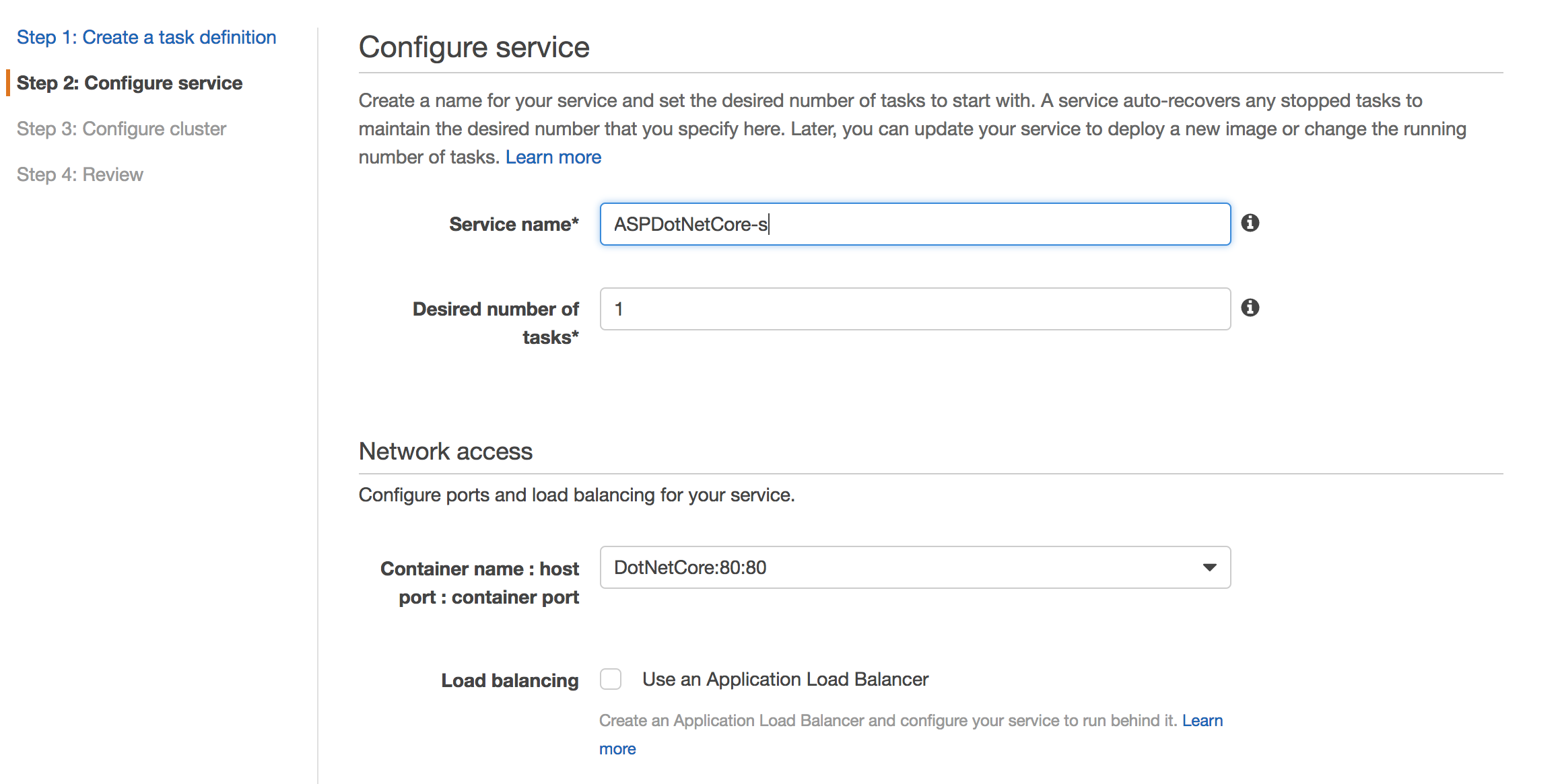
1. On the ECS page, click on *Get Started*.



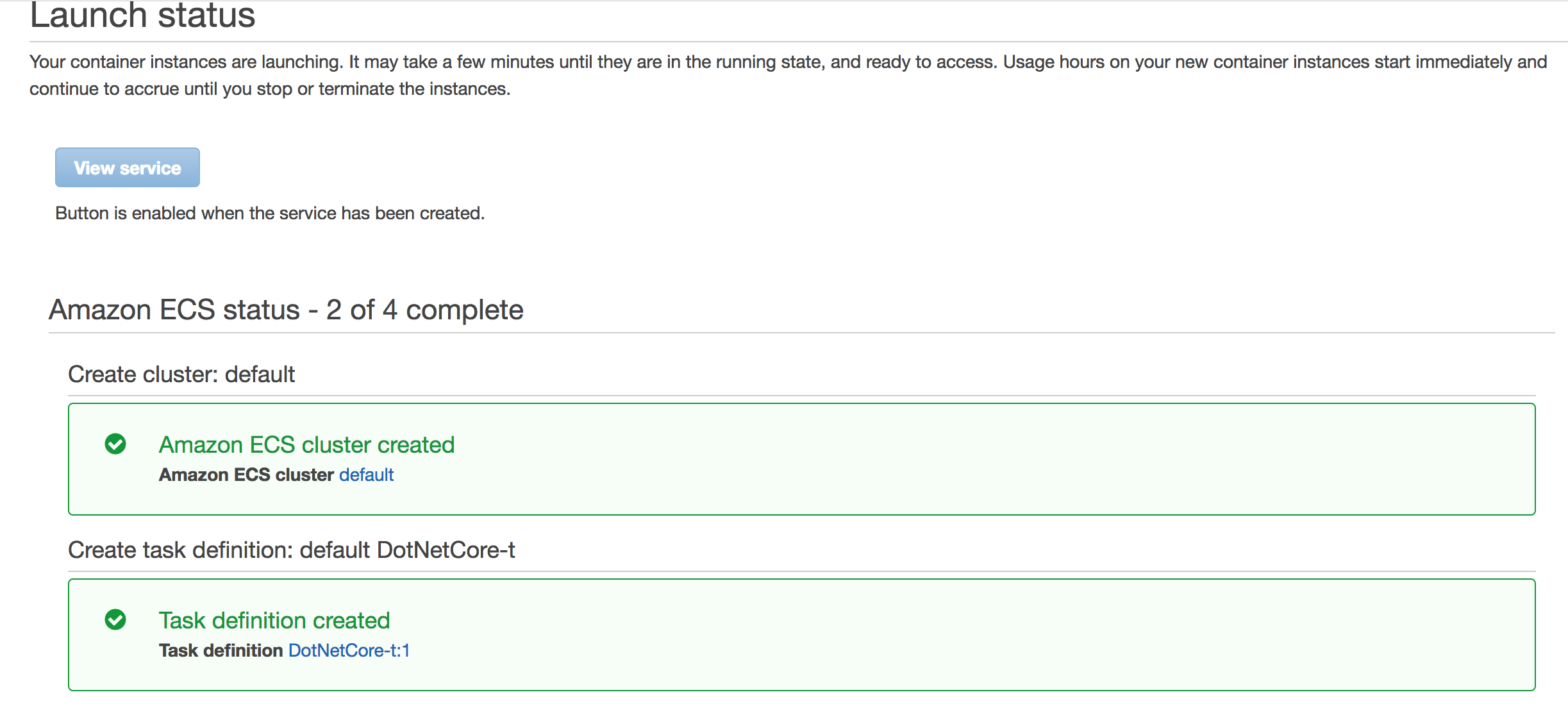
1. Under the *Container definition* section of the screen, choose *sample-app*, which selects the Docker image to use.
2. Create a *Task Definition* by clicking the *Edit* button under *Task definition*.
3. We only need to update two fields: *Task Definition Name* and *Container Name*. Once you’ve entered values for the fields, click *Next Step*.



1. Configure Service and click *Next*



1. On *Configure Cluster* keep all the default values and click *Review & Launch*.
2. Review the setting and click *Launch instance & run service*. Once the services are ready all the tasks will turn green.



### Side-bar

* Time to Complete—5 mins

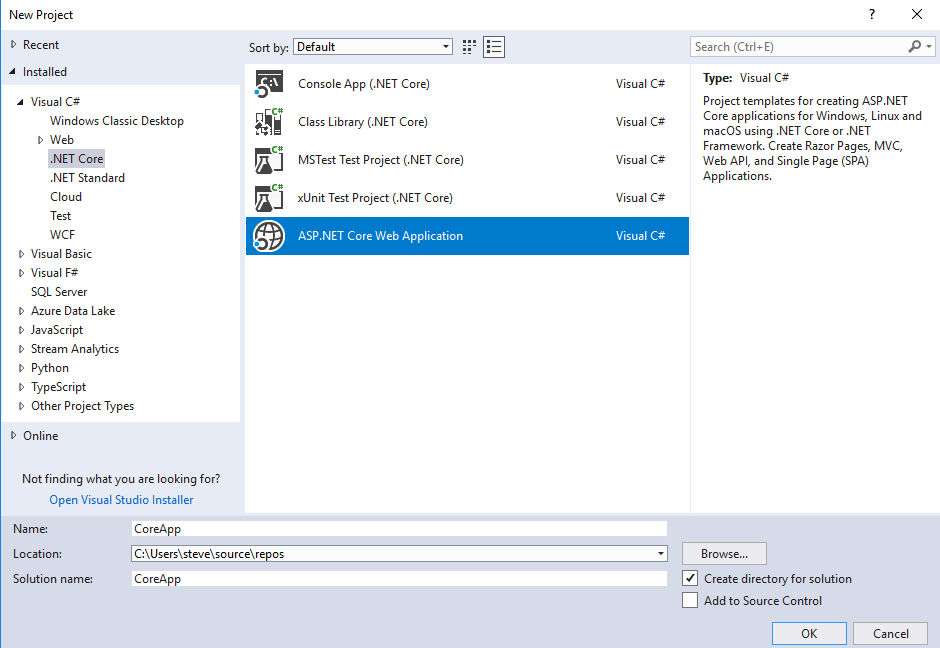
# Module 3: Create ASP.NET COre Application

In this module we will create the ASP.NET Core application in Visual Studio.

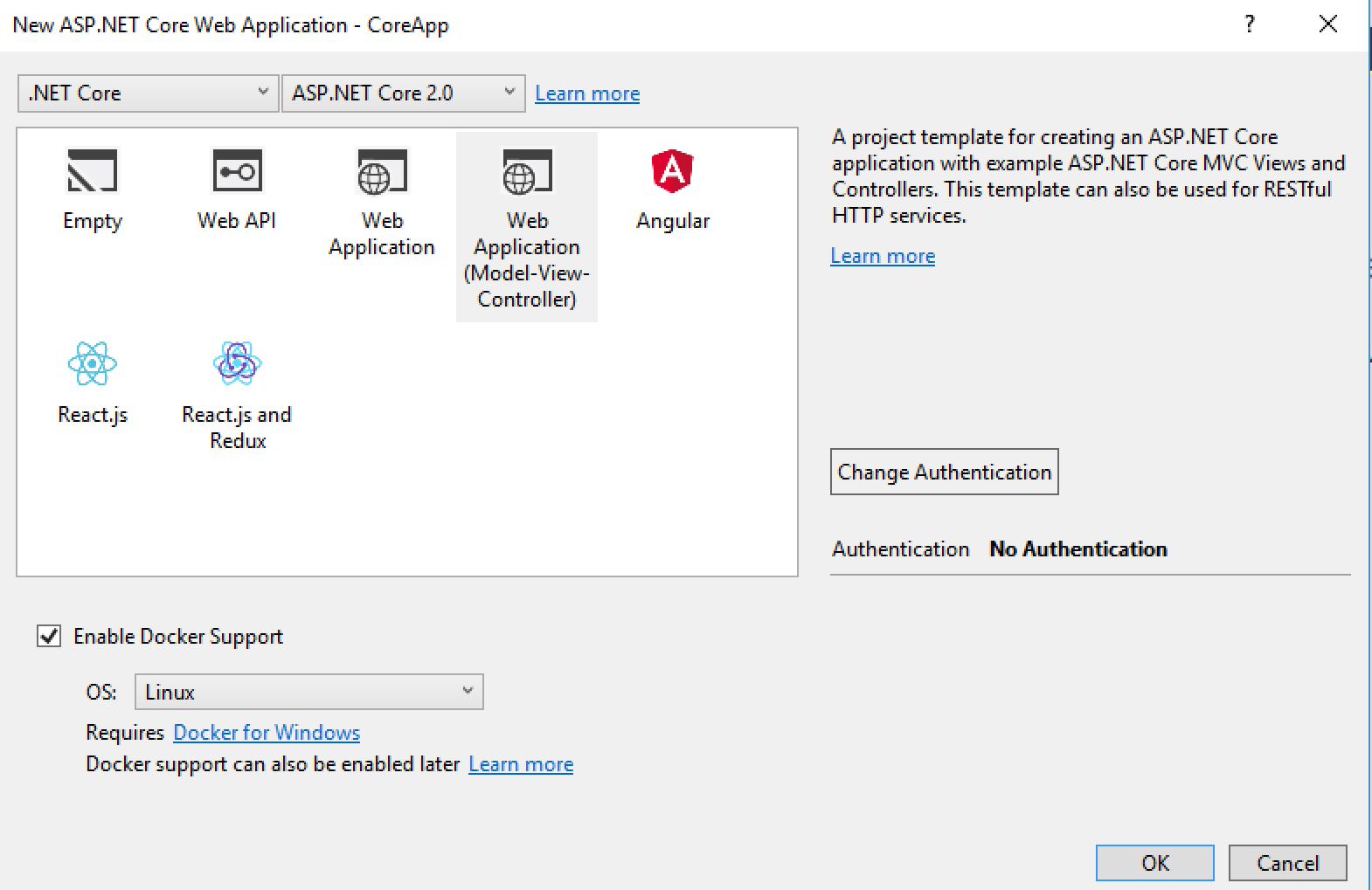
## Implementation Instructions

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

1. Open Visual Studio and create a new project
2. Select the *ASP.NET Core Web Application* template.



1. Chose the MVC version and enable Docker.



### Step 1: Create Application in .NET Core CLI for Windows, MAC OS X, or Linux

### Side-bar

* Time to Complete—5 mins

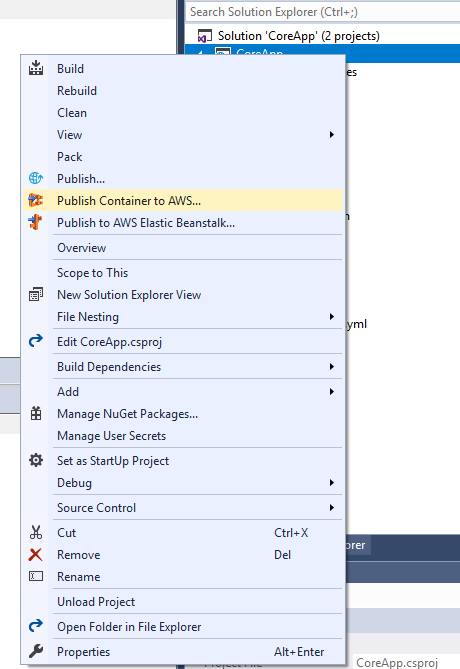
# Module 4: Deploy application to ECS

In this module we will use AWS toolkit for Visual Studio to deploy the ASP.NET Core application to ECS.

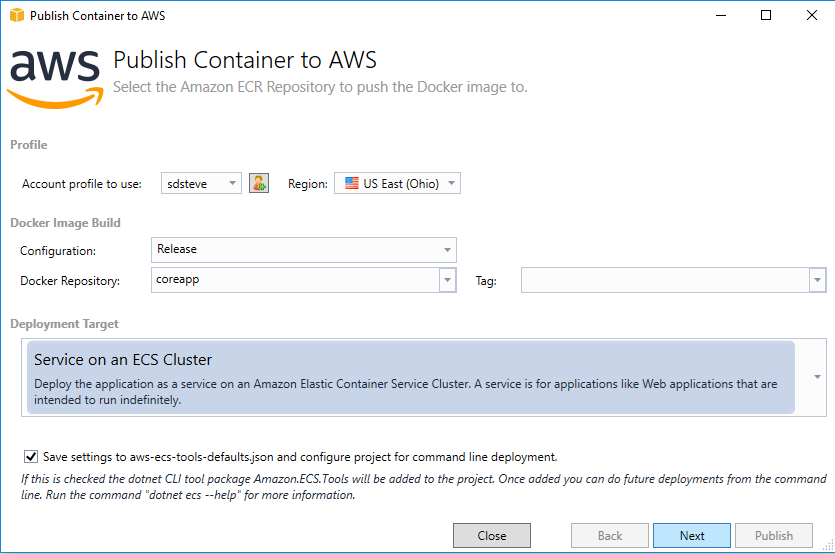
## Implementation Instructions

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

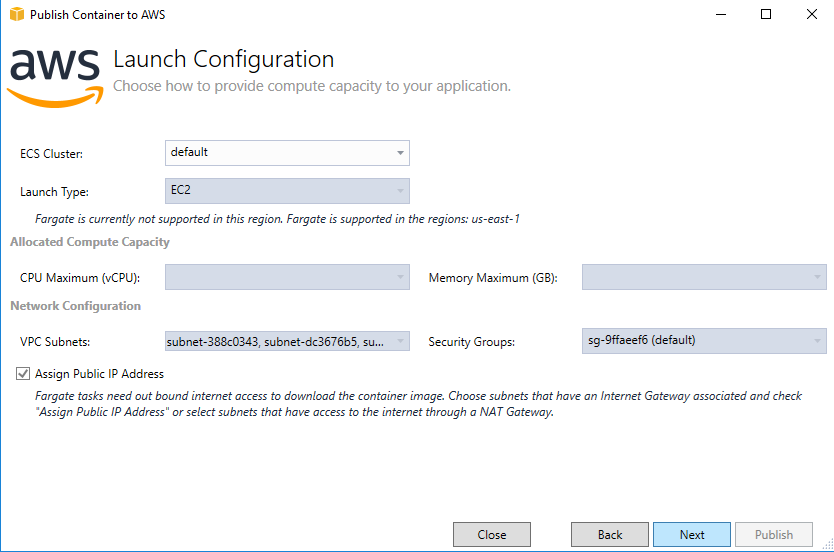
1. Right click on the project in the *Solution Explorer*



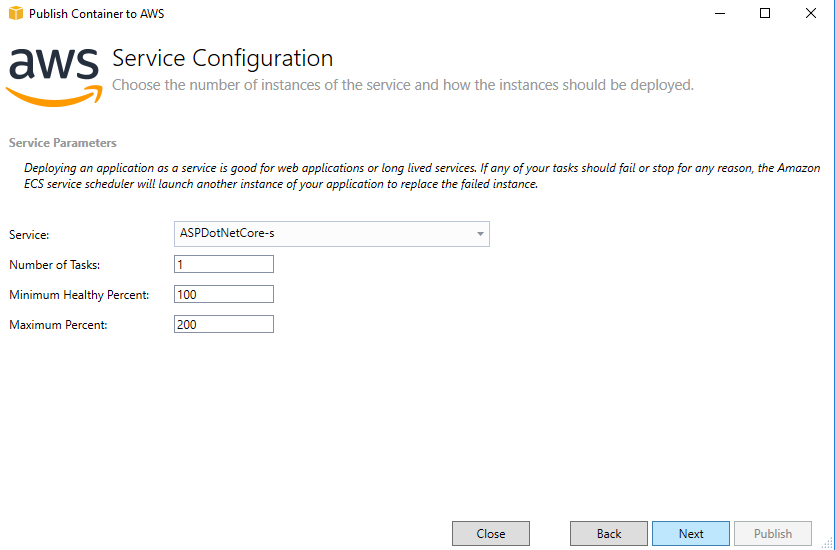
1. The *Publish Container* wizard will start. Chose *Publish to Container* options and click *Next*.



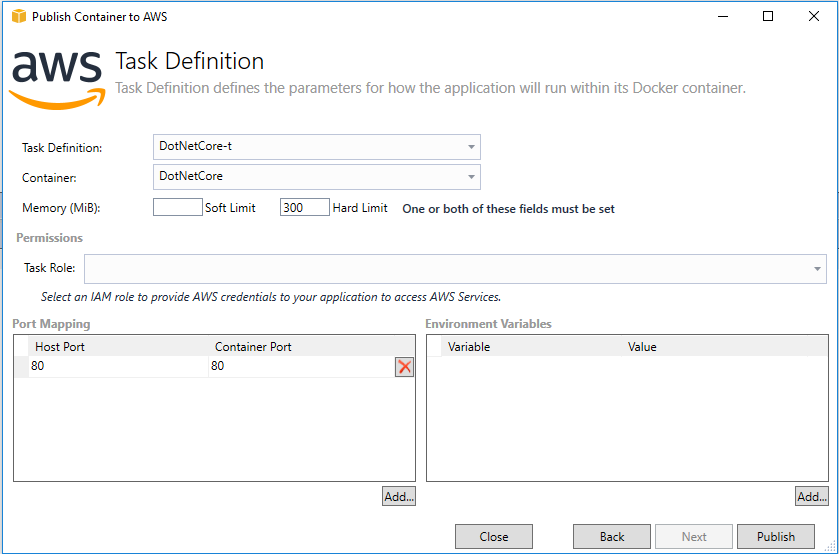
1. Keep default settings for the Launch configuration.



1. Select the service you want to use. I chose the service we created when creating the cluster. Click *Next*.



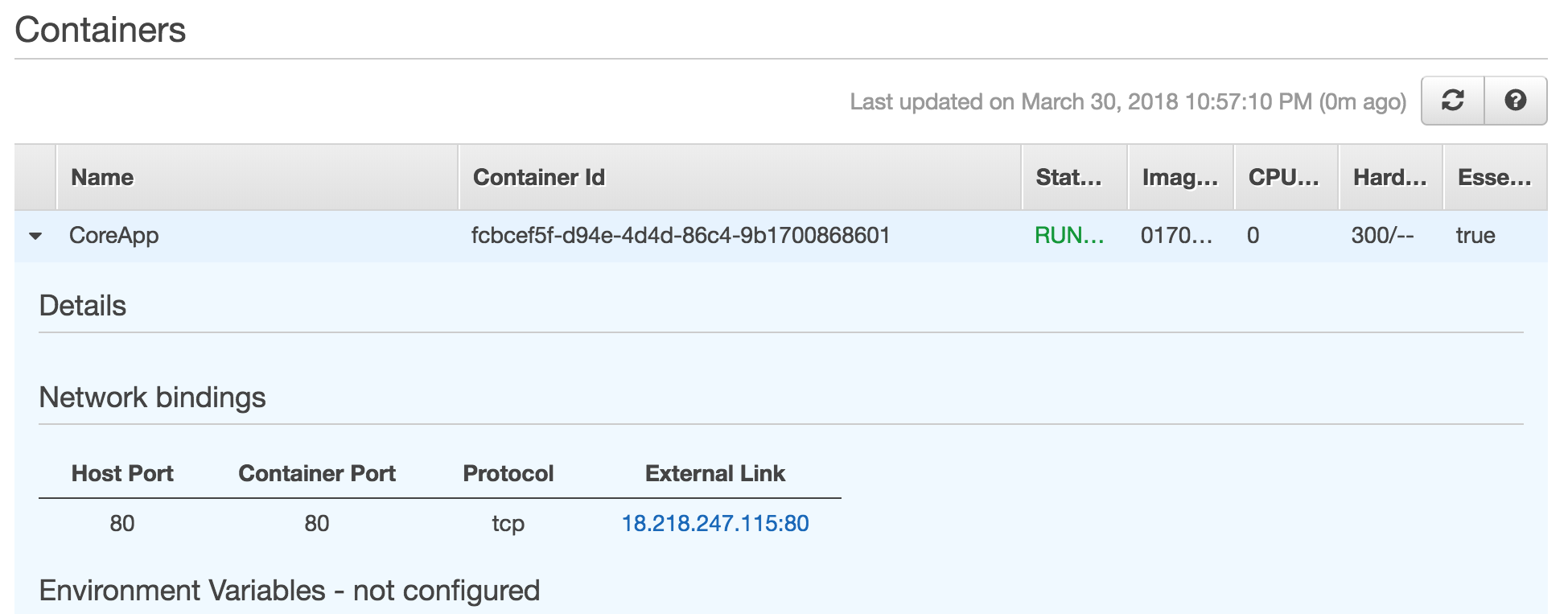
1. Chose *Task Definition*. Create a new task and container. This will let the task use the container we push to ECR via the publishing wizard. Click *Publish*.



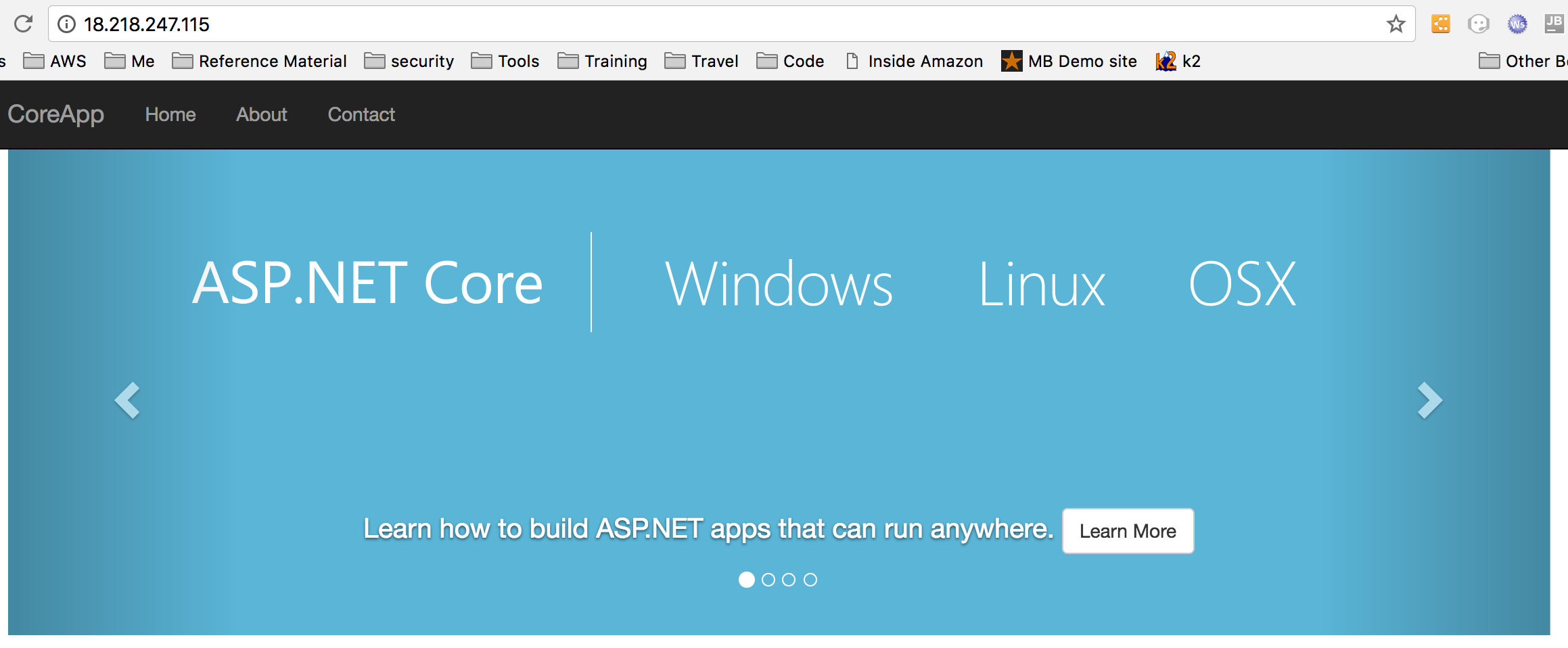
### Step 1: Deploy using .NET CORE CLI on Windows, MAC OS X, or LINUX

### Step 2: test Deployment

Once Publishing is complete you can view your application in the browser. To get the IP go back to the ECS service and click the task. There will be an IP:port to click.



The site will load.



### Side-bar

* Time to Complete—5 mins