

This is a note to remind me how to publish ASP.NET Core website to AWS via Visual studio.
PS: My Visual studio version is Enterprise 2017. Hope it also helps you, good luck.

Step 1: AWS setting

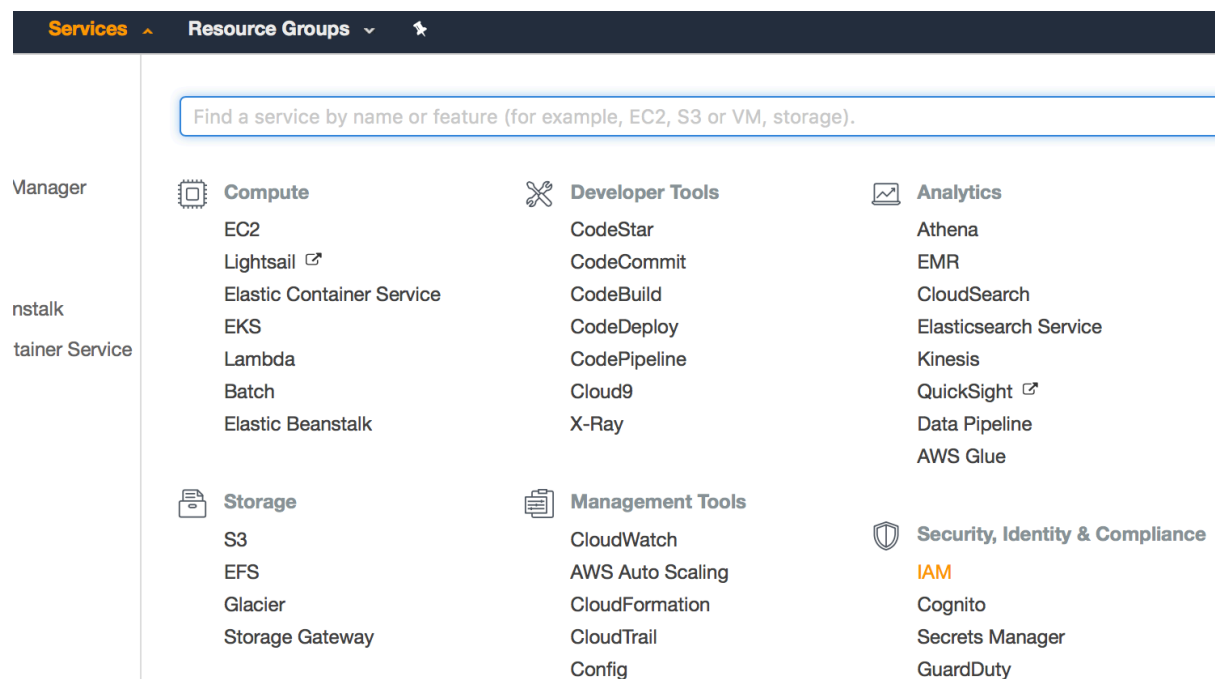
If you already create a user and add access policy for that user, please go to step 2.

If you don't have an AWS account, click the link to create an AWS account:

<https://portal.aws.amazon.com/billing/signup#/start>

Please note that, for authorize your AWS account, you have to bind a valid bank account to your AWS account which can use online and it has at least \$1.

After your AWS account was authorized, login to your account 'Service' -> 'Security, Identity & Compliance' -> 'IAM'



In left hand side, select 'Groups' -> 'Create New Group' (if you don't want to put user in a group, you can create new user and attach policy directly.)

aws Services ▾ Resource Groups ▾ ⚙

Search IAM

Dashboard

Groups

Users

Roles

Policies

Identity providers

Account settings

Credential report

Create New Group Group Actions ▾

Filter


<input type="checkbox"/>	Group Name ↕	Users
<input type="checkbox"/>	ASP.NETuser	1

Input your group name, click 'Next' to attach policy, select 'AmazonEC2FullAccess' & 'AWSElasticBeanstalkFullAccess'

Attach Policy

Select one or more policies to attach. Each group can have up to 10 policies attached.


Filter: Policy Type ▾

	Policy Name ↕	Attached Entities ↕	Creation Time ↕
<input checked="" type="checkbox"/>	 AmazonEC2FullAccess	1	2015-02-07 07:40 UTC+1200

Attach Policy

Select one or more policies to attach. Each group can have up to 10 policies attached.

Filter: Policy Type ▾

	Policy Name ↕	Attached Entities ↕
<input checked="" type="checkbox"/>	 AWSElasticBeanstalkFullAccess	1

After you created group, select 'User' in the left-hand side bar, click 'Add New User', input user name, the access type chose 'Programmatic access'

Add user

1 2 3 4

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Access type* ☒ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☐ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.


* Required


[Cancel](#) [Next: Permissions](#)


Add user to the group, it's ok if you don't have a group, but don't forget to attach access policy for your user, just the same as attach access policy to a group.

Add user

Set permissions for Ivy-aspnet

 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job funct

Add user to group

[Create group](#) [Refresh](#)

Group ▾	Attached policies
<input checked="" type="checkbox"/> ASP.NETUser	AmazonEC2FullAccess and 1 more

After you add user successfully, you will have 'Access key ID' & 'Secret access key', remember this, you will need this for you AWS setting in visual studio.

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://638982746235.signin.aws.amazon.com/console>

[Download .csv](#)

User	Access key ID	Secret access key
lv-y-asp.net		

[Close](#)

Step 2: Visual studio setting

For configure the Visual Studio environment, you have to install 'AWSSDK.EC2' & 'AWSSDK.'

There two ways to install 'AWSSDK.EC2':


The first way is install it via 'Visual studio' -> 'Tools' -> 'NuGet package Manager', you won't have 'AWS explorer'

```
PM> Install-Package AWSSDK.EC2
```

```
PM> Install-Package AWSSDK
```

Another way is to download 'MSI Installer'. The installer will include the AWS Toolkit for Visual Studio and AWS Tools for PowerShell, you will get a 'AWS explorer' in your VS.

Secure | <https://aws.amazon.com/sdk-for-net/>

[Contact Sales](#)[Products](#)[Solutions](#)[Pricing](#)[Getting Started](#)[More](#)[English](#)[My Account](#)[Create an AWS Account](#)

RESOURCES
[AWS SDK for .NET](#)
RELATED LINKS
[Documentation](#)
[Tools](#)

Get Started for Free
[Create Free Account](#)


AWS SDK for .NET

Get started quickly using AWS with the AWS SDK for .NET. The SDK helps take the complexity out of coding by providing .NET APIs for AWS services including Amazon S3, Amazon EC2, Amazon DynamoDB and more. The SDK can be downloaded from NuGet or installed using the MSI package, which also includes the AWS Toolkit for Microsoft Visual Studio 2013 and 2015 editions and the AWS Tools for Windows PowerShell.


The AWS SDK for .NET is now distributed as multiple service-specific packages on NuGet. Example package names include AWSSDK.EC2, AWSSDK.S3, and AWSSDK.DynamoDB. Each of these depends on the AWSSDK.Core, which will automatically be installed in your project if you reference any of the service packages in the NuGet Package Manager.

1


Getting Started »



Developer Guide »



API Reference »



Developer Blog »

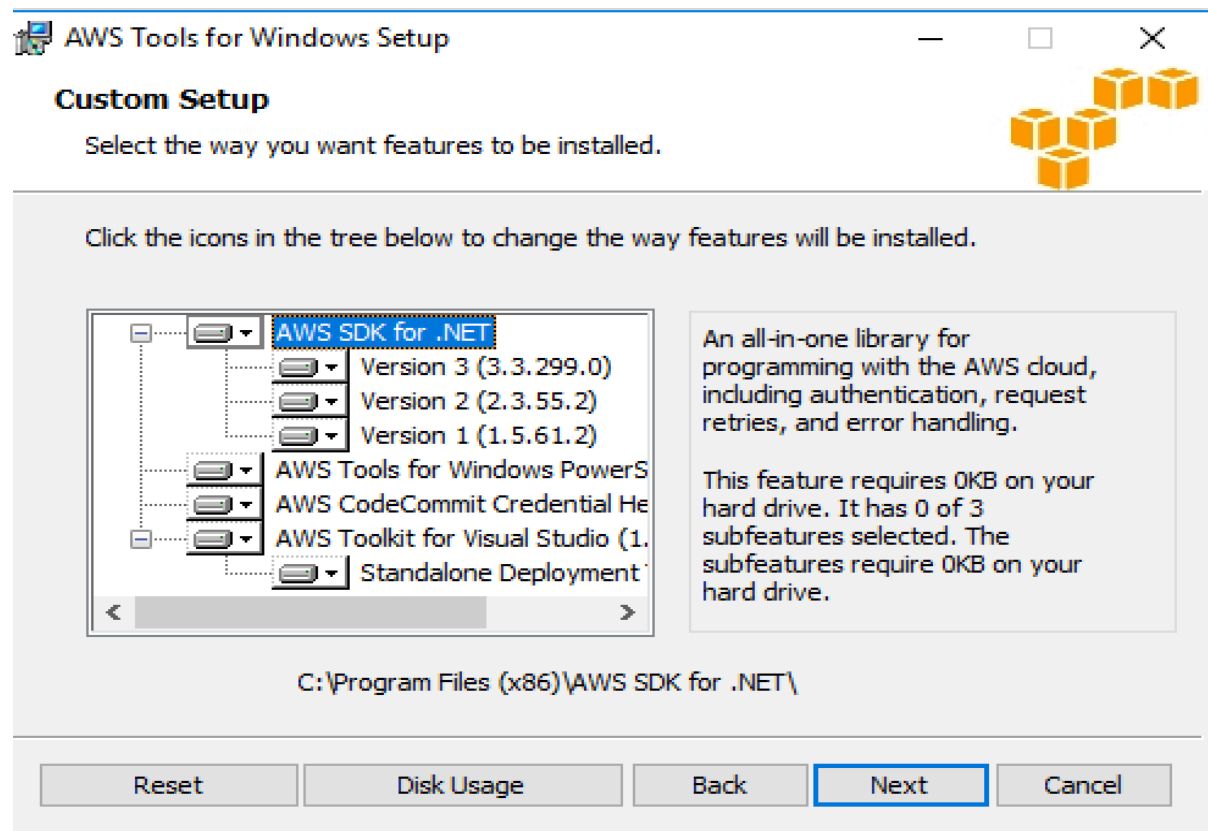
Install
[Install from NuGet »](#)

NuGet is the recommended way to install AWS SDK for .NET packages. NuGet Package Manager will install the correct assemblies for your project type - .NET 3.5, 4.5, or Portable Class Library.

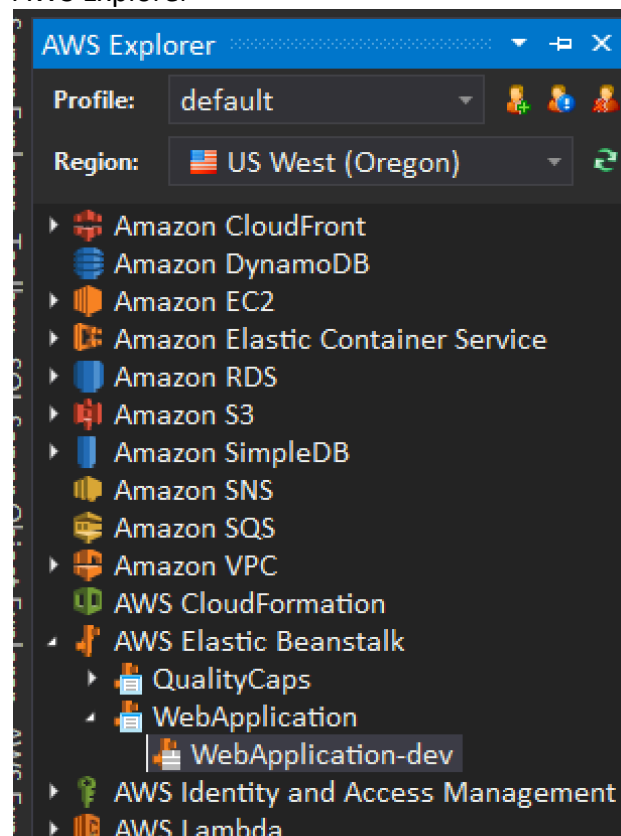
[Download MSI Installer »](#)

If you prefer to use the MSI installer, you can download it by clicking the above button. The installer also includes the AWS Toolkit for Visual Studio and AWS Tools for Windows PowerShell. The installer includes assemblies for .NET 3.5 and 4.5.

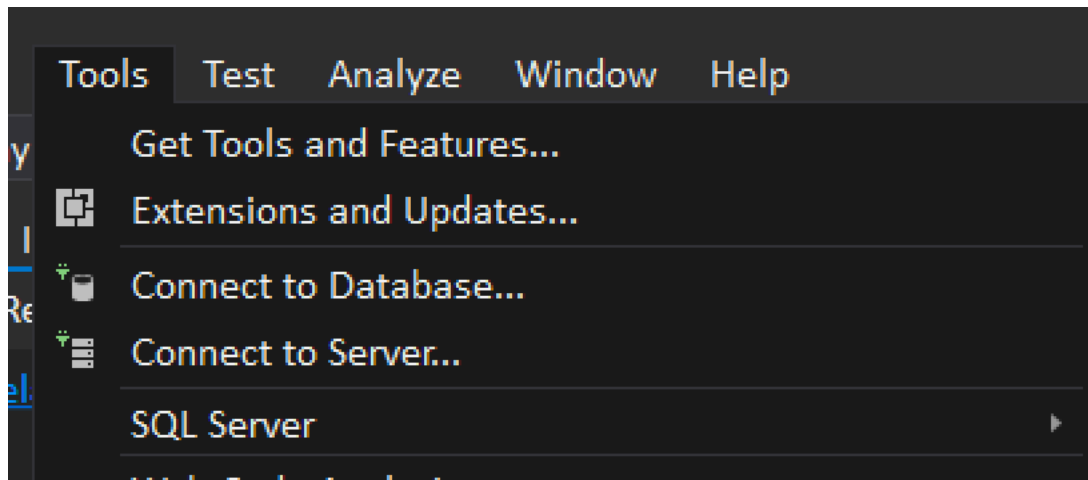
Installing AWS Tools ...



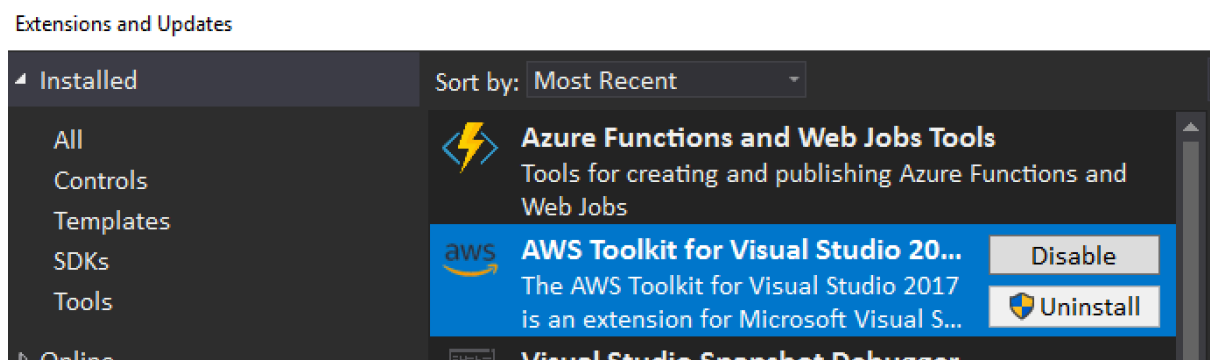
AWS Explorer



Install 'AWS Toolkit' via visual studio, 'Tools' -> 'Extensions and Updates'

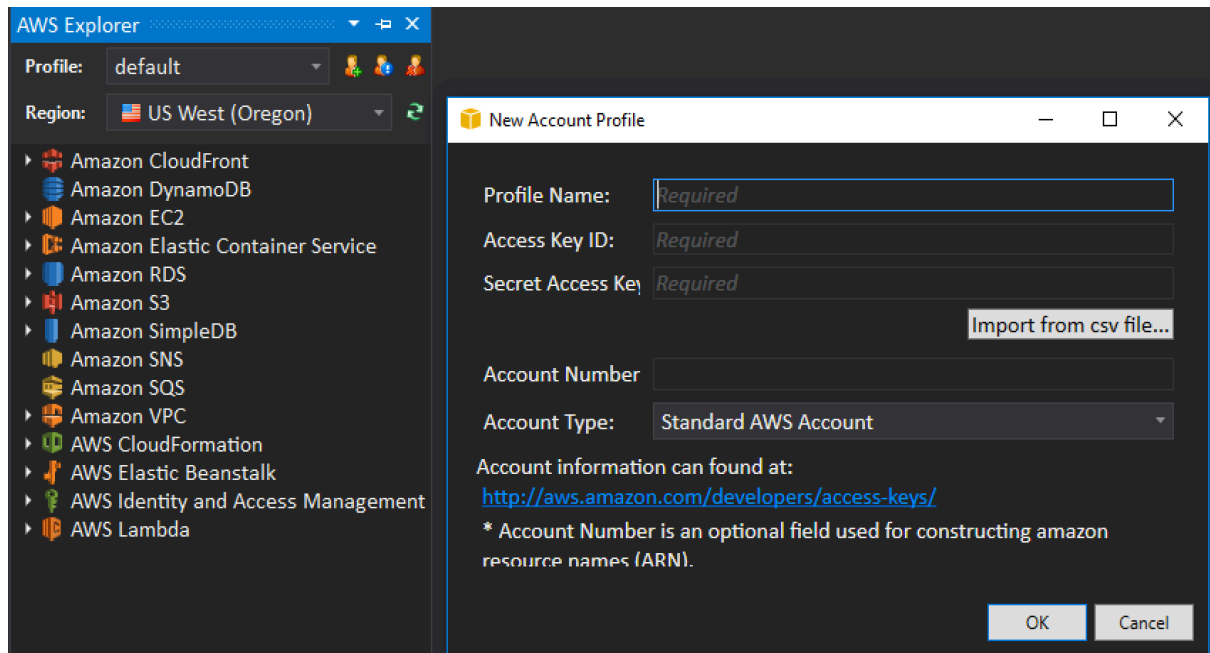


Once you install it, you see it in your 'Extensions and Updates' -> 'Installed' page



All tools are installed.

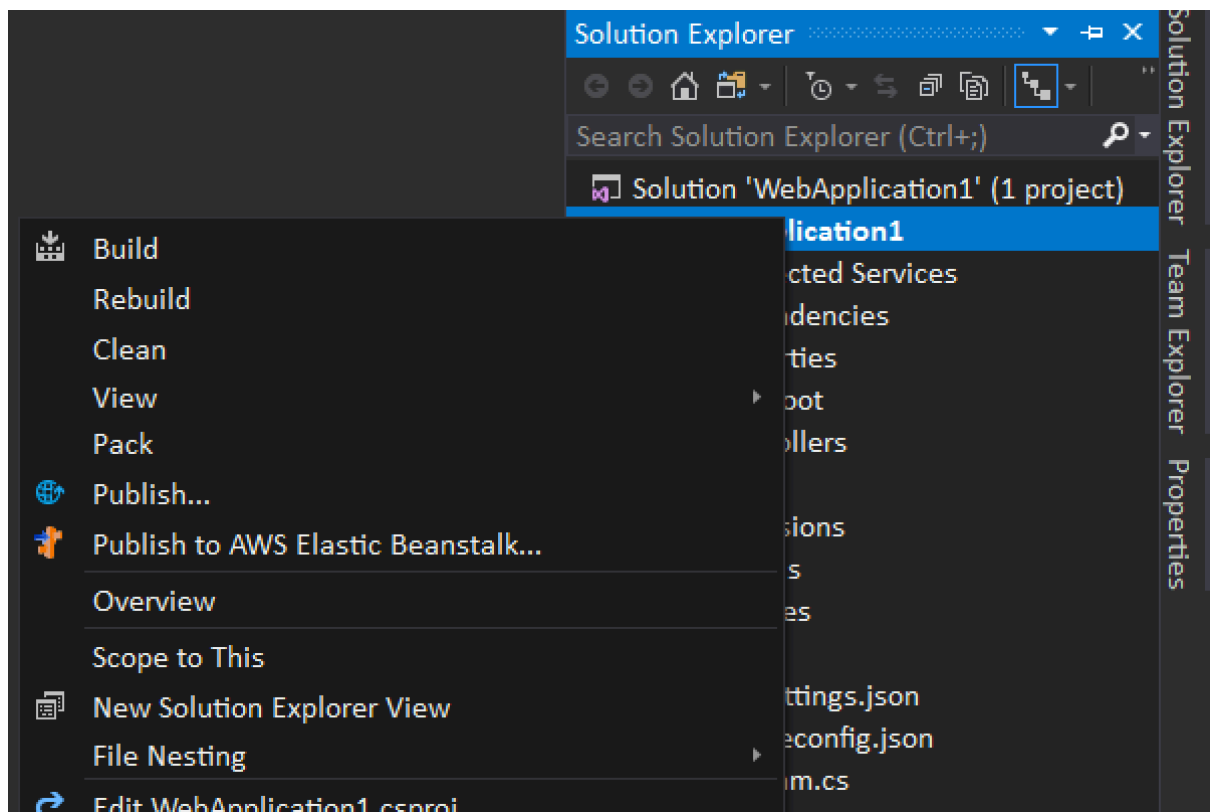
Add profile to your 'AWS Explorer', click '+', 'New Account Profile' window will show. Input a profile name, the have 'Access key ID' & 'Secret access key' is the keys you got from your 'AWS'-> 'IAM'-> 'User'



Everything is done, let's start to publish our website to AWS via visual studio.

Step 3: Publish website to AWS via visual studio

Right click on your project, select 'Publish to AWS Elastic Beanstalk'...



You can choose different account profile, for me, I just have one profile which connect to AWS account. You can choose 'Redeploy to and existing environment' or 'new environment'

The screenshot shows the 'Publish to AWS Elastic Beanstalk' window. On the left is a sidebar with navigation links: Application, Environment, AWS Options, VPC, Updates, Permissions, Options, and Review. The main area is titled 'Publish to AWS Elastic Beanstalk' and includes the text 'Publish can create a new application/environment or redeploy to an existing environment'. Under the 'Profile' section, 'Account profile to use:' is set to 'default' and 'Region:' is 'US West (Oregon)'. The 'Deployment Target' section has two radio buttons: 'Create a new application environment' (selected) and 'Redeploy to an existing environment:'. Below the radio buttons, a list shows an existing application 'QualityCaps' with a sub-environment 'QualityCaps-dev' that is 'Ready' and has the URL 'qualitycaps-dev.us-west-2.elasticbeanstalk.com'. At the bottom are buttons for 'Close', 'Back', 'Next', and 'Finish'.

Check the URL is available or not

The screenshot shows the 'Application Environment' window. The sidebar on the left has 'Environment' selected. The main area is titled 'Application Environment' and says 'Enter the details for your new application environment. To create a new environment for an existing application, select the appropriate application'. Under the 'Application' section, 'Name:' is 'WebApplicationTest'. Under the 'Environment' section, 'Name:' is 'WebApplicationTest-dev'. Under the 'URL' section, the 'http:' field contains 'webapplicationtest-dev' and the '.elasticbeanstalk.com' field is empty. A 'Check availability...' button is to the right. Below the URL fields, a green message says '✓ The requested URL is available'. At the bottom are buttons for 'Close', 'Back', 'Next', and 'Finish'.

I create a new key pair, just click dropdown arrow -> 'Create a new key pair'

The screenshot shows the 'Publish to Amazon Web Services' dialog box with the 'AWS Options' tab selected. The left sidebar contains a navigation menu with 'AWS Options' highlighted. The main content area is titled 'Amazon EC2 Launch Configuration' and includes the following fields and options:

- Container type ***: 64bit Windows Server 2016 v1.2.0 running IIS 10.0
- Instance type ***: t2.micro
- Key pair ***: testappkeypair
- Use custom AMI**: (empty text field)
- Options**: ☐ Use non-default VPC, ☒ Single instance environment, ☐ Enable Rolling Deployments

Below these fields is a section titled 'Relational Database Access' with the instruction: 'Select the Amazon RDS security groups to be modified to permit access from the EC2 instance(s) hosting your application.' This section contains an empty dropdown menu.

At the bottom right, there are four buttons: 'Close', 'Back', 'Next', and 'Finish'.

The Deployed Application Permissions will auto fill-up for you

The screenshot shows the 'Publish to Amazon Web Services' dialog box with the 'Permissions' tab selected. The left sidebar contains a navigation menu with 'Permissions' highlighted. The main content area is titled 'Permissions' and includes the following fields and options:

- Deployed Application Permissions**:
 - Role:** aws-elasticbeanstalk-ec2-role
 - This role is used to delivery AWS credentials to your application so that it can access AWS resources. The permissions for the Identity and Access Management role can be updated after the environment is created.*
- Service Permissions**:
 - Role:** aws-elasticbeanstalk-service-role
 - A service role allows the Elastic Beanstalk service to monitor environment resources on your behalf. See [Roles and Instance Profiles] in the Elastic Beanstalk developer guide for details.*

At the bottom right, there are four buttons: 'Close', 'Back', 'Next', and 'Finish'.

click 'Next' -> 'Deploy' button, it will start to publish your project, you can see this info from 'AWS Explorer'

The screenshot shows the AWS Management Console for the 'WebApplication-dev' environment. The status is 'Launching'. The URL is <http://webapplication-dev.us-west-2.elasticbeanstalk.com/>. The application is 'WebApplication' with a running version of 'v20180614035933'. The container type is '64bit Windows Server 2016 v1.2.0 running IIS 10.0'. The environment was created on 6/14/2018 at 4:05:50 PM and updated on 6/14/2018 at 4:09:52 PM. The 'Events' tab is selected, showing a table of events:

Event Time	Event Type	Version Label	Event Details
6/14/2018 4:07:18 PM	INFO		Waiting for EC2 instances to launch. This may take a few minutes.
6/14/2018 4:06:27 PM	INFO		Created EIP: 52.88.14.233
6/14/2018 4:06:12 PM	INFO		Created security group named: sg-03d0aab804e90ae02
6/14/2018 4:05:50 PM	INFO		Using elasticbeanstalk-us-west-2-638982746235 as Amazon S3 storage bucket for environment data.
6/14/2018 4:05:49 PM	INFO		createEnvironment is starting.

Once the project was done, the 'AWS Explorer' window will show 'INFO Successfully launched environment: xxxxx', you can click the URL show in the top of the window.

The screenshot shows the AWS Management Console for the 'WebApplication-dev' environment. The status is now 'Ready'. The URL is <http://webapplication-dev.us-west-2.elasticbeanstalk.com/>. The application is 'WebApplication' with a running version of 'v20180614035933'. The container type is '64bit Windows Server 2016 v1.2.0 running IIS 10.0'. The environment was created on 6/14/2018 at 4:05:50 PM and updated on 6/14/2018 at 4:11:39 PM. The 'Events' tab is selected, showing a table of events:

Event Time	Event Type	Version Label	Event Details
6/14/2018 4:11:39 PM	INFO		Successfully launched environment: WebApplication-dev
6/14/2018 4:07:18 PM	INFO		Waiting for EC2 instances to launch. This may take a few minutes.
6/14/2018 4:06:27 PM	INFO		Created EIP: 52.88.14.233
6/14/2018 4:06:12 PM	INFO		Created security group named: sg-03d0aab804e90ae02
6/14/2018 4:05:50 PM	INFO		Using elasticbeanstalk-us-west-2-638982746235 as Amazon S3 storage bucket for environment data.
6/14/2018 4:05:49 PM	INFO		createEnvironment is starting.

My sample project was published successfully. So far, everything is done, enjoy your journey at C# .NET Core and AWS...

The screenshot shows a web application running on AWS Elastic Beanstalk. The browser address bar shows webapplication-dev.us-west-2.elasticbeanstalk.com/. The application has a navigation bar with links: 'WebApplication1', 'Home', 'About', 'Contact', 'Register', and 'Log in'. The main content area features a Microsoft Azure banner with the text 'Learn how Microsoft's Azure cloud platform allows you to build, deploy, and scale web apps.' and a 'Learn More' button. Below the banner, there are four columns of content:

- Application uses**
 - Sample pages using ASP.NET Core MVC
 - Theming using Bootstrap
- How to**
 - Add a Controller and View
 - Manage User Secrets using Secret Manager
- Overview**
 - Conceptual overview of what is ASP.NET Core
 - Fundamentals of ASP.NET Core
- Run & Deploy**
 - Run your app
 - Run tools such as EF migrations and more
 - Publish to Microsoft Azure Web