**Overview**

AWS Elastic Beanstalk is a service that simplifies the process of provisioning AWS resources for your application. Elastic Beanstalk provides all of the AWS infrastructure required to deploy your application.

This infrastructure includes:

* Amazon EC2 instances that host the executables and content for your application.
* An Auto Scaling group to maintain the appropriate number of Amazon EC2 instances to support your application.
* An Elastic Load Balancing load balancer that routes incoming traffic to the Amazon EC2 instance with the most bandwidth.

The Toolkit for Visual Studio provides a wizard that simplifies publishing applications through Elastic Beanstalk. You can use Elastic Beanstalk to deploy your .NET web application projects to AWS. You can deploy your application to a single instance environment or to a fully load balanced, automatically scaled environment from within the IDE.

If your application uses SQL Server in Amazon RDS, the deployment wizard can also set up the connectivity between your application environment in Elastic Beanstalk and the database instance in Amazon RDS.

In this guide you are going to use Visual Studio IDE to deploy a .NET Core Web Application to Elastic Beanstalk by performing following steps:

* Module 1 - Installing Toolkit for Visual Studio
* Module 2 - Providing AWS Credentials by adding a new profile to the SDK Credential Store
* Module 3 - Creating a sample .NET Core application starter project
* Module 4 - Publishing .NET Core application using “Publish to Elastic Beanstalk Wizard”
* Module 5 - Testing the .NET Core Application

**Module 1 - Installing Toolkit for Visual Studio:**

You have several options to install the Toolkit for Visual Studio.

Option 1 – Install using Visual Studio Extensions and Updates:

* Open Visual Studio 2017 and click **Tools** -> **Extensions and Updates**
* In the search dialog box, type “**AWS Toolkit for Visual Studio 201**7”
* Click **Enable**

Option 2 – Install using Visual Studio Marketplace

* Navigate to the [Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=AmazonWebServices.AWSToolkitforVisualStudio2017)
* Click **Download**
* Run downloaded AWSToolkitPackage.vsix package.

**Module 2- Providing AWS Credentials by adding a new profile to the SDK Credential Store**

Before you can use the Toolkit for Visual Studio, you must provide one or more sets of valid AWS credentials. These credentials allow you to access your AWS resources through the Toolkit for Visual Studio.

The Toolkit for Visual Studio supports multiple sets of credentials from any number of accounts. Each set is referred to as a profile. When you add a profile to Toolkit for Visual Studio, the credentials are encrypted and stored in the SDK Credential Store. This is also used by the AWS SDK for .NET and the AWS Tools for Windows PowerShell. The SDK Credential Store is specific to your Windows user account on your machine and can't be decrypted or used elsewhere.

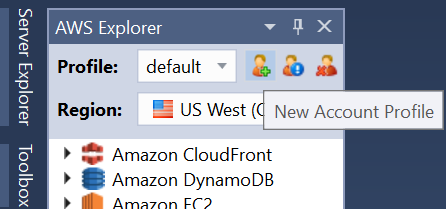
Before creating a profile, you will need to download IAM user details into a CSV file.

* Navigate to the [AWS IAM Console](https://console.aws.amazon.com/iam/home)
* Click **Users** on the left navigation pane click “**Add User**”
* Provide a user name and select “**Programmatic Access**” as the Access Type. Click “**Next:Permissions**”
* On the **Set Permission** dashboard, click “**Attach existing policies directly**” and select “**AWSElasticBeanstalkFullAccess**”. This policy provides full access to AWS Elastic Beanstalk and underlying services that it requires such as S3 and EC2. Click “**Next:Review**”
* On the review page, click **Create User**.
* Once the user is created, click “**Download .csv**” button to download csv file including access Key ID and Secret Access Key.

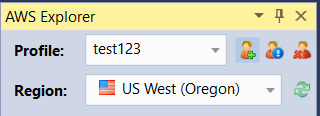


Now you can create a new profile within the Visual Studio:

* Open **Visual Studio**, on the **View** menu, choose **AWS Explorer**.
* Choose the **New Account Profile** icon to the right of the **Profile** list.



* In the **New Account Profile** dialog box, following fields are required:
  + Profile Name:
  + Access Key ID:
  + Secret Access Key:
* Provide a Profile Name and thenclick **Import from CSV file** and choose the CSV file you downloaded in previous step. Click **OK**
* Make sure new profile is created in **AWS Explorer.**



**Module 3 - Creating a sample .NET Core application starter project**

In this module, you will be creating a sample .NET Core Web API application using Visual Studio.

* In Visual Studio, from the **File** menu, choose **New**, and then choose **Project**.
* In the navigation pane of the **New Project** dialog box, expand **Installed**, expand **Visual C#** and then choose .**NET Core**
* In the list of project templates, choose **ASP.NET Core Web Application** template.
* In the **Name** box, type “EBNETCoreApplicationDemo”
* In the **Location** box, type the path to a solution folder on your development machine and choose **OK**.
* In the “**New ASP.NET Core Web Application**” dialog box, choose **Web API** and **uncheck** “**Enable Docker Support**” checkbox.
* Choose **OK**, Visual Studio will create a solution and project and then display Solution Explorer where the new solution and project appear.

**Module 4 - Publishing .NET Core application using “Publish to Elastic Beanstalk Wizard”**

In this module, you will set up Visual Studio Toolkit to publish .NET Core application to AWS Beanstalk.

* In Solution Explorer, open the context (right-click) menu for the EBNETCoreApplicationDemo project folder for the project you created in the previous section, or open the context menu for the project folder for your own application, and choose **Publish to AWS Elastic Beanstalk**.
* In **Profile**, from the **Account profile** **to use** drop-down list, choose the AWS account profile you created in previous steps.
* From the **Region** drop-down list, choose the region to which you want Elastic Beanstalk to deploy the application.
* In **Deployment Target,** choose **Create a new application environment** to perform the first deployment of your application. Choose **Next**
* On the **Application Environment** page, in the **Application** area, the **Name** drop-down list proposes a default name for the application.
* In the **Environment** area, in the **Name** drop-down list, type **EBNETCoreApplicationDemo-dev.**
* In the **URL** box, type a unique subdomain name that will be the URL for your web application. Choose **Check Availability** to make sure the URL for your web application is not already in use. Click **Next**
* On the **AWS Options** page, in **Amazon EC2 Launch Configuration**, from the **Container type** dropdown list, choose **64bit Windows Server 2016 v1.2.0 running IIS 10.0**.
* In the **Instance type** drop-down list, specify t2.micro as the Amazon EC2 instance type to use. This will minimize the cost associated with running the instance.
* In the **Key pair** drop-down list, choose an existing Amazon EC2 instance key pair to use to sign in to the instances that will be used for your application.

In this window, you will also see additional optional configuration options as follows:

* **Use non-default VPC**

This option will allow you to deploy application environment in a VPC. The VPC must have already been created including at least one public and one private subnet. Elastic Load Balancer for your application will be deployed to public subnet which is associated with a routing table that has an entry that points to an internet gateway. Instances created for your application will be placed in the private subnet.

* **Single Instance environment**

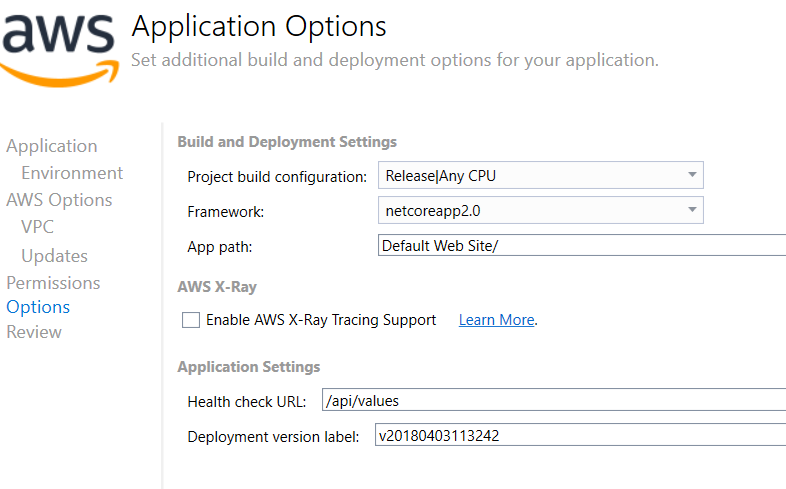
This option allows you to launch only a single Amazon EC2 instance rather than a fully load balanced, automatically scaled environment.

* **Enable rolling deployments**

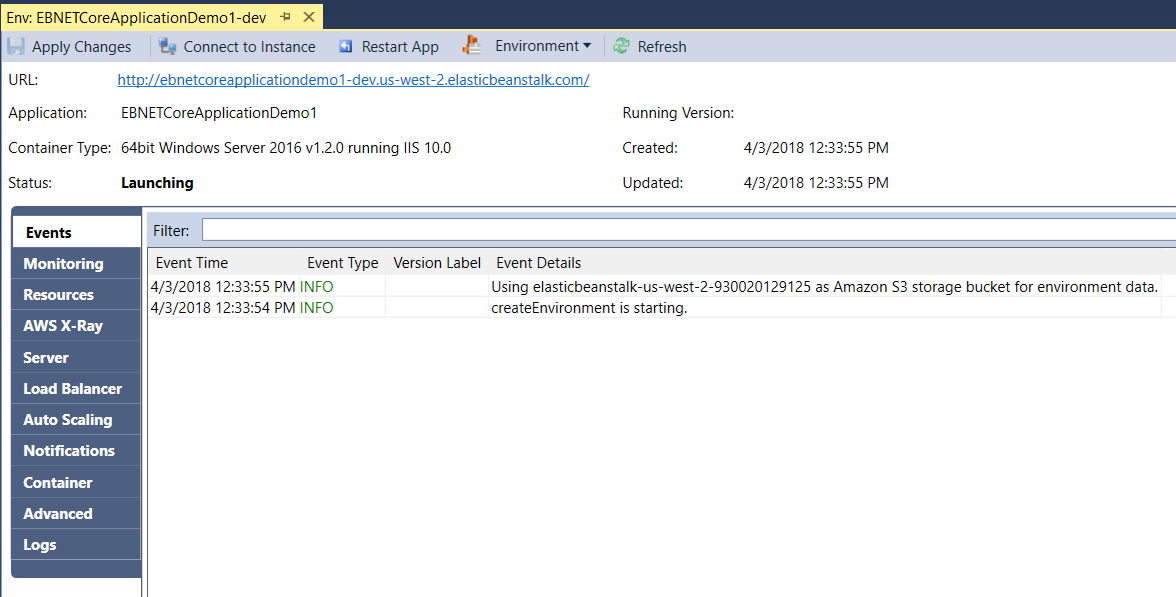
AWS Elastic Beanstalk provides several options for how deployments are processed. With rolling deployments, Elastic Beanstalk splits the environment's EC2 instances into batches and deploys the new version of the application to one batch at a time, leaving the rest of the instances in the environment running the old version of the application. During a rolling deployment, some instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version.

For this guide, uncheck all boxes.

* On the **Permissions** page, accept default values **aws-elasticbeanstalk-ec2-role** and **aws-elasticbeanstalk-service-role**. **Deployed Application Permissions** will be used to delivery AWS credentials to your applications so that it can access AWS resources. **Service Permissions** will allow Elastic Beanstalk service to monitor environment.
* On the **Application Options** page, in the **Build and IIS Deployment Settings** area, specify target build configuration as **Release**. In the **Framework** drop-down list, choose **netcoreapp2.0**. In App Path box, accept the default path (**Default Web Site/**) that IIS will use to deploy application.
* In the **Health Check URL** box, type /api/values. Elastic Beanstalk will use this URL to determine if your web application is still responsive.
* The toolkit will also provide a deployment version label which is based on the current date and time. Accept provided label and click **Finish.**
* Click **Deploy.**

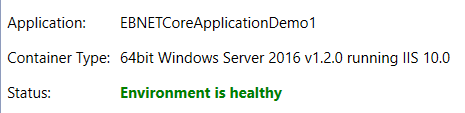


Status page for the deployment will open. The deployment may take a few minutes.



**Module 5 - Testing the .NET Core Application**

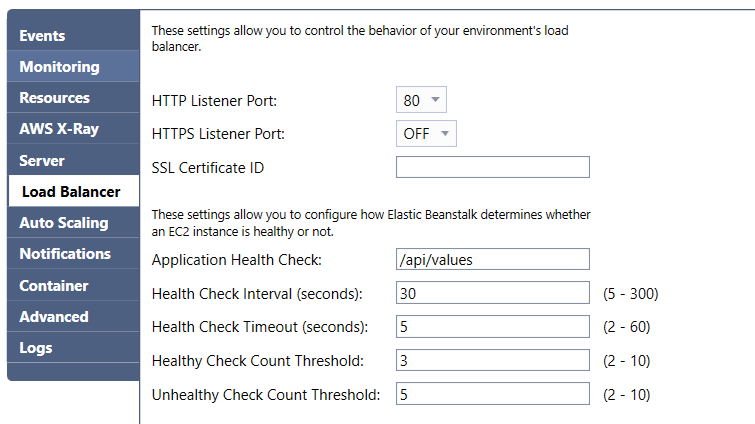
When the deployment is complete, you should see Status: Healthy.



The toolkit created multiple resources to host your sample .NET core application. You can navigate left pane to discover those resources such as:

* EC2 Instances
* Load Balancer
* Auto-Scaling Group

Settings for those resources can also be configured using the Toolkit.



Once the application status is healthy, you can click URL to test your application.

Add **/api/values** at the end of the URL**.** Sample .NET Core Web API application should return following page:

