AWS Toolkit for Visual Studio User Guide

Release 1.0

Amazon Web Services

Dec 01, 2016

Contents

CHAPTER 1

Using the Toolkit for Visual Studio

1.1 The Toolkit for Visual Studio

The Toolkit for Visual Studio is a plugin for the Visual Studio IDE that makes it easier for developers to develop, debug, and deploy .NET applications that use Amazon Web Services. For details on how to download and install the kit, see *Installation*.

The following AWS Toolkit features enhance the development experience:

AWS Explorer AWS Explorer enables you to interact with many of the AWS services from inside the Visual Studio IDE. Supported data services include Amazon Simple Storage Service (Amazon S3), Amazon SimpleDB, Amazon Simple Notification Service (Amazon SNS), Amazon Simple Queue Service (Amazon SQS), and Amazon CloudFront. AWS Explorer also provides access to Amazon Elastic Compute Cloud (Amazon EC2) management, AWS Identity and Access Management (IAM) user and policy management, and deployment to CloudFormation. AWS Explorer supports multiple AWS accounts, including IAM user accounts, and enables you to easily change the displayed view from one account to another.

Amazon EC2 From AWS Explorer, you can view available Amazon Machine Images (AMIs), create Amazon EC2 instances from those AMIs, and then connect to those instances by using Windows Remote Desktop. AWS Explorer also enables supporting functionality, such as the capability to create and manage key pairs and security groups.

Amazon DynamoDB Amazon DynamoDB is a fast, highly scalable, highly available, cost-effective, nonrelational database service. The Toolkit for Visual Studio provides functionality for working with Amazon DynamoDB in a development context. With the Toolkit, you can create and edit attributes in Amazon DynamoDB tables and run Scan operations on tables.

CloudFormation CloudFormation makes it easy for you to deploy your .NET Framework application to AWS. CloudFormation provisions the AWS resources required by your application, which frees you to focus on developing the application's functionality. The Toolkit for Visual Studio includes two ready-to-use CloudFormation templates.

AWS Identity and Access Management (IAM) From AWS Explorer, you can create IAM users and policies, and attach policies to users.

AWS SDK for .NET The Toolkit for Visual Studio installs the latest version of the AWS SDK for .NET. From Visual Studio, you can easily modify, build, and run any of the samples included in the SDK.

Note: Toolkit for Visual Studio for Visual Studio 2008 is still available, but not supported. For more information, see *Installation*.

1.2 What's New in Version 1.3

Added support for deployment to Elastic Beanstalk In addition to its existing deployment support for CloudFormation, the Toolkit for Visual Studio now supports deployment of web applications and websites to Elastic Beanstalk. To deploy to either service, in Solution Explorer, right-click your project and choose *Publish to AWS*. You can then use the deployment wizard to choose the required service. If you have Amazon RDS instances, the deployment wizard for Elastic Beanstalk can also be used to allow connectivity between your deployed application and selected Amazon RDS instances.

Fast redeployment For previously deployed projects, a new *Republish to* command is available in the context menu for a project in Solution Explorer. The command name changes to show where the project was last deployed (AWS Elastic Beanstalk environment or CloudFormation stack) together with the environment or stack name. Choosing the command will display a dialog box that summarizes the deployment options that were last used. Choosing the *Deploy* button will start project redeployment, without the use the deployment wizard.

Support for Amazon RDS and Microsoft SQL Server Amazon RDS support has been added to the AWS Explorer to allow you to manage Amazon RDS assets in Visual Studio. Amazon RDS instances that use Microsoft SQL Server can also be added to Visual Studio's Server Explorer.

AWS standalone deployment tool additions The standalone AWS deployment tool has been updated to support deployments to Elastic Beanstalk and CloudFormation. For CloudFormation stacks, the tool now also supports `update stack` functionality.

1.3 What's New in Version 1.1

AWS standalone deployment tool The Toolkit for Visual Studio includes a command-line tool you can use to deploy your application to CloudFormation from outside of the Microsoft Visual Studio development environment. With the deployment tool, you can make deployment an automatic part of your build process or include deployment in other scripting scenarios.

Redeployment to CloudFormation Both the deployment wizard and the deployment tool can redeploy a new instance of your application over an already running instance.

AWS GovCloud (US) support You can designate AWS accounts as AWS GovCloud (US) users. These users are then able to use the AWS GovCloud (US) region.

Server-side encryption You can specify whether an Amazon S3 object should use server-side encryption. You can specify this feature at the time you upload the object or afterward in the object's properties dialog box.

Customize columns in AMI, instance, and volume views In AWS Explorer, you can customize which columns are displayed when you are viewing Amazon Machine Images (AMIs), Amazon EC2

instances, and EBS volumes.

Tagging of AMIs, instances, and volumes From AWS Explorer, you can add tags and tag values to AMIs, Amazon EC2 instances, and EBS volumes. The tags you add are then added as columns in AWS Explorer views. As with other columns, you can hide these columns if you choose.

Pagination of result set returned by SDB When you execute a query in Amazon SimpleDB, the Toolkit for Visual Studio displays only a single "page" of results—either the first 100 results or the number of results specified by the *LIMIT* parameter, if it is included in the query. The Toolkit for Visual Studio now enables you to fetch either an additional page of results or an additional ten pages of results.

Time-delayed message delivery in SQS When you send an Amazon SQS message from the Toolkit for Visual Studio, you can now specify a time delay before the message appears in the Amazon SQS queue.

Export SDB results to CSV You can export the results of your Amazon SimpleDB queries to a CSV file.

1.4 About Amazon Web Services

Amazon Web Services (AWS) is a collection of digital infrastructure services that developers can leverage when developing their applications. The services include computing, storage, database, and application synchronization (messaging and queuing). AWS uses a pay-as-you-go service model. You are charged only for the services that you—or your applications—use. Also, to make AWS more approachable as a platform for prototyping and experimentation, AWS offers a free usage tier. On this tier, services are free below a certain level of usage. For more information, go to Getting Started with AWS.

CHAPTER 2

Setting Up the Toolkit for Visual Studio

Follow the steps in this topic to install and configure the Toolkit for Visual Studio.

2.1 Prerequisites

The Toolkit for Visual Studio has the following prerequisites.

- An AWS account. To get an AWS account, go to the AWS home page and choose *Create an AWS Account*. This account will enable you to use AWS services.
- Supported operating systems: Microsoft Windows 8, Windows 7, and Windows Vista.

  We recommend that you install the latest service packs and updates for the version of Windows you are using.
- Visual Studio 2010 or later.

  We recommend that you install the latest service packs and updates.

Note: You can install the Toolkit for Visual Studio on Visual Studio Express, but the installation includes only the AWS project templates and the *standalone deployment tool*. Visual Studio Express does not support third-party extensions, such as AWS Explorer.

2.2 Installation

The Toolkit for Visual Studio is part of the AWS Tools for Windows. If you have Visual Studio 2010 or later, install the AWS Tools for Windows as follows:

To install the AWS Tools for Windows

1. Go to Toolkit for Visual Studio.
2. In the *Download* section, choose Toolkit for Visual Studio to download the installer.
3. To start the installation, run the downloaded installer and follow the instructions.

Tip: By default, the Toolkit for Visual Studio is installed in the *Program Files* directory, which requires administrator privileges. To install the Toolkit for Visual Studio as a non-administrator, specify a different installation directory.

Note: You can install the Toolkit for Visual Studio for Visual Studio 2008 from http://sdk-for-net.amazonwebservices.com/latest/AWSToolkitForVisualStudio2008.msi, but this version of the toolkit is no longer supported.

2.3 Specifying Credentials

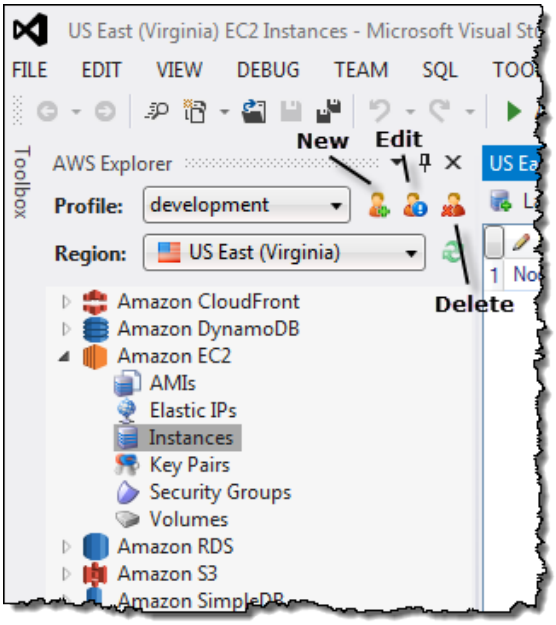
Before you can use the Toolkit for Visual Studio, you must provide one or more sets of valid AWS credentials. These credentials allow you to access your AWS resources through the Toolkit for Visual Studio. They are also used to sign programmatic web services requests, so AWS can verify the request comes from an authorized source.

Important: AWS credentials consist of an access key and a secret key. We recommend that you do not use your account's root credentials. Instead, create one or more IAM users, and then use those credentials. For more information, see Using IAM Users and Best Practices for Managing AWS Access Keys.

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 Store, which is also used by the AWS SDK for .NET and Tools for Windows PowerShell User Guide. The SDK Store is separate from your project directories so that it cannot be unintentionally committed to a public repository. To use the Toolkit for Visual Studio, you must add at least one profile to the SDK Store.

To add a profile to the SDK Store

1. Open AWS Explorer. In Visual Studio, choose the *View* menu, and then choose *AWS Explorer* or press `Ctrl+K`, and then press the `A` key.
2. Choose the *New Account Profile* icon to the right of the *Profile* list.



3. In the *New Account Profile* dialog box, type the following data:

   Profile Name (Required) The profile's display name.

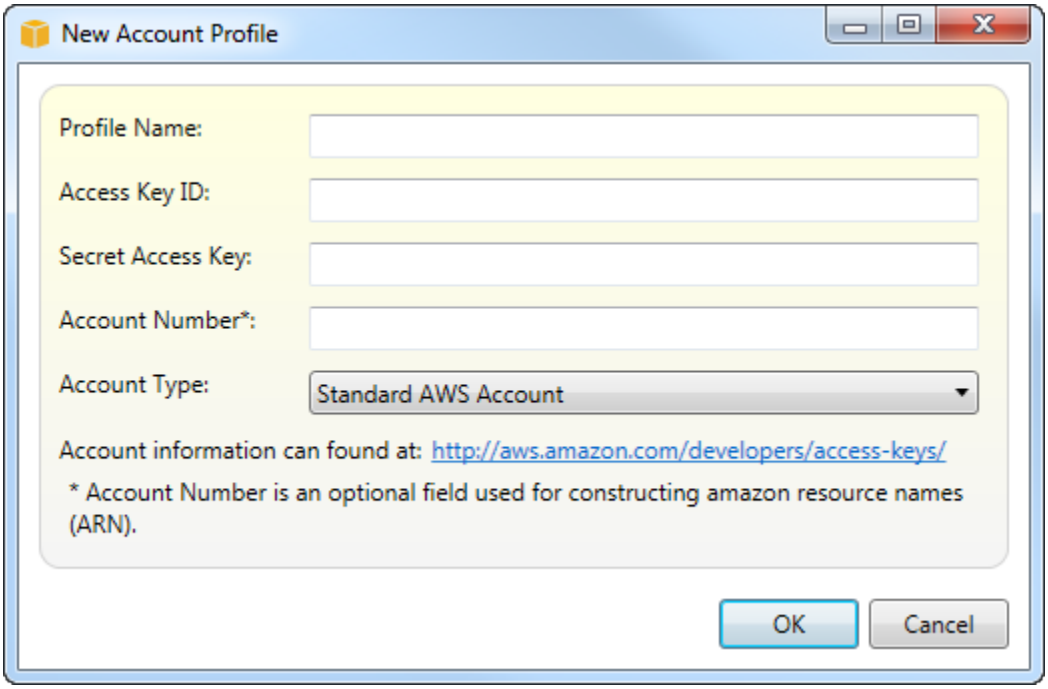
   Access Key ID (Required) The access key.

   Secret Access Key (Required) The secret key.

   Account Number (Optional) The credential's account number. The Toolkit for Visual Studio uses the account number to construct Amazon resource names (ARNs).

   Account Type (Required) The account type. This entry determines which regions are displayed in AWS Explorer when you specify this profile.

   - *Standard AWS Account*
     - If you choose AWS GovCloud (US) Account, AWS Explorer displays only the AWS GovCloud (US) region.
     - If you choose *Amazon AWS Account – China (Beijing) Region*, AWS Explorer displays only the China (Beijing) Region.



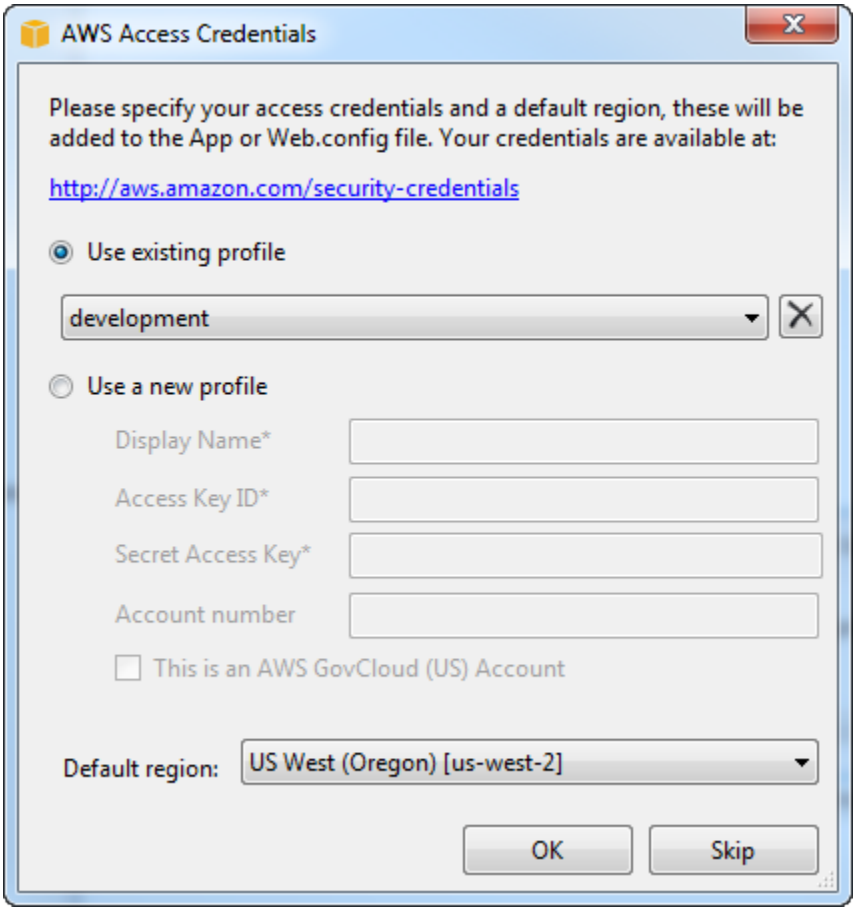
4. To add the profile to the SDK Store, choose *OK*. To use a profile in your project, choose the profile name. Toolkit for Visual Studio adds a reference to the profile to the project's `App.config` or `Web.config` file.

After you have added the first profile:

- To add another profile, repeat the procedure.
- To delete a profile, choose it, and then choose the *Delete Profile* icon.
- To edit a profile, choose the *Edit Profile* icon to display the *Edit Profile* dialog box.

  For example, if you have rotated an IAM user's credentials—a recommended practice—you can edit the profile to update the user's credentials in the SDK Store. For more information, see IAM Credential Rotation.

You can also add profiles to the SDK Store when you create an AWS project. Before Visual Studio creates the project files, it displays the *AWS Access Credentials* dialog box. You can choose a profile from the SDK Store or create one.



2.4 Uninstalling

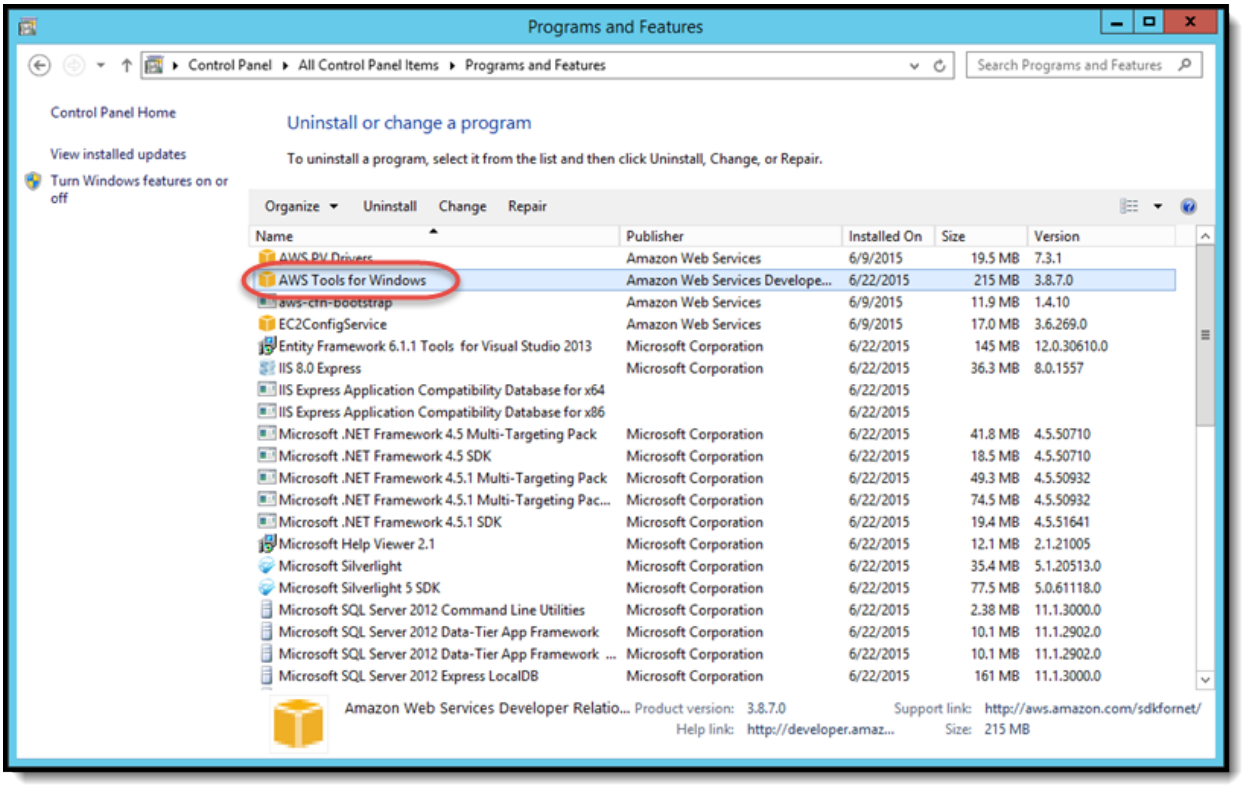
To uninstall the Toolkit for Visual Studio, you must uninstall the AWS Tools for Windows. To uninstall the AWS Tools for Windows, perform the following steps:

To uninstall the AWS Tools for Windows

1. In Control Panel, open *Programs and Features*.

   Tip: To open *Programs and Features* directly, from a command prompt, run the following: `appwiz.cpl`

2. Choose :guilabel:AWS Tools for Windows, and then choose *Uninstall*.



3. If prompted, choose *Yes*.

Uninstalling the AWS Tools for Windows does not remove the Samples directory. This directory is preserved in case you have modified the samples. You will have to manually remove this directory.

CHAPTER 3

Working with AWS Services

AWS Explorer gives you a view of, and allows you to manipulate, multiple Amazon Web Services simultaneously. This section provides information about how to access and use the AWS Explorer view in Visual Studio.

It assumes that you've already installed the AWS Toolkit for Visual Studio on your system.

3.1 Managing Amazon EC2 Instances

AWS Explorer provides detailed views of Amazon Machine Images (AMI) and Amazon Elastic Compute Cloud (Amazon EC2) instances. From these views, you can launch an Amazon EC2 instance from an AMI, connect to that instance, and either stop or terminate the instance, all from inside the Visual Studio development environment. You can use the instances view to create AMIs from your instances. For more information, see *Create an AMI from an Amazon EC2 Instance*.

3.1.1 The Amazon Machine Images and Amazon EC2 Instances Views

From AWS Explorer, you can display views of Amazon Machine Images (AMIs) and Amazon EC2 instances. In AWS Explorer, expand the *Amazon EC2* node.

To display the AMIs view, on the first subnode, *AMIs*, open the context (right-click) menu and then choose *View*.

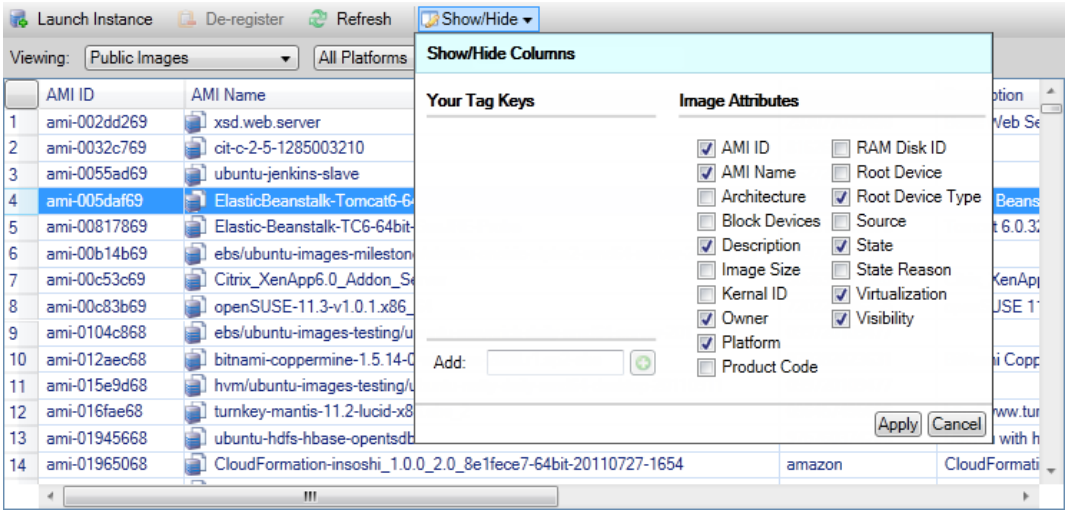
To display the Amazon EC2 instances view, on the *Instances* node, open the context (right-click) menu and then choose *View*.

You can also display either view by double-clicking the appropriate node.

- The views are scoped to the region specified in AWS Explorer (for example, the US West (N. California) region).
- You can rearrange columns by clicking and dragging. To sort the values in a column, click the column heading.
- You can use the drop-down lists and filter box in *Viewing* to configure views. The initial view displays AMIs of any platform type (Windows or Linux) that are owned by the account specified in AWS Explorer.

Show/Hide Columns

You can also choose the *Show/Hide* drop-down at the top of the view to configure which columns are displayed. Your choice of columns will persist if you close the view and reopen it.



Show/Hide Columns UI for AMI and Instances views

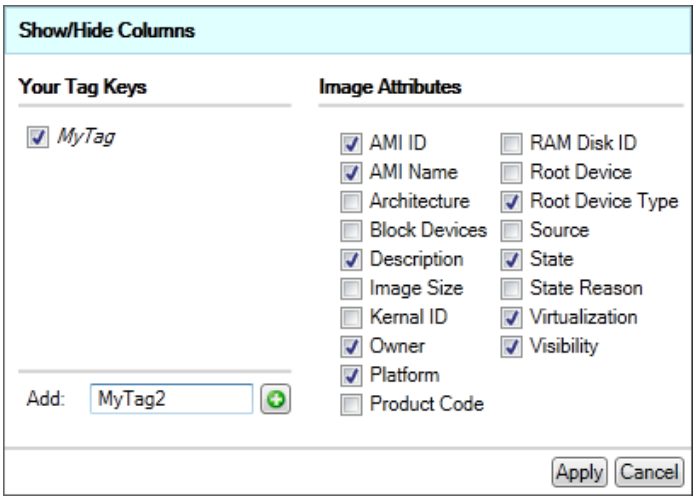
Tagging AMIs, Instances, and Volumes

You can also use the *Show/Hide* drop-down list to add tags for AMIs, Amazon EC2 instances, or volumes you own. Tags are name-value pairs that enable you to attach metadata to your AMIs, instances, and volumes. Tag names are scoped both to your account and also separately to your AMIs and instances. For example, there would be no conflict if you used the same tag name for your AMIs and your instances. Tag names are not case-sensitive.

For more information about tags, go to Using Tags in the Amazon EC2 User Guide for Linux Instances.

To add a tag

1. In the *Add* box, type a name for the tag. Choose the green button with the plus sign (+), and then choose *Apply*.



Add a tag to an AMI or Amazon EC2 instance

The new tag is displayed in italic, which indicates no values have yet been associated with that tag.

In the list view, the tag name appears as a new column. When at least one value has been associated with the tag, the tag will be visible in the AWS Console.

2. To add a value for the tag, double-click a cell in the column for that tag, and type a value. To delete the tag value, double-click the cell and delete the text.

   If you clear the tag in the *Show/Hide* drop-down list, the corresponding column disappears from the view. The tag is preserved, along with any tag values associated with AMIs, instances, or volumes.

Note: If you clear a tag in the *Show/Hide* drop-down list that has no associated values, the AWS Toolkit will delete the tag entirely. It will no longer appear in the list view or in the *Show/Hide* drop-down list. To use that tag again, use the *Show/Hide* dialog box to re-create it.

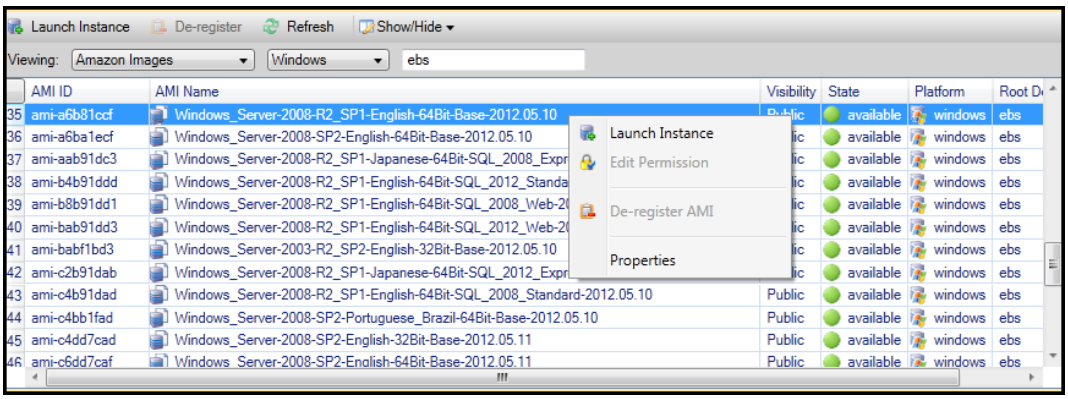
3.1.2 Launching an Amazon EC2 Instance

AWS Explorer provides all of the functionality required to launch an Amazon EC2 instance. In this section, we'll select an Amazon Machine Image (AMI), configure it, and then start it as an Amazon EC2 instance.

To launch a Windows Server Amazon EC2 instance

1. At the top of the AMIs view, in the drop-down list on the left, choose *Amazon Images*. In the drop-down list on the right, choose *Windows*. In the filter box, type `ebs` for Elastic Block Storage. It may take a few moments for the view to be refreshed.

2. Choose an AMI in the list, open the context (right-click) menu, and then choose *Launch Instance*. .



AMI list

3. In the *Launch New Amazon EC2 Instance* dialog box, configure the AMI for your application.

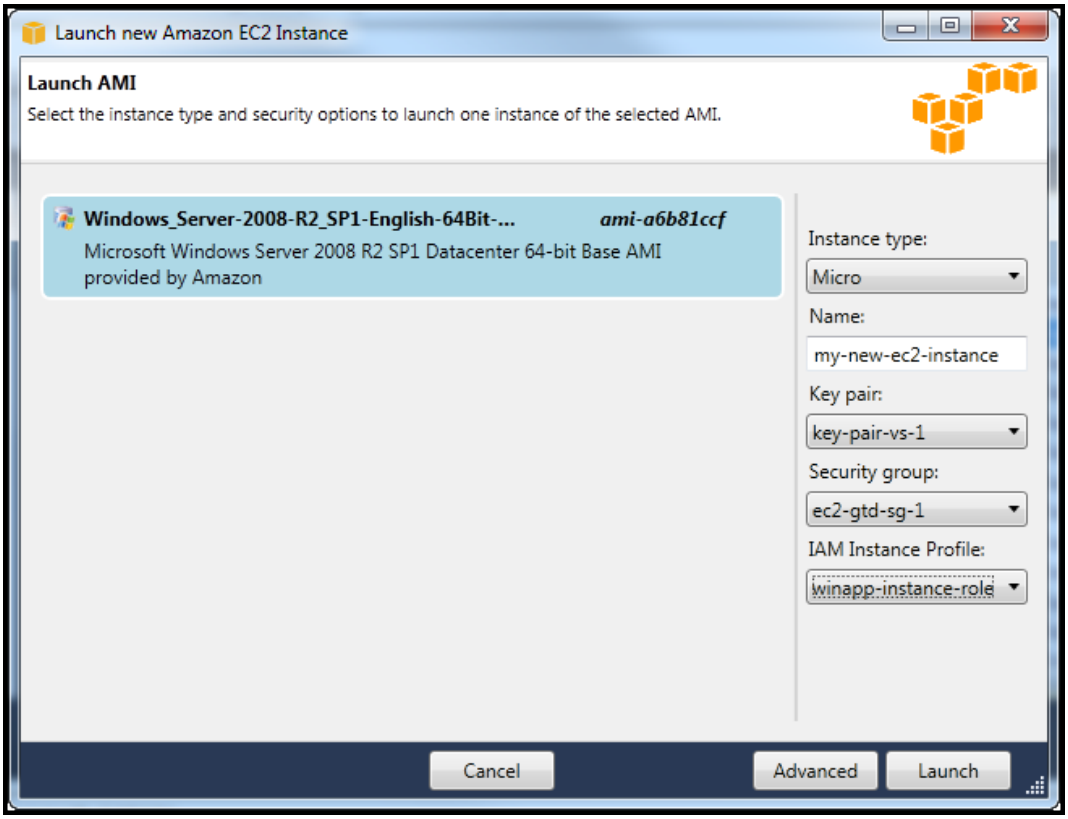
   Instance Type Choose the type of the EC2 instance to launch. You can find a list of instance types and pricing information on the *EC2 Pricing* page.

   Name Type a name for your instance. This name cannot be more than 256 characters.

Key Pair A key pair is used to obtain the Windows password that you use to log in to the EC2 instance using Remote Desktop Protocol (RDP). Choose a key pair for which you have access to the private key, or choose the option to create a key pair. If you create the key pair in the Toolkit, the Toolkit can store the private key for you.

Security Group The security group controls the type of network traffic the EC2 instance will accept. Choose a security group that will allow incoming traffic on port 3389, the port used by RDP, so that you can connect to the EC2 instance. For information about how to use the Toolkit to create security groups, see *Creating a Security Group*.

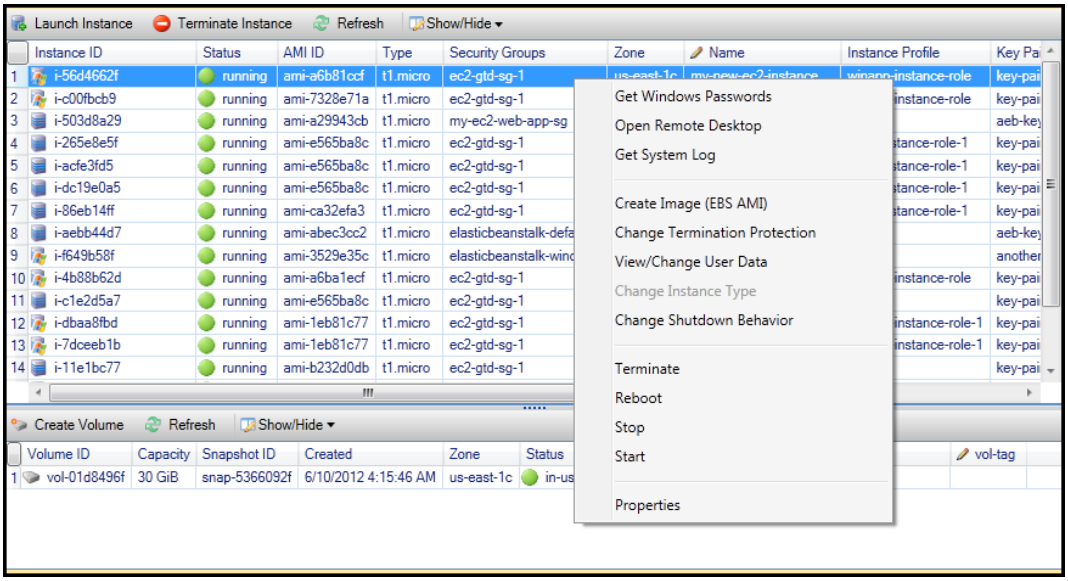
Instance Profile The instance profile is a logical container for an IAM role. When you choose an instance profile, you associate the corresponding IAM role with the EC2 instance. IAM roles are configured with policies that specify access to AWS services and account resources. When an EC2 instance is associated with an IAM role, application software that runs on the instance runs with the permissions specified by the IAM role. This enables the application software to run without having to specify any AWS credentials of its own, which makes the software more secure. For more information about IAM roles, go to the IAM User Guide.



EC2 *Launch AMI* dialog box

4. Choose *Launch*.

   In AWS Explorer, on the *Instances* subnode of *Amazon EC2*, open the context (right-click) menu and then choose *View*. The AWS Toolkit displays the list of Amazon EC2 instances associated with the active account. You may need to choose *Refresh* to see your new instance. When the instance first appears, it may be in a pending state, but after a few moments, it transitions to a running state.



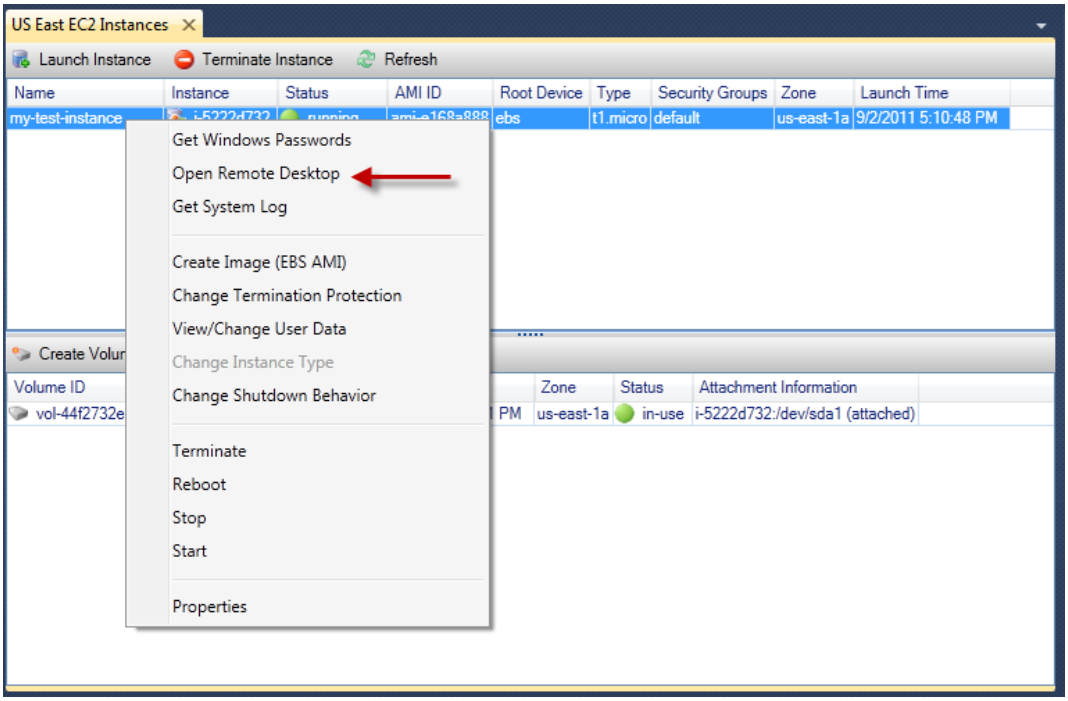
3.1.3 Connecting to an Amazon EC2 Instance

You can use Windows Remote Desktop to connect to a Windows Server instance. For authentication, the AWS Toolkit enables you to retrieve the administrator password for the instance, or you can simply use the stored key pair associated with the instance. In the following procedure, we'll use the stored key pair.

To connect to a Windows Server instance using Windows Remote Desktop

1. In the EC2 instance list, right-click the Windows Server instance to which you want to connect. From the context menu, choose *Open Remote Desktop*.

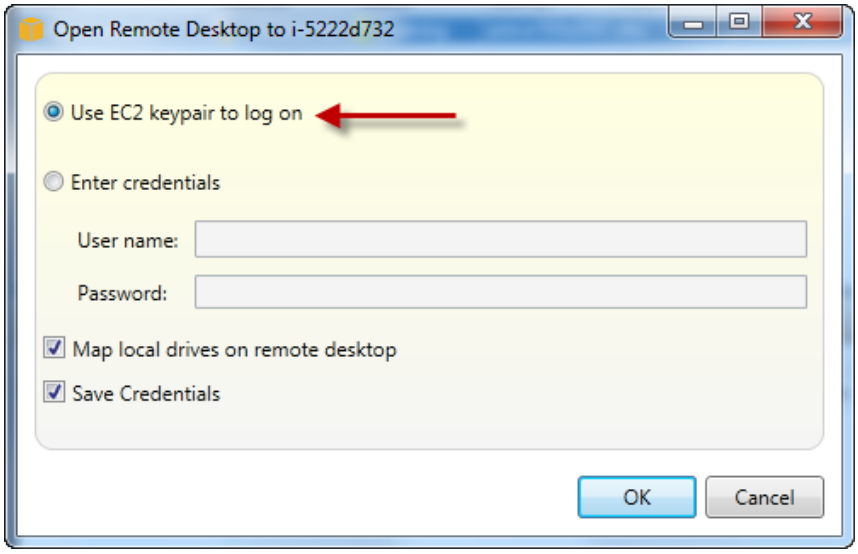
   If you want to authenticate using the administrator password, you would choose *Get Windows Passwords*.



EC2 Instance context menu

2. In the *Open Remote Desktop* dialog box, choose *Use EC2 keypair to log on*, and then choose *OK*.

   If you did not store a key pair with the AWS Toolkit, specify the PEM file that contains the private key.

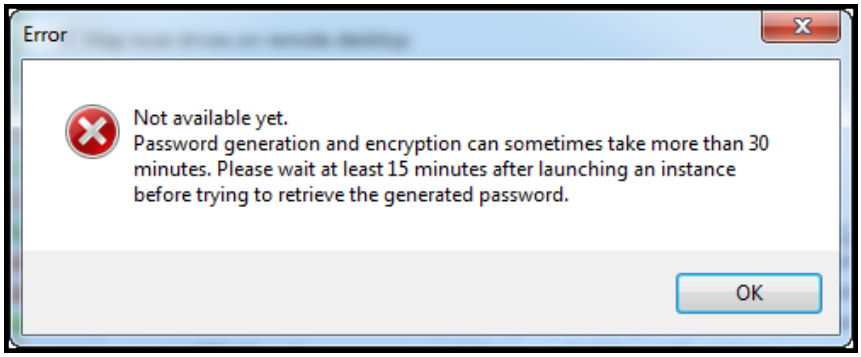


Open Remote Desktop dialog box

3. The *Remote Desktop* window will open. You do not need to sign in because authentication occurred with the key pair. You will be running as the administrator on the Amazon EC2 instance.

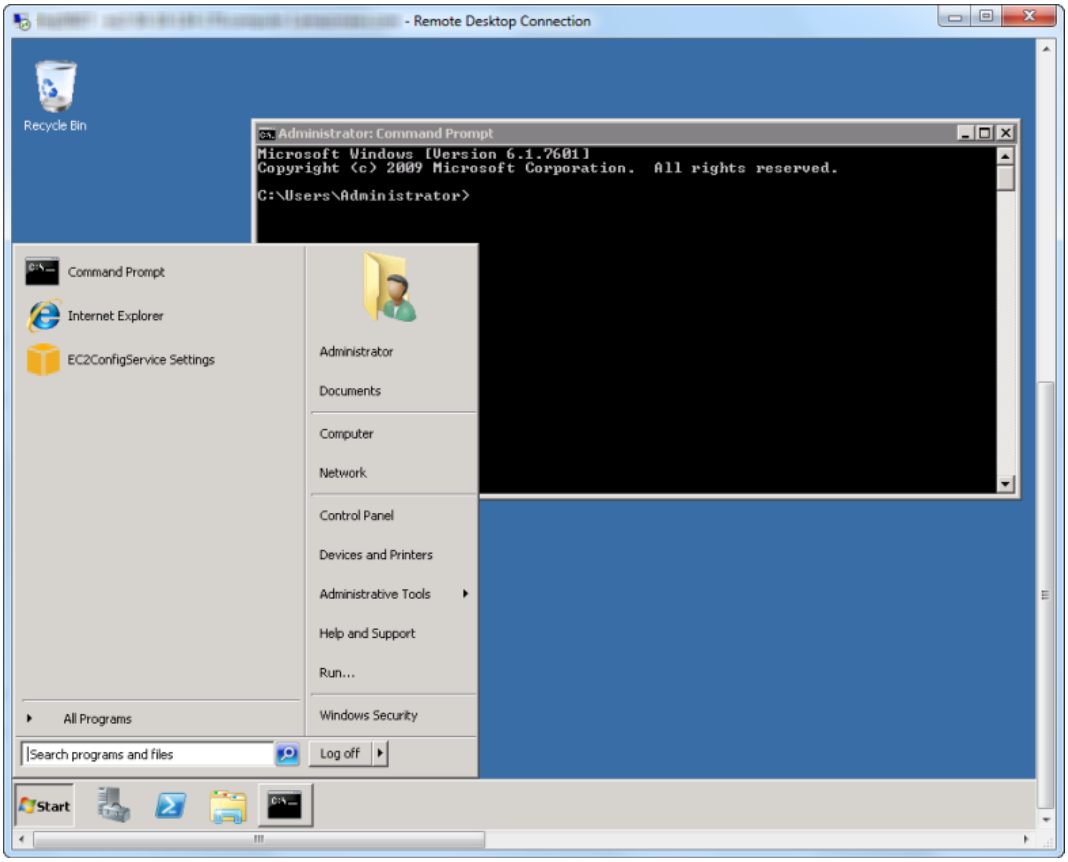
If the EC2 instance has only recently started, you may not be able to connect for two possible reasons:

- The Remote Desktop service might not yet be up and running. Wait a few minutes and try again.
- Password information might not yet have been transferred to the instance. In this case, you will see a message box similar to the following.



Password not yet available

The following screenshot shows a user connected as administrator through Remote Desktop.



Remote Desktop

3.1.4 Ending an Amazon EC2 Instance

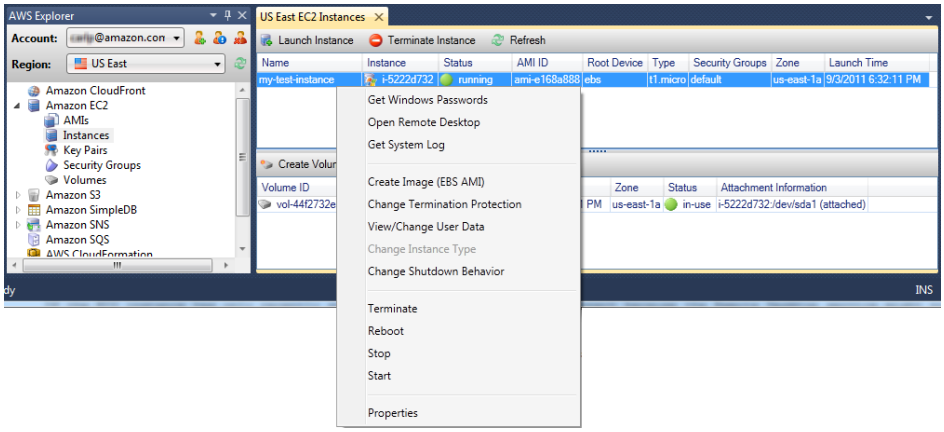
Using the AWS Toolkit, you can stop or terminate a running Amazon EC2 instance from Visual Studio. To stop the instance, the EC2 instance must be using an Amazon EBS volume. If the EC2 instance is not using an Amazon EBS volume, then your only option is to terminate the instance.

If you stop the instance, data stored on the EBS volume is retained. If you terminate the instance, all data stored on the local storage device of the instance will be lost. In either case, stop or terminate, you will not continue to be charged for the EC2 instance. However, if you stop an instance, you will continue to be charged for the EBS storage that persists after the instance is stopped.

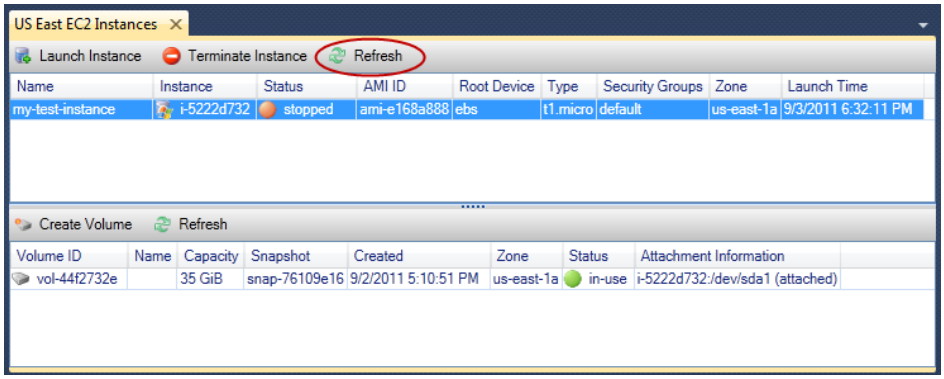
Another possible way to end an instance is to use Remote Desktop to connect to the instance, and then from the Windows *Start* menu, use *Shutdown*. You can configure the instance to either stop or terminate in this scenario.

To stop an Amazon EC2 instance

1. In AWS Explorer, expand the *Amazon EC2* node, open the context (right-click) menu for *Instances*, and then choose *View*. In the *Instances* list, right-click the instance you want to stop and choose *Stop* from the context menu. Choose *Yes* to confirm you want to stop the instance.

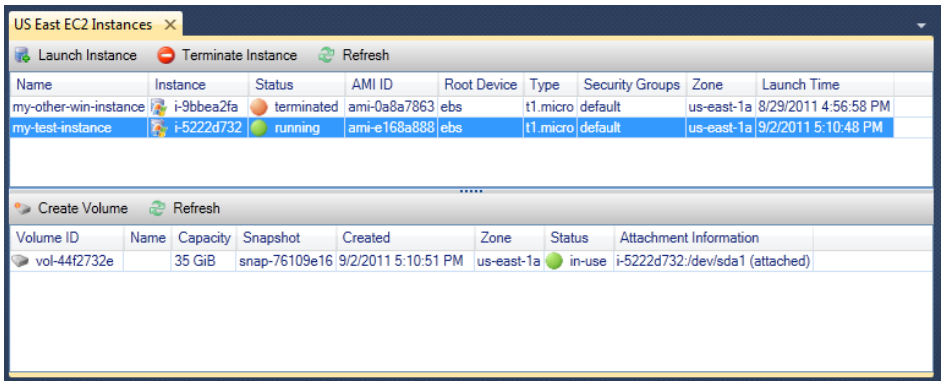


2. At the top of the *Instances* list, choose *Refresh* to see the change in the status of the Amazon EC2 instance. Because we stopped rather than terminated the instance, the EBS volume associated with the instance is still active.



Terminated Instances Remain Visible

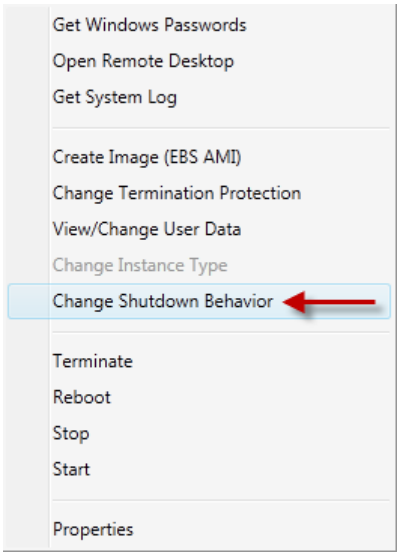
If you terminate an instance, it will continue to appear in the *Instance* list alongside running or stopped instances. Eventually, AWS reclaims these instances and they disappear from the list. You are not charged for instances in a terminated state.



To specify the behavior of an EC2 instance at shutdown

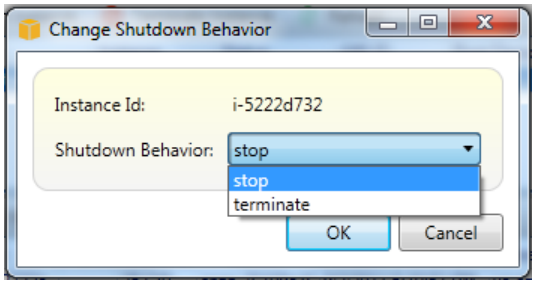
The AWS Toolkit enables you to specify whether an Amazon EC2 instance will stop or terminate if *Shutdown* is selected from the *Start* menu.

1. In the *Instances* list, right-click an Amazon EC2 instance, and then choose *Change shutdown behavior*.



Change Shutdown Behavior menu item

2. In the *Change Shutdown Behavior* dialog box, from the *Shutdown Behavior* drop-down list, choose *Stop* or *Terminate*.



3.2 Managing Security Groups from AWS Explorer

The Toolkit for Visual Studio enables you to create and configure security groups to use with Amazon Elastic Compute Cloud (Amazon EC2) instances and CloudFormation. When you launch Amazon EC2 instances or deploy an application to CloudFormation, you specify a security group to associate with the Amazon EC2 instances. (Deployment to CloudFormation creates Amazon EC2 instances.)

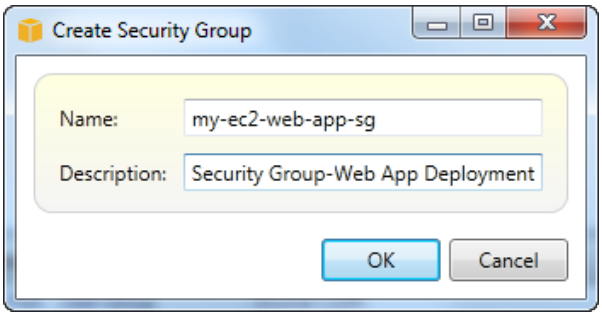
A security group acts like a firewall on incoming network traffic. The security group specifies which types of network traffic are allowed on an Amazon EC2 instance. It can also specify that incoming traffic will be accepted from certain IP addresses only or from specified users or other security groups only.

3.2.1 Creating a Security Group

In this section, we'll create a security group. After it has been created, the security group will not have any permissions configured. Configuring permissions is handled through an additional operation.

To create a security group

1. In AWS Explorer, under the *Amazon EC2* node, open the context (right-click) menu on the *Security Groups* node, and then choose *View*.
2. On the *EC2 Security Groups* tab, choose *Create Security Group*.
3. In the *Create Security Group* dialog box, type a name and description for the security group, and then choose *OK*.

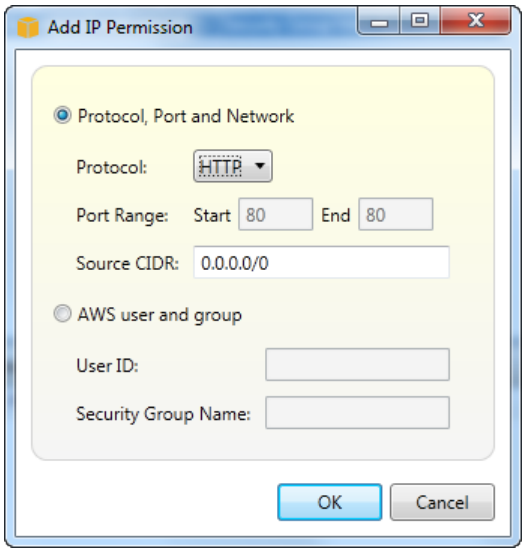


3.2.2 Adding Permissions to Security Groups

In this section, we'll add permissions to the security group to allow web traffic through the HTTP and HTTPS protocols. We'll also allow other computers to connect by using Windows Remote Desktop Protocol (RDP).

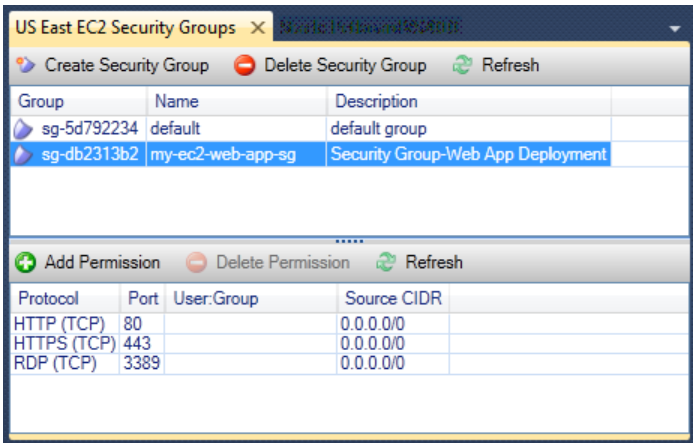
To add permissions to a security group

1. On the *EC2 Security Groups* tab, choose a security group and then choose the *Add Permission* button.
2. In the *Add IP Permission* dialog box, choose the *Protocol, Port and Network* radio button, and then from the *Protocol* drop-down list, choose *HTTP*. The port range automatically adjusts to port 80, the default port for HTTP. The *Source CIDR* field defaults to 0.0.0.0/0, which specifies that HTTP network traffic will be accepted from any external IP address. Choose *OK*.



Open port 80 (HTTP) for this security group

3. Repeat this process for HTTPS and RDP. Your security groups permissions should now look like the following.



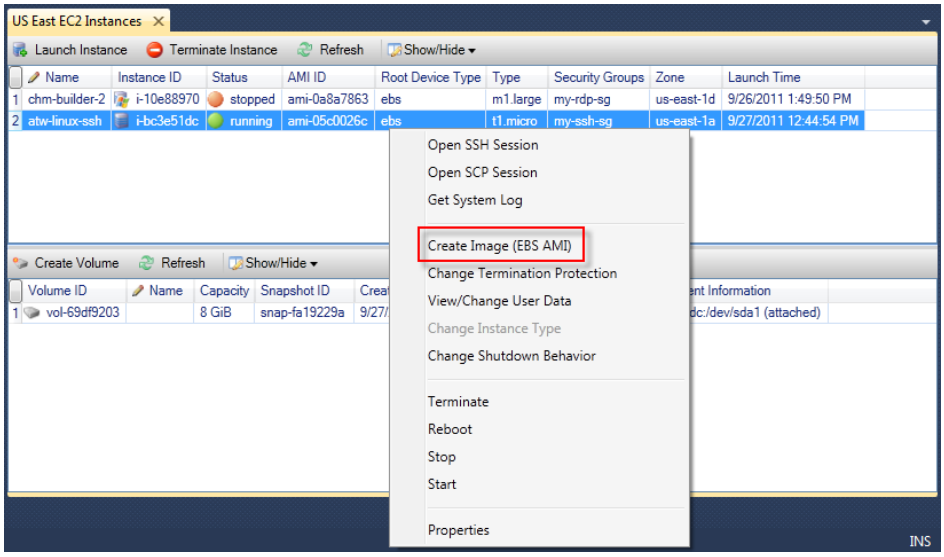
You can also set permissions in the security group by specifying a user ID and security group name. In this case, Amazon EC2 instances in this security group will accept all incoming network traffic from Amazon EC2 instances in the specified security group. You must also specify the user ID as a way to disambiguate the security group name; security group names are not required to be unique across all of AWS. For more information about security groups, go to the EC2 documentation.

3.3 Create an AMI from an Amazon EC2 Instance

From the *Amazon EC2 Instances* view, you can create Amazon Machine Images (AMIs) from either running or stopped instances.

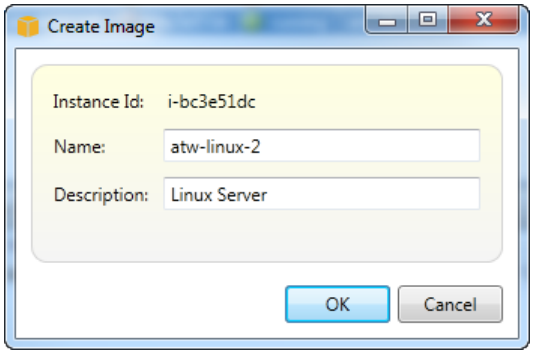
To create an AMI from an instance

1. Right-click the instance you want to use as the basis for your AMI, and choose *Create Image (EBS AMI)* from the context menu.



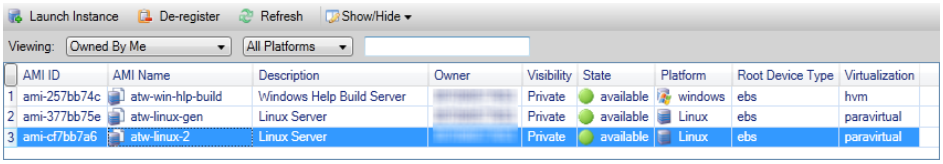
Create Image (EBS AMI) context menu

2. In the *Create Image* dialog box, type a unique name and description, and then choose *OK*.



Create Image dialog box

It may take a few minutes for the AMI to be created. After it is created, it will appear in the *AMIs* view in AWS Explorer. To display this view, double-click the *Amazon EC2 | AMIs* node in AWS Explorer. To see your AMIs, from the *Viewing* drop-down list, choose *Owned By Me*. You may need to choose *Refresh* to see your AMI. When the AMI first appears, it may be in a pending state, but after a few moments, it transitions to an available state.



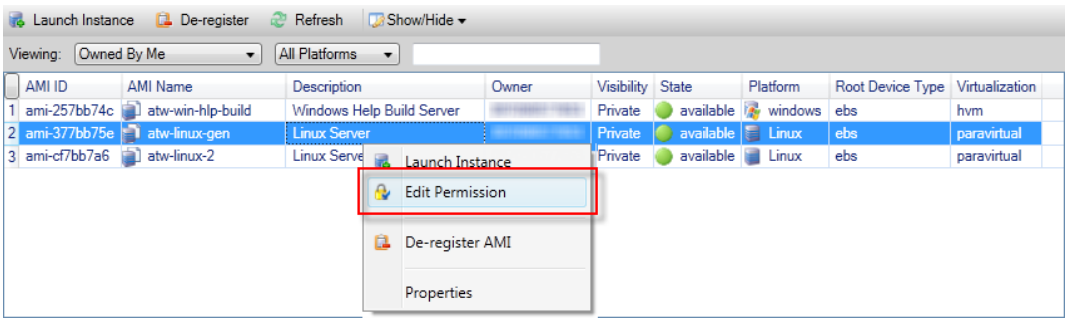
List of created AMIs

3.4 Setting Launch Permissions on an Amazon Machine Image

You can set launch permissions on your Amazon Machine Images (AMIs) from the *AMIs* view in AWS Explorer. You can use the *Set AMI Permissions* dialog box to copy permissions from AMIs.

To set permissions on an AMI

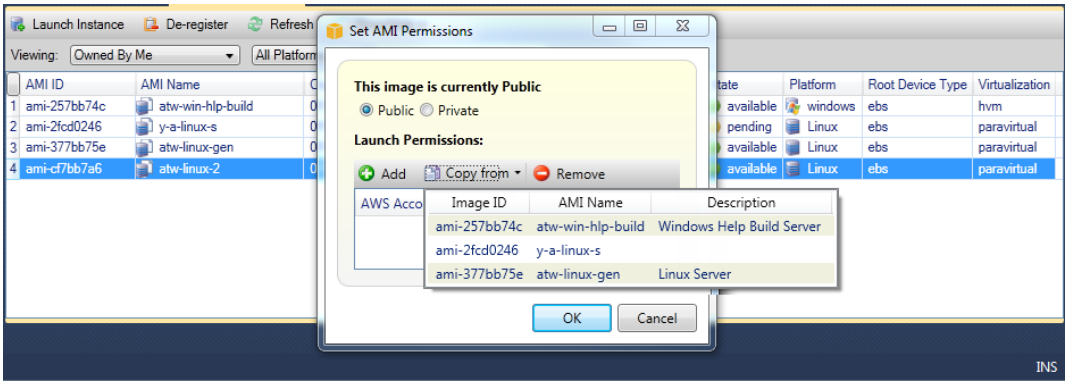
1. In the *AMIs* view in AWS Explorer, open the context (right-click) menu on an AMI, and then choose *Edit Permission*.



2. There are three options available in the *Set AMI Permissions* dialog box:

   - To give launch permission, choose *Add*, and type the account number for the AWS user to whom you are giving launch permission.
   - To remove launch permission, choose the account number for the AWS user from whom you are removing launch permission, and choose *Remove*.
   - To copy permissions from one AMI to another, choose an AMI from the list, and choose *Copy from*. The users who have launch permissions on the AMI you chose will be given launch permissions on the current AMI. You can repeat this process with other AMIs in the *Copy-from* list to copy permissions from multiple AMIs into the target AMI.

     The *Copy-from* list contains only those AMIs owned by the account that was active when the *AMIs* view was displayed from AWS Explorer. As a result, the *Copy-from* list might not display any AMIs if no other AMIs are owned by the active account.



Copy AMI permissions dialog box

3.5 Amazon Virtual Private Cloud (VPC)

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch Amazon Web Services (AWS) resources into a virtual network you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. For more information, go to the Amazon VPC User Guide.

The Toolkit for Visual Studio enables a developer to access VPC functionality similar to that exposed by the AWS Management Console but from the Visual Studio development environment. The *Amazon VPC* node of AWS Explorer includes subnodes for the following areas.

- VPCs
- Subnets
- Elastic IPs
- Internet Gateways
- Network ACLs

- Route Tables
- Security Groups

3.5.1 Walkthrough: How to Create a Public-Private VPC for Deployment with AWS Elastic Beanstalk

This section describes how to create an Amazon VPC that contains both public and private subnets. The public subnet contains an Amazon EC2 instance that performs network address translation (NAT) to enable instances in the private subnet to communicate with the public internet. The two subnets must reside in the same Availability Zone (AZ).

This is the minimal VPC configuration required to deploy an AWS Elastic Beanstalk environment in a VPC. In this scenario, the Amazon EC2 instances that host your application reside in the private subnet; the Elastic Load Balancing load balancer that routes incoming traffic to your application resides in the public subnet.

For more information about network address translation (NAT), go to NAT Instances in the *Amazon Virtual Private Cloud User Guide*. For an example of how to configure your deployment to use a VPC, see *Deploying to Elastic Beanstalk*.

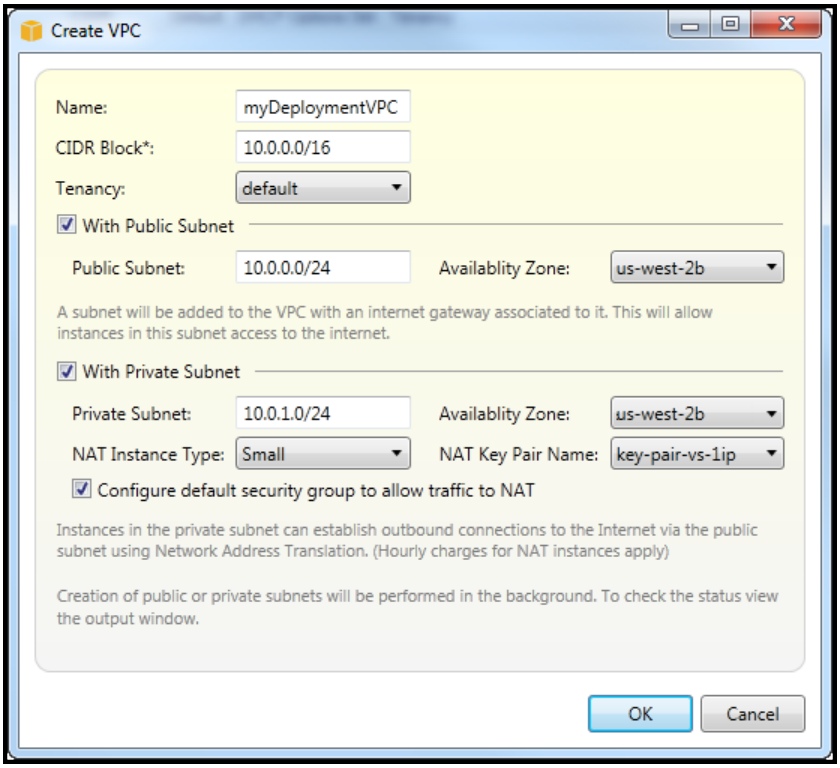
To create a public-private subnet VPC

1. In the *Amazon VPC* node in AWS Explorer, open the *VPCs* subnode, then choose *Create VPC*.

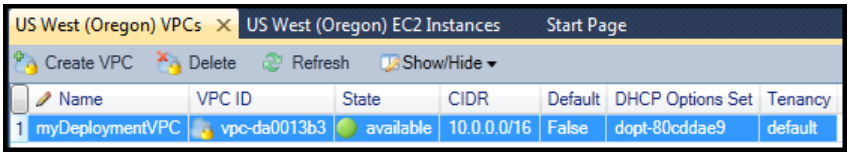
2. Configure the VPC as follows:
   - Type a name for your VPC.
   - Select the *With Public Subnet* and the *With Private Subnet* check boxes.
   - From the *Availability Zone* drop-down list box for each subnet, choose an Availability Zone. Be sure to use the same AZ for both subnets.
   - For the private subnet, in *NAT Key Pair Name*, provide a key pair. This key pair is used for the Amazon EC2 instance that performs network address translation from the private subnet to the public Internet.
   - Select the *Configure default security group to allow traffic to NAT* check box.

Type a name for your VPC. Select the *With Public Subnet* and the *With Private Subnet* check boxes. From the *Availability Zone* drop-down list box for each subnet, choose an Availability Zone. Be sure to use the same AZ for both subnets. For the private subnet, in *NAT Key Pair Name*, provide a key pair. This key pair is used for the Amazon EC2 instance that performs network address translation from the private subnet to the public Internet. Select the *Configure default security group to allow traffic to NAT* check box.

Choose *OK*.

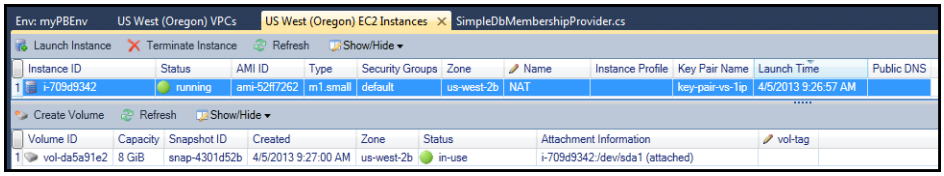


You can view the new VPC in the *VPCs* tab in AWS Explorer.



The NAT instance might take a few minutes to launch. When it is available, you can view it by expanding the *Amazon EC2* node in AWS Explorer and then opening the *Instances* subnode.

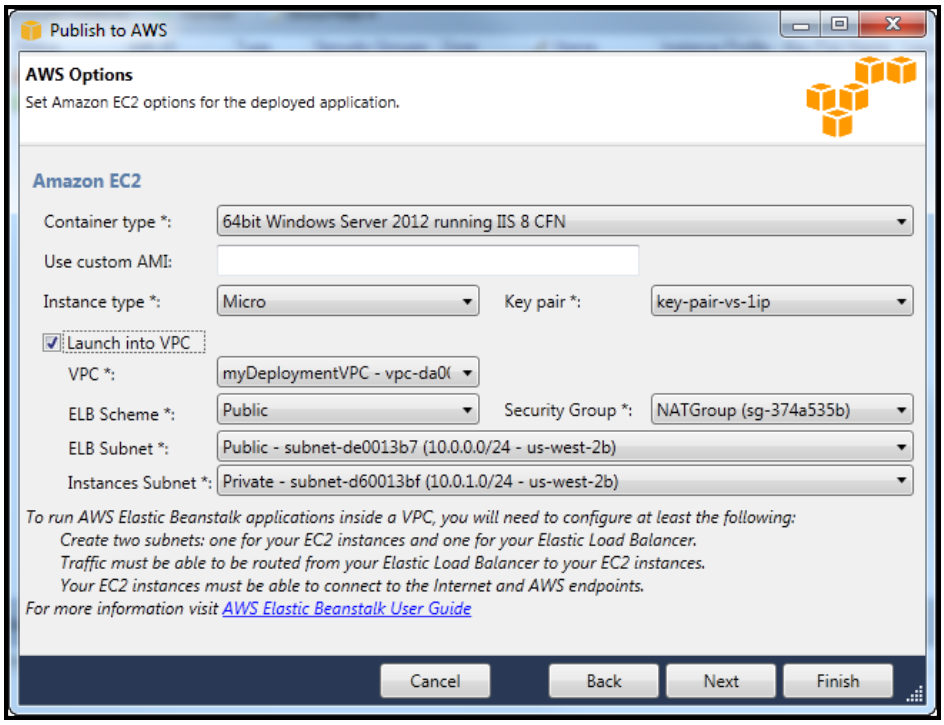
An AWS Elastic Beanstalk (Amazon EBS) volume is created for the NAT instance automatically. For more information about Elastic Beanstalk, go to AWS Elastic Beanstalk (EBS) in the Amazon EC2 User Guide for Linux Instances.



If you *deploy an application to an AWS Elastic Beanstalk environment* and choose to launch the environment in a VPC, the Toolkit will populate the *Publish to AWS* dialog box with the configuration information for your VPC.

The Toolkit populates the dialog box with information only from VPCs that were created in the Toolkit, not from VPCs created using the AWS Management Console. This is because when the Toolkit creates a VPC, it tags the components of the VPC so that it can access their information.

The following screenshot from the Deployment Wizard shows an example of a dialog box populated with values from a VPC created in the Toolkit.



To delete a VPC

To delete the VPC, you must first terminate any Amazon EC2 instances in the VPC.

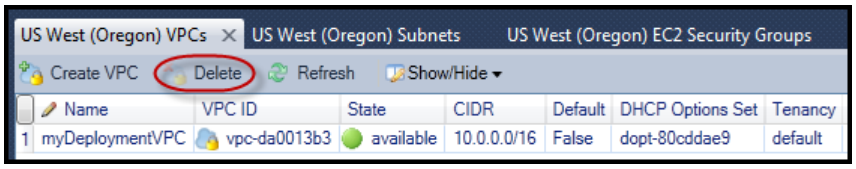
1. If you have deployed an application to an AWS Elastic Beanstalk environment in the VPC, delete the environment. This will terminate any Amazon EC2 instances hosting your application along with the Elastic Load Balancing load balancer.

   If you attempt to directly terminate the instances hosting your application without deleting the environment, the Auto Scaling service will automatically create new instances to replace the deleted ones. For more information, go to the Auto Scaling Developer Guide.

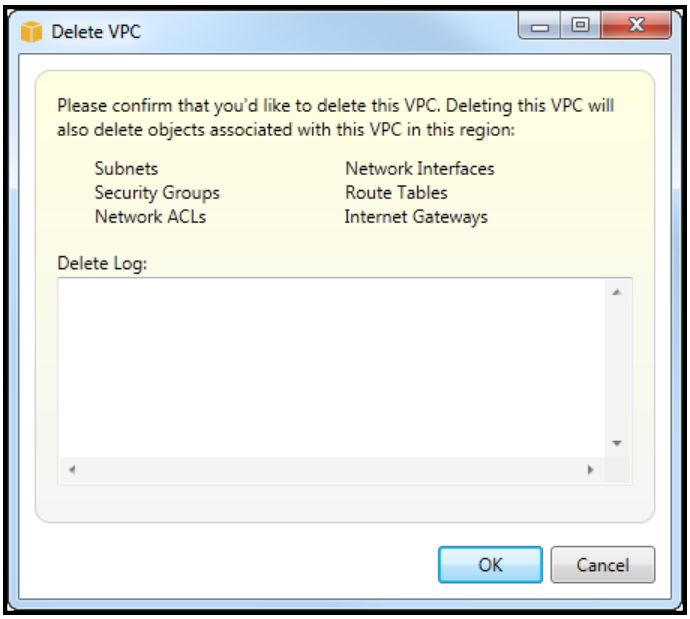
2. Delete the NAT instance for the VPC.

   You do not need to delete the Amazon EBS volume associated with the NAT instance in order to delete the VPC. However, if you do not delete the volume, you will continue to be charged for it even if you delete the NAT instance and the VPC.

3. On the *VPC* tab, choose the *Delete* link to delete the VPC.



4. In the *Delete VPC* dialog box, choose *OK*.



3.6 Deployment Using the AWS Toolkit

The Toolkit for Visual Studio supports application deployment to AWS Elastic Beanstalk containers or CloudFormation stacks.

- *Deploying to Elastic Beanstalk* describes how to use the Visual Studio IDE to deploy applications to Elastic Beanstalk.
- *Standalone Deployment Tool* describes how to use the standalone deployment tool to deploy to either Elastic Beanstalk containers or CloudFormation stacks from a command window.

Note: If you are using Visual Studio Express Edition:

- You can use the *standalone deployment tool* to deploy applications to Elastic Beanstalk containers.

- You can use the AWS Management Console to deploy applications to Elastic Beanstalk containers.

For either approach, you must first create a web deployment package. For more information, see How to: Create a Web Deployment Package in Visual Studio.

3.6.1 Deploying to Elastic Beanstalk

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. This wizard is described in the following sections.

For more information about Elastic Beanstalk, go to the Elastic Beanstalk documentation.

Deploy a Traditional ASP.NET Application to Elastic Beanstalk

This section describes how to use the *Publish to Elastic Beanstalk* wizard, provided as part of the Toolkit for Visual Studio, to deploy an application through Elastic Beanstalk. To practice, you can use an instance of a web application starter project that is built in to Visual Studio or you can use your own project.

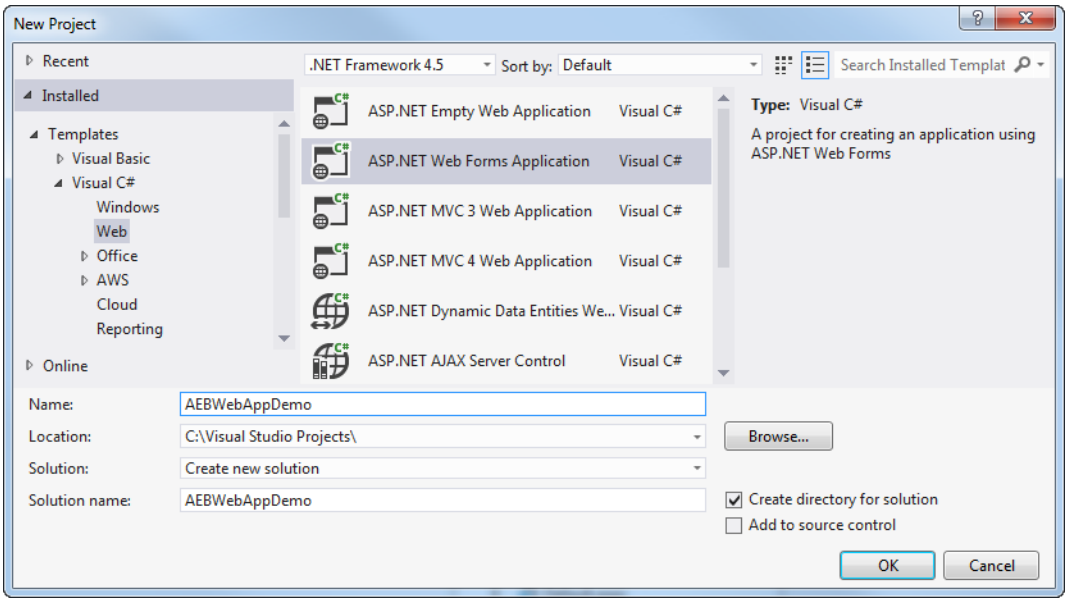
Note: This topic describes using the wizard to deploy traditional ASP.NET applications. The wizard also supports deploying ASP.NET Core applications. For information about ASP.NET Core, see *Deploying an ASP.NET Core Application to Elastic Beanstalk*.

Note: Before you can use the *Publish to Elastic Beanstalk* wizard, you must download and install Web Deploy. The wizard relies on Web Deploy to deploy web applications and websites to Internet Information Services (IIS) web servers.

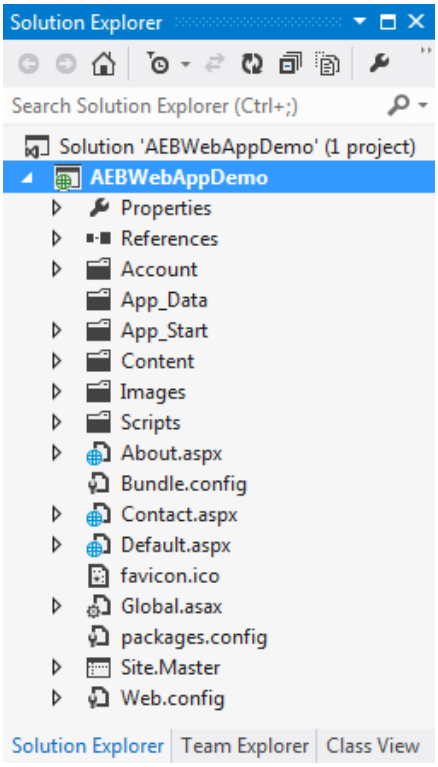
To create a sample web application starter project

1. In Visual Studio, from the *File* menu, choose *New*, and then choose *Project*.
2. In the navigation pane of the *New Project* dialog box, expand *Installed*, expand *Templates*, expand *Visual C#*, and then choose *Web*.

3. In the list of web project templates, choose any template containing the words `Web` and `Application` in its description. For this example, choose *ASP.NET Web Forms Application*.

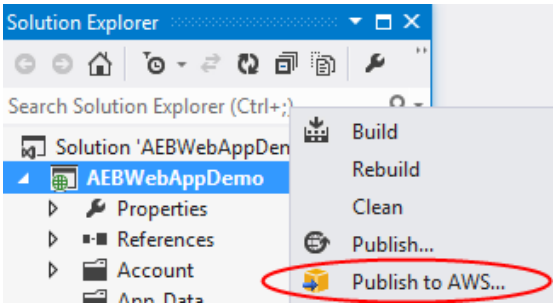


4. In the *Name* box, type `AEBWebAppDemo`.
5. In the *Location* box, type the path to a solution folder on your development machine or choose *Browse*, and then browse to and choose a solution folder, and choose *Select Folder*.
6. Confirm the *Create directory for solution* box is selected. In the *Solution* drop-down list, confirm *Create new solution* is selected, and then choose *OK*. Visual Studio will create a solution and project based on the ASP.NET Web Forms Application project template. Visual Studio will then display Solution Explorer where the new solution and project appear.

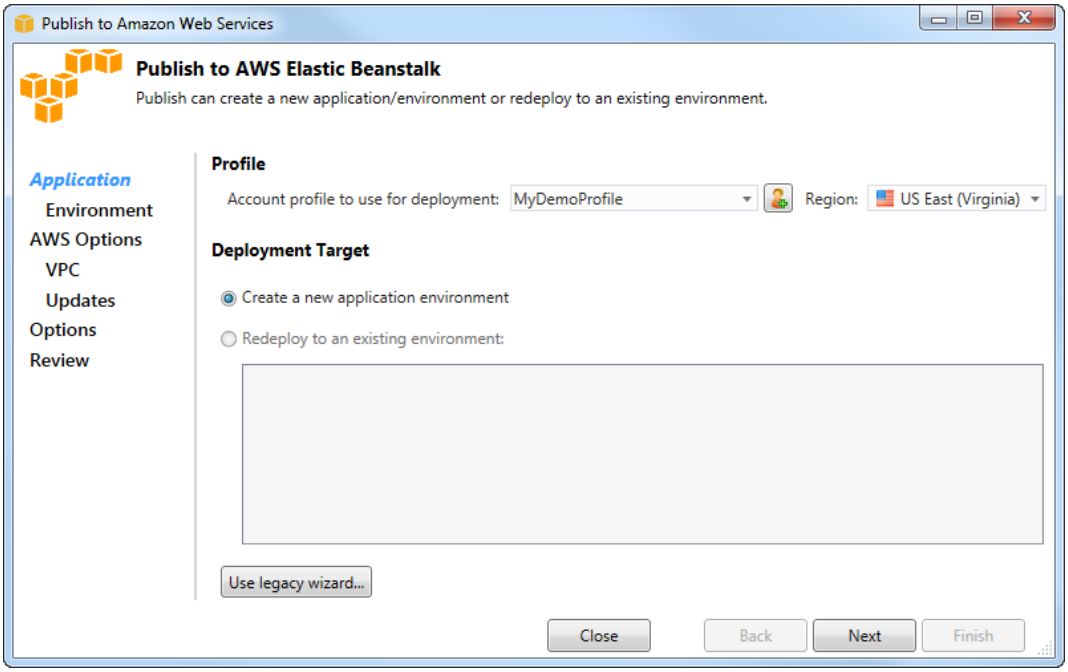


To deploy an application by using the Publish to Elastic Beanstalk wizard

1. In Solution Explorer, open the context (right-click) menu for the *AEBWebAppDemo* 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*.



   The *Publish to Elastic Beanstalk* wizard appears.



2. In *Profile*, from the *Account profile to use for deployment* drop-down list, choose the AWS account profile you want to use for the deployment.

   Optionally, if you have an AWS account you want to use, but you haven't yet created an AWS account profile for it, you can choose the button with the plus symbol (+) to add an AWS account profile.

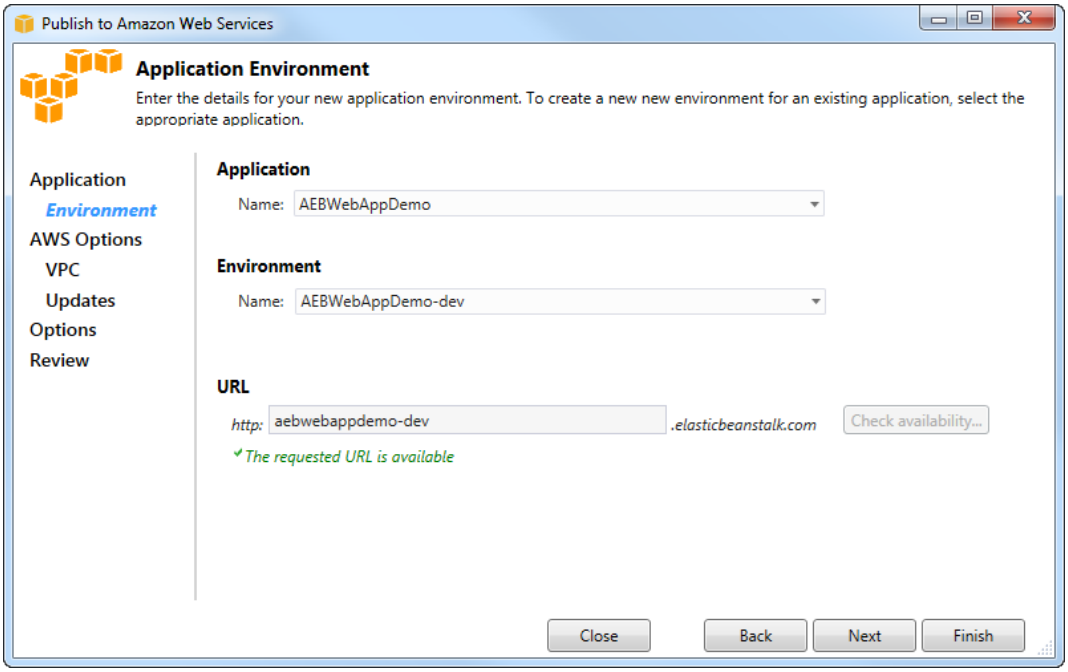
3. From the *Region* drop-down list, choose the region to which you want Elastic Beanstalk to deploy the application.

4. In *Deployment Target*, you can choose either *Create a new application environment* to perform an initial deployment of an application or *Redeploy to an existing environment* to redeploy a previously deployed application. (The previous deployments may have been performed with either the wizard or the *Standalone Deployment Tool*.) If you choose *Redeploy to an existing environment*, there may be a delay while the wizard retrieves information from previous deployments that are currently running.

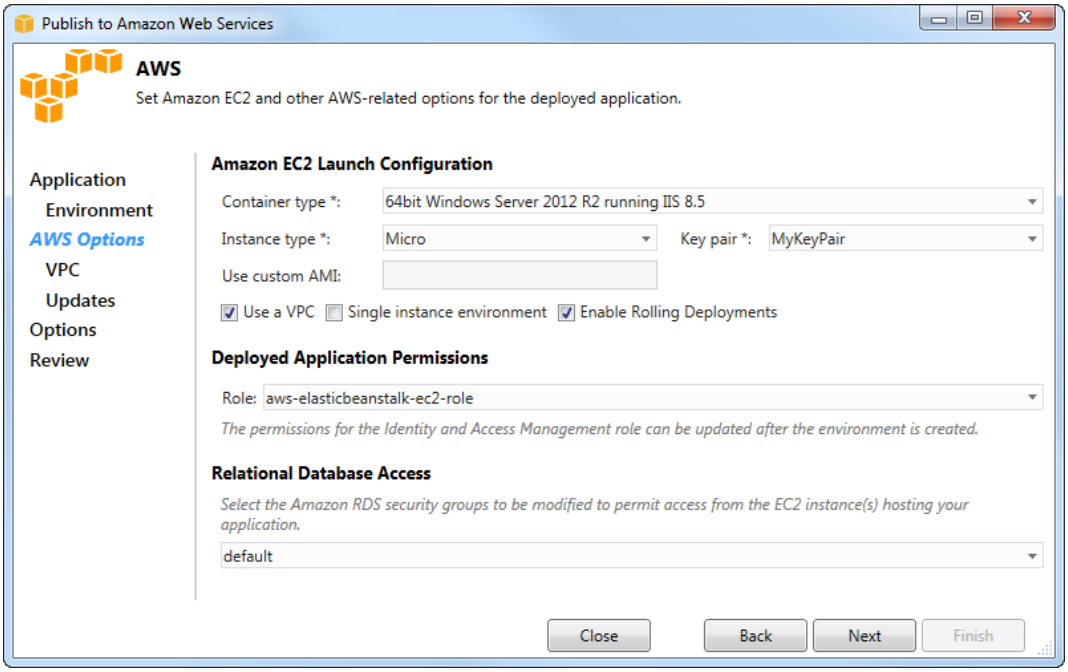
   Note: If you choose *Redeploy to an existing environment*, choose an environment in the list, and then choose *Next*, the wizard will take you directly to the *Application Options* page. If you go this route, skip ahead to the instructions later in this section that describe how to use the *Application Options* page.

   ---

5. Choose *Next*.



6. On the *Application Environment* page, in the *Application* area, the *Name* drop-down list proposes a default name for the application. You can change the default name by choosing a different name from the drop-down list.
7. In the *Environment* area, in the *Name* drop-down list, type a name for your Elastic Beanstalk environment. In this context, the term *environment* refers to the infrastructure Elastic Beanstalk provisions for your application. A default name may already be proposed in this drop-down list. If a default name is not already proposed, you can type one or choose one from the drop-down list, if any additional names are available. The environment name cannot be longer than 23 characters.
8. In the *URL* area, the box proposes a default subdomain of `.elasticbeanstalk.com` that will be the URL for your web application. You can change the default subdomain by typing a new subdomain name.
9. Choose *Check availability* to make sure the URL for your web application is not already in use.
10. If the URL for your web application is okay to use, choose *Next*.

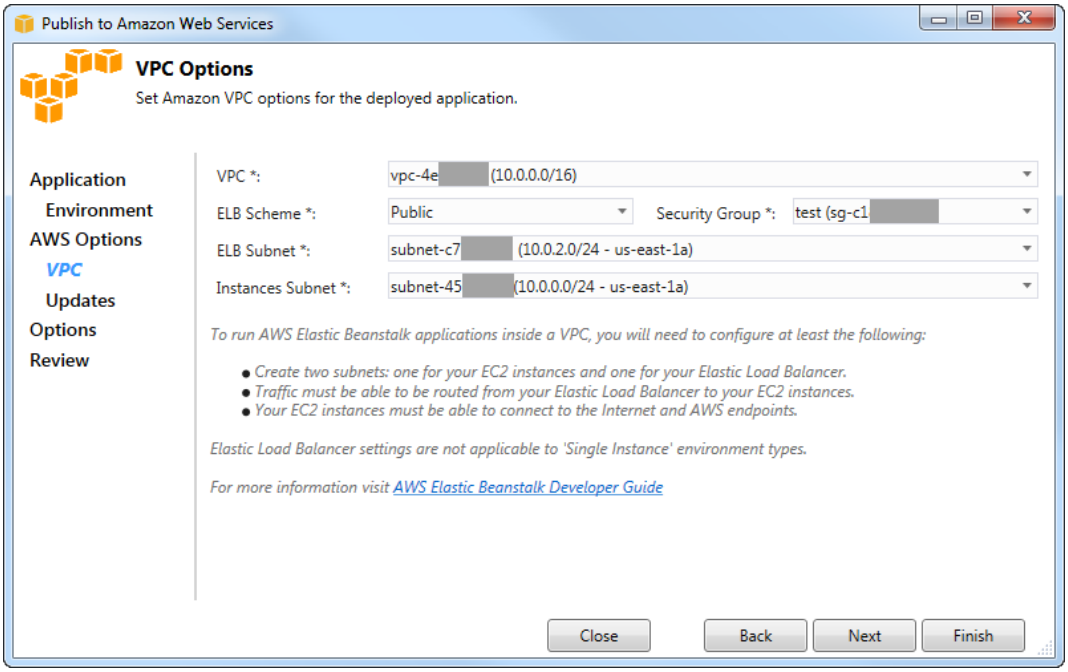


11. On the *AWS Options* page, in *Amazon EC2 Launch Configuration*, from the *Container type* drop-down list, choose an Amazon Machine Image (AMI) type that will be used for your application.
12. In the *Instance type* drop-down list, specify an Amazon EC2 instance type to use. For this example, we recommend you use *Micro*. This will minimize the cost associated with running the instance. For more information about Amazon EC2 costs, go to the *EC2 Pricing* page.
13. In the *Key pair* drop-down list, choose an Amazon EC2 instance key pair to use to sign in to the instances that will be used for your application.
14. Optionally, in the *Use custom AMI* box, you can specify a custom AMI that will override the AMI specified in the *Container type* drop-down list. For more information about how to create a custom AMI, go to Using Custom AMIs in the Elastic Beanstalk Developer Guide and *Create an AMI from an Amazon EC2 Instance*.
15. Optionally, if you want to launch your instances in a VPC, select the *Use a VPC* box.
16. Optionally, if you want to launch a single Amazon EC2 instance and then deploy your application to it, select the *Single instance environment* box.

    If you select this box, Elastic Beanstalk will still create an Auto Scaling group, but will not configure it. If you want to configure the Auto Scaling group later, you can use the AWS Management Console.
17. Optionally, if you want to control the conditions under which your application is deployed to the instances, select the *Enable Rolling Deployments* box. You can select this box only if you have not selected the *Single instance environment* box.
18. If your application uses AWS services such as Amazon S3 and DynamoDB, the best way to provide credentials is to use an IAM role. In the *Deployed Application Permissions* area, you can either choose an existing IAM role or create one the wizard will use to launch your environment.

Applications using the AWS SDK for .NET will automatically use the credentials provided by this IAM role when making a request to an AWS service.

19. If your application accesses an Amazon RDS database, in the drop-down list in the *Relational Database Access* area, select the boxes next to any Amazon RDS security groups the wizard will update so that your Amazon EC2 instances can access that database.
20. Choose *Next*.
    - If you selected *Use a VPC*, the *VPC Options* page will appear.
    - If you selected *Enable Rolling Deployments*, but did not select *Use a VPC*, the *Rolling Deployments* page will appear. Skip ahead to the instructions later in this section that describe how to use the *Rolling Deployments* page.
    - If you did not select *Use a VPC* or *Enable Rolling Deployments*, the *Application Options* page will appear. Skip ahead to the instructions later in this section that describe how to use the *Application Options* page.
21. If you selected *Use a VPC*, specify information on the *VPC Options* page to launch your application into a VPC.



The VPC must have already been created. If you created the VPC in the Toolkit for Visual Studio, the Toolkit for Visual Studio will populate this page for you. If you created the VPC in the AWS Management Console, type information about your VPC into this page.

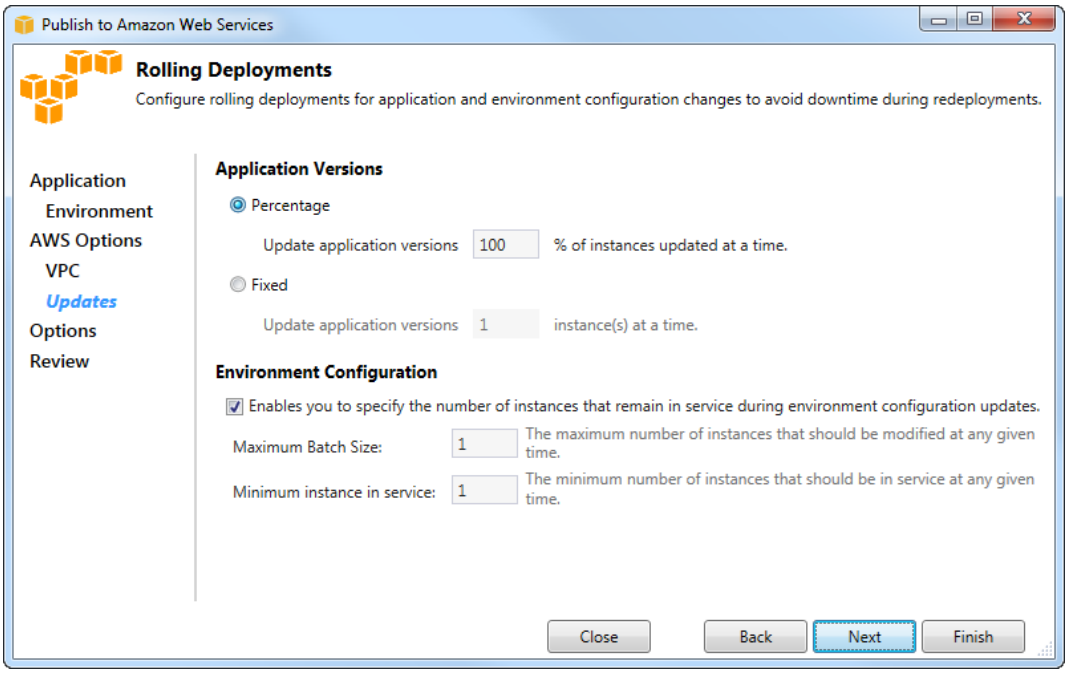
Key considerations for deployment to a VPC

- Your VPC needs at least one public and one private subnet.

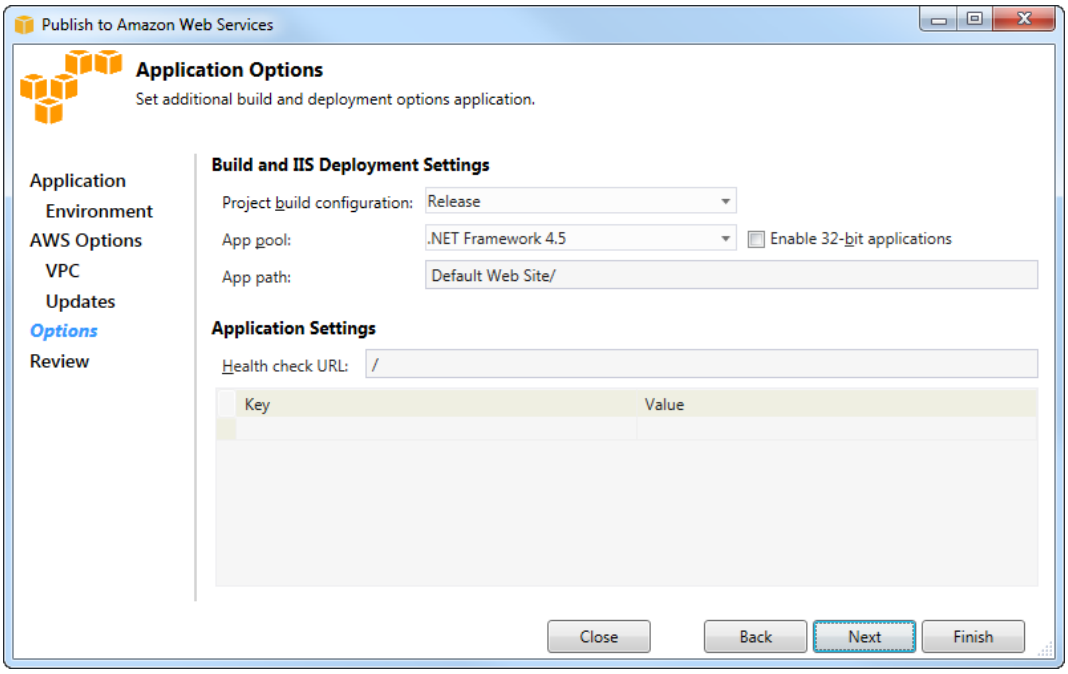
- In the *ELB Subnet* drop-down list, specify the public subnet. The Toolkit for Visual Studio deploys the Elastic Load Balancing load balancer for your application to the public subnet. The public subnet is associated with a routing table that has an entry that points to an Internet gateway. You can recognize an Internet gateway because it has an ID that begins with `igw-` (for example, `igw-83cddaex`). Public subnets that you create by using the Toolkit for Visual Studio have tag values that identify them as public.
- In the *Instances Subnet* drop-down list, specify the private subnet. The Toolkit for Visual Studio deploys the Amazon EC2 instances for your application to the private subnet.
- The Amazon EC2 instances for your application communicate from the private subnet to the Internet through an Amazon EC2 instance in the public subnet that performs network address translation (NAT). To enable this communication, you will need a VPC security group that allows traffic to flow from the private subnet to the NAT instance. Specify this VPC security group in the *Security Group* drop-down list.

For more information about how to deploy an Elastic Beanstalk application to a VPC, go to the Elastic Beanstalk Developer Guide.

22. After you have filled in all of the information on the *VPC Options* page, choose *Next*.
    - If you selected *Enable Rolling Deployments*, the *Rolling Deployments* page will appear.
    - If you did not select *Enable Rolling Deployments*, the *Application Options* page will appear. Skip ahead to the instructions later in this section that describe how to use the *Application Options* page.
23. If you selected *Enable Rolling Deployments*, you specify information on the *Rolling Deployments* page to configure how new versions of your applications are deployed to the instances in a load-balanced environment. For example, if you have four instances in your environment and you want to change the instance type, you can configure the environment to change two instances at a time. This helps ensure your application is still running while changes are being made.



24. In the *Application Versions* area, choose an option to control deployments to either a percentage or number of instances at a time. Specify either the desired percentage or number.
25. Optionally, in the *Environment Configuration* area, select the box if you want to specify the number of instances that remain in service during deployments. If you select this box, specify the maximum number of instances that should be modified at a time, the minimum number of instances that should remain in service at a time, or both.
26. Choose *Next*.
27. On the *Application Options* page, you specify information about build, Internet Information Services (IIS), and application settings.



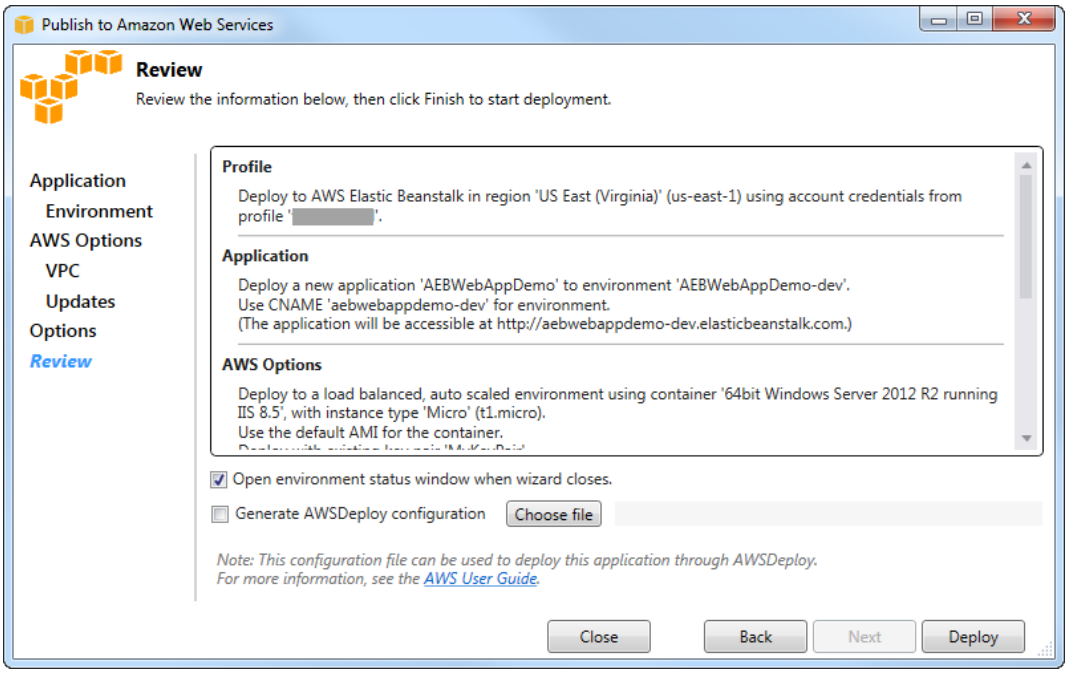
28. In the *Build and IIS Deployment Settings* area, in the *Project build configuration* drop-down list, choose the target build configuration. If the wizard can find it, *Release* appears otherwise, the active configuration is displayed in this box.
29. In the *App pool* drop-down list, choose the version of the .NET Framework required by your application. The correct .NET Framework version should already be displayed.
30. If your application is 32-bit, select the *Enable 32-bit applications* box.
31. In the *App path* box, specify the path IIS will use to deploy the application. By default, *Default Web Site/* is specified, which typically translates to the path `c:\inetpub\wwwroot`. If you specify a path other than *Default Web Site/*, the wizard will place a redirect in the *Default Web Site/* path that points to the path you specified.
32. In the *Application Settings* area, in the *Health check URL* box, type a URL for Elastic Beanstalk to check to determine if your web application is still responsive. This URL is relative to the root server URL. The root server URL is specified by default. For example, if the full URL is `example.com/site-is-up.html`, you would type `/site-is-up.html`.

33. In the area for *Key* and *Value*, you can specify any key and value pairs you want to add to your application's `Web.config` file.

Note: Although not recommended, you can use the area for *Key* and *Value*, to specify AWS credentials under which your application should run. The preferred approach is to specify an IAM role in the *Identity and Access Management Role* drop-down list on the *AWS Options* page. However, if you must use AWS credentials instead of an IAM role to run your application, in the *Key* row, choose *AWSAccessKey*. In the *Value* row, type the access key. Repeat these steps for *AWSSecretKey*.

---

34. Choose *Next*.



35. On the *Review* page, review the options you configured, and select the *Open environment status window when wizard closes* box.
36. Optionally, you can save the deployment configuration to a text file that you can then use with the *standalone deployment tool*. To save the configuration, select *Generate AWSDeploy configuration*, choose *Choose File*, and then specify a file to which to save the configuration. You can also save the deployment configuration to a text file after the deployment is complete. In AWS Explorer, open the context (right-click) menu for the deployment and then choose *Save Configuration*.
37. If everything looks correct, choose *Deploy*.

Note: When you deploy the application, the active account will incur charges for the AWS resources used by the application.

Information about the deployment will appear in the Visual Studio status bar and the *Output* window. It may take several minutes. When the deployment is complete, a confirmation message will appear

in the *Output* window.

38. To delete the deployment, in AWS Explorer, expand the *Elastic Beanstalk* node, open the context (right-click) menu for the subnode for the deployment, and then choose *Delete*. The deletion process might take a few minutes.

Deploying an ASP.NET Core Application to Elastic Beanstalk

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

The Toolkit for Visual Studio supports deploying ASP.NET Core applications to AWS using Elastic Beanstalk. ASP.NET Core is the redesign of ASP.NET with a modularized architecture that minimizes dependency overhead and streamlines your application to run in the cloud.

AWS Elastic Beanstalk makes it easy to deploy applications in a variety of different languages to AWS. Elastic Beanstalk supports both traditional ASP.NET applications and ASP.NET Core applications. This topic describes deploying ASP.NET Core applications.

Using the Deployment Wizard

The easiest way to deploy ASP.NET Core applications to Elastic Beanstalk is with the Toolkit for Visual Studio.

If you have used the toolkit before to deploy traditional ASP. NET applications, you'll find the experience for ASP.NET Core to be very similar. In the steps below, we'll walk through the deployment experience.

If you have never used the toolkit before, the first thing you'll need to do after installing the toolkit is register your AWS credentials with the toolkit. See *How to Specify the AWS Security Credentials for Your Application* for Visual Studio documentation for details on how to do so.

To deploy an ASP.NET Core web application, right-click the project in the Solution Explorer and select *Publish to AWS...*.

On the first page of the Publish to AWS Elastic Beanstalk deployment wizard, choose to create a new Elastic Beanstalk application. An Elastic Beanstalk application is a logical collection of Elastic Beanstalk components, including environments, versions, and environment configurations. The deployment wizard generates an application that in turn contains a collection of application versions and environments. The environments contain the actual AWS resources that run an application version. Every time you deploy an application, a new application version is created and the wizard points the environment to that version. You can learn more about these concepts in Elastic Beanstalk Components..

Next, set names for the application and its first environment. Each environment has a unique CNAME associated with it that you can use to access the application when the deployment is complete.

The next page, **AWS Options**, allows you to configure the type of AWS resources to use. For this example, leave the default values, except for the *Key pair* section. Key pairs allow you retrieve the Windows administrator password so you can log on to the machine. If you haven't already created a key pair you might want to select *Create new key pair*.

Permissions

The **Permissions** page is used for assigning AWS credentials to the EC2 instance(s) running your application. This is important if your application uses the AWS SDK for .NET to access other AWS services. If you are not using any other services from your application then you can leave this page at its default.

Application Options

The details on the **Application Options** page are different from those specified when deploying traditional ASP.NET applications. Here, you specify the build configuration and framework used to package the application, and also specify the IIS resource path for the application.

After completing the **Application Options** page, click *Next* to review the settings, then click *Deploy* to begin the deployment process.

Checking Environment Status

After the application is packaged and uploaded to AWS, you can check the status of the Elastic Beanstalk environment by opening the environment status view from the AWS Explorer in Visual Studio.

Events are displayed in the status bar as the environment is coming online. Once everything is complete, the environment status will move to healthy state. You can click on the URL to view the site. From here, you can also pull the logs from the environment, or remote desktop into the Amazon EC2 instance(s) that are part of your Elastic Beanstalk environment.

The first deployment of any application will take a bit longer than subsequent re-deployments, as it creates new AWS resources. As you iterate on your application during development, you can quickly re-deploy by going back through the wizard, or selecting the *Republish* option when you right click the project.

Republish packages your application using the settings from the previous run through the deployment wizard and uploads the application bundle to the existing Elastic Beanstalk environment.

How to Specify the AWS Security Credentials for Your Application

The AWS account you specify in the *Publish to Elastic Beanstalk* wizard (or the legacy version of this wizard, *Publish to Amazon Web Services*) is the AWS account the wizard will use for deployment to Elastic Beanstalk.

Although not recommended, you may also need to specify AWS account credentials that your application will use to access AWS services after it has been deployed. The preferred approach is to specify an IAM role. In the *Publish to Elastic Beanstalk* wizard, you do this through the *Identity and Access Management Role* drop-down list on the *AWS Options* page. In the legacy *Publish to Amazon Web Services* wizard, you do this through the *IAM Role* drop-down list on the *AWS Options* page.

If you must use AWS account credentials instead of an IAM role, you can specify the AWS account credentials for your application in one of the following ways:

- Reference a profile corresponding to the AWS account credentials in the `appSettings` element of the project's `Web.config` file. (To create a profile, see Configuring AWS Credentials.) The following example specifies credentials whose profile name is `myProfile`.

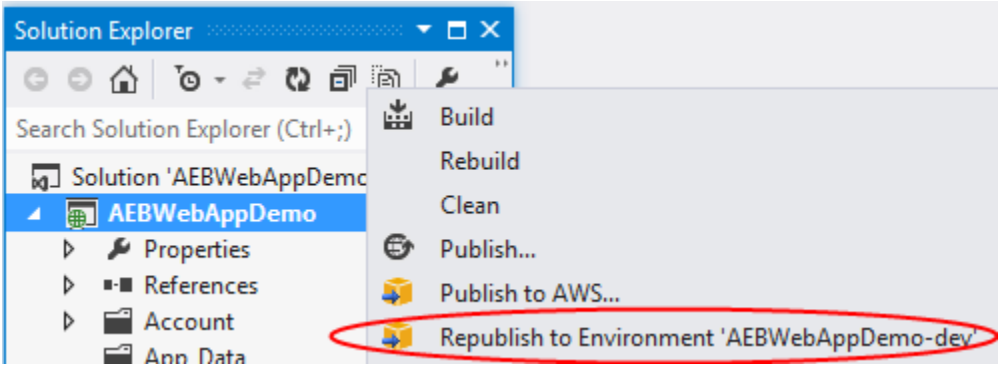
```
<appSettings>
  <!-- AWS CREDENTIALS -->
  <add key="AWSProfileName" value="myProfile"/>
</appSettings>
```

- If you're using the *Publish to Elastic Beanstalk* wizard, on the *Application Options* page, in the *Key* row of the *Key* and *Value* area, choose *AWSAccessKey*. In the *Value* row, type the access key. Repeat these steps for *AWSSecretKey*.
- If you're using the legacy *Publish to Amazon Web Services* wizard, on the *Application Options* page, in the *Application Credentials* area, choose *Use these credentials*, and then type the access key and secret access key into the *Access Key* and *Secret Key* boxes.

How to Republish Your Application to an Elastic Beanstalk Environment

You can iterate on your application by making discrete changes and then republishing a new version to your already launched Elastic Beanstalk environment.

1. In Solution Explorer, open the context (right-click) menu for the project node and then choose *Republish to Environment '<environment_name>'*.



2. On the page that appears, choose *Deploy*. The new version of your application will be published to the current environment.

When you republish, you do not have the option to use a new or different environment. If you need to change any aspect of your deployment other than the options offered on the settings page, you must choose *Publish to AWS* (instead of *Republish to Environment '<environment_name>'*) in step 1 and use the deployment wizard to make changes.

You cannot republish if your application is in the process of launching or terminating.

Custom Elastic Beanstalk Application Deployments

This topic describes how the deployment manifest for Elastic Beanstalk's Microsoft Windows container supports custom application deployments.

Custom application deployments are a powerful feature for advanced users who want to leverage the power of Elastic Beanstalk to create and manage their AWS resources, but want complete control on how their application is deployed. For a custom application deployment, you create Windows PowerShell scripts for the three different actions Elastic Beanstalk performs. The install action is used when a deployment is initiated, restart is used when the `RestartAppServer` API is called from either the toolkit or the web console, and uninstall which is invoked on any previous deployment whenever a new deployment occurs.

For example, you might have an ASP.NET application that you want to deploy while your documentation team has written a static website that they want included with the deployment. You can do that by writing your deployment manifest like this:

```
{
  "manifestVersion": 1,
  "deployments": {

    "msDeploy": [
      {
        "name": "app",
        "parameters": {
          "appBundle": "CoolApp.zip",
          "iisPath": "/"
        }
      }
    ],
    "custom": [
      {
        "name": "PowerShellDocs",
        "scripts": {
          "install": {
            "file": "install.ps1"
          },
          "restart": {
            "file": "restart.ps1"
          },
          "uninstall": {
            "file": "uninstall.ps1"
          }
        }
      }
    ]
  }
}
```

The scripts listed for each action must be in the application bundle relative to the deployment manifest file. For this example, the application bundle will also contain a documentation.zip file which contains a static website created by your documentation team.

The `install.ps1` script extracts the zip file and sets up the IIS Path.

```
Add-Type -assembly "system.io.compression.filesystem"
[io.compression.zipfile]::ExtractToDirectory('./documentation.zip', 'c:
↪\inetpub\wwwroot\documentation')

powershell.exe -Command {New-WebApplication -Name documentation -PhysicalPath␣
↪ c:\inetpub\wwwroot\documentation -Force}
```

Since your application is running in IIS, the restart action will invoke an IIS reset.

```
iisreset /timeout:1
```

For uninstall scripts, it is important to clean up all settings and files used during the install stage. That way during the install phase for the new version, you can avoid any collision with previous deployments. For this example, you need to remove the IIS application for the static website and remove the website files.

```
powershell.exe -Command {Remove-WebApplication -Name documentation}
Remove-Item -Recurse -Force 'c:\inetpub\wwwroot\documentation'
```

With these script files and the documentation.zip file included in your application bundle, the deployment creates the ASP.NET application and then deploys the documentation site.

For this example, we choose a simple example that deploys a simple static website, but with custom application deployment you can deploy any type of application and let Elastic Beanstalk manage the AWS resources for it.

Custom ASP.NET Core Elastic Beanstalk Deployments

This topic describes how deployment works and what you can do customize deployments when creating ASP.NET Core applications with Elastic Beanstalk and the Toolkit for Visual Studio.

After you complete the deployment wizard in the Toolkit for Visual Studio, the toolkit bundles the application and sends it to Elastic Beanstalk. Your first step in creating the application bundle is to use the new dotnet CLI to prepare the application for publishing by using the **publish** command. The framework and configuration are passed down from the settings in the wizard to the **publish** command. So if you selected **Release** for `configuration` and **netcoreapp1.0** for the `framework`, the toolkit will execute the following command:

```
dotnet publish --configuration Release --framework netcoreapp1.0
```

When the **publish** command finishes, the toolkit writes the new deployment manifest into the publishing folder. The deployment manifest is a JSON file named **aws-windows-deployment-manifest.json**, which the Elastic Beanstalk Windows container (version 1.2 or later) reads to determine how to deploy the application. For example, for an ASP.NET Core application you want to be deploy at the root of IIS, the toolkit generates a manifest file that looks like this:

```
{
  "manifestVersion": 1,
  "deployments": {

    "aspNetCoreWeb": [
      {
```

```
        "name": "app",
        "parameters": {
          "appBundle": ".",
          "iisPath": "/",
          "iisWebSite": "Default Web Site"
        }
      }
    ]
  }
}
```

The `appBundle` property indicates where the application bits are in relation to the manifest file. This property can point to either a directory or a ZIP archive. The `iisPath` and `iisWebSite` properties indicate where in IIS to host the application.

Customizing the Manifest

The toolkit only writes the manifest file if one doesn't already exist in the publishing folder. If the file does exist, the toolkit updates the `appBundle`, `iisPath` and `iisWebSite` properties in the first application listed under the `aspNetCoreWeb` section of the manifest. This allows you to add the **aws-windows-deployment-manifest.json** to your project and customize the manifest. To do this for an ASP.NET Core Web application in Visual Studio add a new JSON file to the root of the project and name it **aws-windows-deployment-manifest.json**.

The manifest must be named **aws-windows-deployment-manifest.json** and it must be at the root of the project. The Elastic Beanstalk container looks for the manifest in the root and if it finds it will invoke the deployment tooling. If the file doesn't exist, the Elastic Beanstalk container falls back to the older deployment tooling, which assumes the archive is an **msdeploy** archive.

To ensure the dotnet CLI `publish` command includes the manifest, update the `project.json` file to include the manifest file in the include section under `include` in `publishOptions`.

```
{
   "publishOptions": {
     "include": [
       "wwwroot",
       "Views",
       "Areas/**/Views",
       "appsettings.json",
       "web.config",
       "aws-windows-deployment-manifest.json"
     ]
   }
 }
```

Now that you've declared the manifest so that it's included in the app bundle, you can further configure how you want to deploy the application. You can customize deployment beyond what the deployment wizard supports. AWS has defined a JSON schema for the **aws-windows-deployment-manifest.json file**, and when you installed the Toolkit for Visual Studio, the setup registered the URL for the schema.

When you open `windows-deployment-manifest.json`, you'll see the schema URL selected in the Schema drop down box. You can navigate to the URL to get a full description of what can be set in the manifest. With the schema selected, Visual Studio will provide IntelliSense while you're editing the manifest.

One customization you can do is to configure the IIS application pool under which the application will run. The following example shows how you can define an IIS Application pool ("customPool") that recycles the process every 60 minutes, and assigns it to the application using `"appPool": "customPool"`.

```
{
  "manifestVersion": 1,
  "iisConfig": {
    "appPools": [
      {
        "name": "customPool",
        "recycling": {
          "regularTimeInterval": 60
        }
      }
    ]
  },
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "app",
        "parameters": {
          "appPool": "customPool"
        }
      }
    ]
  }
}
```

Additionally, the manifest can declare Windows PowerShell scripts to run before and after the install, restart and uninstall actions. For example, the following manifest runs the Windows PowerShell script `PostInstallSetup.ps1` to do further setup work after the ASP.NET Core application is deployed to IIS. When adding scripts like this, make sure the scripts are added to the include section under publishOptions in the `project.json` file, just as you did with the `aws-windows-deployment-manifest.json` file. If you don't, the scripts won't be included as part of the dotnet CLI **publish** command.

```
{
  "manifestVersion": 1,
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "app",
        "scripts": {
          "postInstall": {
            "file": "SetupScripts/PostInstallSetup.ps1"
          }
        }
```

```
        }
      ]
    }
}
```

What about ebextensions?

The Elastic Beanstalk **ebextensions** configuration files are supported as with all the other Elastic Beanstalk containers. To include ebextensions in an ASP.NET Core application, add the `.ebextensions` directory to the `include` section under `publishOptions` in the `project.json` file. For further information about ebextensions checkout the Elastic Beanstalk Developer Guide.

Multiple Application Support for .NET and Elastic Beanstalk

Using the deployment manifest, you have the ability to deploy multiple applications to the same Elastic Beanstalk environment.

The deployment manifest supports ASP.NET Core web applications as well as msdeploy archives for traditional ASP.NET applications. Imagine a scenario where you have written a new amazing application using ASP.NET Core for the frontend and a Web API project for an extensions API. You also have an admin app that you wrote using traditional ASP.NET.

The toolkit's deployment wizard focuses on deploying a single project. To take advantage of multiple application deployment, you have to construct the application bundle by hand. To start, write the manifest. For this example, you will write the manifest at the root of your solution.

The deployment section in the manifest has two children: an array of ASP.NET Core web applications to deploy, and an array of msdeploy archives to deploy. For each application, you set the IIS path and the location of the application's bits relative to the manifest.

```
{
  "manifestVersion": 1,
  "deployments": {

    "aspNetCoreWeb": [
      {
        "name": "frontend",
        "parameters": {
          "appBundle": "./frontend",
          "iisPath": "/frontend"
        }
      },
      {
        "name": "ext-api",
        "parameters": {
          "appBundle": "./ext-api",
          "iisPath": "/ext-api"
        }
      }
```

```
    ],
    "msDeploy": [
      {
        "name": "admin",
        "parameters": {
          "appBundle": "AmazingAdmin.zip",
          "iisPath": "/admin"
        }
      }
    ]
  }
}
```

With the manifest written, you'll use Windows PowerShell to create the application bundle and update an existing Elastic Beanstalk environment to run it. The script is written assuming that it will be run from the folder containing your Visual Studio solution.

The first thing you need to do in the script is setup a workspace folder in which to create the application bundle.

```
$publishFolder = "c:\temp\publish"

$publishWorkspace = [System.IO.Path]::Combine($publishFolder, "workspace")
$appBundle = [System.IO.Path]::Combine($publishFolder, "app-bundle.zip")

If (Test-Path $publishWorkspace){
     Remove-Item $publishWorkspace -Confirm:$false -Force
}
If (Test-Path $appBundle){
     Remove-Item $appBundle -Confirm:$false -Force
}
```

Once you've created the folder, it is time to get the frontend ready. As with the deployment wizard, use the dotnet CLI to publish the application.

```
Write-Host 'Publish the ASP.NET Core frontend'
$publishFrontendFolder = [System.IO.Path]::Combine($publishWorkspace,
 ↪"frontend")
dotnet publish .\src\AmazingFrontend\project.json -o $publishFrontendFolder -
 ↪c Release -f netcoreapp1.0
```

Notice that the subfolder "frontend" was used for the output folder, matching the folder you set in the manifest. Now you need to do the same for the Web API project.

```
Write-Host 'Publish the ASP.NET Core extensibility API'
$publishExtAPIFolder = [System.IO.Path]::Combine($publishWorkspace, "ext-api")
dotnet publish .\src\AmazingExtensibleAPI\project.json -o
 ↪$publishExtAPIFolder -c Release -f netcoreapp1.0
```

The admin site is a traditional ASP.NET application, so you can't use the dotnet CLI. For the admin application, you should use msbuild, passing in the build target package to create the msdeploy archive. By default the package target creates the msdeploy archive under the `obj\Release\Package` folder, so

you will need to copy the archive to the publish workspace.

```
Write-Host 'Create msdeploy archive for admin site'
msbuild .\src\AmazingAdmin\AmazingAdmin.csproj /t:package /p:
 ↪Configuration=Release
Copy-Item .\src\AmazingAdmin\obj\Release\Package\AmazingAdmin.zip
 ↪$publishWorkspace
```

To tell the Elastic Beanstalk environment what to do with all these applications, copy the manifest from your solution to the publish workspace and then zip up the folder.

```
Write-Host 'Copy deployment manifest'
Copy-Item .\aws-windows-deployment-manifest.json $publishWorkspace

Write-Host 'Zipping up publish workspace to create app bundle'
Add-Type -assembly "system.io.compression.filesystem"
[io.compression.zipfile]::CreateFromDirectory( $publishWorkspace, $appBundle)
```

Now that you have the application bundle, you could go to the web console and upload the archive to a Elastic Beanstalk environment. Alternatively, you can continue to use the AWS PowerShell cmdlets to update the Elastic Beanstalk environment with the application bundle. Make sure you have set the current profile and region to the profile and region that contains your Elastic Beanstalk environment by using `Set-AWSCredentials` and `Set-DefaultAWSRegion` cmdlets.

```
Write-Host 'Write application bundle to S3'
# Determine S3 bucket to store application bundle
$s3Bucket = New-EBStorageLocation
Write-S3Object -BucketName $s3Bucket -File $appBundle


$applicationName = "ASPNETCoreOnAWS"
$environmentName = "ASPNETCoreOnAWS-dev"
$versionLabel = [System.DateTime]::Now.Ticks.ToString()

Write-Host 'Update Beanstalk environment for new application bundle'
New-EBApplicationVersion -ApplicationName $applicationName -VersionLabel
 ↪$versionLabel -SourceBundle_S3Bucket $s3Bucket -SourceBundle_S3Key app-
 ↪bundle.zip
Update-EBEnvironment -ApplicationName $applicationName -EnvironmentName
 ↪$environmentName -VersionLabel $versionLabel
```

Now, check the status of the update using either the Elastic Beanstalk environment status page in either the toolkit or the web console. Once complete you will be able to navigate to each of the applications you deployed at the IIS path set in the deployment manifest.

Deploying to CloudFormation (Legacy)

Note: The information in this topic refers to the *Publish to Amazon Web Services* wizard, which has been replaced by deploying through Elastic Beanstalk through the use of the *Publish to Elastic Beanstalk*

wizard. The following information is provided for those who prefer to, or must, use the legacy wizard to deploy through CloudFormation.

For information about using the preferred *Publish to Elastic Beanstalk* wizard, see *Deploying to Elastic Beanstalk*.

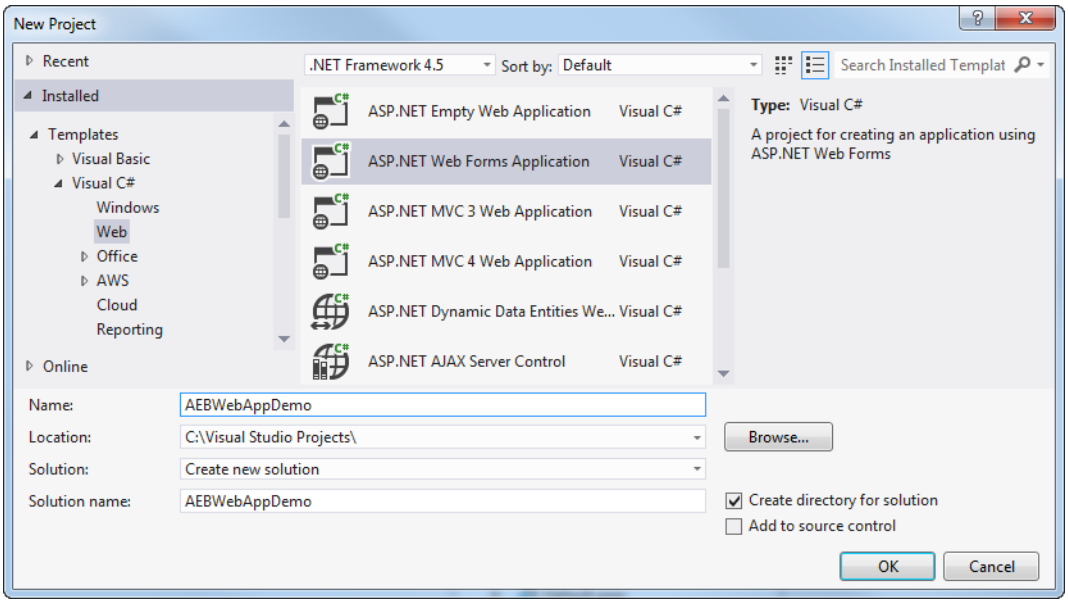
CloudFormation is a service that simplifies the process of provisioning AWS resources for your application. The AWS resources are described in a template file. The CloudFormation service consumes this template and automatically provisions the required resources for you. For more information, go to the CloudFormation documentation.

We'll deploy an application to AWS and use CloudFormation to provision the resources for the application. To practice, you can use an instance of a web application starter project that is built in to Visual Studio or you can use your own project.

To create a sample web application starter project

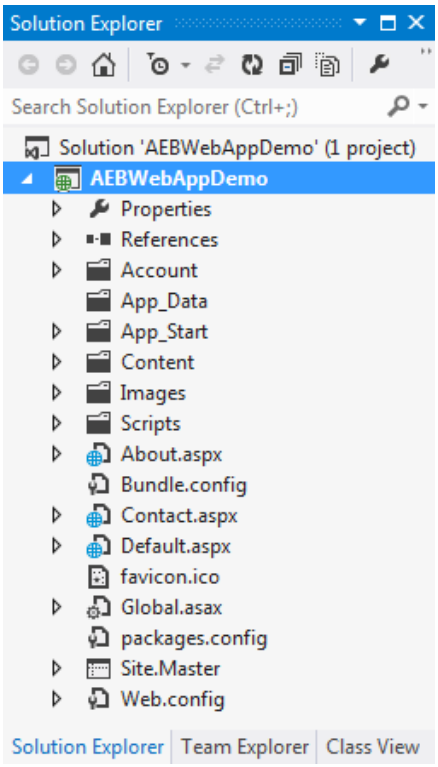
Follow these steps if you do not have project ready to deploy.

1. In Visual Studio, from the *File* menu, choose *New*, and then choose *Project*.
2. In the navigation pane of the *New Project* dialog box, expand *Installed*, expand *Templates*, expand *Visual C#*, and then choose *Web*.
3. In the list of available web project templates, select any template containing the words `Web` and `Application` in its description. For this example, choose *ASP.NET Web Forms Application*.



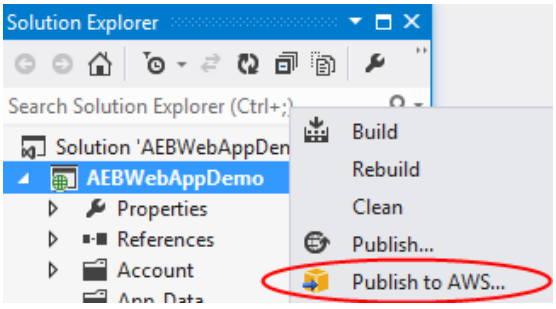
4. In the *Name* box, type `AEBWebAppDemo`.
5. In the *Location* box, type the path to a solution folder on your development machine or choose *Browse*, and then browse to and choose a solution folder, and choose *Select Folder*.

6. Confirm the *Create directory for solution* box is selected. In the *Solution* drop-down list, confirm *Create new solution* is selected, and then choose *OK*. Visual Studio will create a solution and project based on the ASP.NET Web Forms Application project template.

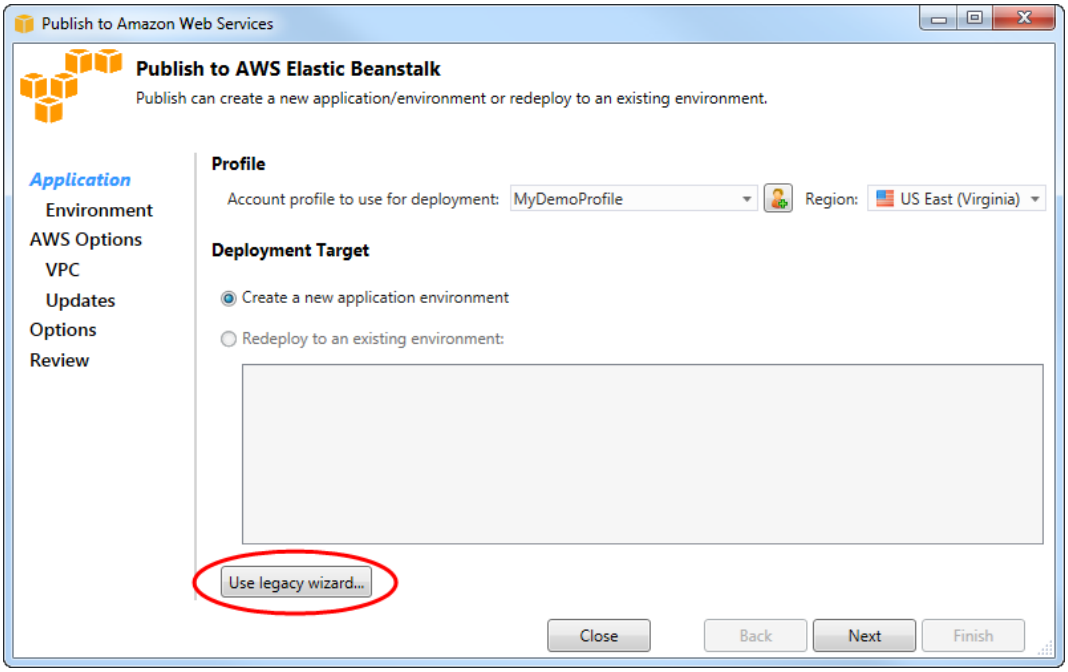


To deploy an application by using the legacy Publish to Amazon Web Services wizard

1. In Solution Explorer, open the context (right-click) menu for the *AEBWebAppDemo* project folder (or your own project folder), and then choose *Publish to AWS*.



2. On the *Publish to AWS Elastic Beanstalk* page, choose *Use legacy wizard*.

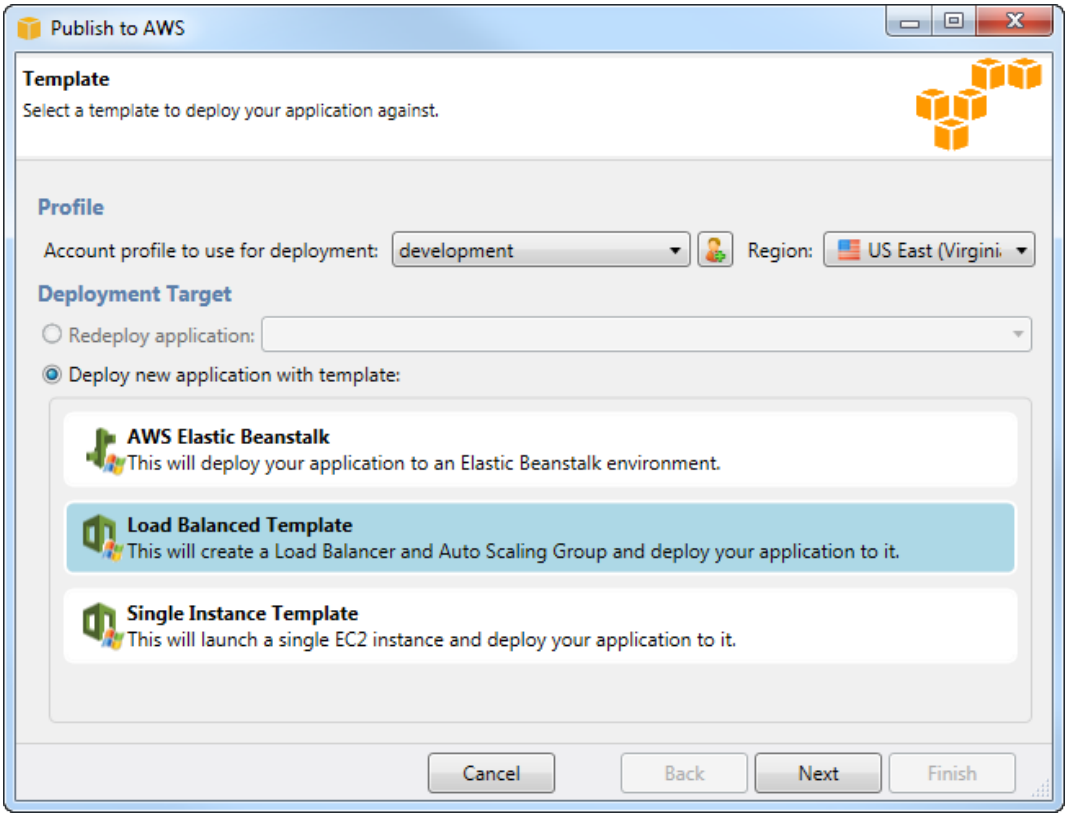


3. On the *Template* page of the wizard, choose the profile you will use for the deployment. To add a new profile, choose *Other*. For more information about profiles, see *Specifying Credentials*.

4. There are options to deploy a new application or redeploy an application that was deployed previously through either the deployment wizard or the standalone deployment tool. If you choose a redeployment, there may be a delay while the wizard retrieves information from the previous deployment.

   The *Load Balanced Template* and *Single Instance Template* are included with the Toolkit for Visual Studio. *Load Balanced Template* provisions an Amazon EC2 instance with an Elastic Load Balancing load balancer and an Auto Scaling group. *Single Instance Template* provisions just a single Amazon EC2 instance.

   For this example, choose *Load Balanced Template*, and then choose *Next*.

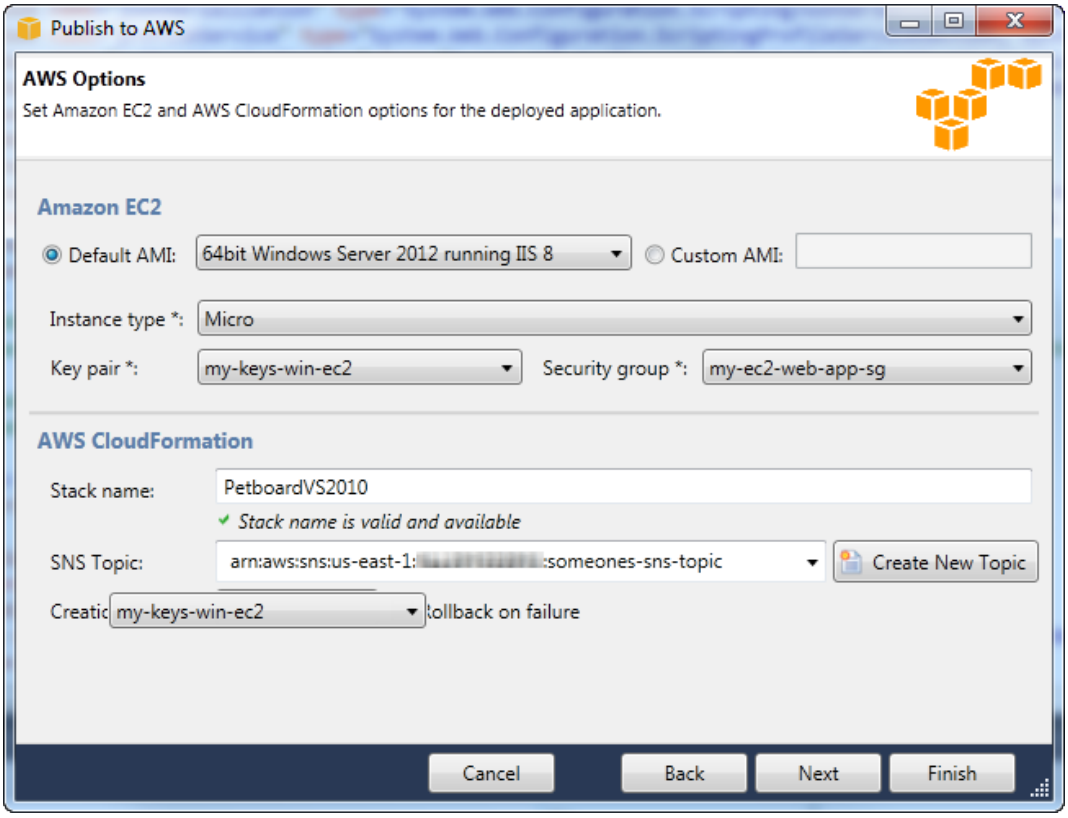


5. On the *AWS Options* page, configure the following:

   - From the *Key pair* drop-down list, choose an Amazon EC2 key pair.
   - Leave *SNS Topic* blank. If you specify an SNS topic, CloudFormation will send status notifications during the deployment.
   - Leave the *Custom AMI* field blank. The CloudFormation template includes an AMI.
   - From the *Instance type* drop-down list, leave the default set to *Micro*. This will minimize the cost associated with running the instance. For more information about Amazon EC2 costs, go to the *EC2 Pricing* page.
   - From the *Security group* drop-down list, choose a security group that has port 80 open. If you have already configured a security group with port 80 open, then choose it. The *default* selection in this drop-down list does not have port 80 open.

     Applications deployed to CloudFormation must have port 80 open because CloudFormation uses this port to relay information about the deployment. If the security group you choose does not have port 80 open, the wizard will ask if it should open it. If you say yes, port 80 will be open for any Amazon EC2 instances that use that security group. For more information about creating a security group, see *Creating a Security Group*.

   Choose *Next*.



6. On the *Application Options* page, in the *Application Credentials* section, choose the profile under which the application (in this example, `PetBoard`) should run. It could be different from the profile used to deploy to CloudFormation (that is, the profile you specified on the first page of the wizard).

   To use a different set of credentials, choose *Use these credentials* and then type the access key and secret key in the fields provided.

   To use the same credentials, choose *Use credentials from profile profile_name* where {profile_name} is the profile you specified on the first page of the wizard.

   To use the credentials for an AWS Identity and Access Management (IAM) user, choose *Use an IAM user*, and then specify the user.

   To use an IAM user, you must have:

   - created the IAM user in the Toolkit for Visual Studio.
   - stored the secret key for the user with the Toolkit for Visual Studio.

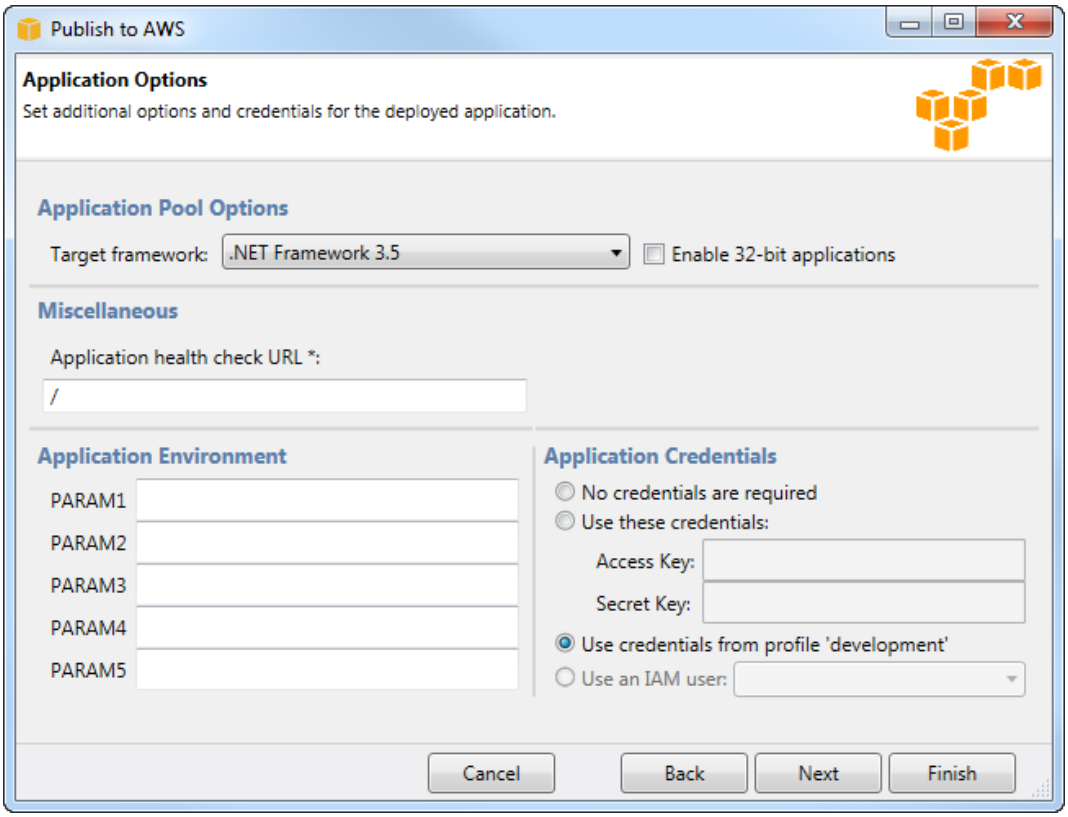
   For more information, see *Create and Configure an IAM User* and *Generate Credentials for an IAM User*.

   An IAM user could have more than one set of credentials stored with the Toolkit. If that is the case, you will need to choose the credentials to use. The root account could rotate the credentials for the IAM user, which would invalidate the credentials. In this scenario, you would need to redeploy the application and then manually enter new credentials for the IAM user.

The following table describes other options available on the *Application Options* page. For `PetBoard`, you can leave the defaults.

Key and Value	Description
PARAM1, PARAM2, PARAM3, PARAM4, PARAM5	These values are made available to the deployed application through the `appSettings` element in the Web.config file. For more information, go to the Microsoft MSDN library.
Target framework	Specifies the version of the .NET Framework targeted by the application. Possible values are: .NET Framework 2.0, .NET Framework 3.0, .NET Framework 3.5, .NET Framework 4.0, .NET Framework 4.5
Enable 32-bit applications	Select if the application is 32-bit. Otherwise, leave the box cleared.
Application health check URL	This URL is relative to the root server URL. For example, if the full path to the URL is `example.com/site-is-up.html`, you would type `/site-is-up.html`. This setting applies only when you use the Load Balanced template. It is ignored when you use the Single Instance template.

Choose *Finish*.



7. On the *Review* page, select *Open environment status window when wizard closes*.

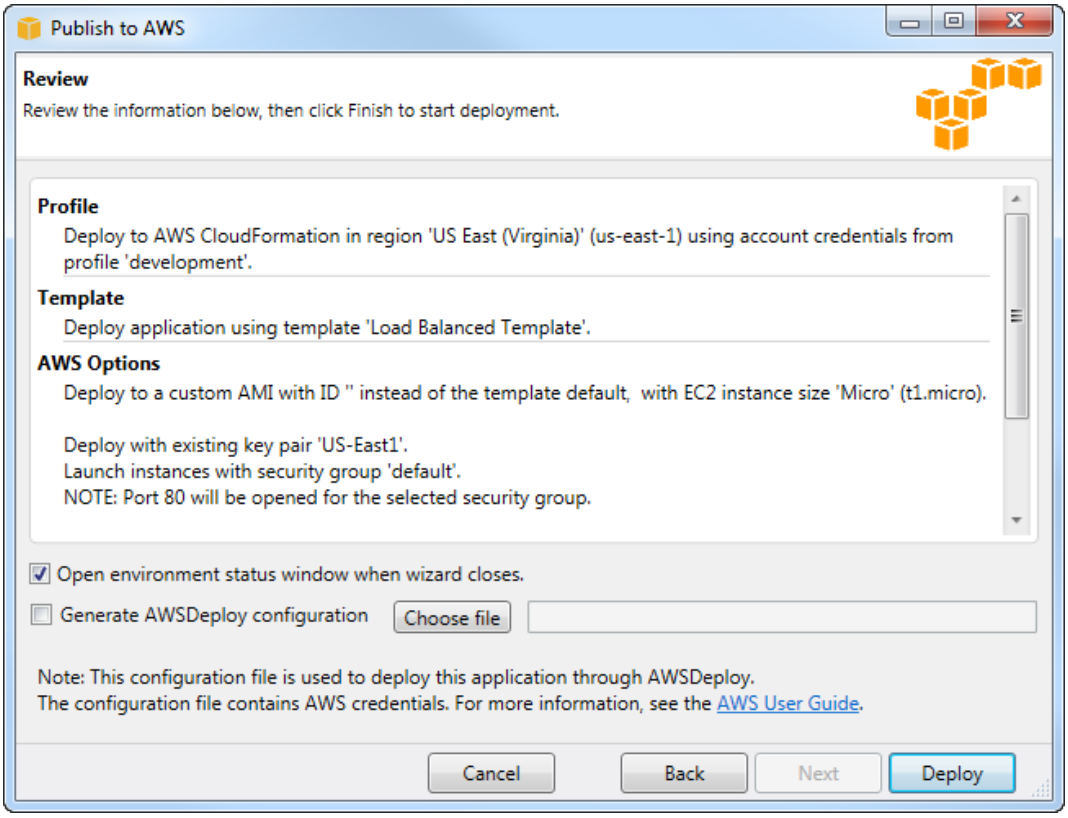
   You can save the deployment configuration to a text file to use with standalone deployment tool. To save the configuration, select *Generate AWSDeploy configuration*. Choose *Choose File* and then specify a file to which to save the configuration. You can also save the deployment configuration

after the deployment is complete. In AWS Explorer, open the context (right-click) menu for the deployment and choose *Save Configuration*.

Note: Because the deployment configuration includes the credentials that were used for deployment, you should keep the configuration file in a secure location.

Choose *Deploy*.

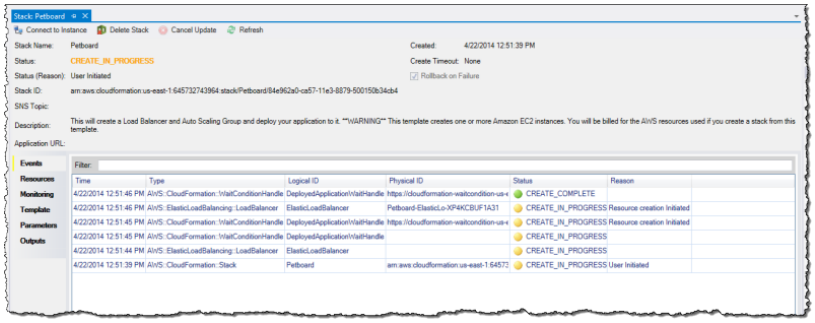
Note: When you deploy the application, the active account will incur charges for the AWS resources used by the application.



8. A status page for the deployment will open. The deployment may take a few minutes.

   When the deployment is complete, the Toolkit will display an alert. This is useful because it allows you to focus on other tasks while the deployment is in progress.

   When the deployment is complete, the status displayed in the Toolkit for Visual Studio will be *CREATE_COMPLETE*.



Choose the *Application URL* link to connect to the application.

9. To delete the deployment, in AWS Explorer, expand the *CloudFormation* node and open the context (right-click) menu for the subnode for the deployment and choose *Delete*. CloudFormation will begin the deletion process, which might take a few minutes. If you specified an SNS topic for the deployment, CloudFormation will send status notifications to this topic.

3.6.2 Standalone Deployment Tool

Note: Standalone Deployment Tool options related to CloudFormation deployments and incremental deployments to Elastic Beanstalk are obsolete in the current version and should not be used.

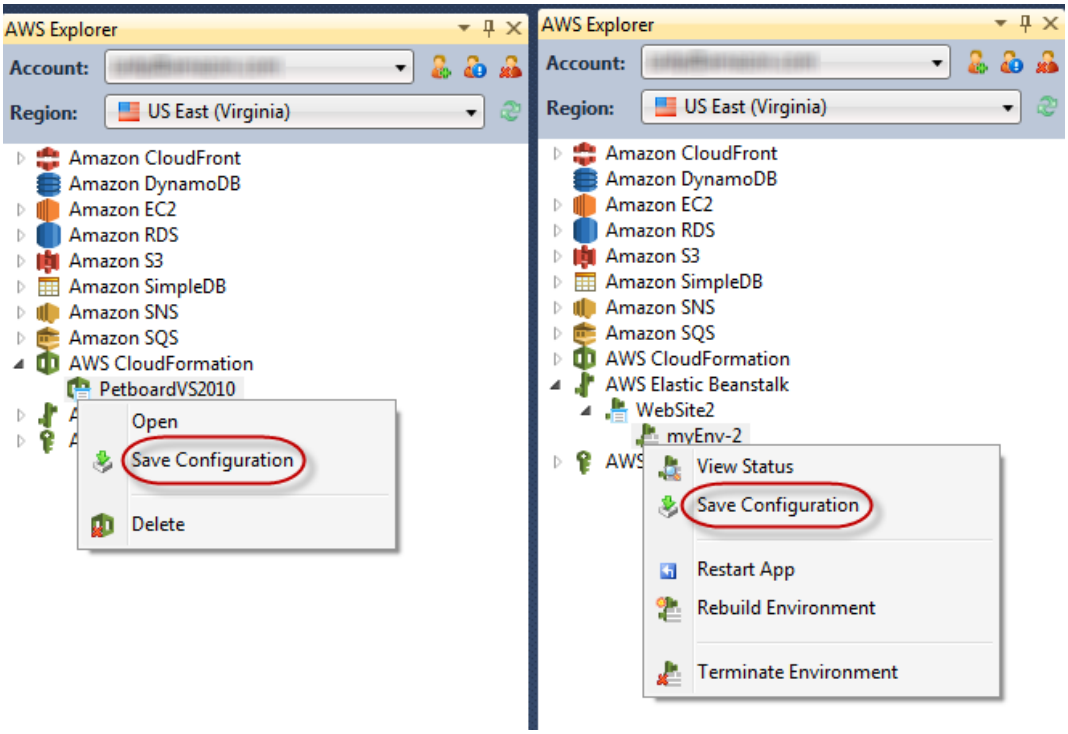
For information about using the preferred *Publish to Elastic Beanstalk* wizard, see *Deploying to Elastic Beanstalk*.

The Toolkit for Visual Studio includes a command line tool that provides the same functionality as the deployment wizard. You can use the standalone deployment tool in your build pipeline or in other scripts to automate deployments to Elastic Beanstalk.

The deployment tool supports both initial deployments and redeployments. If you used the deployment tool to deploy your application, you can use the deployment wizard in Visual Studio to redeploy it, and vice versa.

The deployment tool consumes a configuration file that specifies parameter values for the deployment. If you used the deployment wizard in Visual Studio to deploy your application, you can generate a configuration file either from AWS Explorer or the last step in the wizard.

Note: Because the deployment configuration includes the credentials that were used for deployment, you should keep the configuration file in a secure location.



To to deploy your web application with the deployment tool, package the application in a .zip file. For more information about how to package your application for deployment, go to How to: Create a Web Deployment Package in Visual Studio on MSDN.

Deployment Tool Installation and Invocation

The deployment tool is typically installed in the following directory:

```
C:\Program Files\AWS Tools\Deployment Tool\awsdeploy.exe
```

Or, on Microsoft Windows 64-bit system, in the following directory:

```
C:\Program Files (x86)\AWS Tools\Deployment Tool\awsdeploy.exe
```

Invocation Syntax

```
awsdeploy [options] configFile
```

The configuration file must be the last item specified on the command line.

Command line options can be specified using a forward slash (/) or hyphen (-).

Except for the `D` option, each command line option has a long form and a single letter abbreviation. For example, you can specify silent mode in any of the following ways.

```
/s
-s
/silent
-silent
```

Other command line options follow a similar form.

The following table shows the available command line options.

Option	Description
/s, /silent, -s, -silent	Do not output messages to the console.
/v, /verbose, -v, -verbose	Send more detailed information about the deployment to the console.
/r, /redeploy, -r, -redeploy	Do not create stack. Deploy to existing stack. This option does not change the CloudFormation configuration.
/u, /updateStack, -u, -updateStack	Update the CloudFormation configuration for an existing deployment. Do not redeploy the application. ** (Obsolete. Do not use.) **
/w, /wait, -w, -wait	Block until deployment is complete. This option is useful for scripts that need to take some action after the deployment is complete.
/l <logfile>, /log <logfile>, -l <logfile>, -log <logfile>	Log debugging information to the specified log file.
/D<key>=<value>, -D<key>=<value>	Override a configuration setting from the command line. For more information, see the section of the configuration file.

Output and Exit Codes

Warnings and errors are output to the console. If the log option is specified, additional logging output is sent to the log file.

The deployment tool uses the following exit codes.

Key and Value	Description
0	Success
1	Invalid argument
3	Failed deployment

If the deployment is successful, the deployment tool will output the URL for the deployed application.

Configuration File Samples

You use a configuration file to specify the action of the deployment tool. The Toolkit for Visual Studio includes three sample configuration files:

- Elastic Beanstalk deployment
- CloudFormation single instance deployment
- CloudFormation load-balanced deployment

Sample Web App

A sample web app (in a .zip file archive) that you can deploy using the deployment tool is also included in the Toolkit for Visual Studio. You can find these files in the `Samples` subdirectory of the directory where the deployment tool is installed.

You can use the `D` command line option to override settings in the configuration file:

```
/D<key>=<value>
```

or

```
-D<key>=<value>
```

You can specify the `D` option multiple times to override multiple configuration file settings. If you repeat the same key with different values on the command line, the deployment tool will use the last value specified.

Deployment Tool Configuration File Format

The configuration files provide the same information you would specify in the deployment wizard. The formatting of the configuration files divides the configuration into sections that correspond to the pages in the deployment wizard.

Elastic Beanstalk Deployment Configuration File

The following configuration parameters are for deployments using Elastic Beanstalk.

For a walkthrough of the use of the standalone deployment tool to deploy to Elastic Beanstalk, go to the Developer Guide.

General Settings

```
/Daws:autoscaling:launchconfiguration.SecurityGroups=RDPOnly,HTTPOnly
```

Key and Value	Description
DeploymentPackage = archive.zip	Relative path to the web deployment archive. This path is relative to your working directory (that is, the directory from which you invoke the deployment tool).
IncrementalPushLocation	(Obsolete: Do not use) If specified, incremental deployment is enabled. The value specifies a location (for example, `C:\Temp\VS2008App1`) where a Git repository will be created to store the versioned contents of the deployment package.
Template = ElasticBeanstalk	Can be `Elastic Beanstalk` or `ElasticBeanstalk`.
Application.Name	Specifies a name for the application. This value is required.
Application.Description	Specifies an optional description for the application.
Application.Version	Specifies a version string for the application. If you are using incremental deployment, this value is ignored. Elastic Beanstalk uses the Git commit ID for the version string.
Region = us-east-1	Target Regions and Endpoints.
UploadBucket = awsdeployment-us-east-1-samples	Amazon S3 bucket where the deployment materials will be stored. If this bucket doesn't exist, it will be created. If you use the deployment wizard, it generates the bucket name for you.
KeyPair = default	Amazon EC2 key pair for signing in to the instance. The key pair must exist before deployment. (The deployment wizard allows you to create the key pair during deployment.)
AWSAccessKey = DEPLOYMENT_CREDENTIALS_HERE AWSSecretKey = DEPLOYMENT_CREDENTIALS_HERE	AWS access key and secret key used to create the stack and deploy the application to Elastic Beanstalk. We do not recommend using these parameters to specify credentials. Instead, create a profile for the credentials and use `AWSProfileName` to reference the profile. For more information, see *Specifying Credentials*.
AWSProfileName = {profile_name}	The profile used to create the stack and deploy the application to Elastic Beanstalk.
aws:autoscaling:launchconfiguration.SecurityGroups = default	The names of the security groups for the Amazon EC2 instance. If you specify multiple security groups, separate them with commas. `/Daws:autoscaling:launchconfiguration.Security` The security groups must already exist and must allow ingress on port 80 (HTTP). For information about how to create security groups, see *Managing Security Groups from AWS Explorer*

Environment Settings

Key and Value	Description
Environment.Name	Specifies a name for your Elastic Beanstalk environment. This value is required.
Environment.Description	Optional. Specifies a description for your environment.
Environment.CNAME	Optional. Specifies the URL prefix for your application. If you do not specify this value, Elastic Beanstalk will derive the prefix from your environment name.

Container Settings

Key and Value	Description
Container.TargetRuntime = 4.0	Specifies the target runtime for the .NET Framework. Possible values are 2.0 or 4.0. The following .NET Framework versions are mapped to a target runtime of 2.0: • .NET Framework 2.0 • .NET Framework 3.0 • .NET Framework 3.5 The following .NET Framework versions are mapped to a target runtime of 4.0: • .NET Framework 4.0 • .NET Framework 4.5 The *deployment wizard* in the Toolkit for Visual Studio allows you to specify the .NET Framework version. The wizard then maps the .NET Framework version to the appropriate target runtime version.
Container.Enable32BitApplications = false	If the application is 32-bit, specify `true`. If the application is 64-bit, specify `false`.
Container.ApplicationHealthcheckPath = /	This URL is relative to the root server URL. For example, if the full URL is `example.com/site-is-up.html`, you would type `/site-is-up.html`. The setting applies only when you use the load balanced template. It is ignored when you use the single instance template. The responsiveness of the application at this URL affects into the actions taken by the load balancer and auto scaler. If the application is unresponsive or responds slowly, the load balancer will direct incoming network traffic to other Amazon EC2 instances, and the auto scaler may add additional Amazon EC2 instances.
Container.InstanceType = t1.micro	The type of Amazon EC2 instance to use. The Micro instance shown here is the *EC2 Pricing* type of instance.
Container.AmiID	Specifies a custom Amazon Machine Image (AMI). For more information about how to create a custom AMI, go to Using Custom AMIs in the Elastic Beanstalk Developer Guide and *Create an AMI from an Amazon EC2 Instance*.
Container.NotificationEmail	Optional. Specifies an email address for deployment status notifications.

CloudFormation Deployment Configuration File

Note: Deployments to CloudFormation using the Standalone Deployment Tool are deprecated.

The following configuration parameters are taken from the load balanced template.

General Settings

Key and Value	Description
DeploymentPackage = archive.zip	Relative path to the web deployment archive. This path is relative to your working directory (that is, the directory from which you invoke the deployment tool). If you are updating a deployment (`/updateStack` switch), this parameter is ignored.
Region = **|region_api_default|**	Target region.
Template = LoadBalanced	The value for `Template` can be `SingleInstance` or `LoadBalanced` or a file path to a custom CloudFormation template. For more information, see *Customizing the AWS CloudFormation Template Used for Deployment*
UploadBucket = awsdeployment-us-east-1-samples	Amazon Simple Storage Service (Amazon S3) bucket where the deployment materials will be stored. If the bucket doesn't exist, it will be created. If you use the deployment wizard, it generates this bucket name for you. If you used the wizard for a deployment and are redeploying, this parameter will be ignored. The deployment tool automatically uses the bucket that was used in the original deployment from the wizard.
KeyPair = default	Amazon Elastic Compute Cloud (Amazon EC2) key pair for signing in to the instance. The key pair must exist before deployment. (The deployment wizard allows you to create the key pair during deployment.)
AWSAccessKey = DEPLOYMENT_CREDENTIALS_HERE AWSSecretKey = DEPLOYMENT_CREDENTIALS_HERE	The AWS access key and secret key used to create the stack and deploy the application to CloudFormation. We do not recommend using these parameters to specify credentials. Instead, create a profile for the credentials and use `AWSProfileName` to reference the profile. For more information, see *Specifying Credentials*.
AWSProfileName = {profile_name}	The profile used to create the stack and deploy the application to CloudFormation.

Template Parameters

In addition to the following parameters, the load balanced template supports numerous other parameters to customize load balancing and Auto Scaling behavior.

Key and Value	Description
Template.InstanceType = t1.micro	The type of Amazon EC2 instance to use. The Micro instance shown here is the least expensive type of instance.
Template.SecurityGroup = default	The security group for the Amazon EC2 instance. This security group must have already been created and must allow ingress on port 80 (HTTP). For information about how to create a security groups, see *Managing Security Groups from AWS Explorer*.
Environment.PARAM1 = Environment.PARAM2 = Environment.PARAM3 = Environment.PARAM4 = Environment.PARAM5 =	These values are made available to the deployed application through the `appSettings` in the `Web.config` file. For more information, go to the MSDN library.
Environment.AWSAccessKey = APP_CREDENTIALS_HERE Environment.AWSSecretKey = APP_CREDENTIALS_HERE	The access key and secret key used by the deployed application to access AWS services. We do not recommend using these parameters to specify credentials. Instead, create a profile for the credentials and use `AWSProfileName` to reference the profile. For more information, see *Specifying Credentials*.
AWSProfileName = {profile_name}	The profile used by the deployed application to access AWS services. .

Container Settings

```
SolutionStack="64bit Windows Server 2008 R2 running IIS 7.5"
```

```
SolutionStack="64bit Windows Server 2012 running IIS 8"
```

Key and Value	Description
SolutionStack="64bit Windows Server 2012 running IIS 8"	Specifies the version of Windows Server and Internet Information Services (IIS) to which to deploy. Valid values are: SolutionStack="64bit Windows Server 2008 R2 running IIS 7.5" or SolutionStack="64bit Windows Server 2012 running IIS 8" If not specified, the default is 64bit Windows Server 2012 running IIS 8.0. You can use *Container.Type* as an alias for SolutionStack.
Container.TargetRuntime = 4.0	Specifies the target runtime for the .NET Framework. Possible values are 2.0 or 4.0. **The following .NET Framework versions are mapped to a target** • .NET Framework 2.0 • .NET Framework 3.0 • .NET Framework 3.5 **The following .NET Framework versions are mapped to a target** • .NET Framework 4.0 • .NET Framework 4.5 The *deployment wizard* in the Toolkit for Visual Studio allows you to specify the .NET Framework version. The wizard then maps the .NET Framework version to the appropriate target runtime version.
Container.Enable32BitApplications = false	If the application is 32-bit, specify `true`. If the application is 64-bit, specify `false`.
Container.ApplicationHealthcheckPath = /	This URL is relative to the root server URL. For example, if the full URL is `example.com/site-is-up.html`, you would type `/site-is-up.html`. The setting applies only when you use the load balanced template. It is ignored when you are using the single instance template. The responsiveness of the application at this URL affects the actions taken by the load balancer and auto scaling. If the application is unresponsive or responds slowly, the load balancer will direct incoming network traffic to other Amazon EC2 instances, and the auto scaler may add additional Amazon EC2 instances.

Stack Creation Settings

Key and Value	Description
Settings.SNSTopic	SNS topic to use for deployment messages.
Settings.CreationTimeout = 0	The amount of time to allow for the creation of the stack. A value of zero means there is no time limit.
Settings.RollbackOnFailure = false	If this value is `true`, the deployment tool tears down the stack if the deployment fails.

How to Update the Configuration for an Existing Deployment

You can use the `updateStack` feature of the deployment tool to modify the CloudFormation configuration of an existing deployment. This configuration—the application's environment—includes the cloud resources your application runs on and has access to. The `updateStack` feature does not change or redeploy the application; it only updates the application's environment. In this way, the `updateStack` feature complements the redeployment feature. Redeployment provides a way to update your application without changing the environment.

There are various scenarios in which you might use `updateStack`. For example, if you develop your application using the single instance template, as the application nears production readiness, you could update its configuration to use a load balanced template, either for public beta testing or live release deployment. In a related scenario, a deployment using a load-balanced configuration could be optimized by modifying some of the configuration parameters—for example, by increasing the maximum number of supporting EC2 instances or changing the size of the instances, say from micro to large. You can use the `updateStack` feature of the deployment tool to implement either of these scenarios.

There are scenarios in which you might use both the `/updateStack` option and the `/redeploy` option, effectively modifying both the application itself and the environment in which the application is running. In some cases, this approach is more efficient than just performing a regular deployment. For example, you might change your environment to add an Amazon S3 bucket and then update your application to use that bucket. With a combination of `/updateStack` and `/redeploy`, you could implement both changes, but leave any already provisioned Amazon EC2 instances up and running. A regular deployment would result in all of the environment being taken down and rebuilt.

The `updateStack` feature is available only through the deployment tool. It is not available through the deployment wizard in Visual Studio. You can use `updateStack` to update a deployment that was initially deployed through the deployment wizard, but not vice versa.

The invocation syntax for updating a deployment is similar to the syntax for a new deployment.

```
awsdeploy /updateStack [other options] updatedConfigFile
```

Keep the following in mind when you attempt to update a deployment:

- You cannot update a deployment that is in the process of being created or taken down.
- The specified config file must use the same value for the `StackName` parameter as the original deployment.

- You cannot use `updateStack` to change the region for your deployment. However, you can change the Availability Zones for your deployment.
- If you use `updateStack` to transition your deployment from single instance to load balanced, the endpoint for your deployment will necessarily change. In the single instance case, the endpoint refers to an Amazon EC2 instance. In the load balanced template, the endpoint refers to the Elastic Load Balancing load balancer, a computer that distributes computing load across all EC2 instances. Therefore, if you are using a CNAME record to associate a domain name with your deployment, you should update the CNAME record so that it points to the load balancer of the load balanced template.

The deployment tool implements the `updateStack` feature by calling the CloudFormation UpdateStack API. For more information about CloudFormation, go to the CloudFormation User Guide.

Customizing the AWS CloudFormation Template Used for Deployment

In addition to modifying a deployment by specifying parameters in the deployment wizard—or in the configuration file for the standalone deployment tool—you can also modify the deployment by providing your own custom AWS CloudFormation template. By default, the deployment automatically uses one of a set of templates that are stored in Amazon Simple Storage Service (Amazon S3). This default set of templates includes two templates for each AWS region. One of these two is for deployment to a single Amazon Elastic Compute Cloud (Amazon EC2) instance; the other is for deployment to a load-balanced set of Amazon EC2 instances. You can use these templates as a starting point for creating your own.

To create your own custom template

1. Copy the template that corresponds to your region and the type of deployment that you want to do. Links to each of the templates is provided below.

   Note: Templates are available only for the regions listed below.

   US East (N. Virginia)

SingleInstance.template	LoadBalanced.template

   US West (Oregon)

SingleInstance-us-west-2.template	LoadBalanced-us-west-2.template

   US West (N. California)

SingleInstance-us-west-1.template	LoadBalanced-us-west-1.template

   EU (Ireland)

SingleInstance-eu-west-1.template	LoadBalanced-eu-west-1.template

   Asia Pacific (Singapore)

SingleInstance-ap-southeast-1.template	LoadBalanced-ap-southeast-1.template

   Asia Pacific (Tokyo)

SingleInstance-ap-northeast-1.template	LoadBalanced-ap-northeast-1.template

Asia Pacific (Sydney)

SingleInstance-ap-southeast-2.template	LoadBalanced-ap-southeast-2.template

South America (Sao Paulo)

SingleInstance-sa-east-1.template	LoadBalanced-sa-east-1.template

If you need to create your own links to the templates, the format for each link is as follows:

```
http://vstoolkit.amazonwebservices.com/CloudFormationTemplates/{template-name}
```

For example, for the single instance template for the US West (N. California) region, the link would be:

```
http://vstoolkit.amazonwebservices.com/CloudFormationTemplates/SingleInstance-
↪us-west-1.template
```

The links in the table show the HTTP protocol. The HTTPS protocol is also supported.

2. Edit the template to modify it for your specific needs. The templates are text files, so you can edit them with any standard text editor. The deployment information in the templates is represented in JavaScript Object Notation format. After editing the file, it's wise to revalidate the JSON formatting using a tool such as JSONLint.

   The template file has three sections: Parameters, Resources, and Outputs.

   To add resources to your deployment, add them to the `Resources` section of the template. For example, you could add an Amazon RDS database or an Amazon SNS topic. To configure these resources at deployment time, add parameters to the `Parameters` section of the template. When you add new parameters to the template, the AWS Toolkit adds them to the parameters that are displayed in the deployment wizard. You can specify values for these parameters either in the deployment wizard or in the config file for the standalone deployment tool.

   Similarly, data that you specify in the `Output` section of the template is also displayed in the deployment wizard—as well as in the AWS Management Console. You can use the `Output` section to display post-deployment information about your resources. For example, if you add an Amazon S3 bucket to the `Resources` section of the template, you can use the `Outputs` section to display the autogenerated name for the bucket.

   For more information about editing AWS CloudFormation templates, go to the CloudFormation User Guide.

3. Set the `Template` parameter in the deployment configuration file to the path to your customized template. The `Template` parameter is located under `General Settings` in the config file. The path that you specify could be the path to the file on your local hard drive or it could be a URL that points to the location of the configuration file on a remote server. When you next run a deployment, the tool will use your template.

Required Data in the Template File

The deployment process requires that certain data be specified in the template file. While editing your version of the template, you must ensure that it continues to provide this data. The required data is located only in the `Parameters` and `Outputs` sections of the template.

Parameters Section of Template

The following table shows the required parameters in the Parameters section of the template.

Name	Meaning
InstanceType	The "API name" for the type of the Amazon EC2 instances to use for the deployment. Examples are t1.micro for Micro instances or m1.xlarge for Extra Large instances. For a list of instance types and corresponding API names, see the Amazon EC2 detail page.
KeyPair	Which of your key pairs to use for the Amazon EC2 instances.
SecurityGroup	The security group to use for the Amazon EC2 instances.
BucketName	Amazon S3 bucket where the deployment files are uploaded.
ConfigFile	Name of the config file that the deployment uses.
AmazonMachineImage	The Amazon Machine Image (AMI) that is used for the deployment. For more information about how to create a custom AMI, go to Using Custom AMIs in the Elastic Beanstalk Developer Guide and *Create an AMI from an Amazon EC2 Instance*. Note that the Host Manager software that is installed on AMIs that are used in CloudFormation deployments is now auto-updating. Therefore, if you derive a custom AMI from one of the CloudFormation AMIs, you do not need to maintain the Host Manager software. However, you still need to keep the operating system and application software up to date.
UserData	The user data that the deployment provides to the deployed application.

Outputs Section of Template

The following table shows the required outputs in the Outputs section of the template.

Name	Meaning
Bucket	The Amazon S3 bucket to which the deployment files were uploaded.
ConfigFile	The name of the configuration file that was used for the deployment.
VSToolkitDeployed	Boolean flag set to `true`, which indicates that this stack was created as part of a deployment from the AWS Toolkit for Visual Studio. This flag is also set to `true` if the deployment is done from the standalone deployment tool.
URL	The URL for the deployed application.

3.7 Using the CloudFormation Template Editor for Visual Studio

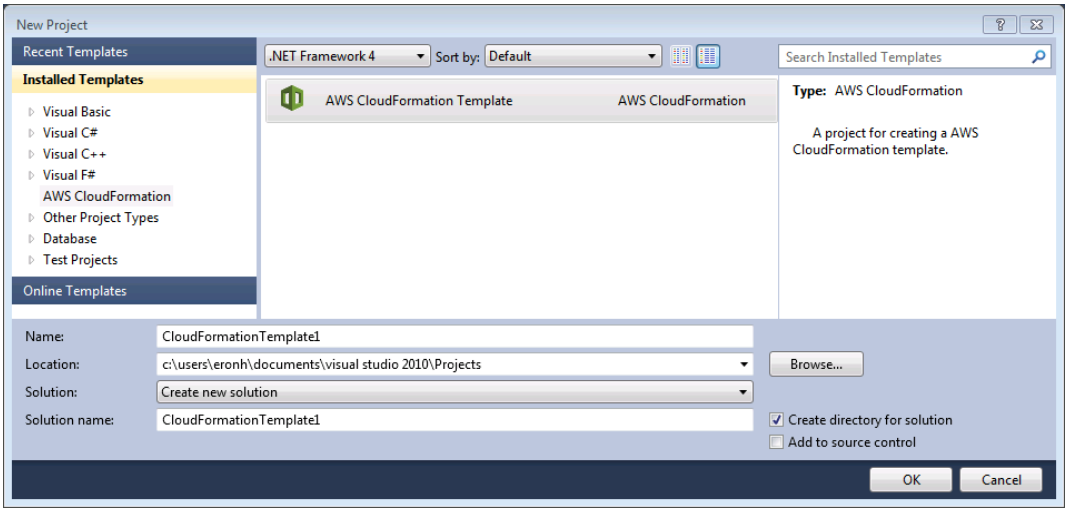
The Toolkit for Visual Studio includes an CloudFormation template editor and CloudFormation template projects for Visual Studio. The supported features include:

- Creating new templates (either empty or copied from an existing stack or sample template) using the supplied CloudFormation template project type.
- Editing templates with automatic JSON validation, auto-completion, code folding, and syntax highlighting.
- Automatic suggestion of intrinsic functions and resource reference parameters for the field values in your template.
- Menu items to perform common actions for your template from Visual Studio: deploying the template, estimating the cost of your template, and formatting your template.

3.7.1 Creating a CloudFormation Template Project in Visual Studio

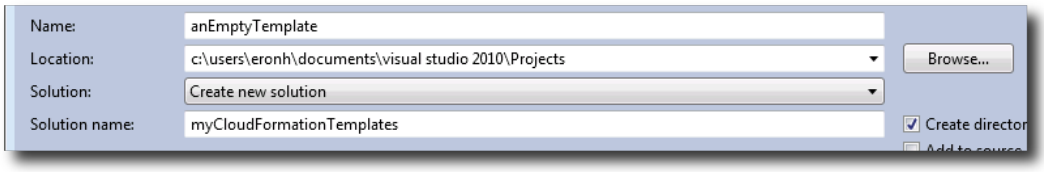
To create a template project

1. In Visual Studio, choose *File*, choose *New*, and then choose *Project*.
2. In the *New Project* dialog box, choose *Installed Templates*, choose *CFN*, and then choose *CFN Template*.

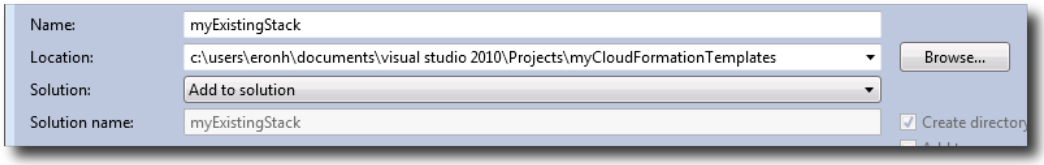


3. In *Name*, type the name of your template project.
4. In the *Solution* drop-down list, choose one of the following:

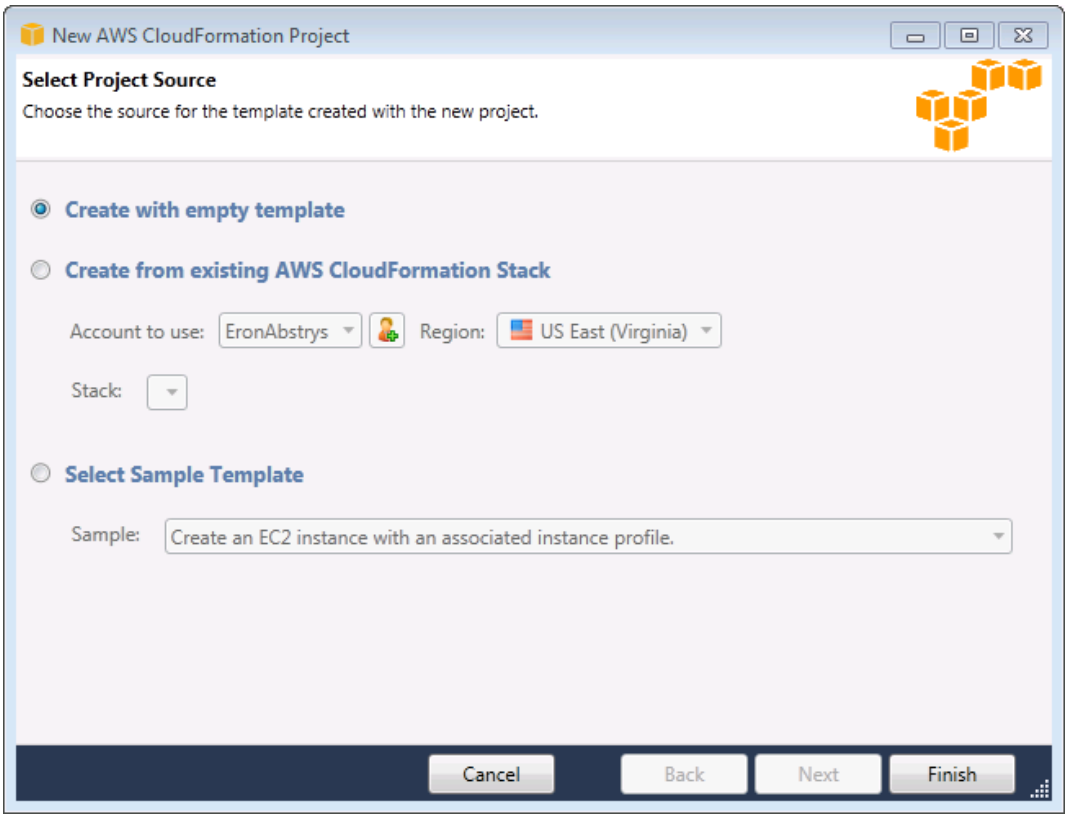
- *Create new solution*. To create a new solution for your template, type a name for the solution in *Solution Name*. (Optional) Choose a location for your project in *Location*.



- *Add to solution*. If you choose to add this template to your currently open solution, the *Location* field will be set to the location of your current solution automatically, and the *Solution Name* field will be unavailable.



5. Choose *OK*.
6. On the *Select Project Source* page, choose the source of the template you will create:
   - *Create with empty template* generates a new, empty CloudFormation template.
   - *Create from existing CFN stack* generates a template from an existing stack in your AWS account. (The stack doesn't need to have a status of `CREATE_COMPLETE`.)
   - *Select sample template* generates a template from one of the CloudFormation sample templates.

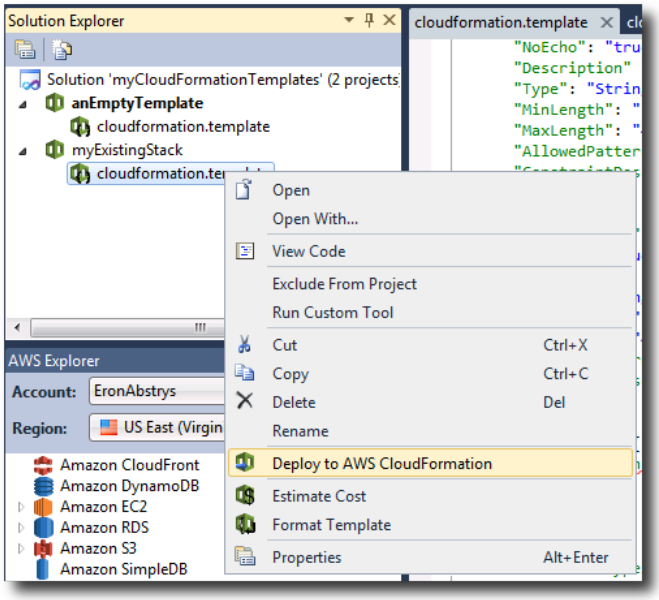


7. To complete the creation of your CloudFormation template project, choose *Finish*.

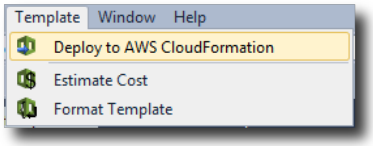
3.7.2 Deploying a CloudFormation Template in Visual Studio

To deploy an CFN template

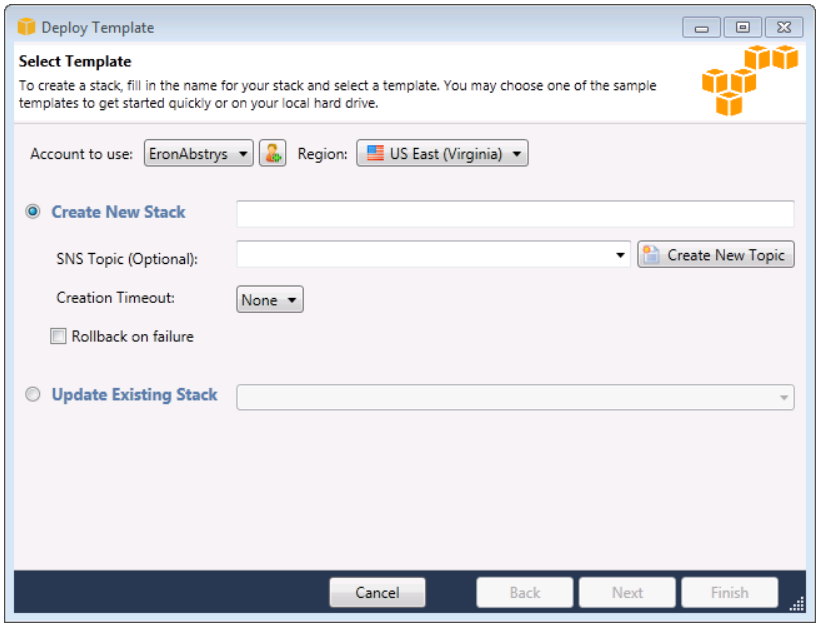
1. In Solution Explorer, open the context (right-click) menu for the template you want to deploy, and choose *Deploy to AWS CloudFormation*.



Alternatively, to deploy the template you're currently editing, from the *Template* menu, choose *Deploy to AWS CloudFormation* .



2. On the *Deploy Template* page, choose the AWS account to use to launch the stack and the region where it will be launched.



3. Choose *Create New Stack* and type a name for your stack.
4. Choose any (or none) of the following options:
   - To receive notifications about the stack's progress, from the *SNS Topic* drop-down list, choose an SNS topic. You can also create an SNS topic by choosing *Create New Topic* and typing an email address in the box.
   - Use *Creation Timeout* to specify how long CloudFormation should allow for the stack to be created before it is declared failed (and rolled back, unless the *Rollback on failure* option is cleared).
   - Use *Rollback on failure* if you want the stack to roll back (that is, delete itself) on failure. Leave this option cleared if you would like the stack to remain active for debugging purposes, even if it has failed to complete the launch.
5. Choose *Finish* to launch the stack.

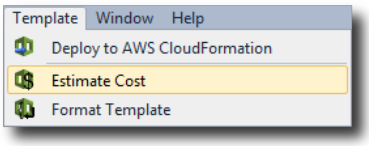
3.7.3 Estimating the Cost of Your CloudFormation Template Project in Visual Studio

With the Toolkit for Visual Studio, you can easily estimate the cost of the CloudFormation stack you are working on before you deploy it. This way you'll have an idea of the monthly operating costs for the resources include in your template.

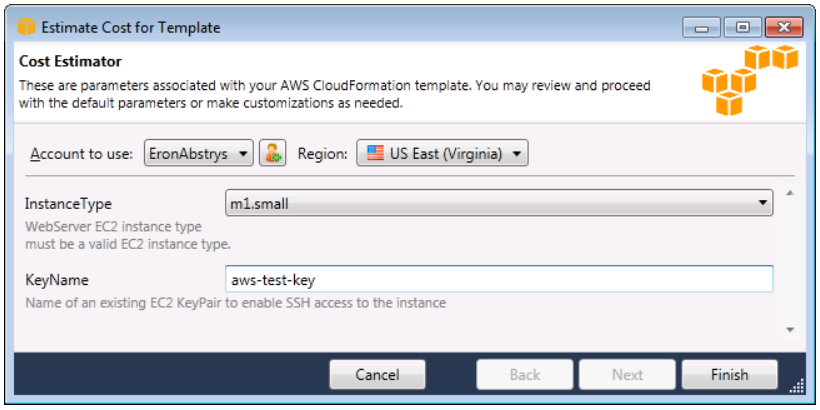
To estimate the cost of your CFN stack

1. In Solution Explorer, open the context (right-click) menu for the template and choose *Estimate Cost*.

   Alternatively, to estimate the cost of the template you're currently editing, from the *Template* menu, choose *Estimate Cost*.

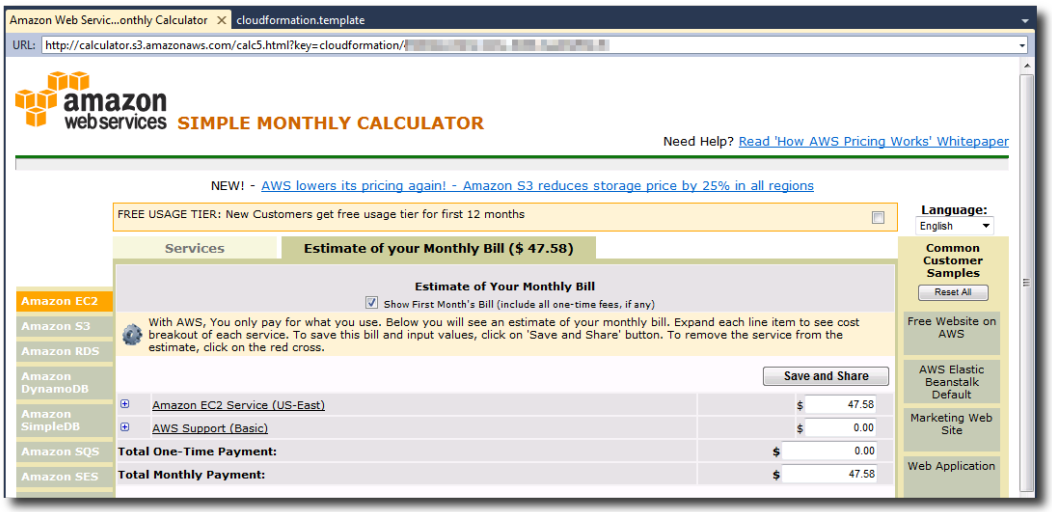


2. Provide values for parameters you have defined for your stack, and choose *Finish*.



3. The AWS Simple Monthly Calculator will be displayed. The values for the form data will be filled in with information pulled from the template you're editing. You can adjust the values, if needed.

   The *Estimate of Your Monthly Bill* tab will display an itemized view of the estimated monthly costs of running your stack.

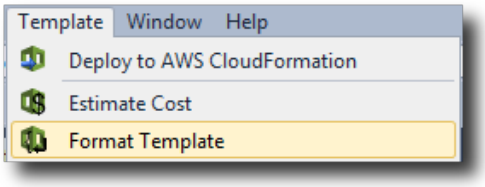


Note: Cost estimates are calculated using the values you provide and the current rates of AWS services, which can vary over time. For more information, see the How AWS Pricing Works whitepaper.

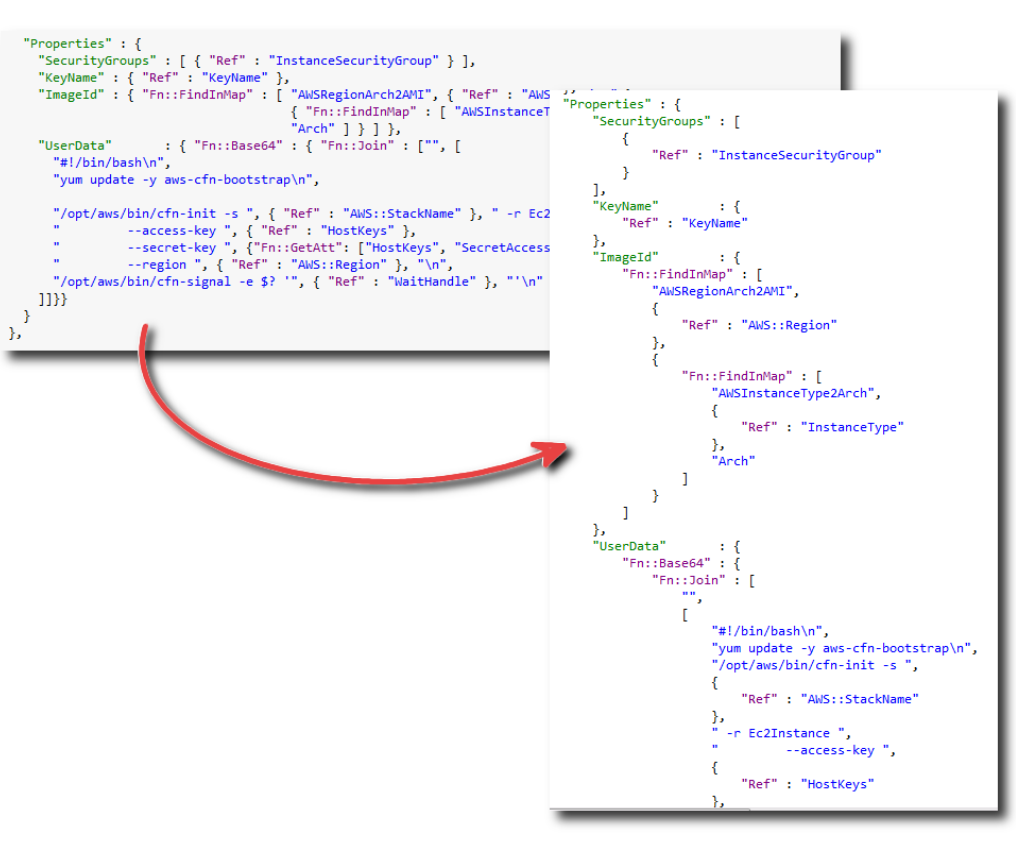
3.7.4 Formatting a CloudFormation Template in Visual Studio

- In Solution Explorer, open the context (right-click) menu for the template and choose *Format Template*.

  Alternatively, to format the template you're currently editing, from the *Template* menu, choose *Format Template*.



  Your JSON code will be formatted so that its structure is clearly presented.



3.8 Using Amazon S3 from AWS Explorer

Amazon Simple Storage Service (Amazon S3) enables you to store and retrieve data from any connection to the Internet. All data you store on Amazon S3 is associated with your account and, by default, can only

be accessed by you. The Toolkit for Visual Studio enables you to store data on Amazon S3 and to view, manage, retrieve, and distribute that data.

Amazon S3 uses the concept of buckets, which you can think of as being similar to file systems or logical drives. Buckets can contain folders, which are similar to directories, and objects, which are similar to files. In this section, we'll be using these concepts as we walk through the Amazon S3 functionality exposed by the Toolkit for Visual Studio.

Note: To use this tool, your IAM policy must grant permissions for the `s3:GetBucketAcl`, `s3:GetBucket`, and `s3:ListBucket` actions. For more information, see Overview of AWS IAM Policies.

3.8.1 Creating an Amazon S3 Bucket

The bucket is most fundamental unit of storage in Amazon S3.

To create an S3 bucket

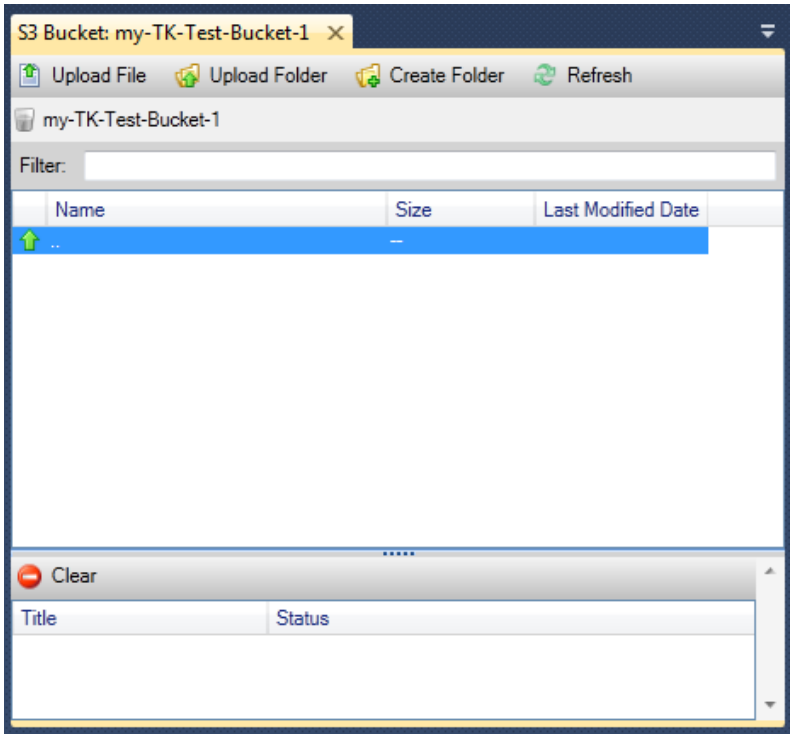
1. In AWS Explorer, open the context (right-click) menu for the *Amazon S3* node, and then choose *Create Bucket*.
2. In the *Create Bucket* dialog box, type a name for the bucket. Bucket names must be unique across AWS. For information about other constraints, go to the Amazon S3 documentation.
3. Choose *OK*.

3.8.2 Managing Amazon S3 Buckets from AWS Explorer

In AWS Explorer, the following operations are available when you open a context (right-click) menu for an Amazon S3 bucket.

Browse

Displays a view of the objects contained in the bucket. From here, you can create folders or upload files or entire directories and folders from your local computer. The lower pane displays status messages about the upload process. To clear these messages, choose the *Clear* icon. You can also access this view of the bucket by double-clicking the bucket name in AWS Explorer.



Properties

Displays a dialog box where you can do the following:

- Set Amazon S3 permissions that scope to:
  - you as the bucket owner.
  - all users who have been authenticated on AWS.
  - everyone with Internet access.
- Turn on logging for the bucket.
- Set up a notification using the Amazon Simple Notification Service (Amazon SNS) so that if you are using Reduced Redundancy Storage (RRS), you are notified if data loss occurs. RRS is an Amazon S3 storage option that provides less durability than standard storage, but at reduced cost. For more information, see S3 FAQs.
- Create a static website using the data in the bucket.

Policy

Enables you to set up AWS Identity and Access Management (IAM) policies for your bucket. For more information, go to the IAM documentation and the use cases for IAM and S3.

Create Pre-Signed URL

Enables you to generate a time-limited URL you can distribute to provide access to the contents of the bucket. For more information, see *How to Create a Pre-Signed URL*.

View Multi-Part Uploads

Enables you to view multipart uploads. Amazon S3 supports breaking large object uploads into parts to make the upload process more efficient. For more information, go to the discussion of multipart uploads in the S3 documentation.

Delete

Enables you to delete the bucket. You can only delete empty buckets.

3.8.3 Uploading Files and Folders to Amazon S3

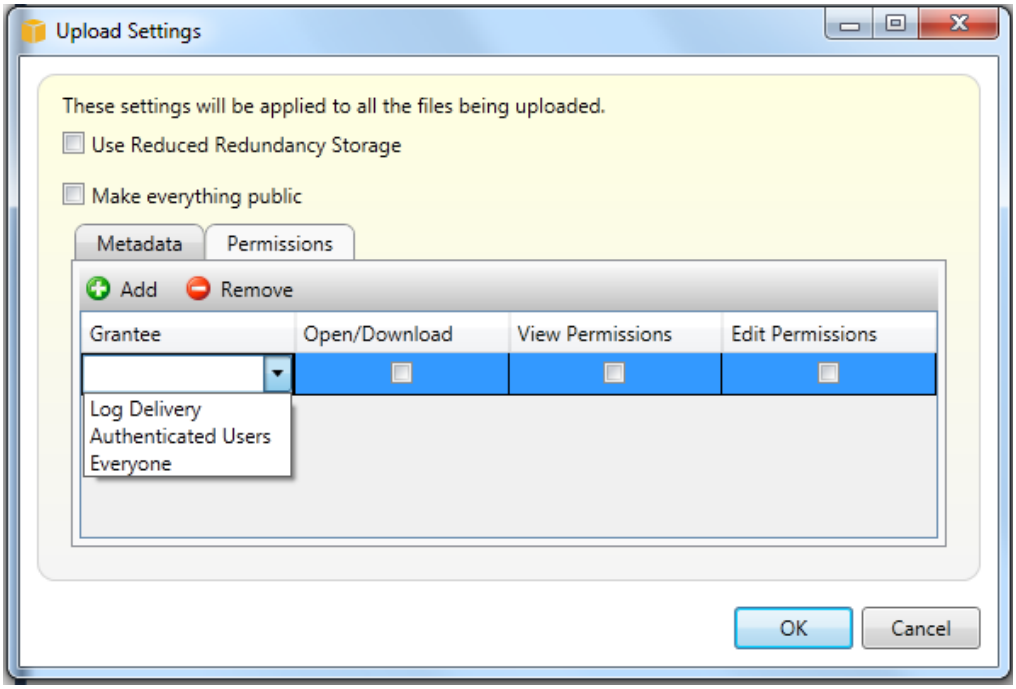
You can use AWS Explorer to transfer files or entire folders from your local computer to any of your buckets.

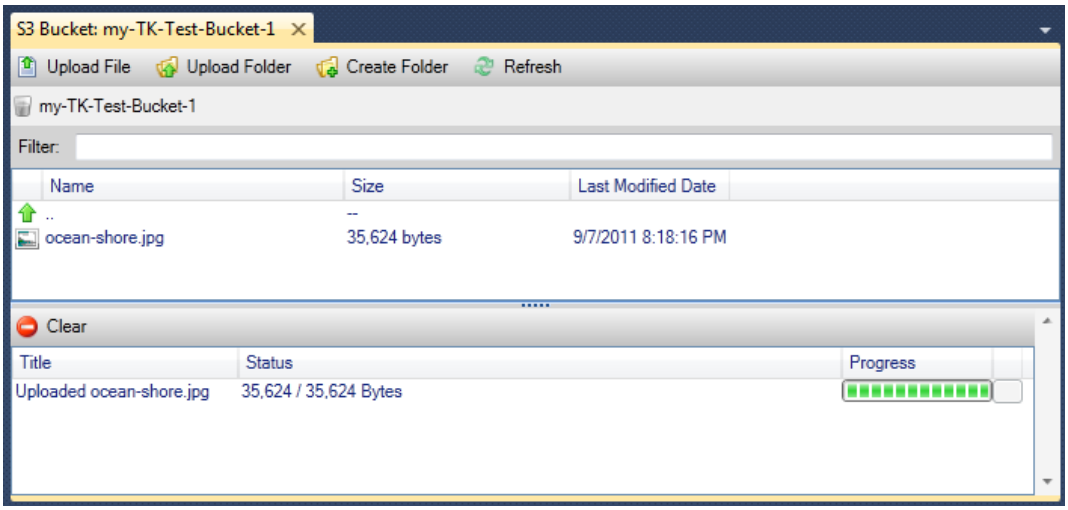
Note: If you upload files or folders that have the same name as files or folders that already exist in the Amazon S3 bucket, your uploaded files will overwrite the existing files without warning.

To upload a file to S3

1. In AWS Explorer, expand the *Amazon S3* node, and double-click a bucket or open the context (right-click) menu for the bucket and choose *Browse*.
2. In the *Browse* view of your bucket, choose *Upload File* or *Upload Folder*.
3. In the *File-Open* dialog box, navigate to the files to upload, choose them, and then choose *Open*. If you are uploading a folder, navigate to and choose that folder, and then choose *Open*.

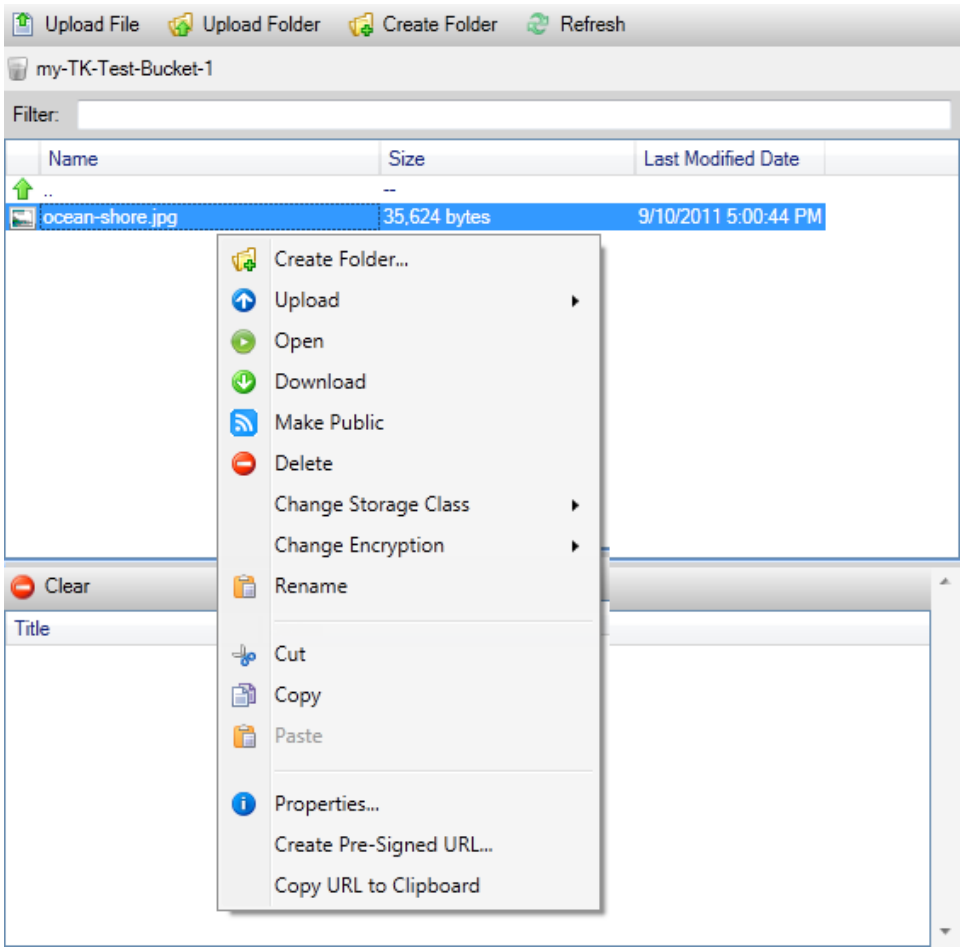
   The *Upload Settings* dialog box enables you to set metadata and permissions on the files or folder you are uploading. Selecting the *Make everything public* check box is equivalent to setting *Open/Download* permissions to *Everyone*. You can select the option to use Reduced Redundancy Storage for the uploaded files.





3.8.4 Amazon S3 File Operations from AWS Toolkit for Visual Studio

If you choose a file in the Amazon S3 view and open the context (right-click) menu, you can perform various operations on the file.



Create Folder

Enables you to create a folder in the current bucket. (Equivalent to choosing the *Create Folder* link.)

Upload

Enables you to upload files or folders. (Equivalent to choosing the *Upload File* or *Upload Folder* links.)

Open

Attempts to open the selected file in your default browser. Depending on the type of file and your default browser's capabilities, the file might not be displayed. It might simply be downloaded by your browser instead.

Download

Opens a *Folder-Tree* dialog box to enable you to download the selected file.

Make Public

Sets permissions on the selected file to *Open/Download* and *Everyone*. (Equivalent to selecting the *Make everything public* check box on the *Upload Settings* dialog box.)

Delete

Deletes the selected files or folders. You can also delete files or folders by choosing them and pressing `Delete`.

Change Storage Class

Sets the storage class to either Standard or Reduced Redundancy Storage (RRS). To view the current storage class setting, choose *Properties*.

Change Encryption

Enables you to set server-side encryption on the file. To view the current encryption setting, choose *Properties*.

Rename

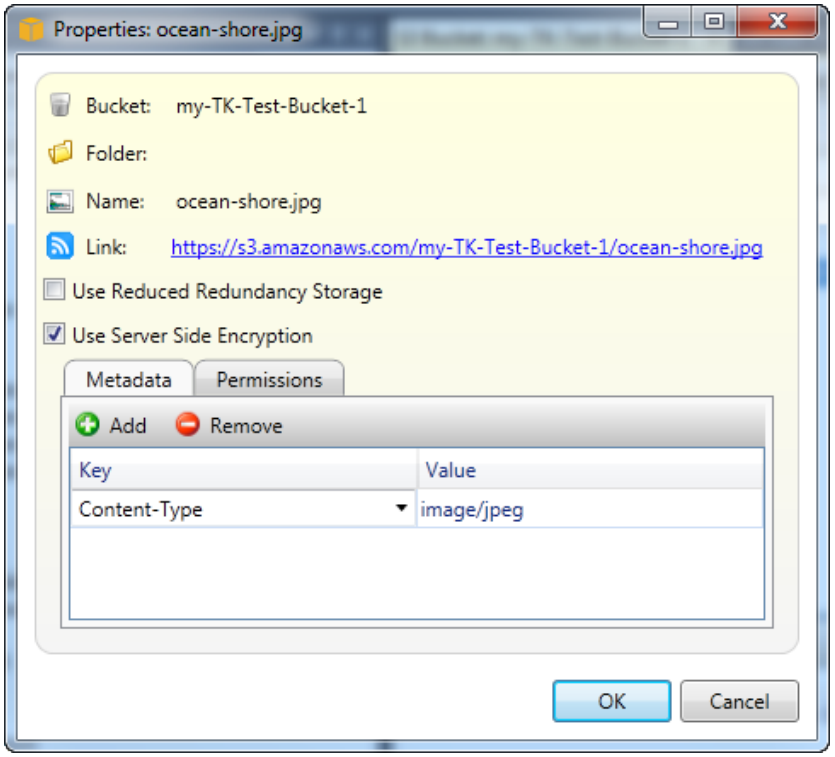
Enables you to rename a file. You cannot rename a folder.

Cut | *Copy* | *Paste*

Enables you to cut, copy, and paste files or folders between folders or between buckets.

Properties

Displays a dialog box that enables you to set metadata and permissions for the file, as well as toggle storage for the file between Reduced Redundancy Storage (RRS) and Standard, and set server-side encryption for the file. This dialog box also displays an https link to the file. If you choose this link, the Toolkit for Visual Studio opens the file in your default browser. If you have permissions on the file set to *Open/Download* and *Everyone*, other people will be able to access the file through this link. Rather than distributing this link, we recommend you create and distribute pre-signed URLs.



Create Pre-Signed URL

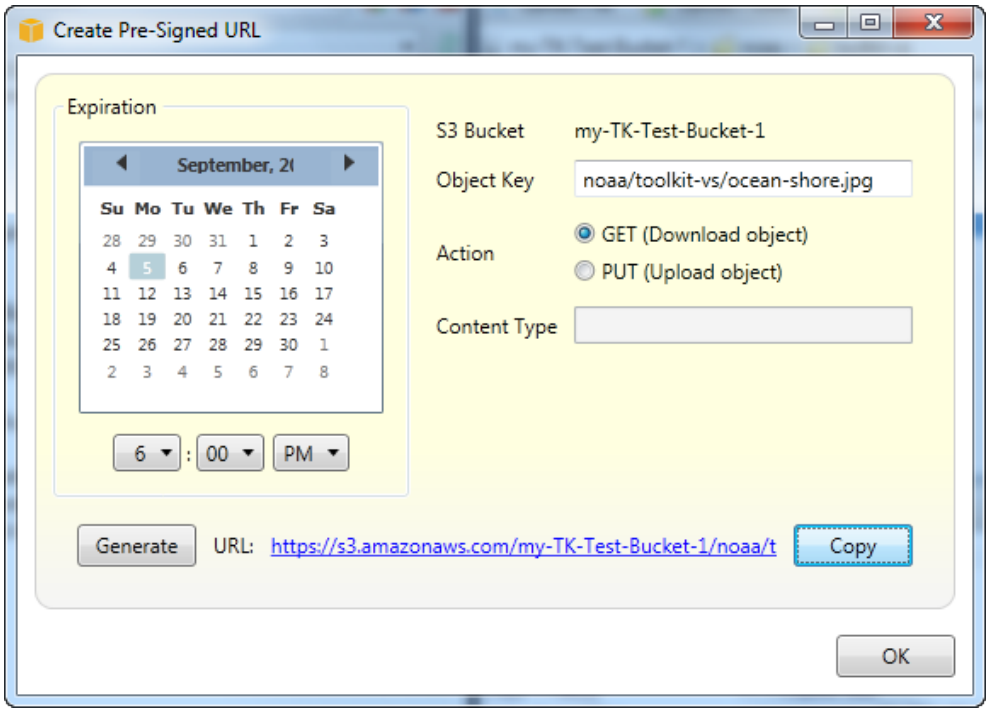
Enables you to create a time-limited pre-signed URL that you can distribute to enable other people to access the content you have stored on Amazon S3.

How to Create a Pre-Signed URL

You can create a pre-signed URL for a bucket or files in a bucket. Other people can then use this URL to access the bucket or file. The URL will expire after a period of time that you specify when you create the URL.

To create a pre-signed URL

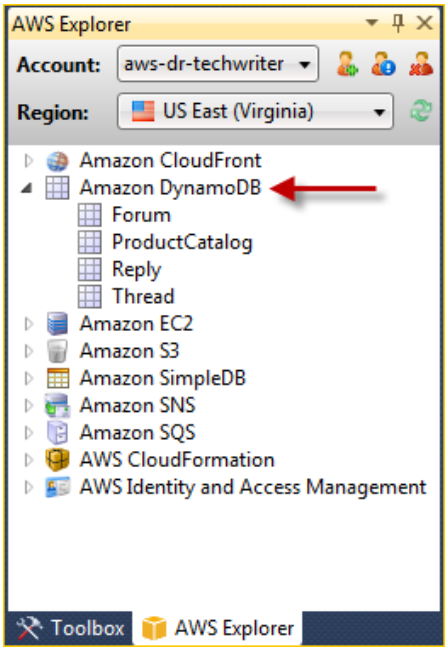
1. In the *Create Pre-Signed URL* dialog box, set the expiration date and time for the URL. The default setting is one hour from the current time.
2. Choose the *Generate* button.
3. To copy the URL to the clipboard, choose *Copy*.



3.9 Using DynamoDB from AWS Explorer

Amazon DynamoDB is a fast, highly scalable, highly available, cost-effective, non-relational database service. DynamoDB removes traditional scalability limitations on data storage while maintaining low latency and predictable performance. The Toolkit for Visual Studio provides functionality for working with DynamoDB in a development context. For more information about DynamoDB, see DynamoDB on the AWS website.

In the Toolkit for Visual Studio, AWS Explorer displays all of the DynamoDB tables associated with the active AWS account.

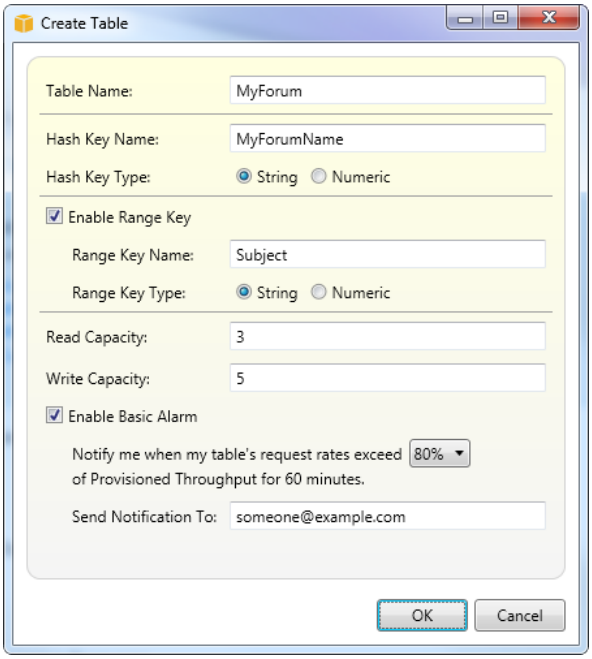


3.9.1 Creating an DynamoDB Table

You can use the Toolkit for Visual Studio to create a DynamoDB table.

To create a table in AWS Explorer

1. In AWS Explorer, open the context (right-click) menu for *Amazon DynamoDB*, and then choose *Create Table*.
2. In the *Create Table* wizard, in *Table Name*, type a name for the table.
3. In the *Hash Key Name* field, type a primary hash key attribute and from the *Hash Key Type* buttons, choose the hash key type. DynamoDB builds an unordered hash index using the primary key attribute and an optional sorted range index using the range primary key attribute. For more information about the primary hash key attribute, go to the Primary Key section in the DynamoDB Developer Guide.
4. (Optional) Select *Enable Range Key*. In the *Range Key Name* field, type a range key attribute, and then from the *Range Key Type* buttons, choose a range key type.
5. In the *Read Capacity* field, type the number of read capacity units. In the *Write Capacity* field, type the number of write capacity units. You must specify a minimum of three read capacity units and five write capacity units. For more information about read and write capacity units, go to Provisioned Throughput in DynamoDB.
6. (Optional) Select *Enable Basic Alarm* to alert you when your table's request rates are too high. Choose the percentage of provisioned throughput per 60 minutes that must be exceeded before the alert is sent. *In Send Notifications To*, type an email address.
7. Click *OK* to create the table.

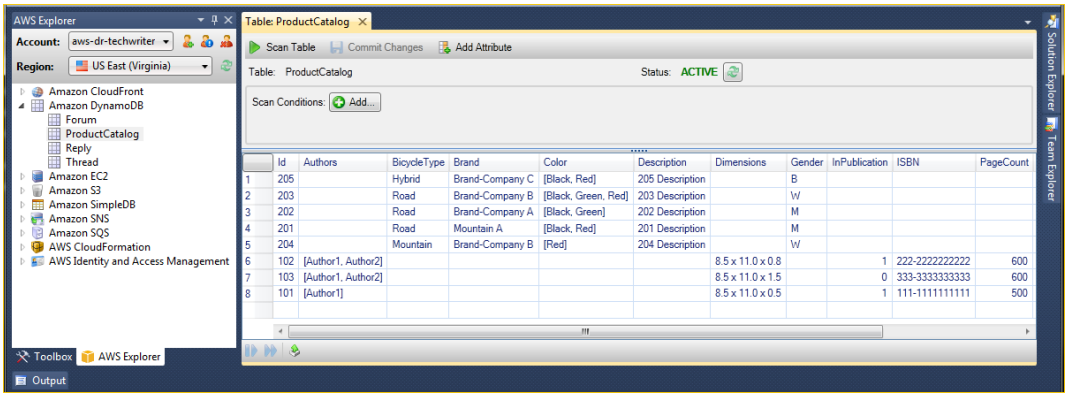


For more information about DynamoDB tables, go to Data Model Concepts - Tables, Items, and Attributes.

3.9.2 Viewing an DynamoDB Table as a Grid

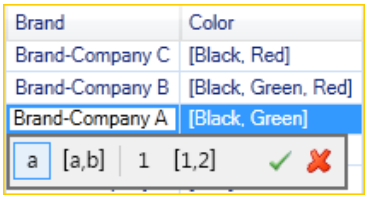
To open a grid view of one of your DynamoDB tables, in AWS Explorer, double-click the subnode that corresponds to the table. From the grid view, you can view the items, attributes, and values stored in the table. Each row corresponds to an item in the table. The table columns correspond to attributes. Each cell of the table holds the values associated with that attribute for that item.

An attribute can have a value that is a string or a number. Some attributes have a value that consists of a *set* of strings or numbers. Set values are displayed as a comma-separated list enclosed by square brackets.

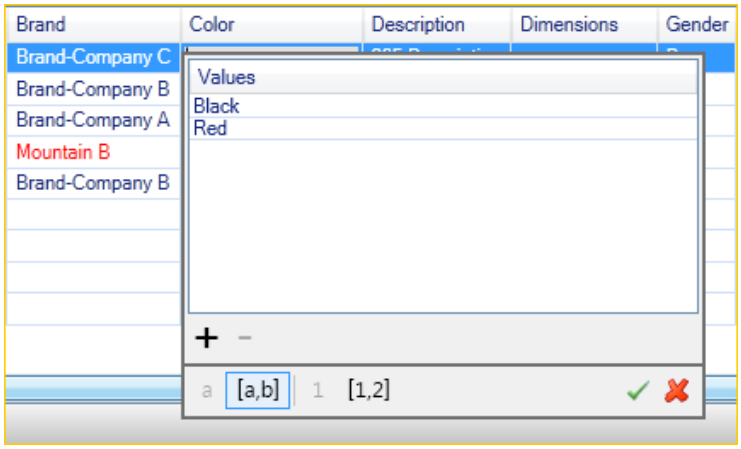


3.9.3 Editing and Adding Attributes and Values

By double-clicking a cell, you can edit the values for the item's corresponding attribute. For set-value attributes, you can also add or delete individual values from the set.



In addition to changing the value of an attribute, you can also, with some limitations, change the format of the value for an attribute. For example, any number value can be converted into a string value. If you have a string value, the content of which is a number, such as 125, the cell editor enables you to convert the format of the value from string to number. You can also convert a single-value to a set-value. However, you cannot generally convert from a set-value to a single-value; an exception is when the set-value has, in fact, only one element in the set.

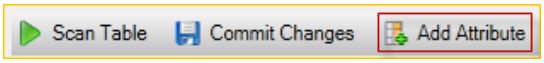


After editing the attribute value, choose the green check mark to confirm your changes. If you want to discard your changes, choose the red X.

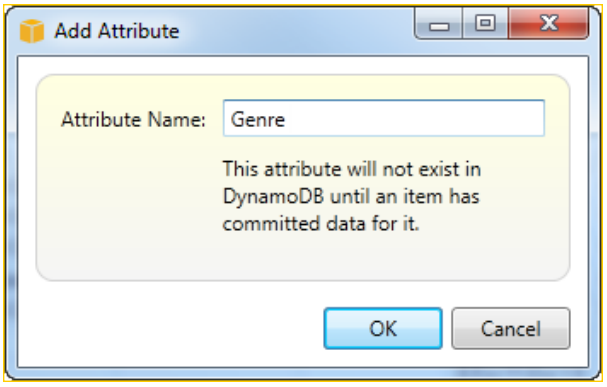
After you have confirmed your changes, the attribute value will be displayed in red. This indicates the attribute has been updated, but that the new value has not been written back to the DynamoDB database. To write your changes back to DynamoDB, choose *Commit Changes*. To discard your changes, choose *Scan Table* and when the Toolkit asks if you would like to commit your changes before the Scan, choose *No*.

Adding an Attribute

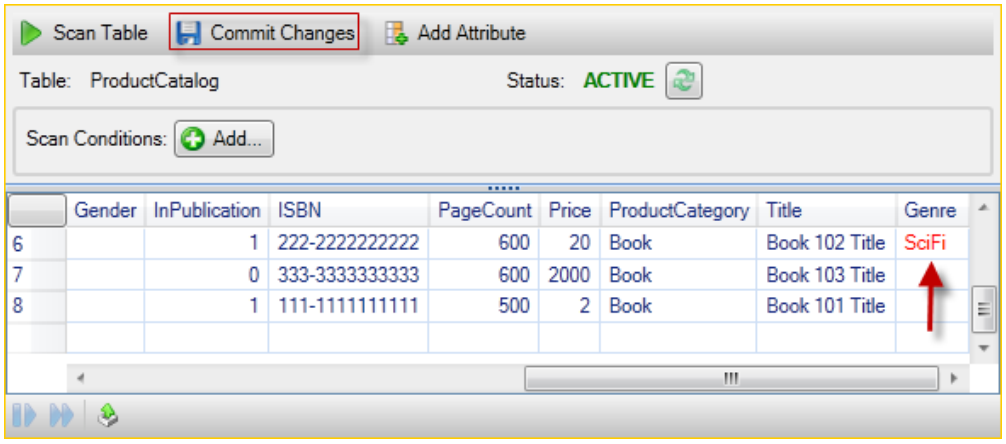
From the grid view, you can also add attributes to the table. To add a new attribute, choose *Add Attribute*.



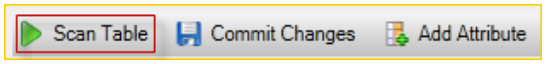
In the *Add Attribute* dialog box, type a name for your attribute, and then choose *OK*.



To make the new attribute become part of the table, you must add a value to it for at least one item and then choose the *Commit Changes* button. To discard the new attribute, just close the grid view of the table without choosing *Commit Changes*.



3.9.4 Scanning an DynamoDB Table



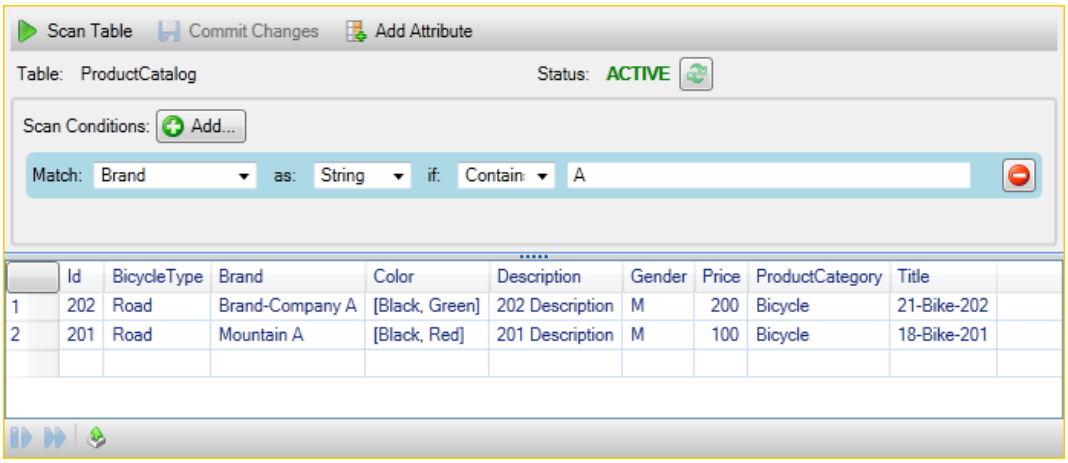
You can perform Scans on your DynamoDB tables from the Toolkit. In a Scan, you define a set of criteria and the Scan returns all items from the table that match your criteria. Scans are expensive operations and should be used with care to avoid disrupting higher priority production traffic on the table. For more information about using the Scan operation, go to the DynamoDB Developer Guide.

To perform a Scan on an DynamoDB table from AWS Explorer

1. In the grid view, choose the *scan conditions: add* button.
2. In the Scan clause editor, choose the attribute to match against, how the value of the attribute should be interpreted (string, number, set value), how it should be matched (for example Begins With or Contains), and the literal value it should match.

3. Add more Scan clauses, as needed, for your search. The Scan will return only those items that match the criteria from all of your Scan clauses. The Scan will perform a case-sensitive comparison when matching against string values.
4. On the button bar at the top of the grid view, choose *Scan Table*.

To remove a Scan clause, choose the red button with the white line to the right of each clause.



To return to the view of the table that includes all items, remove all Scan clauses and choose *Scan Table* again.

Paginating Scan Results

At the bottom of the view are three buttons.

The first two blue buttons provide pagination for Scan results. The first button will display an additional page of results. The second button will display an additional ten pages of results. In this context, a page is equal to 1 MB of content.

Export Scan Result to CSV

The third button exports the results from the current Scan to a CSV file.

3.10 Amazon RDS from AWS Explorer

Amazon Relational Database Service (Amazon RDS) is a service that enables you to provision and manage SQL relational database systems in the cloud. Amazon RDS supports three types of database systems:

- MySQL Community Edition
- Oracle Database Enterprise Edition
- Microsoft SQL Server (Express, Standard, or Web Editions)

For more information, see the Amazon RDS User Guide.

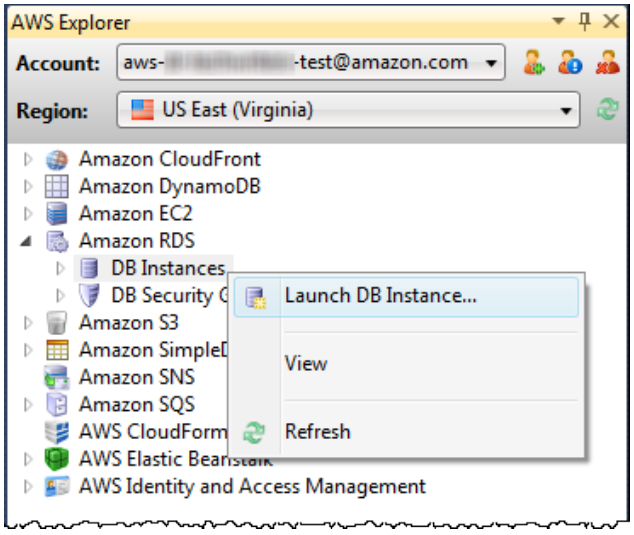
A lot of the functionality discussed here is also available through the AWS Management Console for Amazon RDS.

3.10.1 Launch an Amazon RDS Database Instance

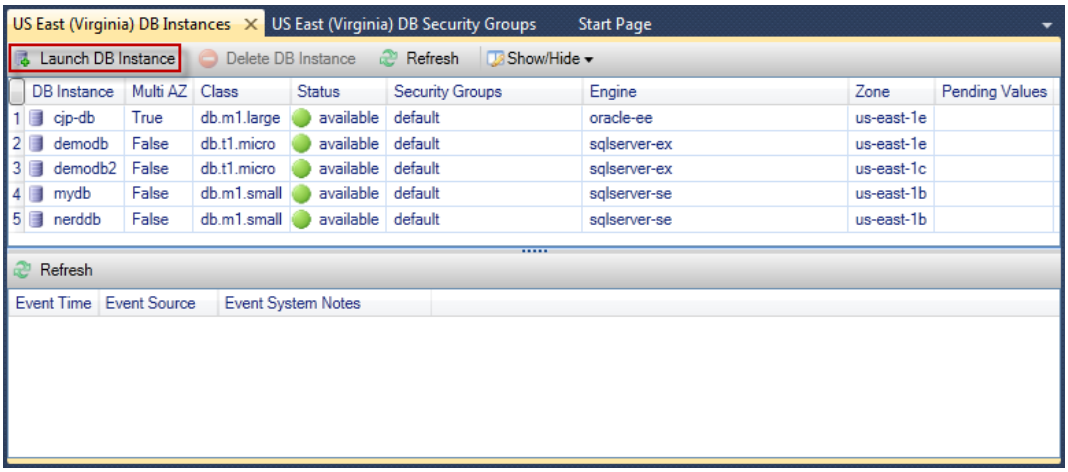
With AWS Explorer, you can launch an instance of any of the database engines supported by Amazon RDS. The following walkthrough shows the user experience for launching an instance of Microsoft SQL Server Standard Edition, but the user experience is similar for all supported engines.

To launch an Amazon RDS instance

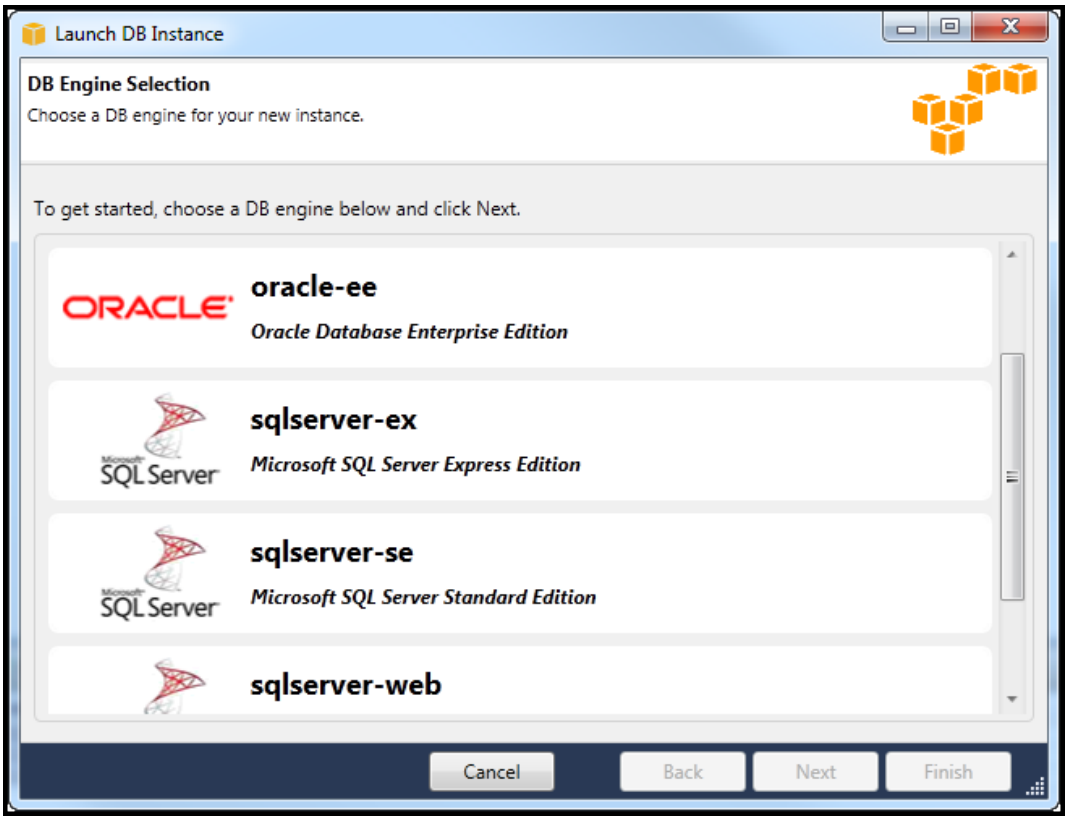
1. In AWS Explorer, open the context (right-click) menu for the *Amazon RDS* node and choose *Launch DB Instance*.



Alternatively, on the *DB Instances* tab, choose *Launch DB Instance*.



2. In the *DB Engine Selection* dialog box, choose the type of database engine to launch. For this walkthrough, choose Microsoft SQL Server Standard Edition (sqlserver-se), and then choose *Next*.



3. In the *DB Engine Instance Options* dialog box, choose configuration options.

   In the *DB Engine Instance Options and Class* section, you can specify the following settings.

   License Model

Engine Type	License
Microsoft SQL Server	license-included
MySql	general-public-license
Oracle	bring-your-own-license

   The license model varies, depending on the type of database engine. Engine Type License Microsoft SQL Server license-included MySql general-public-license Oracle bring-your-own-license

   DB Instance Version Choose the version of the database engine you would like to use. If only one version is supported, it is selected for you.

   DB Instance Class Choose the instance class for the database engine. Pricing for instance classes varies. For more information, see the Amazon RDS Pricing of the Amazon RDS detail page.

   Perform a multi AZ deployment Select this option to create a multi-AZ deployment for enhanced data durability and availability. Amazon RDS provisions and maintains a standby copy of your database in a different Availability Zone for automatic failover in the event of a scheduled or

unplanned outage. For information about pricing for multi-AZ deployments, see the pricing section of the Amazon RDS detail page. This option is not supported for Microsoft SQL Server.

Upgrade minor versions automatically Select this option to have AWS automatically perform minor version updates on your RDS instances for you.

In the *RDS Database Instance* section, you can specify the following settings.

Allocated Storage

Engine	Minimum (GB)	Maximum (GB)
MySQL	5	1024
Oracle Enterprise Edition	10	1024
Microsoft SQL Server Express Edition	30	1024
Microsoft SQL Server Standard Edition	250	1024
Microsoft SQL Server Web Edition	30	1024

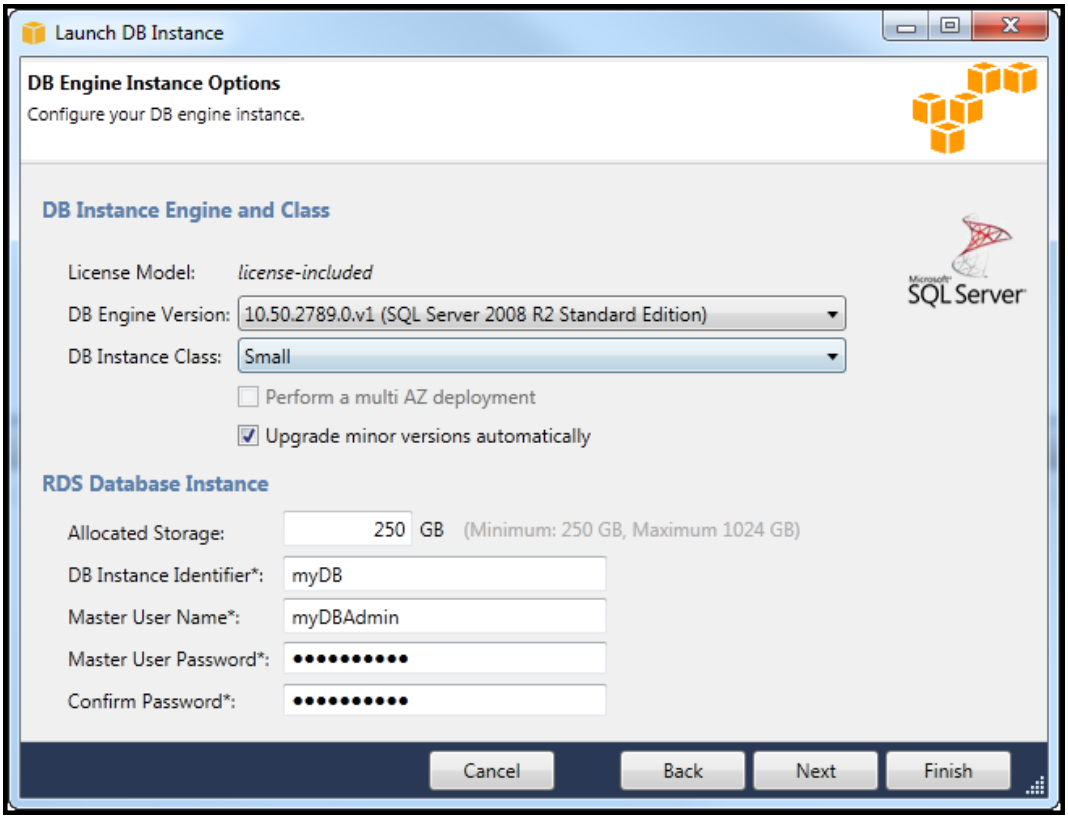
The minimums and maximums for allocated storage depend on the type of database engine. Engine Minimum (GB) Maximum (GB) MySQL 5 1024 Oracle Enterprise Edition 10 1024 Microsoft SQL Server Express Edition 30 1024 Microsoft SQL Server Standard Edition 250 1024 Microsoft SQL Server Web Edition 30 1024

DB Instance Identifier Specify a name for the database instance. This name is not case-sensitive. It will be displayed in lowercase form in AWS Explorer.

Master User Name Type a name for the administrator of the database instance.

Master User Password Type a password for the administrator of the database instance.

Confirm Password Type the password again to verify it is correct.



4. In the *Additional Options* dialog box, you can specify the following settings.

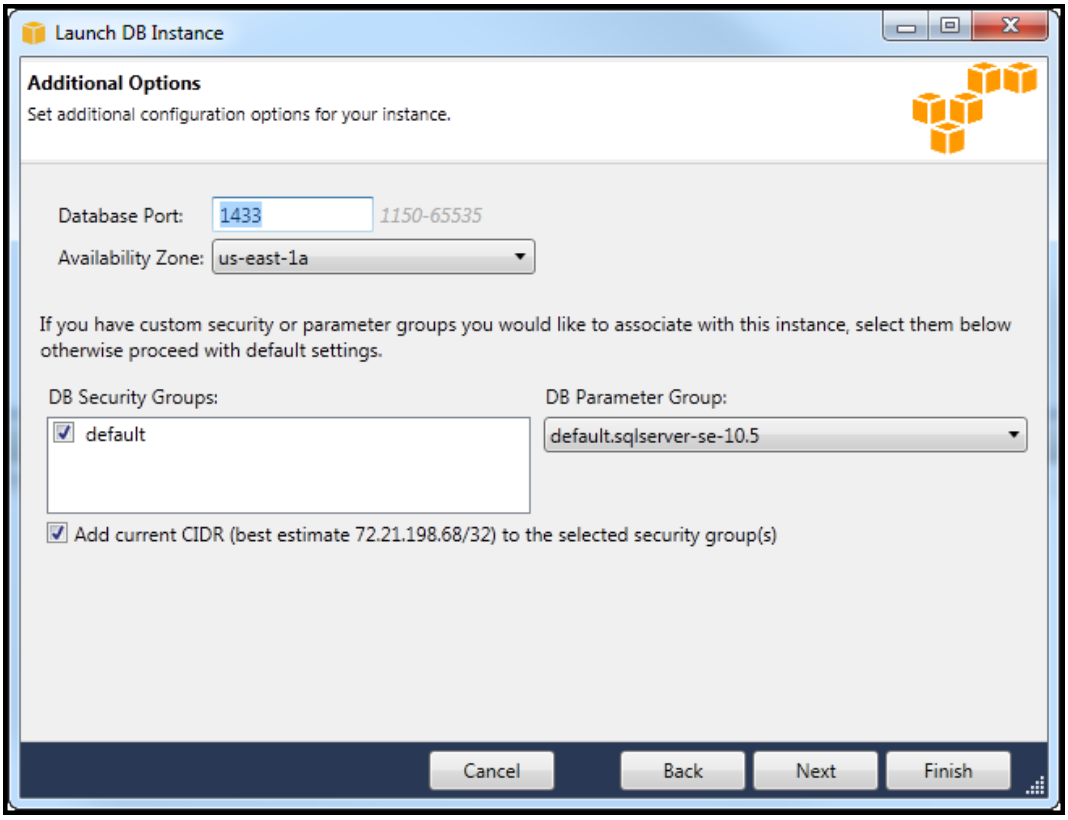
   Database Port This is the TCP port the instance will use to communicate on the network. If your computer accesses the Internet through a firewall, set this value to a port through which your firewall allows traffic.

   Availability Zone Use this option if you want the instance to be launched in a particular Availability Zone in your region. The database instance you have specified might not be available in all Availability Zones in a given region.

   RDS Security Group Select an RDS security group (or groups) to associate with your instance. RDS security groups specify the IP address, Amazon EC2 instances, and AWS accounts that are allowed to access your instance. For more information about RDS security groups, see *Amazon RDS Security Groups*. The Toolkit for Visual Studio attempts to determine your current IP address and provides the option to add this address to the security groups associated with your instance. However, if your computer accesses the Internet through a firewall, the IP address the Toolkit generates for your computer may not be accurate. To determine which IP address to use, consult your system administrator.

   DB Parameter Group (Optional) From this drop-down list, choose a DB parameter group to associate with your instance. DB parameter groups enable you to change the default configuration for the instance. For more information, go to the Amazon Relational Databse Service User Guide and this article.

   When you have specified settings on this dialog box, choose *Next*.

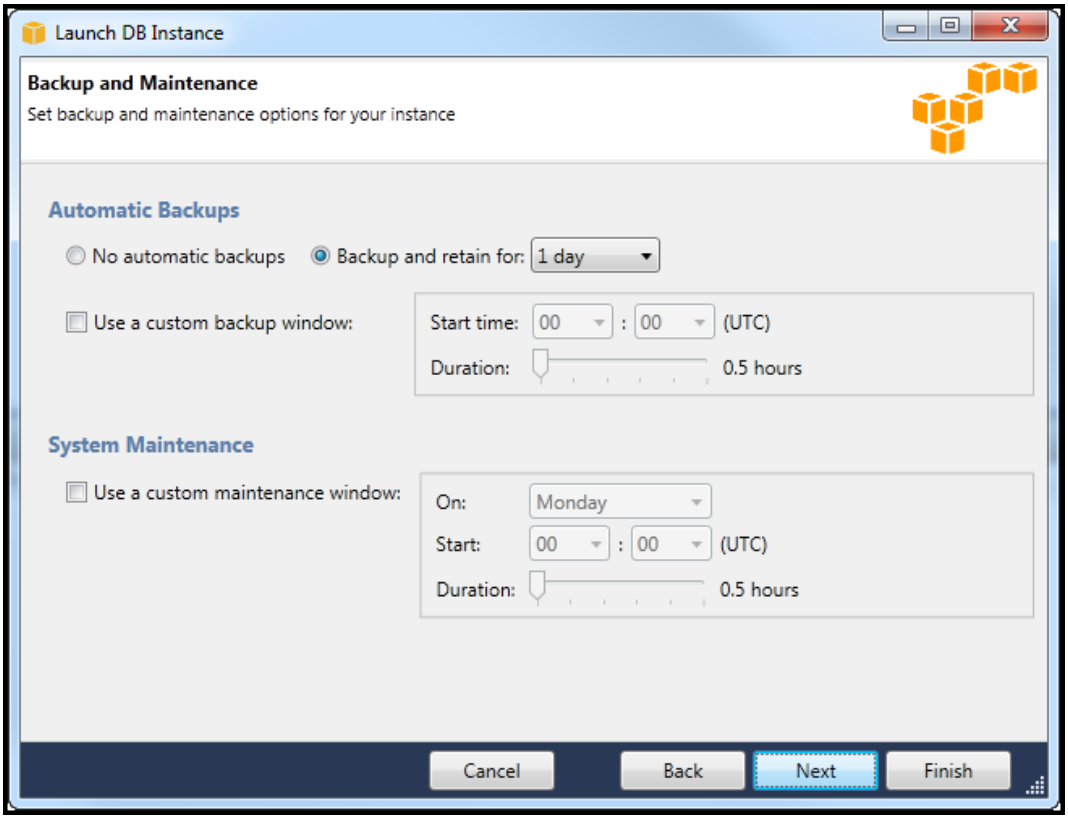


5. The *Backup and Maintenance* dialog box enables you to specify whether Amazon RDS should back up your instance and if so, for how long the backup should be retained. You can also specify a window of time during which the backups should occur.

   This dialog box also enables you to specify if you would like Amazon RDS to perform system maintenance on your instance. Maintenance includes routine patches and minor version upgrades.

   The window of time you specify for system maintenance cannot overlap with the window specified for backups.

   Choose *Next*.



6. The final dialog box in the wizard allows you to review the settings for your instance. If you need to modify settings, use the *Back* button. If all the settings are correct, choose *Launch*.

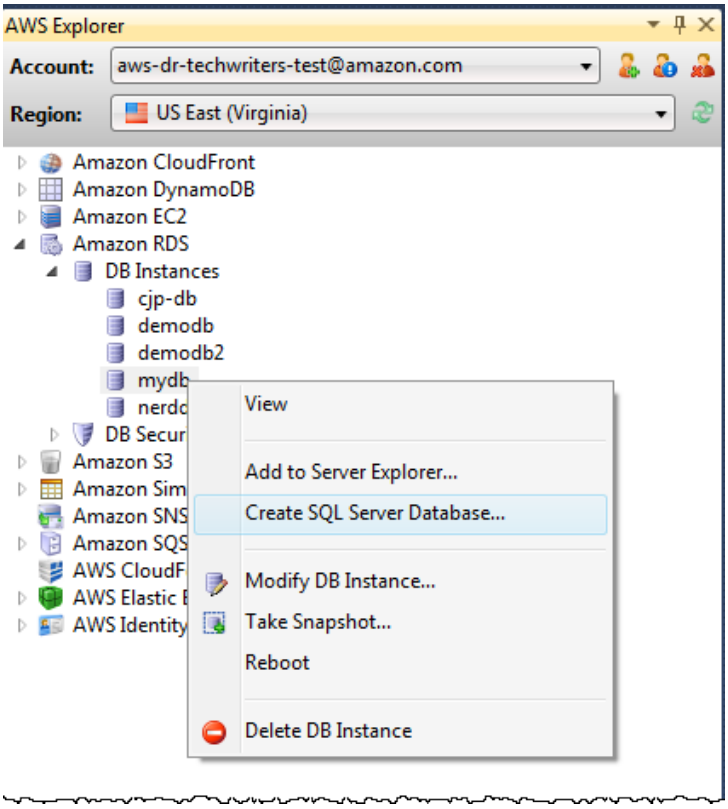
3.10.2 Create a Microsoft SQL Server Database in an RDS Instance

Microsoft SQL Server is designed in such a way that, after launching an Amazon RDS instance, you need to create an SQL Server database in the RDS instance.

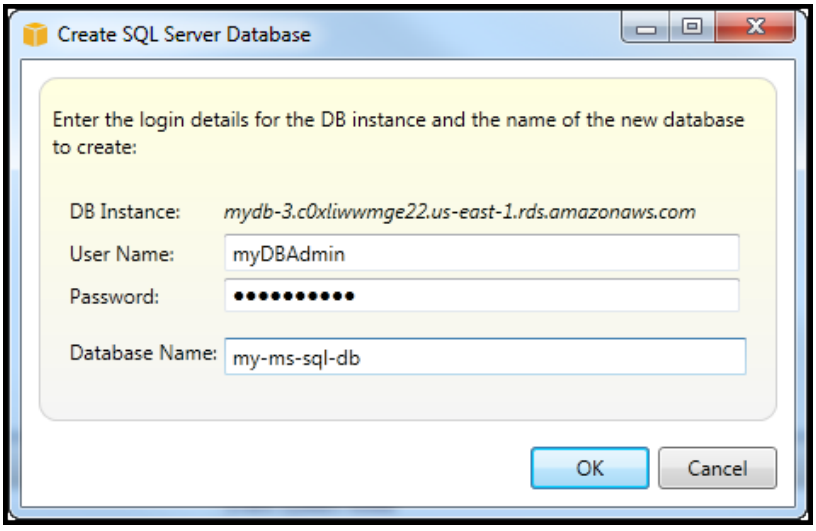
For information about how to create an Amazon RDS instance, see *Launch an Amazon RDS Database Instance*.

To create a Microsoft SQL Server database

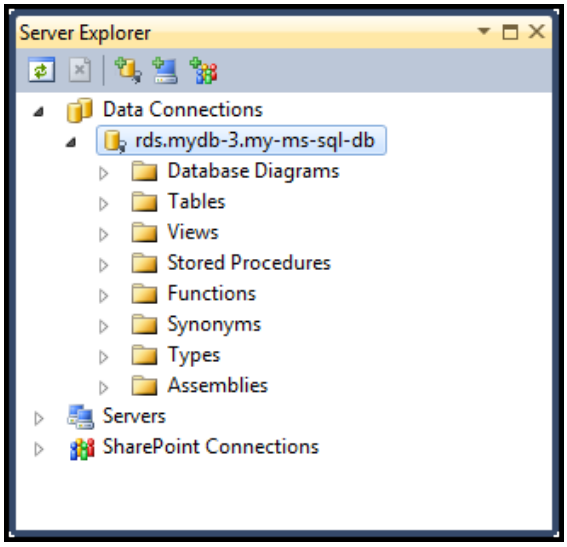
1. In AWS Explorer, open the context (right-click) menu for the node that corresponds to your RDS instance for Microsoft SQL Server, and choose *Create SQL Server Database*.



2. In the *Create SQL Server Database* dialog box, type the password you specified when you created the RDS instance, type a name for the Microsoft SQL Server database, and then choose *OK*.



3. The Toolkit for Visual Studio creates the Microsoft SQL Server database and adds it to the Visual Studio Server Explorer.



3.10.3 Amazon RDS Security Groups

Amazon RDS security groups enable you to manage network access to your Amazon RDS instances. With security groups, you specify sets of IP addresses using CIDR notation, and only network traffic originating from these addresses is recognized by your Amazon RDS instance.

Although they function in a similar way, Amazon RDS security groups are different from Amazon EC2 security groups. It is possible to add an EC2 security group to your RDS security group. Any EC2 instances that are members of the EC2 security group are then able to access the RDS instances that are members of the RDS security group.

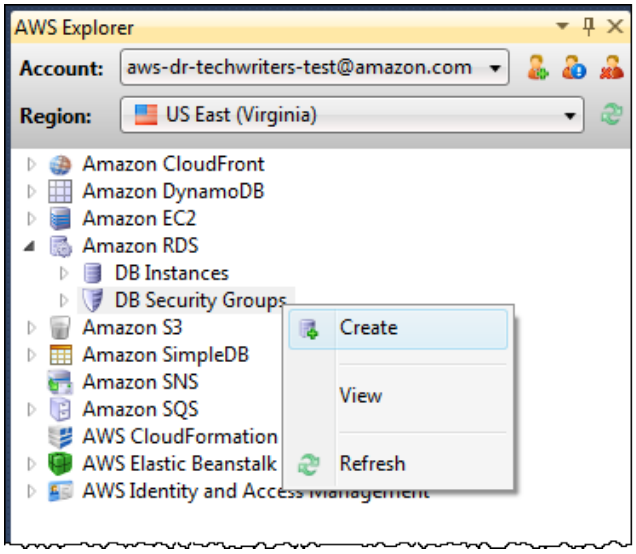
For more information about Amazon RDS security groups, go to the RDS Security Groups. For more information about Amazon EC2 security groups, go to the EC2 User Guide.

Create an Amazon RDS Security Group

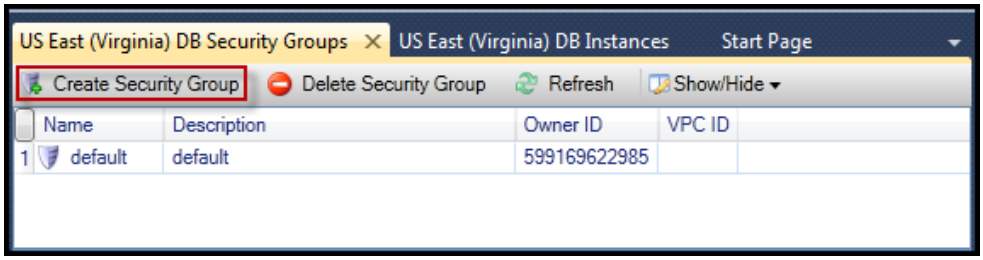
You can use the Toolkit for Visual Studio to create an RDS security group. If you use the AWS Toolkit to launch an RDS instance, the wizard will allow you to specify an RDS security group to use with your instance. You can use the following procedure to create that security group before you start the wizard.

To create an Amazon RDS security group

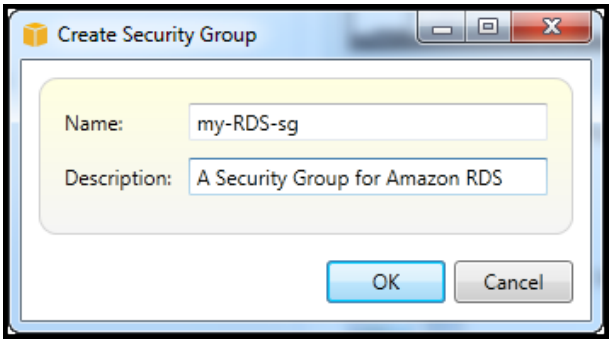
1. In AWS Explorer, expand the *Amazon RDS* node, open the context (right-click) menu for the *DB Security Groups* subnode and choose *Create*.



Alternatively, on the *Security Groups* tab, choose *Create Security Group*. If this tab isn't displayed, open the context (right-click) menu for the *DB Security Groups* subnode and choose *View*.



2. In the *Create Security Group* dialog box, type a name and description for the security group, and then choose *OK*.



Set Access Permissions for an Amazon RDS Security Group

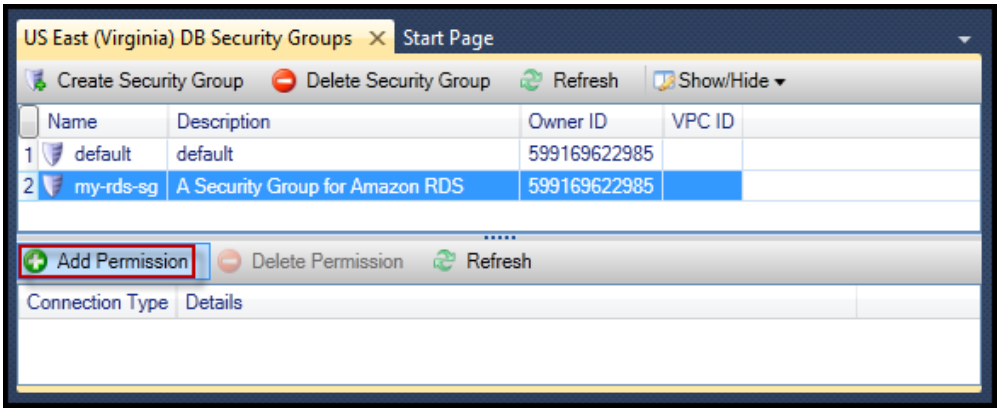
By default, a new Amazon RDS security group provides no network access. To enable access to Amazon RDS instances that use the security group, use the following procedure to set its access permissions.

To set access for an Amazon RDS security group

1. On the *Security Groups* tab, choose the security group from the list view. If your security group does not appear in the list, choose *Refresh*. If your security group still does not appear in the list, verify you are viewing the list for the correct AWS region. *Security Group* tabs in the AWS Toolkit are region-specific.

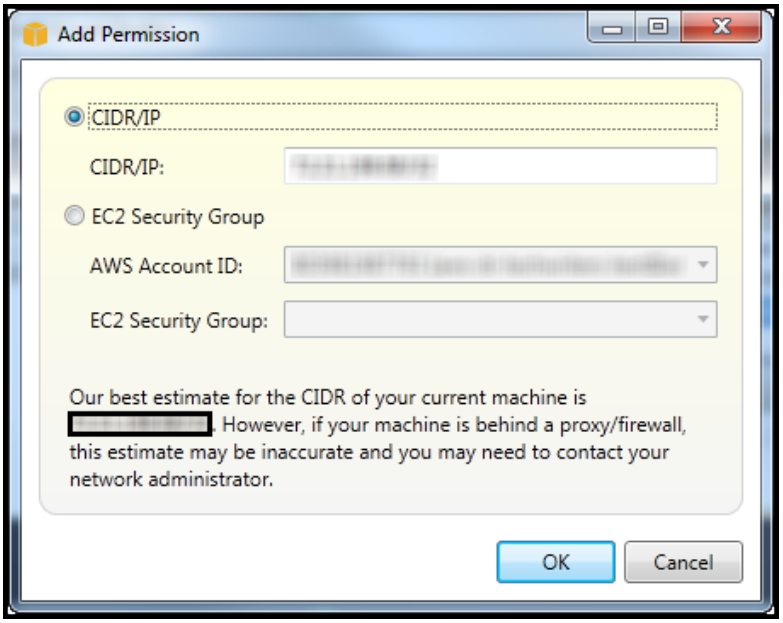
   If no *Security Group* tabs appear, in AWS Explorer, open the context (right-click) menu for the *DB Security Groups* subnode and choose *View*.

2. Choose *Add Permission*.



Add Permissions button on the *Security Groups* tab

3. In the *Add Permission* dialog box, you can use CIDR notation to specify which IP addresses can access your RDS instance, or you can specify which EC2 security groups can access your RDS instance. When you choose *EC2 Security Group*, you can specify access for all EC2 instances associated with an AWS account have access, or you can choose a EC2 security group from the drop-down list.



The AWS Toolkit attempts to determine your IP address and auto-populate the dialog box with the

appropriate CIDR specification. However, if your computer accesses the Internet through a firewall, the CIDR determined by the Toolkit may not be accurate.

3.11 Using Amazon SimpleDB from AWS Explorer

AWS Explorer displays all of the Amazon SimpleDB domains associated with the active AWS account. From AWS Explorer, you can create or delete Amazon SimpleDB domains.

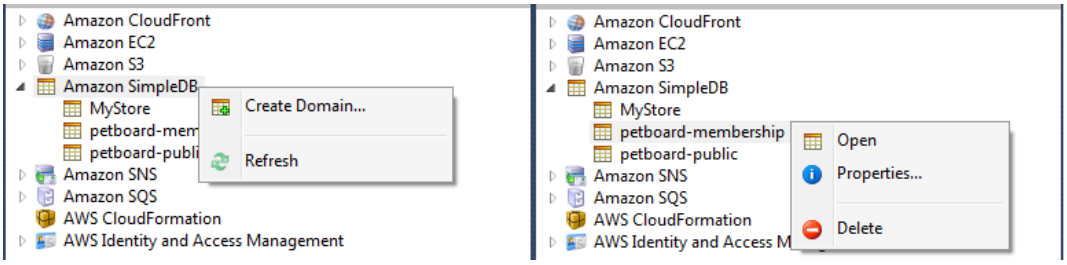


Fig. 3.1: Create, delete, or open Amazon SimpleDB domains associated with your account

Executing Queries and Editing the Results

AWS Explorer can also display a grid view of a Amazon SimpleDB domain from which you can view the items, attributes, and values in that domain. You can execute queries so that only a subset of the domain's items is displayed. By double-clicking a cell, you can edit the values for that item's corresponding attribute. You can also add new attributes to the domain.

The domain displayed here is from the Amazon SimpleDB sample included with the AWS SDK for .NET.

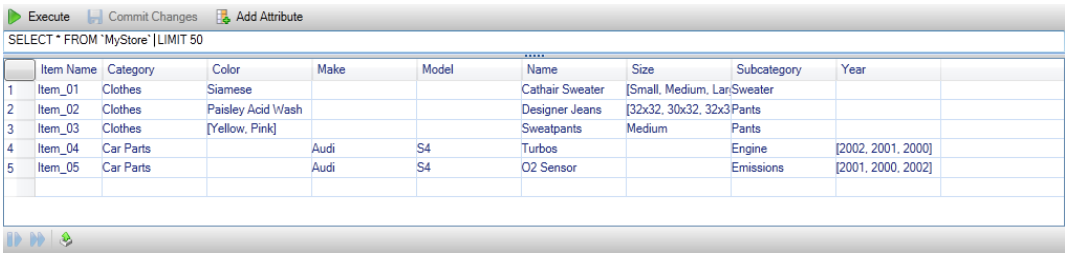


Fig. 3.2: Amazon SimpleDB grid view

To execute a query, edit the query in the text box at the top of the grid view, and then choose *Execute*. The view is filtered to show only the items that match the query.

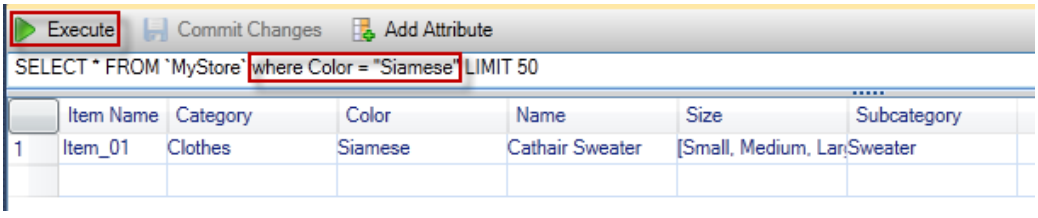


Fig. 3.3: Execute query from AWS Explorer

To edit the values associated with an attribute, double-click the corresponding cell, edit the values, and then choose *Commit Changes*.

Adding an Attribute

To add an attribute, at the top of the view, choose *Add Attribute*.

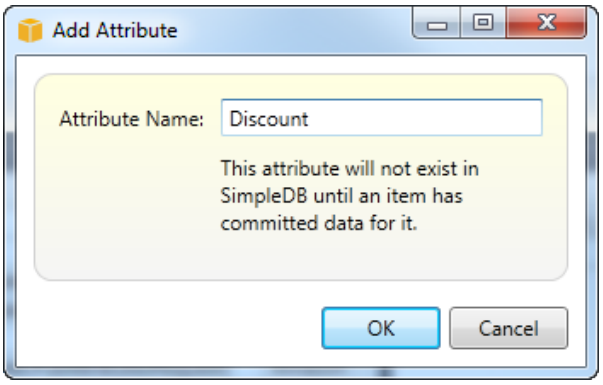


Fig. 3.4: *Add Attribute* dialog box

To make the attribute part of the domain, you must add a value for it to at least one item and then choose *Commit Changes*.

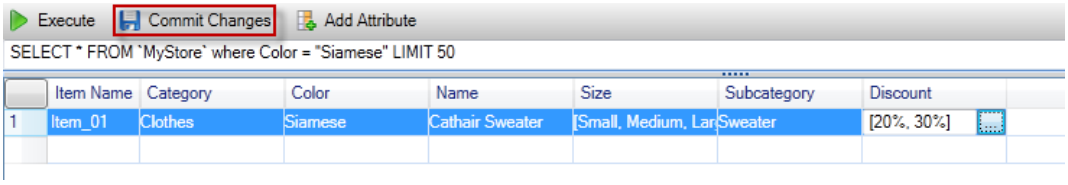


Fig. 3.5: Commit changes for a new attribute

Paginating Query Results

There are three buttons at the bottom of the view.

Fig. 3.6: Paginate and export buttons

The first two buttons provide pagination for query results. To display an additional page of results, choose the first button. To display an additional ten pages of results, choose the second button. In this context, a page is equal to 100 rows or the number of results specified by the LIMIT value, if it is included in the query.

Export to CSV

The last button exports the current results to a CSV file.

3.12 Using Amazon SQS from AWS Explorer

Amazon Simple Queue Service (Amazon SQS) is a flexible queue service that enables message passing between different processes of execution in a software application. Amazon SQS queues are located in the AWS infrastructure, but the processes that are passing messages can be located locally, on Amazon EC2 instances, or on some combination of these. Amazon SQS is ideal for coordinating the distribution of work across multiple computers.

The Toolkit for Visual Studio enables you to view Amazon SQS queues associated with the active account, create and delete queues, and send messages through queues. (By active account, we mean the account selected in AWS Explorer.)

For more information about Amazon SQS, go to Introduction to SQS in the AWS documentation.

3.12.1 Creating a Queue

You can create an Amazon SQS queue from AWS Explorer. The ARN and URL for the queue will be based on the account number for the active account and the queue name you specify at creation.

To create a queue

1. In AWS Explorer, open the context (right-click) menu for the *Amazon SQS* node, and then choose *Create Queue*.
2. In the *Create Queue* dialog box, specify the queue name, the default visibility timeout, and the default delivery delay. The default visibility timeout and the default delivery delay are specified in seconds. The default visibility timeout is the amount of time that a message will be invisible to potential receiving processes after a given process has acquired the message. The default delivery delay is the amount of time from the moment the message is sent to the moment it first becomes visible to potential receiving processes.
3. Choose *OK*. The new queue will appear as a subnode under the *Amazon SQS* node.

3.12.2 Deleting a Queue

You can delete existing queues from AWS Explorer. If you delete a queue, any messages associated with the queue are no longer available.

To delete a queue

1. In AWS Explorer, open the context (right-click) menus for the queue you want to delete, and then choose *Delete*.

3.12.3 Managing Queue Properties

You can view and edit the properties for any of the queues displayed in AWS Explorer. You can also send messages to the queue from this properties view.

To manage queue properties

- In AWS Explorer, open the context (right-click) menu for the queue whose properties you want to manage, and then choose *View Queue*.

  From the queue properties view, you can edit the visibility timeout, the maximum message size, message retention period, and default delivery delay. The default delivery delay can be overridden when you send a message. In the following screenshot, the obscured text is the account number component of the queue ARN and URL.

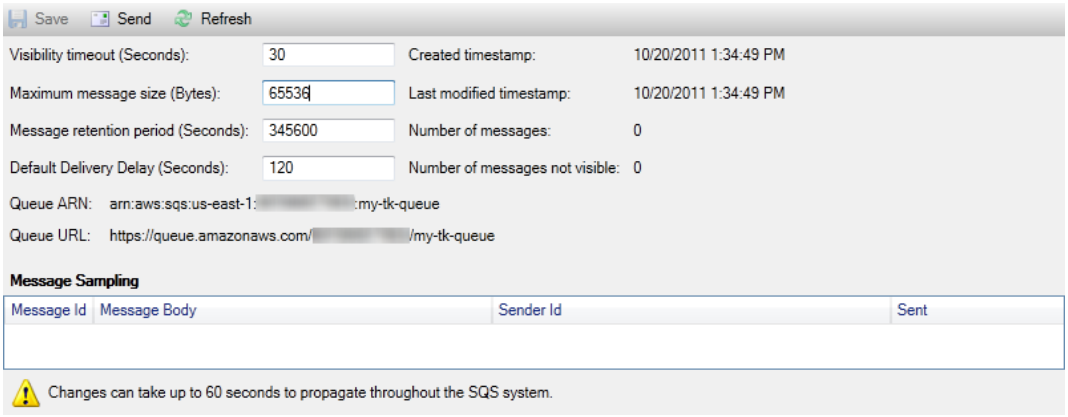


Fig. 3.7: SQS queue properties view

3.12.4 Sending a Message to a Queue

From the queue properties view, you can send a message to the queue.

To send a message

1. At the top of the queue properties view, choose the *Send* button.
2. Type the message. (Optional) Enter a delivery delay that will override the default delivery delay for the queue. In the following example, we have overridden the delay with a value of 240 seconds. Choose *OK*.

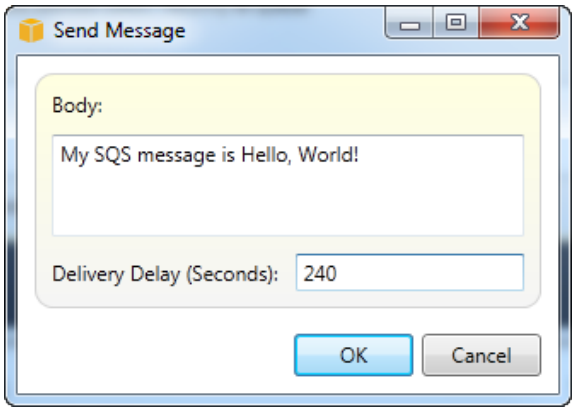


Fig. 3.8: *Send Message* dialog box

3. Wait for approximately 240 seconds (four minutes). The message will appear in the *Message Sampling* section of the of the queue properties view.

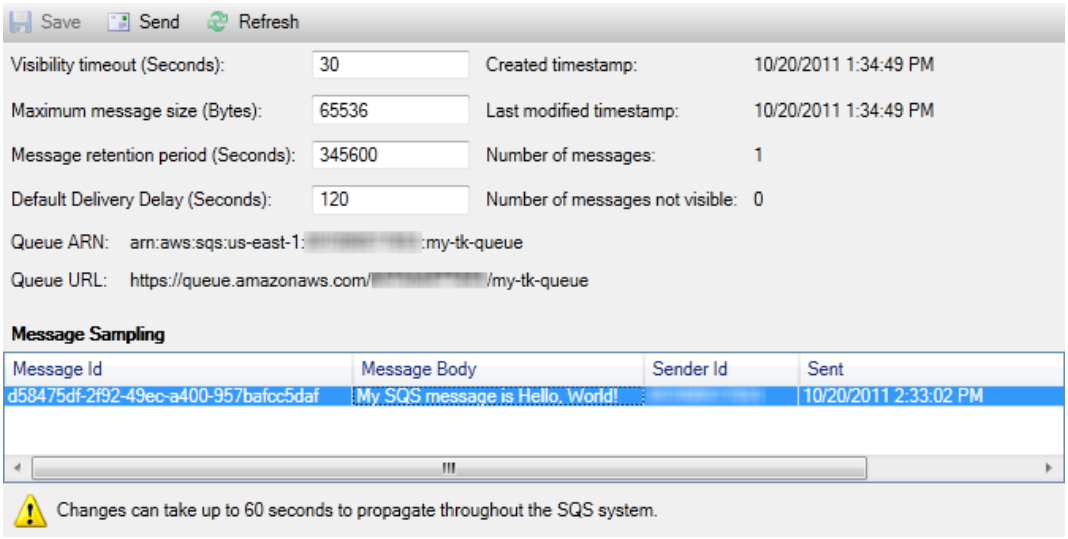


Fig. 3.9: SQS properties view with sent message

The timestamp in the queue properties view is the time you chose the *Send* button. It does not include the delay. Therefore, the time that the message appears in the queue and is available to receivers might be later than this timestamp. The timestamp is displayed in your computer's local time.

3.13 Identity and Access Management

AWS Identity and Access Management (IAM) enables you to more securely manage access to your AWS accounts and resources. With IAM, you can create multiple users in your primary (*root*) AWS account. These users can have their own credentials: password, access key ID, and secret key, but all IAM users share a single account number.

You can manage each IAM user's level of resource access by attaching IAM policies to the user. For example, you can attach a policy to an IAM user that gives the user access to the Amazon S3 service and related resources in your account, but which doesn't provide access to any other services or resources.

For more efficient access management, you can create IAM groups, which are collections of users. When you attach a policy to the group, it will affect all users who are members of that group.

In addition to managing permissions at the user and group level, IAM also supports the concept of IAM roles. Like users and groups, you can attach policies to IAM roles. You can then associate the IAM role with an Amazon EC2 instance. Applications that run on the EC2 instance are able to access AWS using the permissions provided by the IAM role. For more information about using IAM roles with the Toolkit, see *Create an IAM Role*. For more information about IAM, go to the IAM User Guide.

3.13.1 Create and Configure an IAM User

IAM users enable you to grant others access to your AWS account. Because you are able to attach policies to IAM users, you can precisely limit the resources an IAM user can access and the operations they can perform on those resources.

As a best practice, all users who access an AWS account should do so as IAM users—even the owner of the account. This ensures that if the credentials for one of the IAM users are compromised, just those credentials can be deactivated. There is no need to deactivate or change the root credentials for the account.

From the Toolkit for Visual Studio, you can assign permissions to an IAM user either by attaching an IAM policy to the user or by assigning the user to a group. IAM users who are assigned to a group derive their permissions from the policies attached to the group. For more information, see *Create an IAM Group* and *Add an IAM User to an IAM Group*.

From the Toolkit for Visual Studio, you can also generate AWS credentials (access key ID and secret key) for the IAM user. For more information, see *Generate Credentials for an IAM User*

The Toolkit for Visual Studio supports specifying IAM user credentials for accessing services through AWS Explorer. Because IAM users typically do not have full access to all AWS services, some of the functionality in AWS Explorer might not be available. If you use AWS Explorer to change resources while the active account is an IAM user and then switch the active account to the root account, the changes might not be visible until you refresh the view in AWS Explorer. To refresh the view, choose the refresh () button.

For information about how to configure IAM users from the AWS Management Console, go to Working with Users and Groups in the IAM User Guide.

To create an IAM user

1. In AWS Explorer, expand the *AWS Identity and Access Management* node, open the context (right-click) menu for *Users* and then choose *Create User*.
2. In the *Create User* dialog box, type a name for the IAM user and choose *OK*. This is the IAM friendly name. For information about constraints on names for IAM users, go to the IAM User Guide.

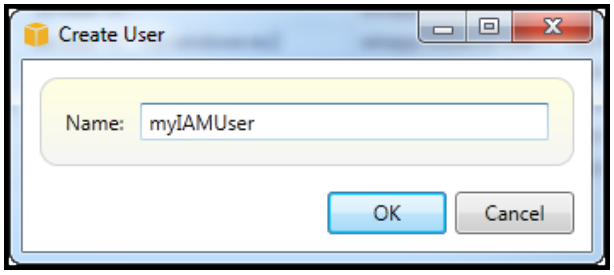


Fig. 3.10: Create an IAM user

The new user will appear as a subnode under *Users* under the *AWS Identity and Access Management* node.

For information about how to create a policy and attach it to the user, see *Create an IAM Policy*.

3.13.2 Create an IAM Group

Groups provide a way of applying IAM policies to a collection of users. For information about how to manage IAM users and groups, go to Working with Users and Groups in the IAM User Guide.

To create an IAM group

1. In AWS Explorer, under *Identity and Access Management*, open the context (right-click) menu for *Groups* and choose *Create Group*.
2. In the *Create Group* dialog box, type a name for the IAM group and choose *OK*.

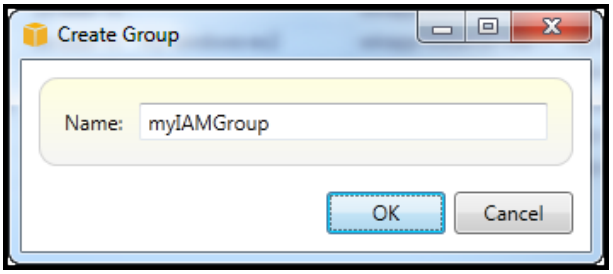


Fig. 3.11: Create IAM group

The new IAM group will appear under the *Groups* subnode of *Identity and Access Management*.

For information about to create a policy and attach it to the IAM group, see *Create an IAM Policy*.

3.13.3 Add an IAM User to an IAM Group

IAM users who are members of an IAM group derive access permissions from the policies attached to the group. The purpose of an IAM group is to make it easier to manage permissions across a collection of IAM users.

For information about how the policies attached to an IAM group interact with the policies attached to IAM users who are members of that IAM group, go to Managing IAM Policies in the IAM User Guide.

In AWS Explorer, you add IAM users to IAM groups from the *Users* subnode, not the *Groups* subnode.

To add an IAM user to a IAM group

1. In AWS Explorer, under *Identity and Access Management*, open the context (right-click) menu for *Users* and choose *Edit*.

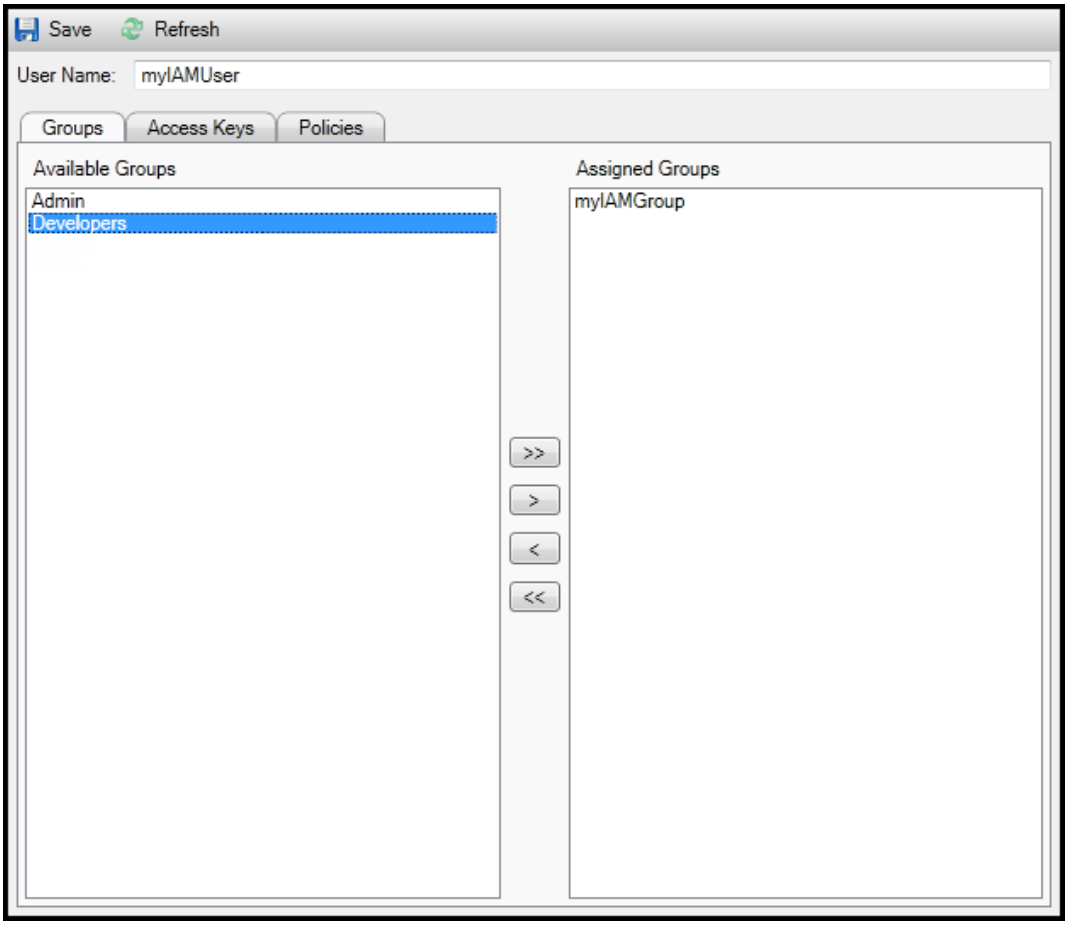


Fig. 3.12: Assign an IAM user to a IAM group

2. The left pane of the *Groups* tab displays the available IAM groups. The right pane displays the groups of which the specified IAM user is already a member.

   To add the IAM user to a group, in the left pane, choose the IAM group and then choose the > button.

   To remove the IAM user from a group, in the right pane, choose the IAM group and then choose the < button.

   To add the IAM user to all of the IAM groups, choose the >> button. Similarly, to remove the IAM user from all of the groups, choose the << button.

   To choose multiple groups, choose them in sequence. You do not need to hold down the Control key. To clear a group from your selection, simply choose it a second time.

3. When you have finished assigning the IAM user to IAM groups, choose *Save*.

3.13.4 Generate Credentials for an IAM User

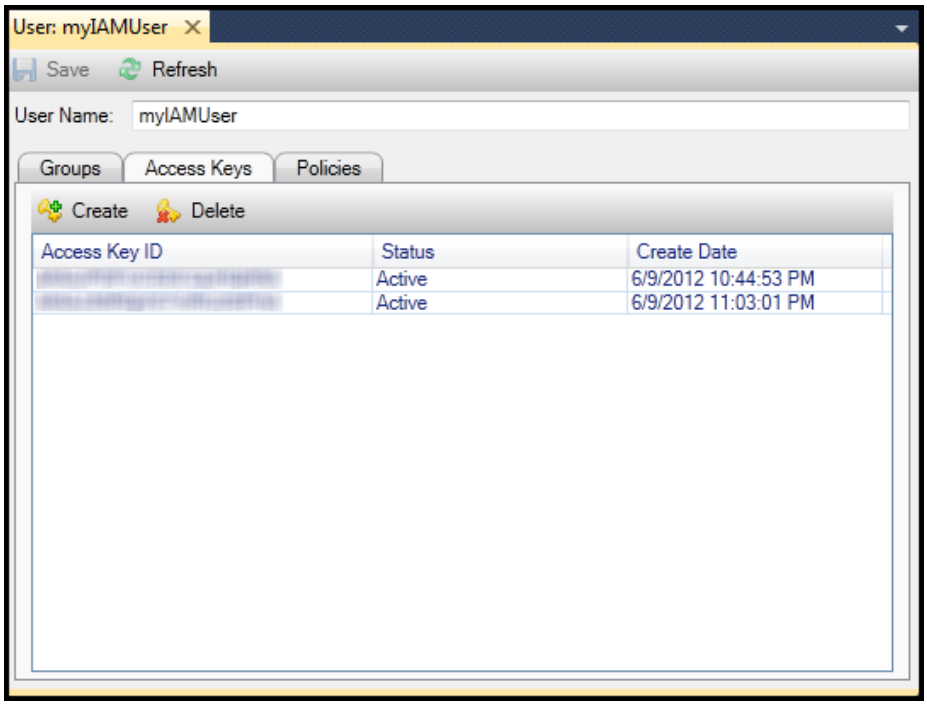
With Toolkit for Visual Studio, you can generate the access key ID and secret key used to make API calls to AWS. These keys can also be specified to access AWS services through the Toolkit. For more

information about how to specify credentials for use with the Toolkit, see *Specifying Credentials*. For more information about how to safely handle credentials, see Best Practices for Managing AWS Access Keys.

The Toolkit cannot be used to generate a password for an IAM user.

To generate credentials for an IAM user

1. In AWS Explorer, open the context (right-click) menu for an IAM user and choose *Edit*.



2. To generate credentials, on the *Access Keys* tab, choose *Create*.

   You can generate only two sets of credentials per IAM user. If you already have two sets of credentials and need to create an additional set, you must delete one of the existing sets.

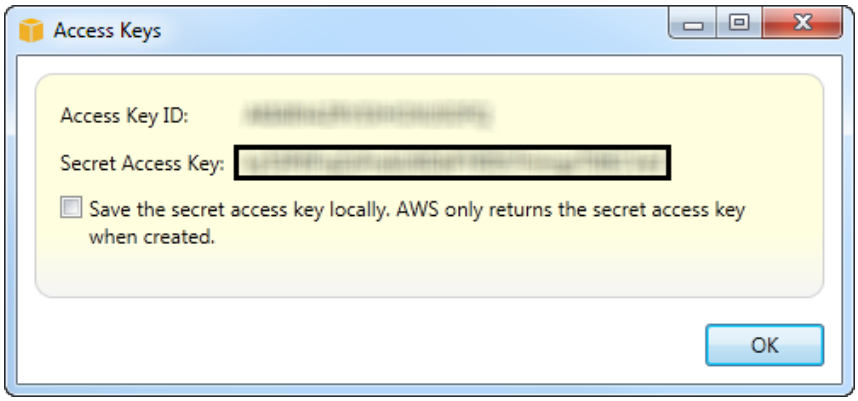


Fig. 3.13: reate credentials for IAM user

If you want the Toolkit to save an encrypted copy of your secret access key to your local drive, select

Save the secret access key locally. AWS only returns the secret access key when created. You can also copy the secret access key from the dialog box and save it in a secure location.

3. Choose *OK*.

After you generate the credentials, you can view them from the *Access Keys* tab. If you selected the option to have the Toolkit save the secret key locally, it will be displayed here.

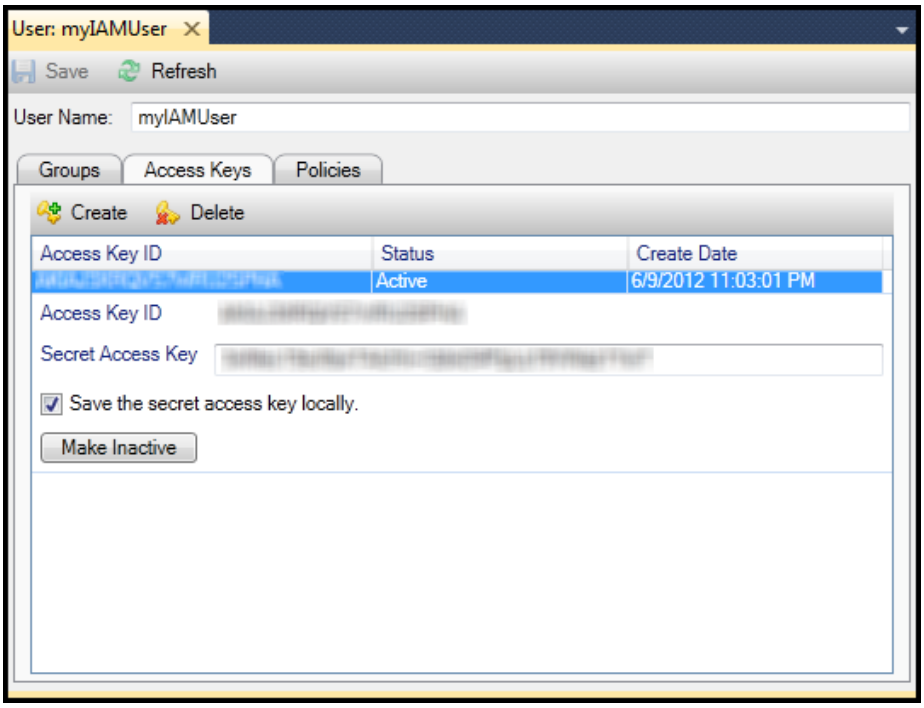


Fig. 3.14: Create credentials for IAM user

If you saved the secret key yourself and would also like the Toolkit to save it, in the *Secret Access Key* box, type the secret access key, and then select *Save the secret access key locally*.

To deactivate the credentials, choose *Make Inactive*. (You might do this if you suspect the credentials have been compromised. You can reactivate the credentials if you receive an assurance they are secure.)

3.13.5 Create an IAM Role

The Toolkit for Visual Studio supports the creation and configuration of IAM roles. Just as with users and groups, you can attach policies to IAM roles. You can then associate the IAM role with an Amazon EC2 instance. The association with the EC2 instance is handled through an *instance profile*, which is a logical container for the role. Applications that run on the EC2 instance are automatically granted the level of access specified by the policy associated with the IAM role. This is true even when the application hasn't specified other AWS credentials.

For example, you can create a role and attach a policy to that role that limits access to Amazon S3 only. After associating this role with an EC2 instance, you can then run an application on that instance and the application will have access to Amazon S3, but not to any other services or resources. The advantage of

this approach is that you don't need to be concerned with securely transferring and storing AWS credentials on the EC2 instance.

For more information about IAM roles, go to Working with IAM Roles in the IAM User Guide. For examples of programs accessing AWS using the IAM role associated with an Amazon EC2 instance, go to the AWS developer guides for Java, .NET, PHP, and Ruby.

To create an IAM role

1. In AWS Explorer, under *Identity and Access Management*, open the context (right-click) menu for *Roles* and then choose *Create Roles*.
2. In the *Create Role* dialog box, type a name for the IAM role and choose *OK*.

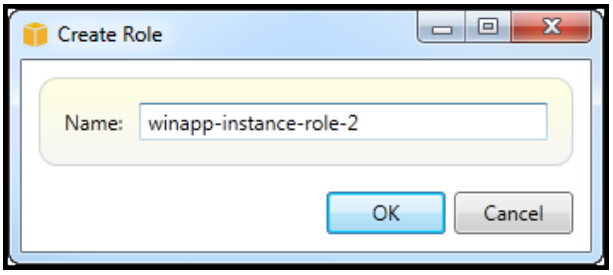


Fig. 3.15: Create IAM role

The new IAM role will appears under *Roles* in *Identity and Access Management*.

For information about how to create a policy and attach it to the role, see *Create an IAM Policy*.

3.13.6 Create an IAM Policy

Policies are fundamental to IAM. Policies can be associated with IAM *entities* such users, groups, or roles. Policies specify the level of access enabled for a user, group, or role.

To create an IAM policy

In AWS Explorer, expand the *AWS Identity and Access Management* node, then expand the node for the type of entity (*Groups*, *Roles*, or *Users*) to which you will attach the policy. For example, open a context menu for an IAM role and choose *Edit*.

A tab associated with the role will appear in the AWS Explorer. Choose the *Add Policy* link.

In the *New Policy Name* dialog box, type a name for the policy (for example, s3-access).

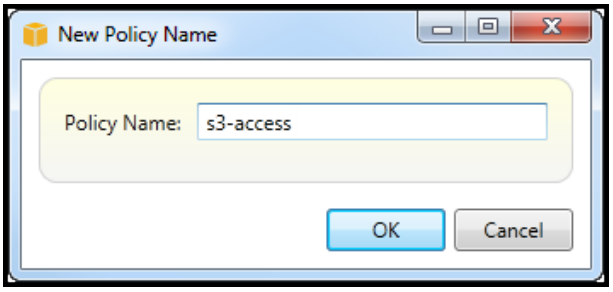


Fig. 3.16: New Policy Name dialog box

In the policy editor, add policy statements to specify the level of access to provide to the role (in this example, winapp-instance-role-2 associated with the policy. In this example, a policy provides full access to Amazon S3, but no access to any other resources.

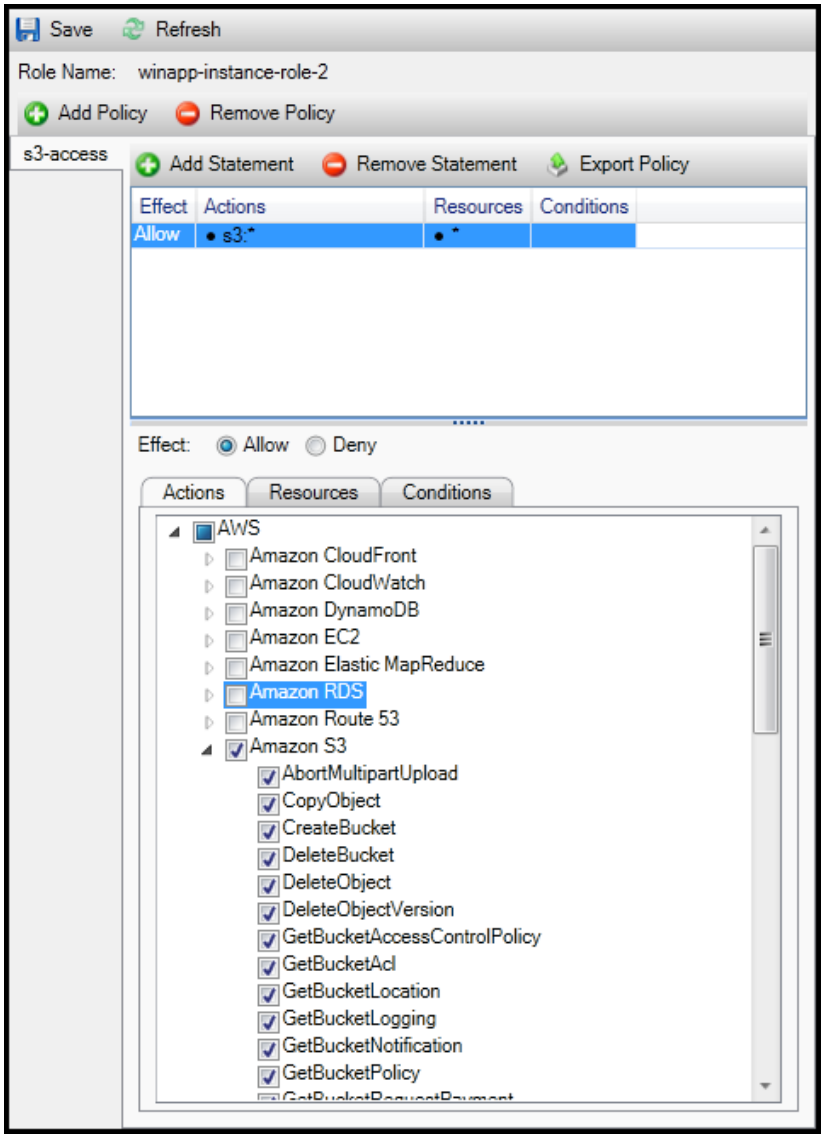


Fig. 3.17: Specify IAM policy

For more precise access control, you can expand the subnodes in the policy editor to allow or disallow actions associated with AWS services.

When you have edited the policy, choose the *Save* link.

3.14 Using AWS Lambda with the AWS Toolkit for Visual Studio

The AWS Toolkit for Visual Studio includes AWS Lambda .NET Core project templates for Visual Studio. You can use the templates to quickly develop and deploy .NET Core-based C# Lambda functions. .NET Core is cross-platform, supporting Windows, macOS, and Linux, and can be used to develop device, cloud, and embedded applications. You must have Visual Studio 2015 Update 3 installed to install .NET Core for Windows and the AWS Toolkit for Visual Studio.

Note: For more information about Microsoft .NET Core, see .NET Core.

For .NET Core prerequisites and installation instructions for the three platforms, see .NET Core Downloads.

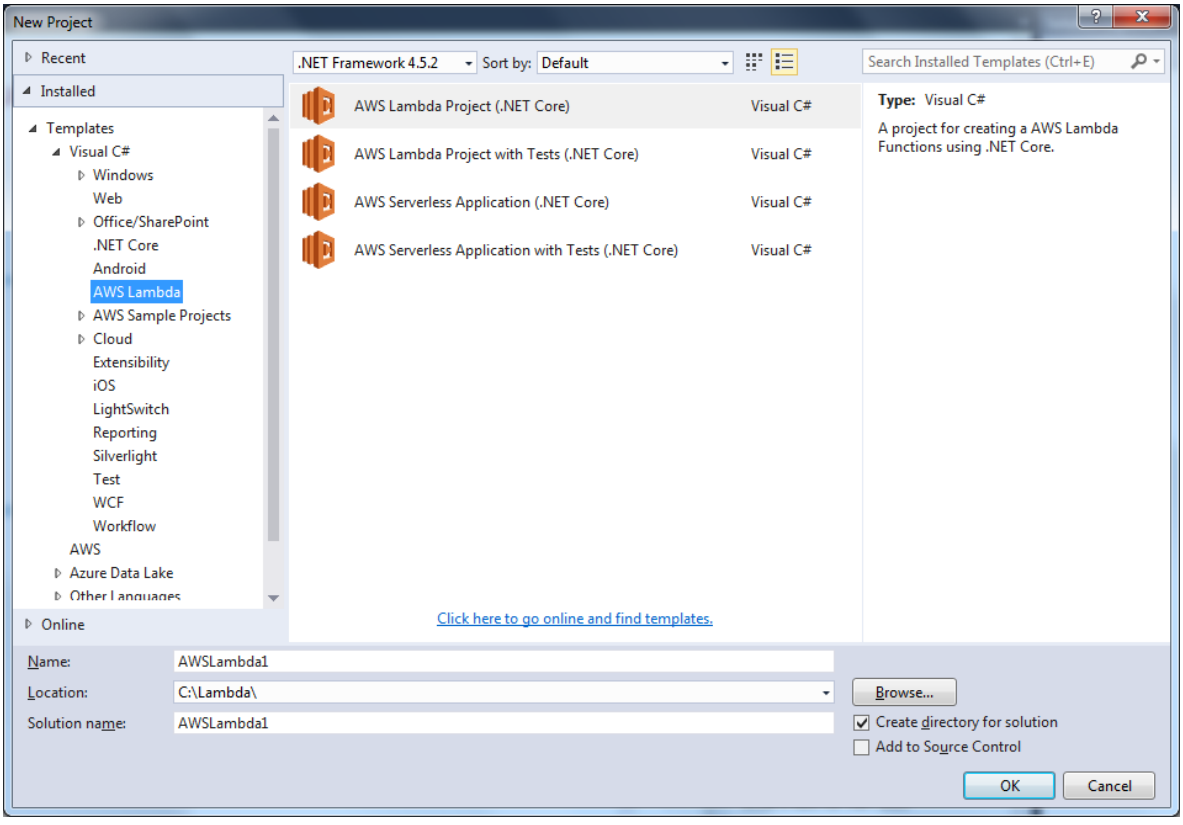
For more information about AWS Lambda functions, see What Is AWS Lambda?

3.14.1 Creating a Visual Studio .NET Core Lambda Project

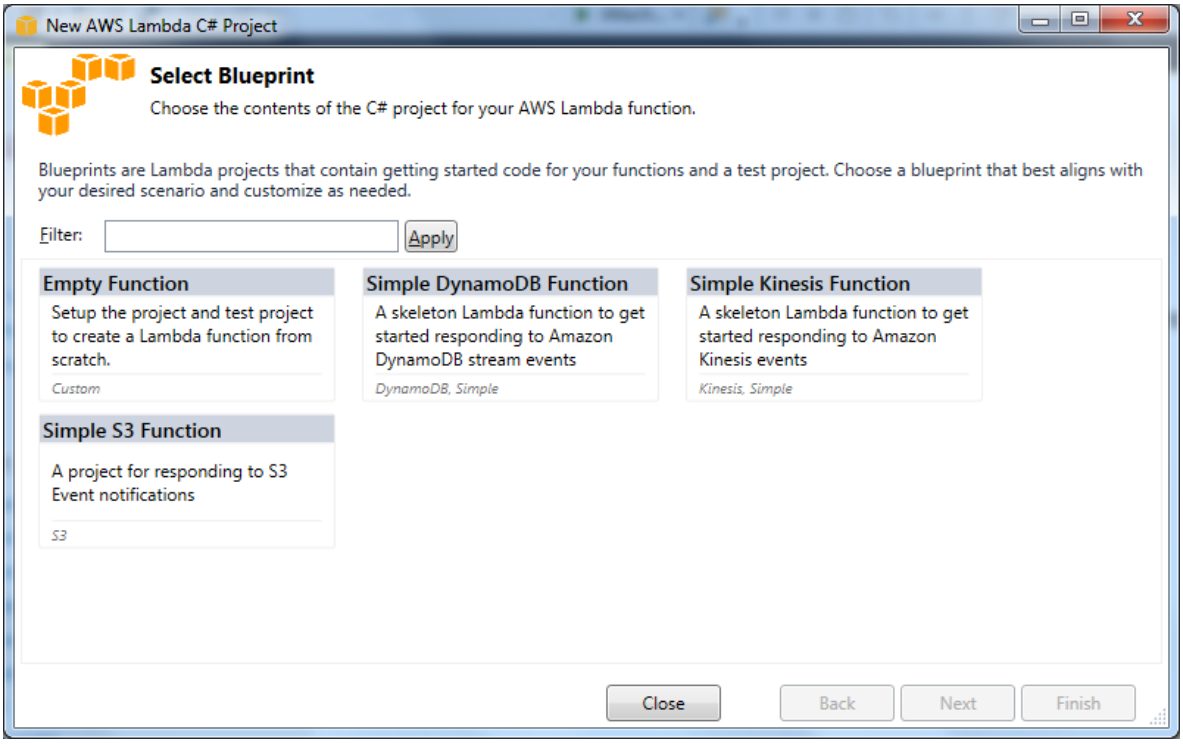
Open Visual Studio and create a new project.

1. In Visual Studio, on the *File* menu, choose *New*, *Project*.
2. In the *Installed* pane, choose Visual C# and the AWS Lambda Project template.

   There are two categories of projects, AWS Lambda projects for creating a project to develop and deploy an individual Lambda function, and AWS Serverless Applications for creating Lambda functions with a serverless AWS CloudFormation template. AWS Serverless Applications enable you to define more than just the function. For example, you can simultaneously create a database, add IAM roles, etc., with serverless deployment. AWS Serverless Applications also enable you to deploy multiple functions at one time.



3. After you select the project type, choose a blueprint. For the *AWS Lambda Project (.NET Core)*, you should see the *Select Blueprint* page showing several Lambda function templates.

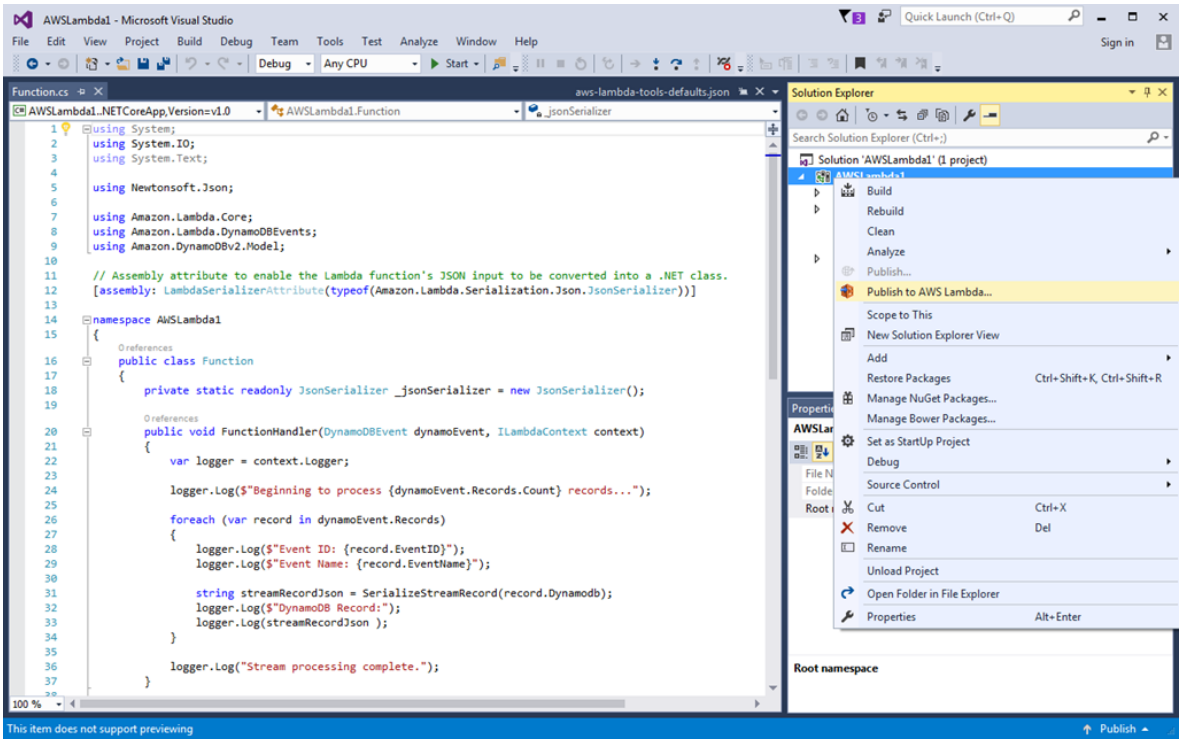


4. Choose the type of Lambda function you want to develop, and then choose *Finish* to create the Visual Studio project.
5. Review the project's structure and code. Your project is now ready to publish to Lambda.

Publishing to Lambda

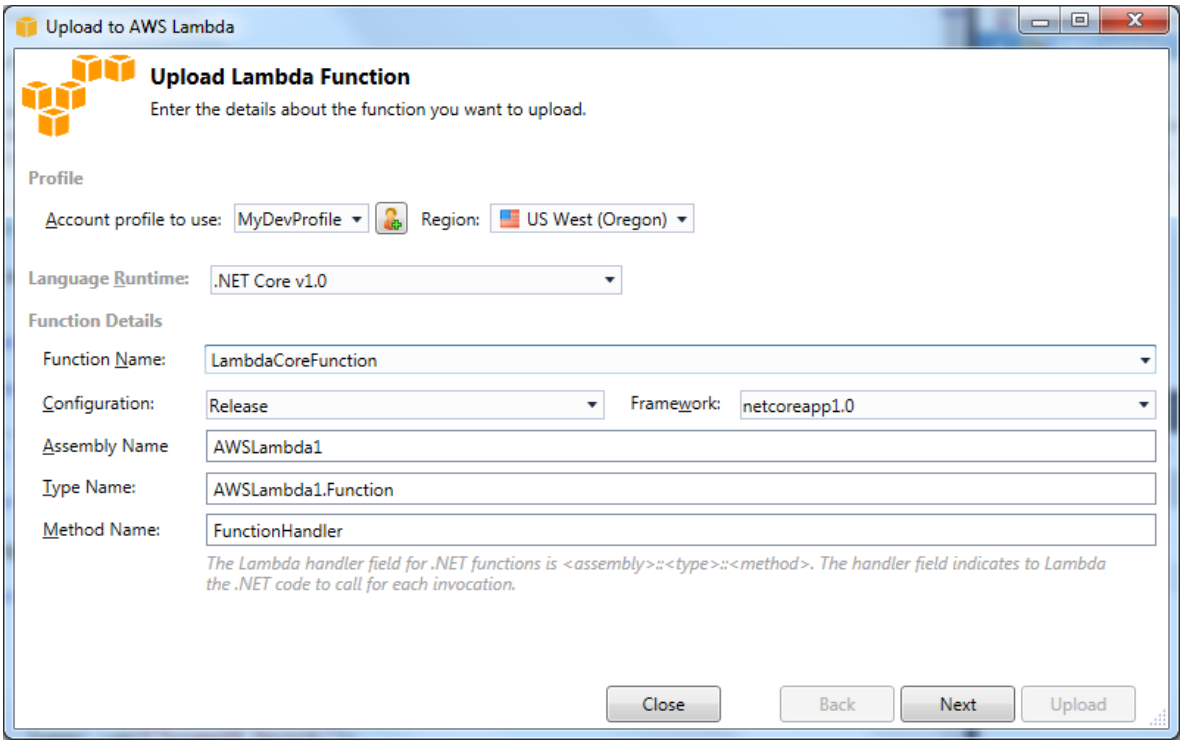
When your function is complete, you can publish it to Lambda.

1. In *Solution Explorer*, right-click the project and choose *Publish to AWS Lambda*.



2. On the *Upload Lambda Function* page, in *Function Name*, type a name for the function or select a previously published function to republish.

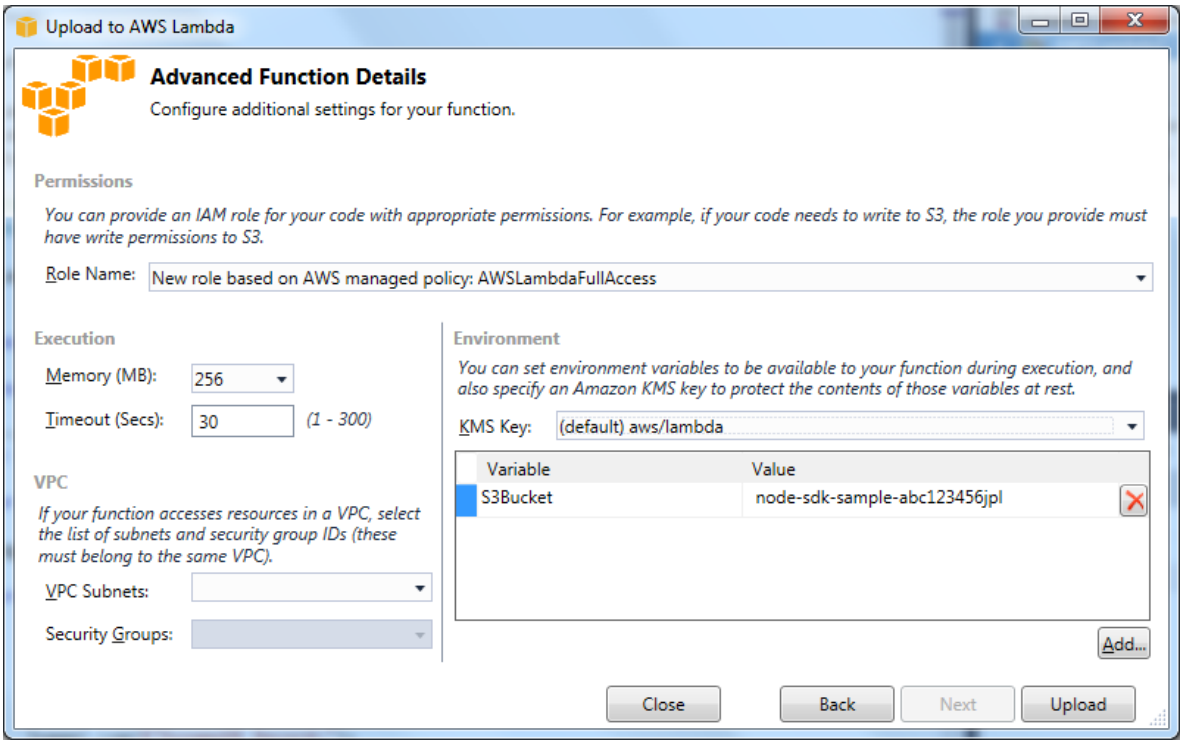
   Choose *Next*.



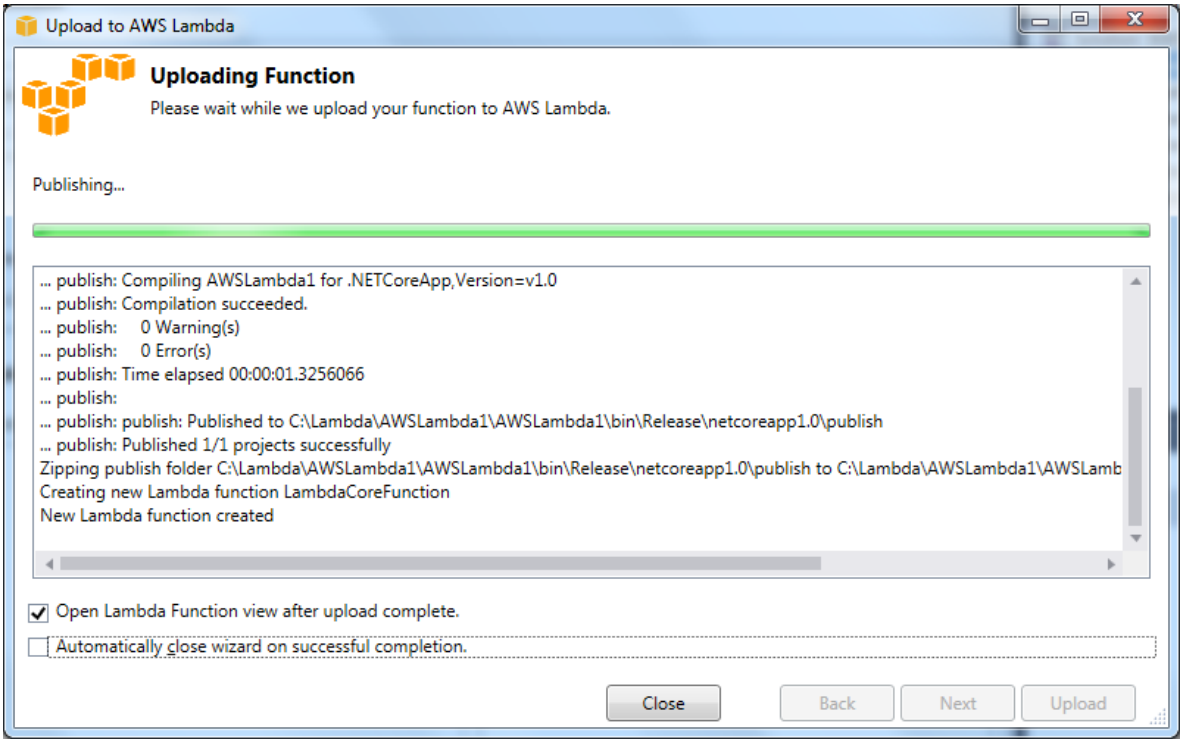
3. Set the fields you want in the *Advanced Function Details* page, as follows:

- **Required:** You must provide a *Role Name*. Select a role associated with your account. You can choose either an existing role or a new role based on an AWS managed policy or your own managed policy. The role will be used to provide credentials for any AWS service calls the code in the function makes.
- *Optional:* If your Lambda function accesses resources on an Amazon VPC, select the subnets and security groups.
- *Optional:* Set any environment variables that your Lambda function requires. The keys are automatically encrypted by the default service key (free) or you can specify an AWS KMS key (a charge). KMS is a managed service you can use to create and control the encryption keys used to encrypt your data. If you have an AWS KMS key, you can select it from the list.

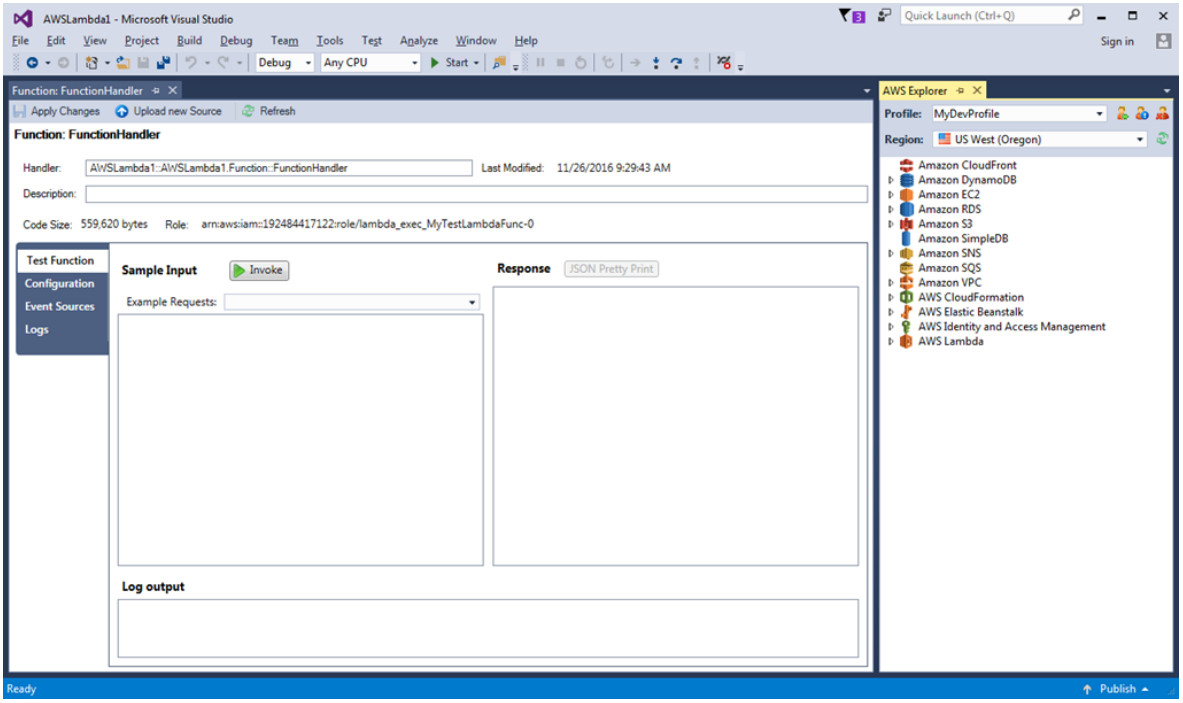
After you set the environment variables in the list that your function needs, choose *Upload*.



4. While the function is uploading, the *Uploading Function* page is shown. The page will automatically close upon completion. To keep the wizard open so you can view the report, clear *Automatically close wizard on successful completion* at the bottom of the form before the upload completes. Close the page when you are finished viewing the report.



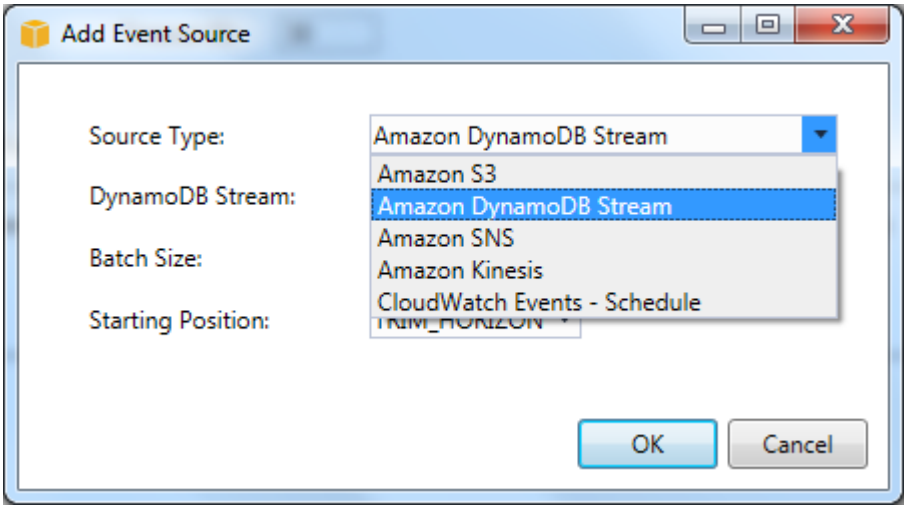
5. After the function is uploaded, the *Function view* page opens. Tabs on the left side of the page enable you to test the function, add event sources, and view the log. The configuration tab enables you to add VPC subnets and security groups, memory, timeout, and environment variables.



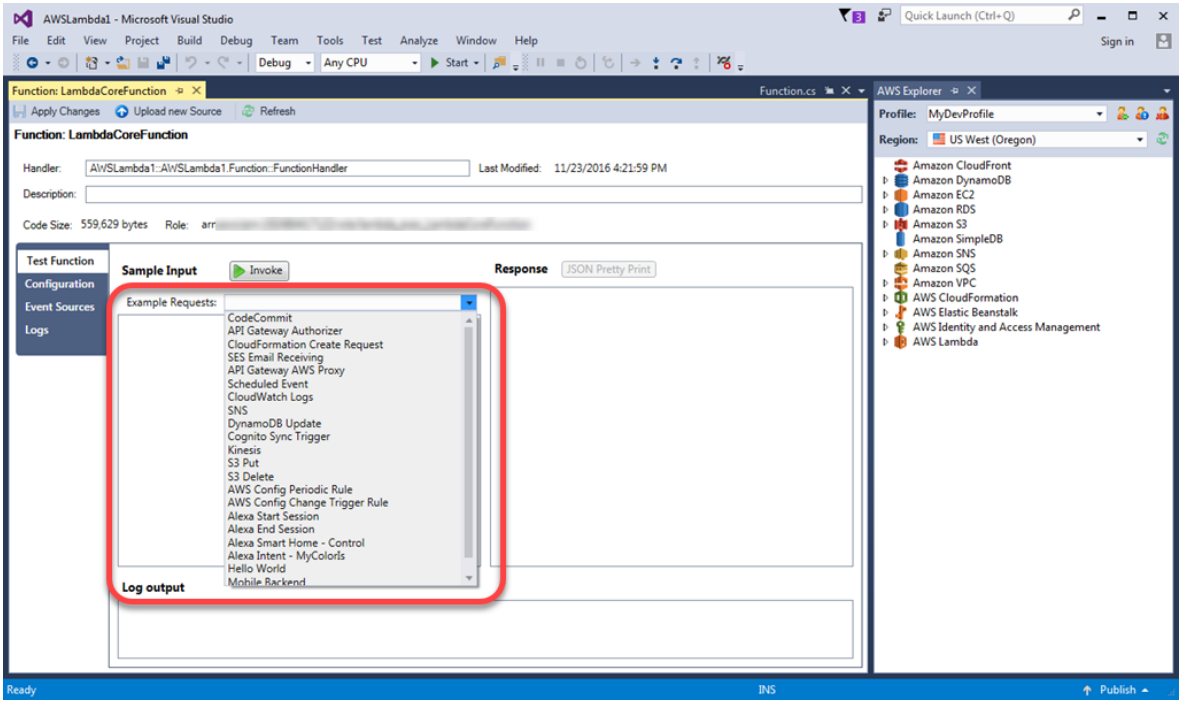
6. To add event sources that can be used to establish a connection between an AWS Resource (such an Amazon S3 bucket, Amazon SNS topic, or Amazon Kinesis Streams streams) and a Lambda function, choose *Event Sources*. On the *Add Event Sources* page, choose *Add* to add the event

sources.

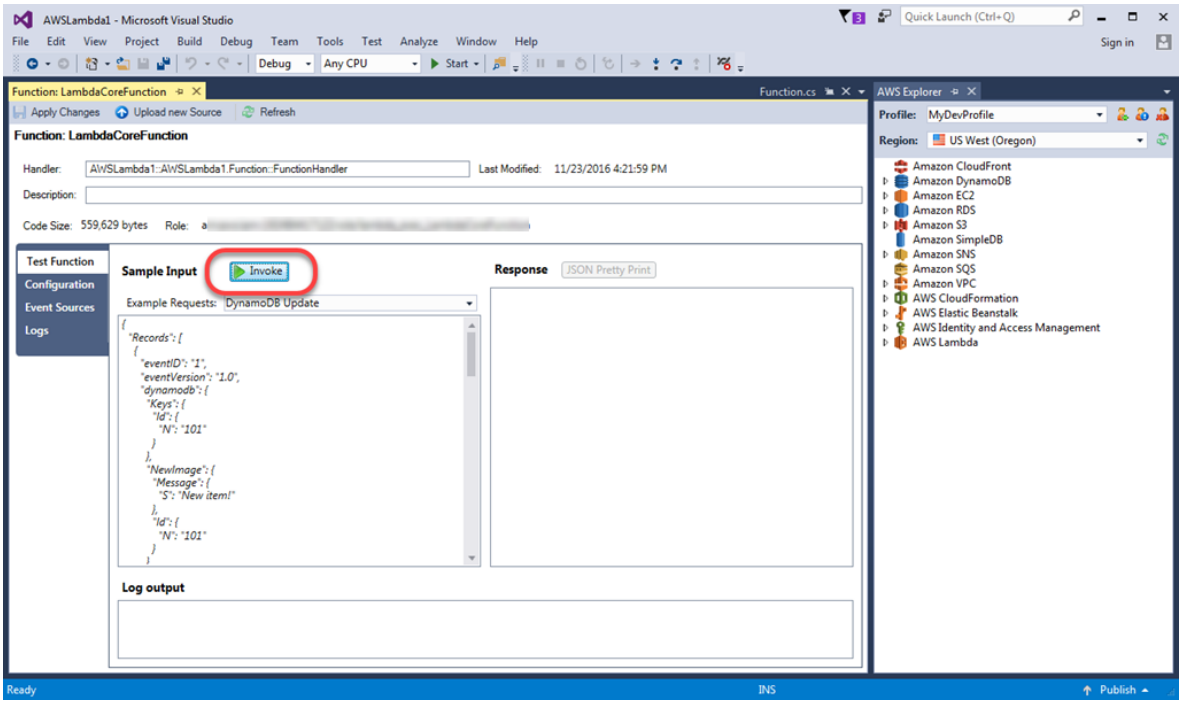
On the *Add Event Sources* page, from *Source Type*, choose the appropriate event source.



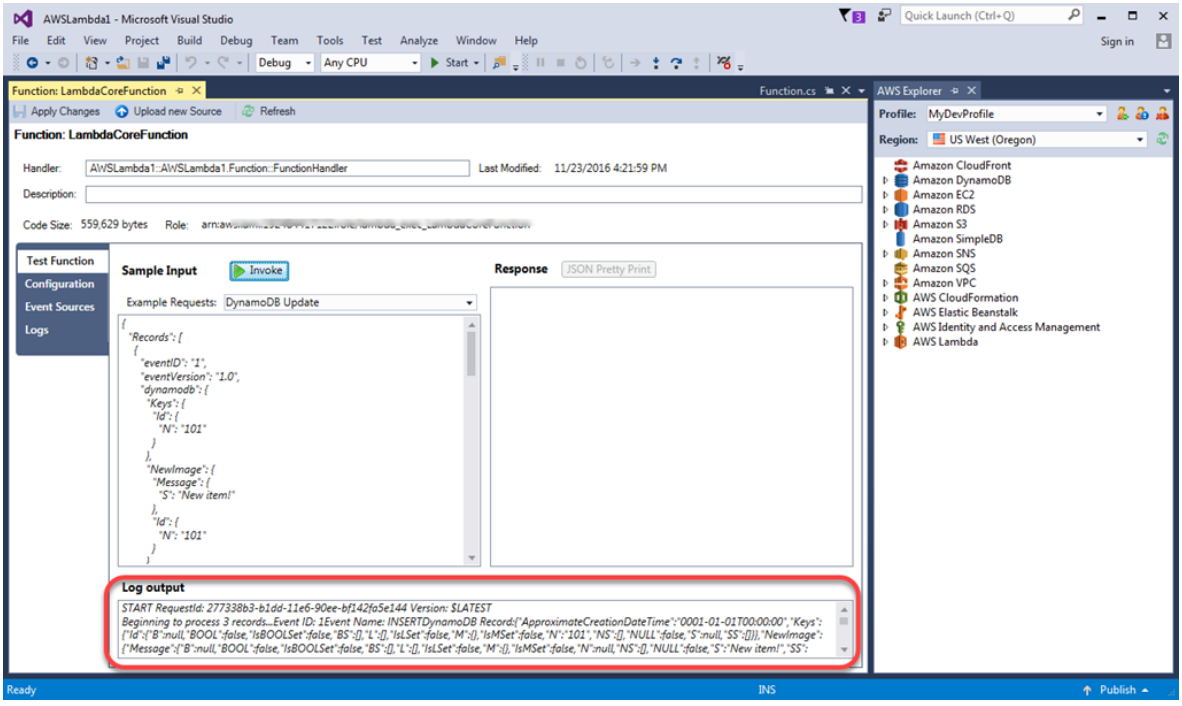
7. To test the function, in *Example Requests*, choose an example request.



8. To run the test, choose *Invoke*.



9. View the output from the test in *Log output*.



After your Lambda function is published, it is ready to use. For example use cases, see Examples of How to Use AWS Lambda.

Lambda automatically monitors Lambda functions for you, reporting metrics through Amazon CloudWatch. To monitor and troubleshoot your Lambda function, see Troubleshooting and Monitoring AWS Lambda Functions with Amazon CloudWatch.

3.15 Deploying an AWS Lambda Project with the .NET Core CLI

The AWS Toolkit for Visual Studio includes AWS Lambda .NET Core project templates for Visual Studio. You must have Visual Studio 2015 Update 3 installed before you can install .NET Core for Windows and the Toolkit for Visual Studio. You can deploy Lambda functions built in Visual Studio using the .NET Core command line interface (CLI).

Note: For information about creating Lambda functions in Visual Studio, see *Using AWS Lambda with the AWS Toolkit for Visual Studio*.

For more information about Microsoft .NET Core, see .NET Core.

For more information about Lambda functions, see What Is AWS Lambda?

3.15.1 Listing the Lambda Commands Available through the CLI

There are a variety of Lambda commands available through the .NET Core CLI.

1. Open a command prompt and navigate to the folder containing a Visual Studio .NET Core Lambda project.
2. Type `dotnet lambda --help`.

```
C:\Lambda\AWSLambda1\AWSLambda1>dotnet lambda --help
    AWS Lambda Tools for .NET Core functions
    Project Home: https://github.com/aws/aws-lambda-dotnet
    .
    Commands to deploy and manage |LAM| functions:
    .
            deploy-function         Deploy the project to |LAM|
            invoke-function         Invoke the function in |LAM|␣
↪with an optional input
            list-functions          List all of your |LAM| functions
            delete-function         Delete a |LAM| function
            get-function-config     Get the current runtime␣
↪configuration for a |LAM| function
            update-function-config  Update the runtime configuration␣
↪for a |LAM| function
    .
    Commands to deploy and manage AWS Serverless applications using␣
↪|CFNlong|:
    .
            deploy-serverless       Deploy an AWS Serverless␣
↪application
            list-serverless         List all of your AWS Serverless␣
↪applications
            delete-serverless       Delete an AWS Serverless␣
↪application
    .
    Other Commands:
```

```
    .
            package                 Package a |LAM| project into a .
↪zip file ready for deployment
    .
    To get help on individual commands execute:

            dotnet lambda help <command>
```

3.15.2 Publishing a .NET Core Lambda Project from the .NET Core CLI

The following instructions assume you've created an AWS Lambda .NET Core function in Visual Studio.

1. Open a command prompt and navigate to the folder containing your Visual Studio .NET Core Lambda project.
2. Type `dotnet lambda deploy-function`.

#. When prompted, type the name of the function to deploy. It can be a new name or the name of an existing function.

1. When prompted, enter the AWS Region (the region to which your Lambda function will be deployed).
2. When prompted, select or create the IAM role that Lambda will assume when executing the function.
3. On successful completion, the message **New Lambda function created** is displayed.

```
C:\Lambda\AWSLambda1\AWSLambda1>dotnet lambda deploy-function
Executing publish command
... invoking 'dotnet publish', working folder 'C:
↪\Lambda\AWSLambda1\AWSLambda1\bin\Release\netcoreapp1.0\publish'
... publish: Publishing AWSLambda1 for .NETCoreApp,Version=v1.0
... publish: Project AWSLambda1 (.NETCoreApp,Version=v1.0) will be␣
↪compiled because expected outputs are missing
... publish: Compiling AWSLambda1 for .NETCoreApp,Version=v1.0
... publish: Compilation succeeded.
... publish:     0 Warning(s)
... publish:     0 Error(s)
... publish: Time elapsed 00:00:01.2479713
... publish:
... publish: publish: Published to C:
↪\Lambda\AWSLambda1\AWSLambda1\bin\Release\netcoreapp1.0\publish
... publish: Published 1/1 projects successfully
Zipping publish folder C:
↪\Lambda\AWSLambda1\AWSLambda1\bin\Release\netcoreapp1.0\publish␣
↪to C:\Lambda\AWSLambda1\AWSLamb
da1\bin\Release\netcoreapp1.0\AWSLambda1.zip
Enter Function Name: (AWS Lambda function name)
DotNetCoreLambdaTest
Enter AWS Region: (The region to connect to AWS services)
us-west-2
Creating new Lambda function
Select IAM Role that Lambda will assume when executing function:
```

```
    1) lambda_exec_LambdaCoreFunction
    2) *** Create new IAM Role ***
1
New Lambda function created
```

If you deploy an existing function, the deploy function asks only for the AWS Region.

```
C:\Lambda\AWSLambda1\AWSLambda1>dotnet lambda deploy-function
Executing publish command
Deleted previous publish folder
... invoking 'dotnet publish', working folder 'C:
↪\Lambda\AWSLambda1\AWSLambda1\bin\Release\netcoreapp1.0\publish'
... publish: Publishing AWSLambda1 for .NETCoreApp,Version=v1.0
... publish: Project AWSLambda1 (.NETCoreApp,Version=v1.0) was␣
↪previously compiled. Skipping compilation.
... publish: publish: Published to C:
↪\Lambda\AWSLambda1\AWSLambda1\bin\Release\netcoreapp1.0\publish
... publish: Published 1/1 projects successfully
Zipping publish folder C:
↪\Lambda\AWSLambda1\AWSLambda1\bin\Release\netcoreapp1.0\publish␣
↪to C:\Lambda\AWSLambda1\AWSLamb
da1\bin\Release\netcoreapp1.0\AWSLambda1.zip
Enter Function Name: (AWS Lambda function name)
DotNetCoreLambdaTest
Enter AWS Region: (The region to connect to AWS services)
us-west-2
Updating code for existing function
```

After your Lambda function is deployed, it is ready to use. See Examples of How to Use AWS Lambda.

Lambda automatically monitors Lambda functions for you, reporting metrics through Amazon CloudWatch. To monitor and troubleshoot your Lambda function, see Troubleshooting and Monitoring AWS Lambda Functions with Amazon CloudWatch.

CHAPTER 4

Document History

The following table describes the important changes since the last release of the Toolkit for Visual Studio User Guide.

Last documentation update: Dec 01, 2016

Change	Description	Release Date
Added ASP.NET Core Details	AWS Elastic Beanstalk deployment wizard now supports ASP.NET Core applications. See *Deploying an ASP.NET Core Application to Elastic Beanstalk* for details.	July 25, 2016
Revised deployment wizards	This release introduces a new *Publish to Elastic Beanstalk* wizard. For more information, see *Deploying to Elastic Beanstalk*. With the introduction of this new wizard, the *Publish to Amazon Web Services* wizard has been moved to legacy status. For more information, see tkv-deploy-beanstalk-legacy and *Deploying to CloudFormation (Legacy)*.	December 17, 2014
Support for Amazon VPC	This release adds support for Amazon Virtual Private Cloud.	April 4, 2013
New release	This is version 3.0 of the Toolkit for Visual Studio User Guide.	June 8, 2012