AWS Elastic Beanstalk

Developer Guide

API Version 2010-12-01

Amazon Web Services

AWS Elastic Beanstalk: Developer Guide

What Is AWS Elastic Beanstalk and Why Do I Need It?

Amazon Web Services (AWS) comprises about 25 services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them.

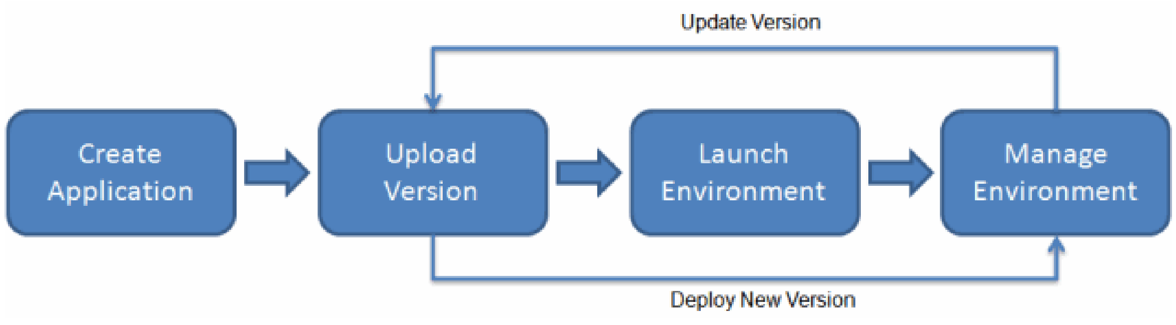
With AWS Elastic Beanstalk, you can quickly deploy and manage applications in the AWS cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and AWS Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring. AWS Elastic Beanstalk uses highly reliable and scalable services that are available in the AWS Free Usage Tier such as:

- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Simple Storage Service (Amazon S3)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon CloudWatch
- Elastic Load Balancing
- Auto Scaling

To learn more about the AWS Free Usage Tier, and how to deploy a sample web application in it using AWS Elastic Beanstalk, go to Deploy a Sample Web Application in the Free Usage Tier.

You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the AWS Elastic Beanstalk web interface.

To use AWS Elastic Beanstalk, you create an application, upload an application version (for example, a Java .war file) to AWS Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions. The following diagram illustrates the workflow of AWS Elastic Beanstalk.



After you create and deploy your application, information about the application—including metrics, events, and environment status—is available through the AWS Management Console, APIs, and CLI. For step-by-step instructions on how to create, deploy, and manage your application using the AWS Management Console, go to Getting Started Using AWS Elastic Beanstalk (p. 5). To learn more about an AWS Elastic Beanstalk application and its components, see AWS Elastic Beanstalk Components (p. 17).

AWS Elastic Beanstalk provides developers and systems administrators an easy, fast way to deploy and manage your application without having to worry about AWS infrastructure. If you already know the AWS resources you want to use and how they work, you might prefer AWS CloudFormation to create your AWS resources by creating a template. You can then use this template to launch new AWS resources in the exact same way without having to recustomize your AWS resources. Once your resources are deployed, you can modify and update the AWS resources in a controlled and predictable way, providing the same sort of version control over your AWS infrastructure that you exercise over your software. For more information about AWS CloudFormation, go to AWS CloudFormation Getting Started Guide.

Supported Platforms

AWS Elastic Beanstalk supports applications developed in Java, PHP, .NET, and Python as well as different container types for each language. AWS Elastic Beanstalk provisions the resources needed to run your application including one or more Amazon EC2 instances. The software stack running on the Amazon EC2 instances is dependent on the container type. A container type defines the infrastructure topology and software stack to be used for that environment. For example, the AWS Elastic Beanstalk for Apache Tomcat 7 container uses the Amazon Linux operating system, Apache web server, and Apache Tomcat software.

Java

You can get started with Java using the AWS Toolkit for Eclipse. The toolkit is a downloadable package that includes the AWS libraries, project templates, code samples, and documentation. The AWS SDK for Java supports developing applications using either Java 5 or Java 6. AWS Elastic Beanstalk supports the following container types:

- 32-bit Amazon Linux running Tomcat 6
- 64-bit Amazon Linux running Tomcat 6
- 32-bit Amazon Linux running Tomcat 7
- 64-bit Amazon Linux running Tomcat 7

PHP

You can get started in minutes using the AWS Management Console or the eb command line interface. You can deploy applications by simply zipping them up and uploading them through the console. To learn how to upload an application using the AWS Management Console, see Creating New Applications (p. 124). To learn how to get started deploying a PHP application to AWS Elastic Beanstalk using eb and Git, see Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git (p. 92). AWS Elastic Beanstalk supports the following container types:

- 32-bit Amazon Linux running PHP 5.3
- 64-bit Amazon Linux running PHP 5.3

Windows and .NET

You can get started in minutes using the AWS Toolkit for Visual Studio. The toolkit includes the AWS libraries, project templates, code samples, and documentation. The SDK for .NET supports the development of applications using .NET Framework 2.0 or later. AWS Elastic Beanstalk supports 64-bit Windows Server 2008 R2 running IIS 7.5.

Python

AWS Elastic Beanstalk supports Python applications running on Apache and WSGI. This includes support for many popular frameworks such as Django and Flask. You can get started in minutes using the AWS Management Console or the eb command line interface. You can deploy applications by simply zipping them up and uploading them through the console. To learn how to upload an application using the AWS Management Console, see Creating New Applications (p. 124). To learn how to get started deploying a Python application to AWS Elastic Beanstalk using eb and Git, see Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git (p. 103). AWS Elastic Beanstalk supports the following container types:

- 32-bit Amazon Linux running Python
- 64-bit Amazon Linux running Python

AWS Elastic Beanstalk supports Python 2.6 running on Amazon Linux.

Storage

AWS Elastic Beanstalk does not restrict your choice of persistent storage and database service options. For more information on AWS storage options, go to Storage Options in the AWS Cloud.

Pricing

There is no additional charge for AWS Elastic Beanstalk; you pay only for the underlying AWS resources that your application consumes. For details about pricing, see the AWS Elastic Beanstalk service detail page.

Community

Customers have built a wide variety of products, services, and applications on top of AWS. Whether you are searching for ideas about what to build, looking for examples, or just want to explore, you can find many solutions at the AWS Customer App Catalog. You can browse by audience, services, and technology. We also invite you to share applications you build with the community. Developer resources produced by the AWS community are at http://aws.amazon.com/resources/.

Where to Go Next

This guide contains conceptual information about the AWS Elastic Beanstalk web service, as well as information about how to use the service to create and deploy new web applications. Separate sections describe how to program with the command line interface (CLI) and how to integrate AWS Elastic Beanstalk with other Amazon Web Services.

We recommend that you first read Getting Started Using AWS Elastic Beanstalk (p. 5) to learn how to start using AWS Elastic Beanstalk. Getting Started steps you through creating, viewing, and updating your AWS Elastic Beanstalk application, as well as editing and terminating your AWS Elastic Beanstalk environment. Getting Started also describes different ways you can access AWS Elastic Beanstalk.

Getting Started Using AWS Elastic Beanstalk

Topics

Getting started with AWS Elastic Beanstalk is simple, and the AWS Management Console makes it easy for you to create, edit, and manage your Java, PHP, .NET, and Python applications in a matter of minutes. The following walkthrough steps you through how to use the console to get started. You can also access AWS Elastic Beanstalk using the AWS Toolkit for Eclipse, AWS Toolkit for Visual Studio, AWS SDKs, APIs, and CLIs. The remainder of this topic provides information about each of these and where to go next.

Walkthrough

The following tasks will help you get started with AWS Elastic Beanstalk to create, view, deploy, update your application as well as edit and terminate your environment. You'll use the AWS Management Console, a point-and-click web-based interface, to complete these tasks.

Step 1: Sign up for the Service

Signing up for AWS Elastic Beanstalk also signs you up for other AWS services you will need, such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), and Amazon Simple Notification Service (Amazon SNS).

To sign up for an AWS Elastic Beanstalk account

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Follow the on-screen instructions.

Note

If you have never registered for Amazon EC2, part of the sign-up procedure for AWS Elastic Beanstalk will include receiving an automated telephone call and entering a PIN using the telephone keypad.

Step 2: Create an Application

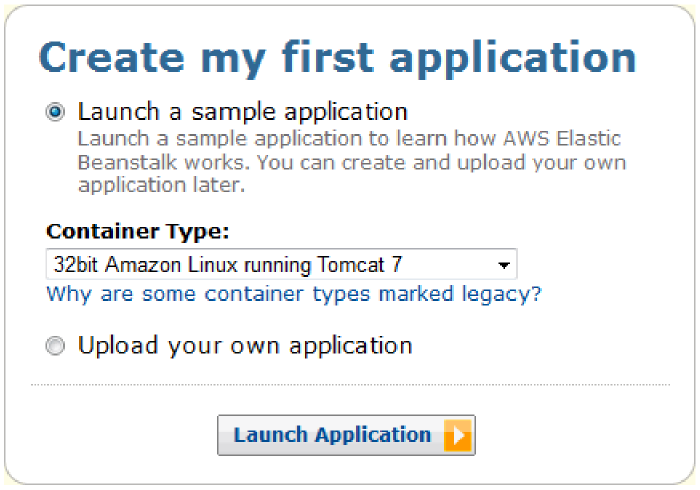
Next, you will create and deploy a sample application. For this step, you use a sample application that is already prepared.

Important

AWS Elastic Beanstalk is free, but the AWS resources that AWS Elastic Beanstalk provides will be live (and not running in a sandbox). You will incur the standard usage fees for these resources until you terminate them in the last task in this tutorial. The total charges will be minimal (typically less than a dollar). For information on how you might minimize any charges, go to http://aws.amazon.com/free/.

To create a sample application

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Click **Launch a sample application** and select a container option from the **Container Type** list. Click **Launch Application** to start the application creation process.



Create my first application

[✓ Launch a sample application]
Launch a sample application to learn how AWS Elastic Beanstalk works. You can create and upload your own application later.

Container Type: 32bit Amazon Linux running Tomcat 7

Why are some container types marked legacy?

Upload your own application

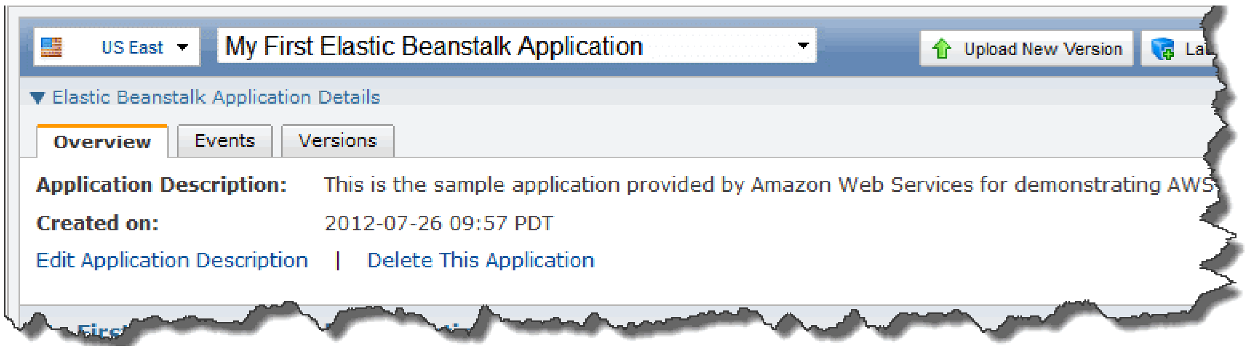
To begin the process of creating the necessary components to run the sample application on AWS resources, AWS Elastic Beanstalk does the following:

- Creates an AWS Elastic Beanstalk application named "My First Elastic Beanstalk Application."
- Creates a new application version labeled "Initial Version" that refers to a default sample application file.
- Launches an environment named "Default-Environment" that provisions the AWS resources to host the application.
- Deploys the "Initial Version" application into the newly created "Default-Environment."

This process may take several minutes to complete.

Step 3: View Application

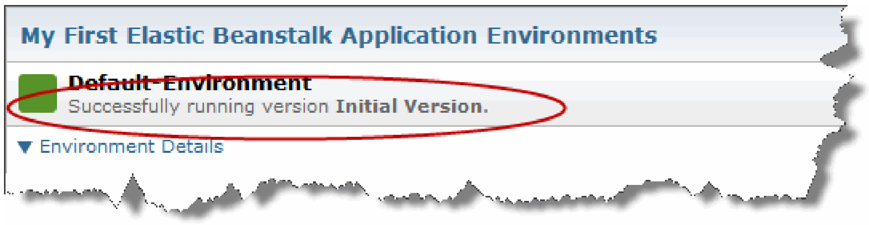
After you create your application, the details and environment for the application appear in the AWS Management Console. The **Application Details** pane on the top of the console provides basic overview information about your application, including events associated with the application and all versions of the application.


US East
My First Elastic Beanstalk Application
Upload New Version
Elastic Beanstalk Application Details
Overview
Events
Versions
Edit Application Description
Delete This Application

Application Description: This is the sample application provided by Amazon Web Services for demonstrating AWS

Created on: 2012-07-26 09:57 PDT

The **Environment** pane below the **Application Details** pane displays information about the Amazon EC2 instances that host your application, along with the AWS resources that AWS Elastic Beanstalk provisions when it launches your environment. While AWS Elastic Beanstalk creates your AWS resources and launches your application, the environment will be in a `Launching` state. Status messages about launch events are displayed on the environment's information bar.



Successfully running version **Initial Version.**

To see the published version of your application

1. Click the **Environment Details** link in the **Environments** pane for your application.
   The details appear for your application's environment.
2. Click the link the **URL** field in the **Overview** tab.
   The application page opens in a new tab.

Step 4: Deploy New Version

You can update your deployed application, even while it is part of a running environment. In this section, you upload a new version of the sample application. For a Java application, you can also use the AWS Toolkit for Eclipse to update your deployed application; for instructions, see Edit the Application and Redeploy (p. 29). For a PHP application, it is easy to update your application using a Git deployment via eb; for instructions, see Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git (p. 92). For a .NET application, you can use the AWS Toolkit for Visual Studio to update your deployed application; for instructions, see Edit the Application and Redeploy (p. 67).

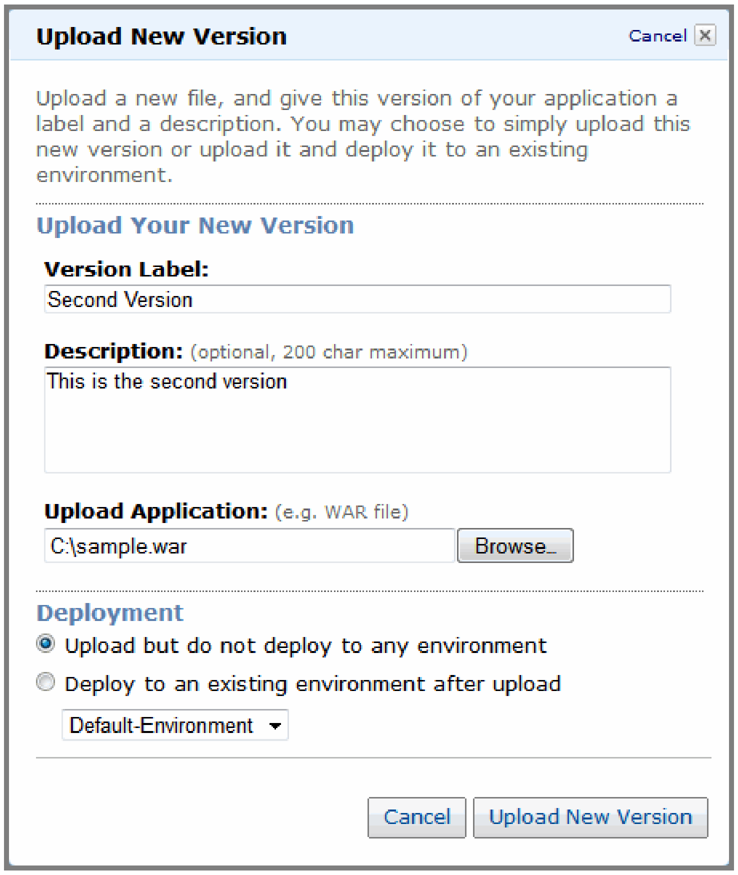
The application version you are running now is labeled **Initial Version**.

To update your application version

1. Do one of the following:

   - For Java, go to https://elasticbeanstalk-us-east-1.s3.amazonaws.com/resources/elasticbeanstalk-sampleapp2.war and save the file as `sample.war`.
   - For PHP, go to https://elasticbeanstalk-samples-us-east-1.s3.amazonaws.com/php-secondsample.zip and save the file as `php-sample.zip`.
   - For .NET, go to https://s3.amazonaws.com/elasticbeanstalk-samples-us-east-1/SecondSampleApp.zip and save the file as `net-sample.zip`.
   - For Python, go to https://elasticbeanstalk-samples-us-east-1.s3.amazonaws.com/python-secondsample.zip and save the file as `python-sample.zip`.

2. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
3. Click the **Versions** tab.
4. Click **Upload New Version**.


Upload New Version
Cancel
Upload a new file, and give this version of your application a label and a description. You may choose to simply upload this new version or upload it and deploy it to an existing environment.
Upload Your New Version
Version Label:
Description: (optional, 200 char maximum)
Upload Application: (e.g. WAR file)
Browse...
Deployment
Cancel
Upload New Version

Version Label: Second Version
Description: This is the second version
Upload Application: C:\sample.war
Deployment: [✓ Upload but do not deploy to any environment] Deploy to an existing environment after upload
Default-Environment

5. Do the following in the **Upload New Version** dialog box:

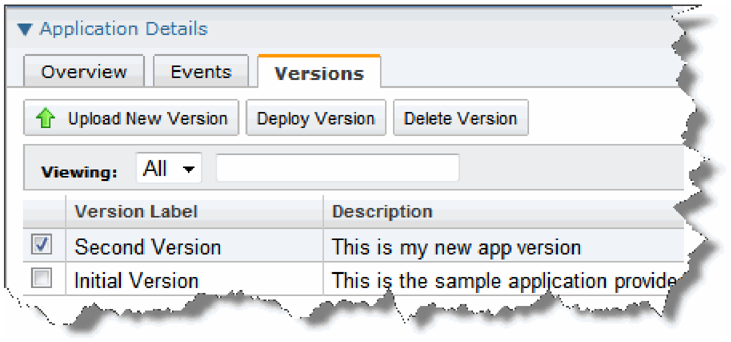
   - Enter a label for this version in the **Version Label** field. For this example, we use **`Second Version`**.

- Enter a brief description for this version in the **Description** field.
- Use the **Upload Application** field to browse for the updated file.
- Select **Upload but do not deploy to any environment**.
- Click **Upload New Version**.

The new version is now available to deploy to a running environment.

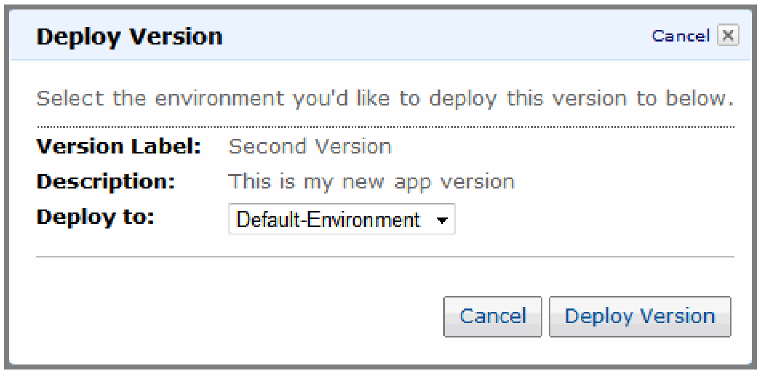
To deploy the new version of your application

1. Click the **Versions** tab in the AWS Elastic Beanstalk console.
2. Select the check box next to the **Second Version** version label.



	Version Label	Description
✓	Second Version	This is my new app version
	Initial Version	This is the sample application provide

3. Click the **Deploy Version** button.



Select the environment you'd like to deploy this version to below.

Version Label: Second Version
Description: This is my new app version
Deploy to: Default-Environment

4. Accept the default environment in the **Deploy to** field and click **Deploy Version**.
5. A dialog box confirms the update to your environment. Click the **Close** button.

AWS Elastic Beanstalk now deploys your file to your Amazon EC2 instances. You will see the environment turn gray and the status changed to "Updating." When the deployment is complete, there's an application health check. The environment returns to green when the application responds to the health check.

To view the new version of your application

1. Select your environment in the **Environments** pane.
2. Click the **Events** tab to view current information on the deployment of the new version.
3. Click the **View Running Version** button in the **Overview** tab to see the new version of your application.

Step 5: Change Configuration

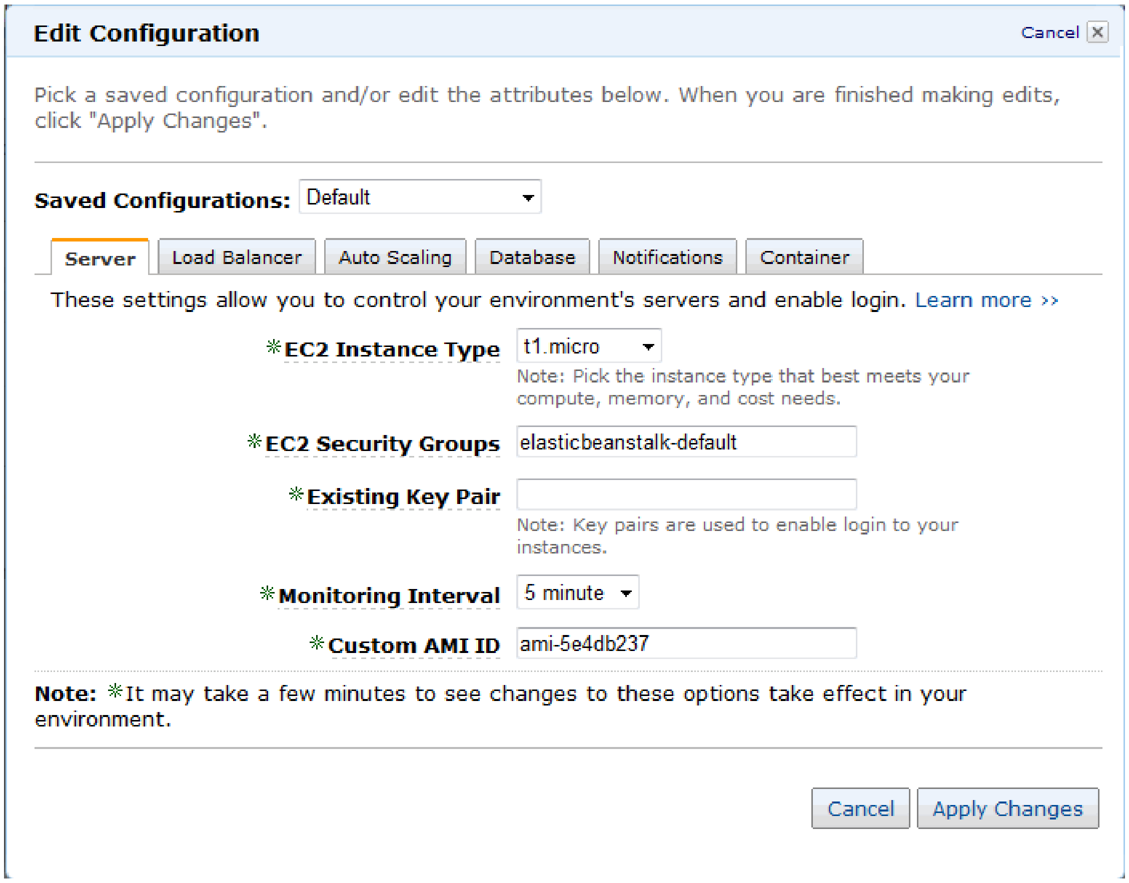
You can customize your environment to better suit your application. For example, if you have a compute-intensive application, you can change the type of Amazon EC2 instance that is running your application.

Some configuration changes are simple and happen quickly. Some changes require AWS Elastic Beanstalk to delete and recreate AWS resources, which can take several minutes. AWS Elastic Beanstalk will warn you about possible application downtime when changing configuration settings.

In this task, you change the minimum instance settings for your Auto Scaling group from one to two and then verify that the change occurred. After the new instance gets created, it will become associated with your load balancer.

To change your environment configuration

1. Click the **Actions** drop-down menu on the right of the **Environment** pane, and select **Edit/Load Configuration**.



Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

These settings allow you to control your environment's servers and enable login.

*EC2 Instance Type: t1.micro
Note: Pick the instance type that best meets your compute, memory, and cost needs.

*EC2 Security Groups: elasticbeanstalk-default

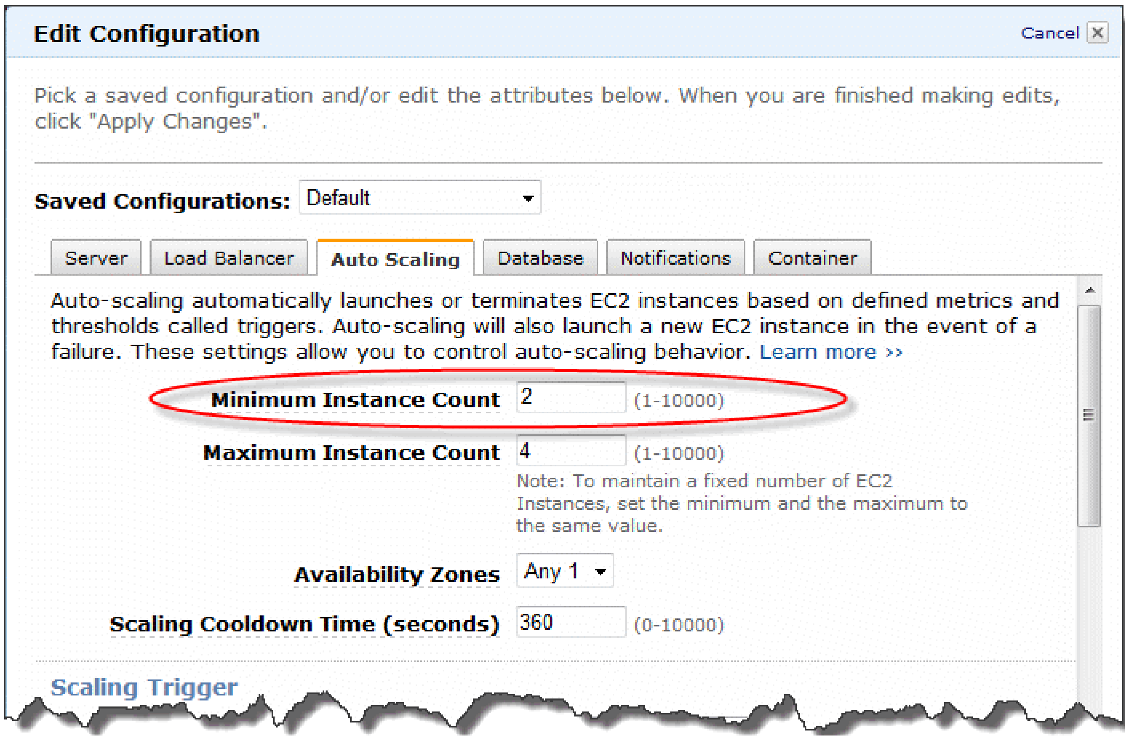
*Existing Key Pair:
Note: Key pairs are used to enable login to your instances.

*Monitoring Interval: 5 minute

*Custom AMI ID: ami-5e4db237

Note: *It may take a few minutes to see changes to these options take effect in your environment.

2. Click the **Auto Scaling** tab in the **Edit Configuration** dialog box.


Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

Auto-scaling automatically launches or terminates EC2 instances based on defined metrics and thresholds called triggers. Auto-scaling will also launch a new EC2 instance in the event of a failure. These settings allow you to control auto-scaling behavior. Learn more >>

Minimum Instance Count	2	(1-10000)
Maximum Instance Count	4	(1-10000)
Availability Zones	Any 1	
Scaling Cooldown Time (seconds)	360	(0-10000)

Note: To maintain a fixed number of EC2 Instances, set the minimum and the maximum to the same value.

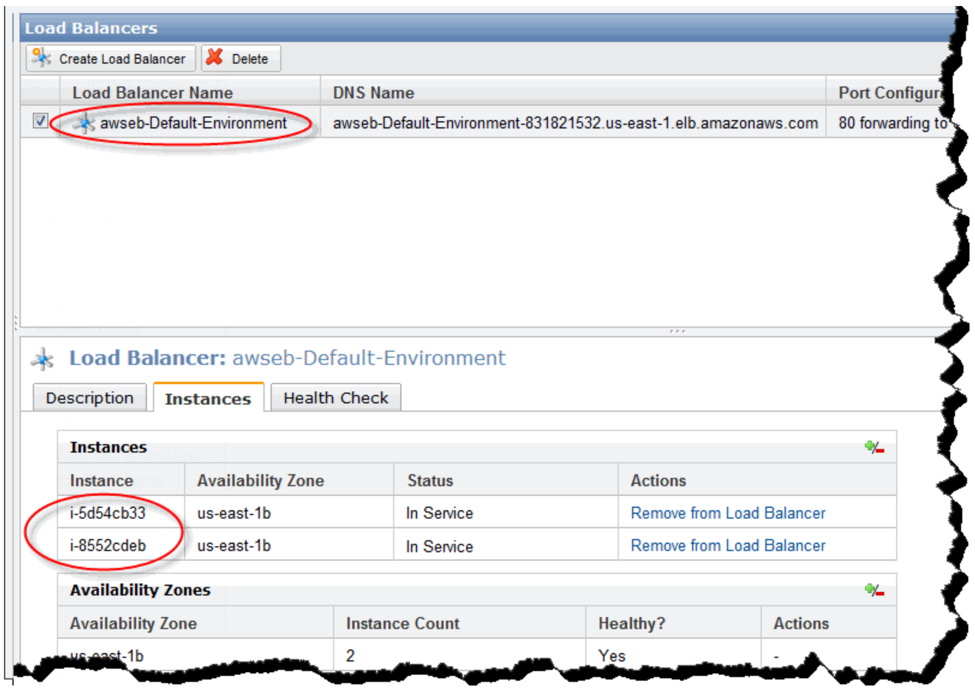
Scaling Trigger

3. Change **Minimum Instance Count** from 1 to 2. This change increases the minimum number of Auto Scaling instances deployed in Amazon EC2.
4. Click **Apply Changes**.

Wait for the environment's status to change from **Updating** to **Ready**, and then go to the next task to verify your changes.

To verify changes to load balancers

1. Select your environment in the **Environments** pane.
2. Click the **Events** tab.
   You should see the event `Successfully completed update Environment activity` in the events list. This confirms that the Auto Scaling minimum instance count has been set to 2. A second instance is launched automatically.
3. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
4. In the **Navigation** pane, under **Networking & Security**, click **Load Balancers**.
5. Click **Load Balancer Name awseb-<your environment name>** in the **Load Balancers** pane.
6. Click the **Instances** tab in the **Load Balancer: awseb-<your environment name>** pane.



Load Balancer Name	DNS Name	Port Configur
awseb-Default-Environment	awseb-Default-Environment-831821532.us-east-1.elb.amazonaws.com	80 forwarding to

Load Balancer: awseb-Default-Environment

Instances

Instance	Availability Zone	Status	Actions
i-5d54cb33	us-east-1b	In Service	Remove from Load Balancer
i-8552cdeb	us-east-1b	In Service	Remove from Load Balancer

Availability Zones

Availability Zone	Instance Count	Healthy?	Actions
us-east-1b	2	Yes	-

The information shows that two instances are associated with this load balancer, corresponding to the increase in Auto Scaling instances.

Step 6: Clean Up

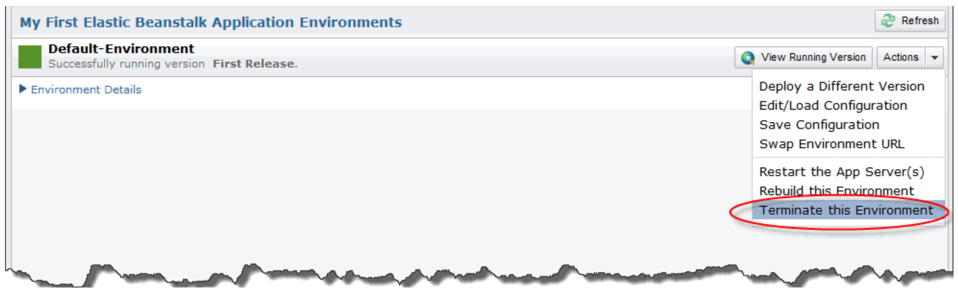
Congratulations! You have completed the AWS Elastic Beanstalk getting started tasks. To make sure you are not charged for any services you don't need, you can clean up by deleting any unwanted applications and environments from AWS Elastic Beanstalk and AWS services.

Verify that you are not using any AWS Elastic Beanstalk resources by reviewing your applications and deleting those you no longer need.

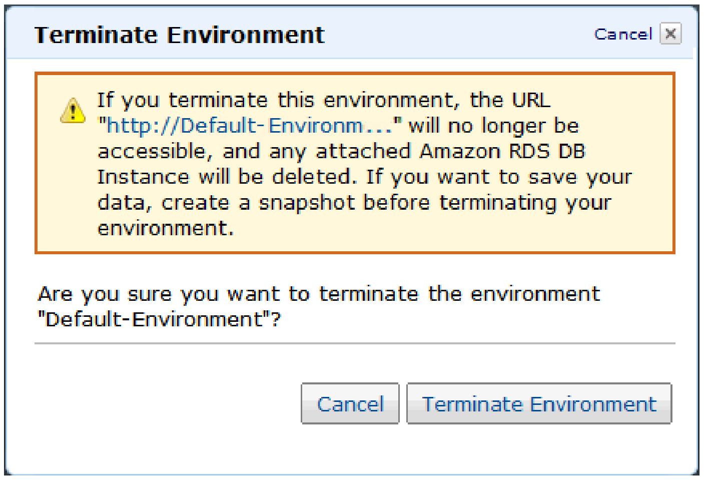
To completely delete the application

1. Terminate the environment:

   a. Click the **Actions** button next to the environment you want to delete and click **Terminate this Environment**.

   
   My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

   The **Terminate Environment** dialog box appears.



Terminate Environment

If you terminate this environment, the URL "http://Default-Environm..." will no longer be accessible, and any attached Amazon RDS DB Instance will be deleted. If you want to save your data, create a snapshot before terminating your environment.

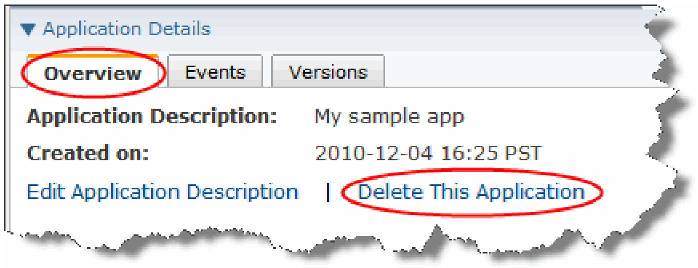
Are you sure you want to terminate the environment "Default-Environment"?

b. Click the **Terminate Environment** button.

2. Delete all application versions:

   a. In the **Application Details** view, click the **Versions** tab.
   b. Select the check box next to your application versions and click the **Delete Version** button.
   c. In the **Delete Application Version** dialog box, select the **Delete Version from Amazon S3 as well** check box.
   d. Click **Yes, Delete**.

3. In the **Application Details** view, click the **Overview** tab.



Application Description: My sample app
Created on: 2010-12-04 16:25 PST

4. Click the **Delete This Application** link.
5. Click **Yes, Delete**.

Accessing AWS Elastic Beanstalk

Topics

- AWS SDK for PHP (p. 16)
- Boto (AWS SDK for Python) (p. 16)
- AWS Elastic Beanstalk API (p. 16)
- Endpoints (p. 16)

You can create an application using one of several different AWS Elastic Beanstalk interfaces: the AWS Elastic Beanstalk console in the AWS Management Console, Git deployment using AWS DevTools, the AWS Elastic Beanstalk command line interface, the Amazon Toolkits for Eclipse and Visual Studio, or programmatically through the AWS SDKs for Java, PHP, .NET, Python, or the AWS Elastic Beanstalk web service API.

The simplest and quickest method for creating an application is to use the AWS Elastic Beanstalk console. This does not require any additional software or tools. The console is a web browser interface that you can use to create and manage your AWS Elastic Beanstalk applications.

The following sections contain overviews of each of the available AWS Elastic Beanstalk interfaces. In order to use the AWS Elastic Beanstalk features, you need to have an AWS account and be signed up for AWS Elastic Beanstalk. For instructions on how to sign up for AWS Elastic Beanstalk, see Step 1: Sign up for the Service (p. 5).

AWS Management Console

The AWS Management Console enables you to manage applications through AWS Elastic Beanstalk from a single web browser interface. The console provides access to all of your deployed applications and gives you the ability to manage and monitor your applications and environments. From the console you can:

- Create and delete applications
- Add and delete application versions
- Create and delete environments
- Identify the running version within an environment
- View operational metrics
- View application and environment logs

The AWS Management Console is available at http://console.aws.amazon.com/elasticbeanstalk.

For more information about getting started with AWS Elastic Beanstalk using the AWS Management Console, go to the Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Git Deployment Via Eb

Eb is an updated command line interface for AWS Elastic Beanstalk that enables you to deploy applications quickly and more easily. To learn how to get started using eb, see Getting Started with Eb (p. 270). For an example of how to use eb for PHP, see Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git (p. 92). For an example of how to use eb for Python, see Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git (p. 103).

For a complete CLI reference for more advanced scenarios, see Operations (p. 286), and see Getting Set Up (p. 275) for instructions on how to get set up.

AWS SDK for Java

The AWS SDK for Java provides a Java API you can use to build applications that use AWS infrastructure services. With the AWS SDK for Java, you can get started in minutes with a single, downloadable package that includes the AWS Java library, code samples, and documentation.

The AWS SDK for Java requires J2SE Development Kit 5.0 or later. You can download the latest Java software from http://developers.sun.com/downloads/. The SDK also requires Apache Commons (Codec, HTTPClient, and Logging), and Saxon-HE third-party packages, which are included in the third-party directory of the SDK.

For more information on using the AWS SDK for Java, go to the AWS SDK for Java.

AWS Toolkit for Eclipse

With the AWS Toolkit for Eclipse plug-in, you can create new AWS Java web projects that are preconfigured with the AWS SDK for Java, and then deploy the web applications to AWS Elastic Beanstalk. The AWS Elastic Beanstalk plug-in builds on top of the Eclipse Web Tools Platform (WTP). The toolkit provides a Travel Log sample web application template that demonstrates the use of Amazon S3, Amazon SimpleDB, and Amazon SNS.

To ensure you have all the WTP dependencies, we recommend that you start with the Java EE distribution of Eclipse, which you can download from http://eclipse.org/downloads/.

For more information on using the AWS Elastic Beanstalk plug-in for Eclipse, go to the AWS Toolkit for Eclipse web page. To get started creating your AWS Elastic Beanstalk application using Eclipse, see Creating and Deploying AWS Elastic Beanstalk Applications in Java Using AWS Toolkit for Eclipse (p. 22).

AWS SDK for .NET

The AWS SDK for .NET enables you to build applications that use AWS infrastructure services. With the AWS SDK for .NET, you can get started in minutes with a single, downloadable package that includes the AWS .NET library, code samples, and documentation.

The AWS SDK for .NET requires .NET Framework 2.0 or later. It will work with any of the following Visual Studio editions:

- Microsoft Visual Studio Professional Edition 2010
- Microsoft Visual C# 2008 Express Edition
- Microsoft Visual Web Developer 2008 Express Edition

For more information on using the AWS SDK for .NET, go to the AWS SDK for .NET web page.

Amazon Toolkit for Visual Studio

With the Amazon Toolkit for Visual Studio plug-in, you can deploy an existing .NET application to AWS Elastic Beanstalk. You can also create new projects using the AWS templates that are preconfigured with the AWS SDK for .NET. The toolkit supports Microsoft Visual Studio 2008 (Standard Edition or higher) or Microsoft Visual Studio 2010 (Professional Edition or higher). It requires .NET Framework 3.5 Service Pack 1 or later, available at the .NET Framework Developer Center. We recommend Microsoft Visual Studio 2010. For more information on publishing your .NET application to AWS Elastic Beanstalk using Visual Studio, go to AWS Toolkit for Visual Studio. To get started creating your AWS Elastic Beanstalk application using Visual Studio, see Creating and Deploying AWS Elastic Beanstalk Applications in .NET Using AWS Toolkit for Visual Studio (p. 51).

AWS SDK for PHP

The AWS SDK for PHP enables you to build applications on top of AWS infrastructure services. With the AWS SDK for PHP, you can get started in minutes with a single, downloadable package that includes the AWS PHP library, code samples, and documentation.

The AWS SDK for PHP requires PHP 5.2 or later.

For more information on using the AWS SDK for PHP, go to the AWS SDK for PHP web page. For instructions on installing the AWS SDK for PHP, go to Installation.

Boto (AWS SDK for Python)

With Boto, you can get started in minutes with a single, downloadable package complete with the AWS Python library, code samples, and documentation. You can build Python applications on top of APIs that take the complexity out of coding directly against web services interfaces. The all-in-one library provides Python developer-friendly APIs that hide much of the lower-level tasks associated with programming for the AWS cloud, including authentication, request retries, and error handling. Practical examples are provided in Python for how to use the libraries to build applications. For information about Boto, sample code, documentation, tools, and additional resources, go to http://aws.amazon.com/python/.

AWS Elastic Beanstalk API

AWS Elastic Beanstalk provides a comprehensive API that enables you to programmatically access AWS Elastic Beanstalk functionality. For more information, go to the AWS Elastic Beanstalk API Reference.

Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the *Amazon Web Services General Reference*.

Where to Go Next

Now that you have learned about AWS Elastic Beanstalk and how to access it, we recommend that you read How Does AWS Elastic Beanstalk Work? (p. 17) This topic provides information about the AWS Elastic Beanstalk components, the architecture, and important design considerations for your AWS Elastic Beanstalk application.

How Does AWS Elastic Beanstalk Work?

Topics

Now that you have a better understanding of what AWS Elastic Beanstalk is, let's take a peek under the hood and see how AWS Elastic Beanstalk works. The following sections discuss the AWS Elastic Beanstalk components, the architecture, and important design considerations for your AWS Elastic Beanstalk application.

AWS Elastic Beanstalk Components

The components that comprise AWS Elastic Beanstalk work together to enable you to easily deploy and manage your application in the cloud. This section discusses those components.

Application

An AWS Elastic Beanstalk *application* is a logical collection of AWS Elastic Beanstalk components, including *environments*, *versions*, and *environment configurations*. In AWS Elastic Beanstalk an application is conceptually similar to a folder.

Version

In AWS Elastic Beanstalk, a *version* refers to a specific, labeled iteration of deployable code. A version points to an Amazon Simple Storage Service (Amazon S3) object that contains the deployable code (e.g., a Java WAR file). A version is part of an application. Applications can have many versions.

Environment

An *environment* is a version that is deployed onto AWS resources. Each environment runs only a single version, however you can run the same version or different versions in many environments at the same time. When you create an environment, AWS Elastic Beanstalk provisions the resources needed to run the application version you specified. For more information about the environment and the resources that are created, see Architectural Overview (p. 18).

Environment Configuration

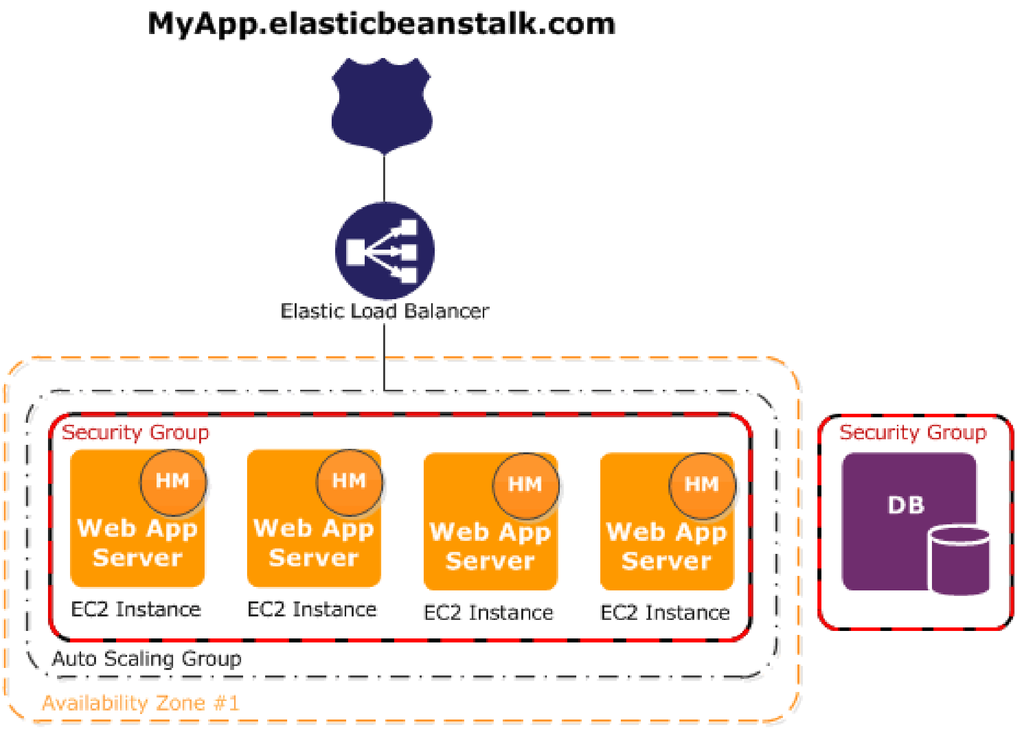
An *environment configuration* identifies a collection of parameters and settings that define how an environment and its associated resources behave. When you update an environment's configuration settings, AWS Elastic Beanstalk automatically applies the changes to existing resources or deletes and deploys new resources (depending on the type of change).

Configuration Template

A *configuration template* is a starting point for creating unique environment configurations. Configuration templates can be created or modified only by using the AWS Elastic Beanstalk command line utilities or APIs.

Architectural Overview

This following diagram illustrates an example AWS Elastic Beanstalk architecture and shows how the components work together. The remainder of this section discusses all the components in more detail.



The environment is the heart of the application. In the diagram, the environment is delineated by the broken yellow line. When you create an environment, AWS Elastic Beanstalk provisions the resources required to run your application. AWS resources created for an environment include one elastic load balancer (ELB in the diagram), an Auto Scaling group, and one or more Amazon EC2 instances.

Every environment has a CNAME (URL) that points to a load balancer. In the diagram the Amazon Route 53 URL (MyApp.elasticbeanstalk.com) is the CNAME. Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. It provides secure and reliable routing to your infrastructure. Your domain name that you registered with your DNS provider will forward requests to the CNAME. The load balancer sits in front of the Amazon EC2 instances, which are part of an Auto Scaling group. (The Auto Scaling group is delineated in the diagram by a broken black line.) Auto Scaling automatically starts additional Amazon EC2 instances to accommodate increasing load on your application. If the load on your application decreases, Auto Scaling stops instances, but always leaves at least one instance running. For information about how to map your root domain to your Elastic Load Balancer, see Using AWS Elastic Beanstalk with Amazon Route 53 to Map Your Root Domain to Your Load Balancer (p. 221).

The software stack running on the Amazon EC2 instances is dependent on the *container type*. A container type defines the infrastructure topology and software stack to be used for that environment. For example, the AWS Elastic Beanstalk for Apache Tomcat 7 container uses the Amazon Linux operating system, Apache web server, and Apache Tomcat software.

Note

AWS Elastic Beanstalk currently supports Apache Tomcat 6, Apache Tomcat 7, PHP 5.3, and IIS 7.5.

Each Amazon EC2 server instance that runs your application uses one of these container types. In addition, a software component called the *host manager (HM)* runs on each Amazon EC2 server instance. (In the diagram, the HM is an orange circle in each EC2 instance.) The host manager is responsible for:

- Deploying the application
- Aggregating events and metrics for retrieval via the console, the API, or the command line
- Generating instance-level events
- Monitoring the application log files for critical errors
- Monitoring the application server
- Patching instance components
- Rotating your application's log files and publishing them to Amazon S3

The host manager reports metrics, errors and events, and server instance status, which are available via the AWS Management Console, APIs, and CLIs.

The Amazon EC2 instances shown in the diagram are part of one security group. A security group defines the firewall rules for your instances. By default, AWS Elastic Beanstalk defines a security group, which allows everyone to connect using port 80 (HTTP). You can define more than one security group. For instance, you can define a security group for your database server. For more information about Amazon EC2 security groups and how to configure them for your AWS Elastic Beanstalk application, see the Amazon EC2 Security Groups (p. 161).

Design Considerations

Because applications deployed using AWS Elastic Beanstalk run on Amazon cloud resources, you should keep five things in mind when designing your application: *scalability*, *security*, *persistent storage*, *fault tolerance*, and *content delivery*. For a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security and economics, go to AWS Cloud Computing Whitepapers.

Scalability

When you're operating in a physical hardware environment, as opposed to a cloud environment, you can approach scalability two ways—you can scale up (vertical scaling) or scale out (horizontal scaling). The scale-up approach requires an investment in powerful hardware as the demands on the business increase, whereas the scale-out approach requires following a distributed model of investment, so hardware and application acquisitions are more targeted, datasets are federated, and design is service-oriented. The scale-up approach could become very expensive, and there's still the risk that demand could outgrow capacity. Although the scale-out approach is usually more effective, it requires predicting the demand at regular intervals and deploying infrastructure in chunks to meet demand. This approach often leads to excessive capacity and requires constant manual monitoring.

By moving to the cloud you can bring the use of your infrastructure into close alignment with demand by leveraging the elasticity of the cloud. Elasticity is the streamlining of resource acquisition and release, so that your infrastructure can rapidly scale in and scale out as demand fluctuates. To implement elasticity, configure your Auto Scaling settings to send triggers to your system to take appropriate actions based on metrics (utilization of the servers or network I/O, for instance). You can use Auto Scaling to automatically add compute capacity when usage rises and remove it when usage drops. Monitor your system metrics (CPU, memory, disk I/O, network I/O) using Amazon CloudWatch so that you can take appropriate actions (such as launching new AMIs dynamically using Auto Scaling) or send notifications. For more instructions on configuring Auto Scaling, see Configuring Auto Scaling with AWS Elastic Beanstalk (p. 170).

AWS Elastic Beanstalk applications should also be as *stateless* as possible, using loosely coupled fault-tolerant components that can be scaled out as needed. For more information about designing scalable application architectures for AWS, go to Architecting for the Cloud: Best Practices.

Security

Physical security is typically handled by your service provider, however network and application-level security is your responsibility. If you need to protect information from your clients to your elastic load balancer, you should configure SSL. You will need a certificate from an external certification authority such as VeriSign or Entrust. The public key included in the certificate authenticates your server to the browser and serves as the basis for creating the shared session key used to encrypt the data in both directions. For instructions on creating and uploading an SSL certificate, go to Creating and Uploading Server Certificates in *Using AWS Identity and Access Management*. For instructions on configuring your SSL ID for your AWS Elastic Beanstalk application, see Ports (p. 165).

Note

Data moving between the Elastic Load Balancer and the Amazon EC2 instances is unencrypted.

Persistent Storage

AWS Elastic Beanstalk applications run on Amazon EC2 instances that have no persistent local storage. When the Amazon EC2 instances terminate, the local file system is not saved, and new Amazon EC2 instances start with a default file system. You should design your application to store data in a persistent data source. Amazon Web Services offers a number of persistent storage options that you can leverage for your application, including:

- Amazon Simple Storage Service (Amazon S3). For more information about Amazon S3, go to the documentation.
- Amazon Elastic Block Store (Amazon EBS). For more information, go to the documentation and see also the article Feature Guide: Elastic Block Store.
- Amazon SimpleDB. For more information, go to the documentation and see also Query 101: Building Amazon SimpleDB Queries.

- Amazon Relational Database Service (Amazon RDS). For more information, go to the documentation and see also Amazon RDS for C# Developers.

Fault Tolerance

As a rule of thumb, you should be a pessimist when designing architecture for the cloud. Always design, implement, and deploy for automated recovery from failure. Use multiple Availability Zones for your Amazon EC2 instances and for Amazon RDS. Availability Zones are conceptually like logical datacenters. Use Amazon CloudWatch to get more visibility into the health of your AWS Elastic Beanstalk application and take appropriate actions in case of hardware failure or performance degradation. Configure your Auto Scaling settings to maintain your fleet of Amazon EC2 instances at a fixed size so that unhealthy Amazon EC2 instances are replaced by new ones. If you are using Amazon RDS, then set the retention period for backups, so that Amazon RDS can perform automated backups.

Content Delivery

When users connect to your website, their requests may be routed through a number of individual networks. As a result users may experience poor performance due to high latency. Amazon CloudFront can help ameliorate latency issues by distributing your web content (such as images, video, and so on) across a network of edge locations around the world. End users are routed to the nearest edge location, so content is delivered with the best possible performance. CloudFront works seamlessly with Amazon S3, which durably stores the original, definitive versions of your files. For more information about Amazon CloudFront, see http://aws.amazon.com/cloudfront.

Software Updates and Patching

AWS Elastic Beanstalk does not currently have a software update mechanism or policy. AWS Elastic Beanstalk periodically updates its default AMIs with new software and patches. Running environments, however, do not get automatically updated. To obtain the latest AMIs, you must launch a new environment. For more information about launching a new environment, see Launching New Environments (p. 135).

Where to Go Next

Now that you have learned some AWS Elastic Beanstalk basics, you are ready to start creating and deploying your applications. If you are a developer, go to one of the following sections:

- **Java** — Creating and Deploying AWS Elastic Beanstalk Applications in Java Using AWS Toolkit for Eclipse (p. 22)
- **PHP** — Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git (p. 92)
- **.NET** — Creating and Deploying AWS Elastic Beanstalk Applications in .NET Using AWS Toolkit for Visual Studio (p. 51)
- **Python** — Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git (p. 103)

If you want to manage and configure your applications and environments using the AWS Management Console, command line interface, or APIs, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Creating and Deploying AWS Elastic Beanstalk Applications in Java Using AWS Toolkit for Eclipse

Topics

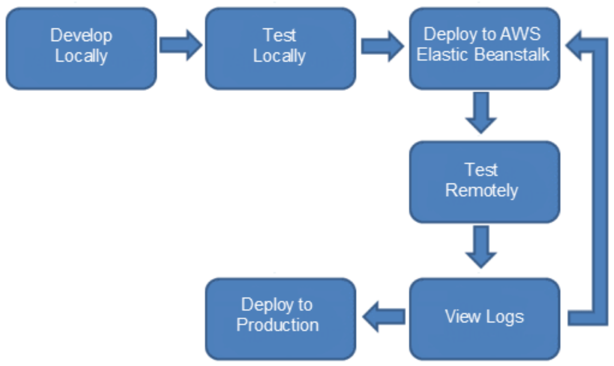
The first part of this topic provides step-by-step instructions for creating, testing, deploying, and redeploying your Java application to AWS Elastic Beanstalk using the AWS Toolkit for Eclipse. The second part of this topic provides information on how you can manage and configure your applications and environments using the AWS Toolkit for Eclipse. For more information about prerequisites and installing the AWS Toolkit for Eclipse, go to http://aws.amazon.com/eclipse. You can also check out the Using AWS Elastic Beanstalk with the AWS Toolkit for Eclipse video. This topic also provides useful information covering tools, how-to topics, and additional resources for Java developers.

Develop, Test, and Deploy

Topics

- Deploy to AWS Elastic Beanstalk (p. 25)
- Test Remotely (p. 28)
- View Logs (p. 28)
- Edit the Application and Redeploy (p. 29)
- Deploy to Production (p. 30)

The following diagram illustrates a typical software development life cycle that includes deploying your application to AWS Elastic Beanstalk.



After developing and testing your application locally, you will typically deploy your application to AWS Elastic Beanstalk. After deployment, your application will be live at a URL such as http://myexampleapp-wpams3yrvj.elasticbeanstalk.com. Because your application is live, you should consider setting up multiple environments, such as a testing environment and a production environment. You can use the AWS Toolkit for Eclipse if you want to set up different AWS accounts for testing, staging, and production. For more information about managing multiple accounts, see Managing Multiple AWS Accounts (p. 35).

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can point your domain name to the Amazon Route 53 CNAME <yourappname>.elasticbeanstalk.com. Contact your DNS provider to set this up. For information about how to map your root domain to your Elastic Load Balancer, see Using AWS Elastic Beanstalk with Amazon Route 53 to Map Your Root Domain to Your Load Balancer (p. 221).

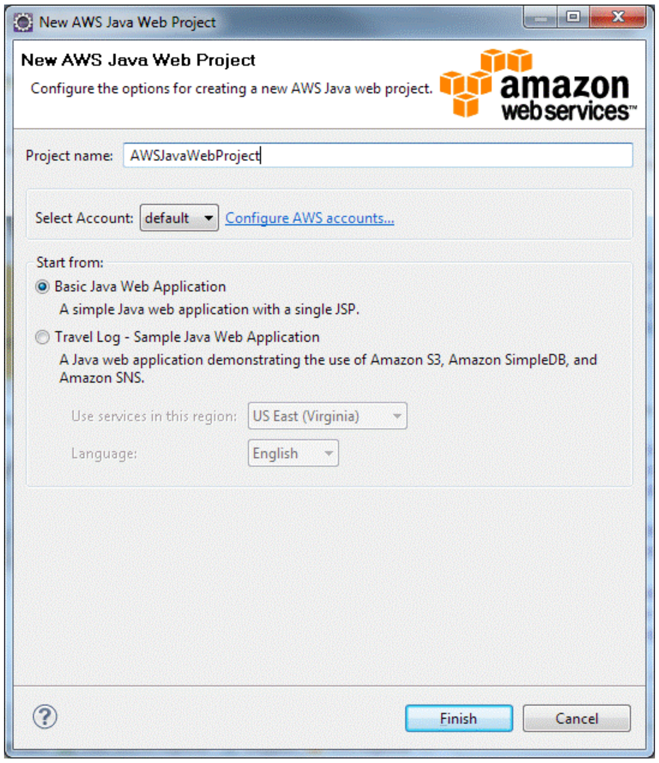
After you remotely test and debug your AWS Elastic Beanstalk application, you can make any updates and then redeploy to AWS Elastic Beanstalk. After you are satisfied with all of your changes, you can upload the latest version to your production environment. The following sections explain each stage of the software development life cycle.

Create Project

The AWS Toolkit provides an AWS Java web project template for use in Eclipse. The template creates a WTP dynamic web project that includes the AWS SDK for Java in the project's classpath. Your AWS Account credentials and a simple index.jsp file are provided to help you get started.

To create a new AWS Java web project in Eclipse

1. Click **File** from the Eclipse file menu, and select **New**.
2. Select AWS Java Web Project. The New AWS Java Web Project wizard appears.


New AWS Java Web Project

New AWS Java Web Project

Configure the options for creating a new AWS Java web project.

Project name: AWSJavaWebProject

Select Account: default Configure AWS accounts...

Start from:

(●) Basic Java Web Application
A simple Java web application with a single JSP.

( ) Travel Log - Sample Java Web Application
A Java web application demonstrating the use of Amazon S3, Amazon SimpleDB, and Amazon SNS.

Use services in this region: US East (Virginia)

Language: English

3. Enter the name of your AWS Java Web Project in the **Project name** field.
4. Select the AWS account you want to use to deploy your application. If you haven't already configured an account, click **Configure AWS accounts** to add a new account. For instructions on how to add a new account, go to Managing Multiple AWS Accounts (p. 35).
5. Select the application you want to start from. If you select Travel Log - Sample Java Web Application, then you need to select the following items:
   - From the **Use services in this region** list, click the region where you want your application to run.
   - From the **Language** list, click the language that you want the application to be displayed in.

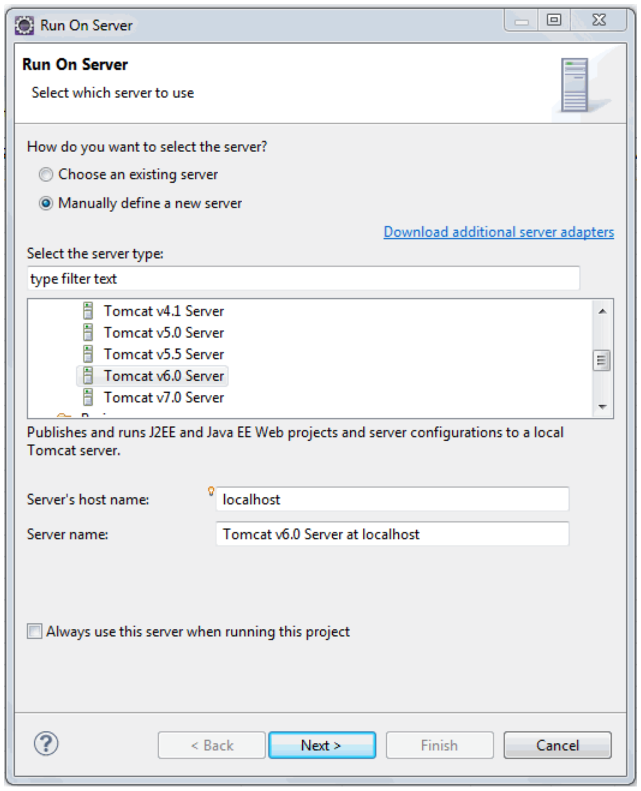
6. Click **Finish.**

A new AWS Java web project is created inside your Eclipse workspace. To help you get started developing your project, a basic index.jsp file is included inside your WebContent folder. Your AWS security credentials are included in the AwsCredentials.properties file inside your src folder.

Test Locally

You can test and debug your application before you deploy it to AWS Elastic Beanstalk using the built-in WTP tools.

1. Right-click your Java web project in the **Project Explorer** view.
2. Select **Run As**, and click **Run On Server**. The **Run On Server** wizard appears.


Run On Server

Run On Server

Select which server to use

How do you want to select the server?

( ) Choose an existing server

(●) Manually define a new server

Download additional server adapters

Select the server type:

type filter text

- Tomcat v4.1 Server
- Tomcat v5.0 Server
- Tomcat v5.5 Server
- Tomcat v6.0 Server
- Tomcat v7.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Tomcat v6.0 Server at localhost

[ ] Always use this server when running this project

3. Select **Manually define a new server**.
4. Expand the **Apache** item in the **Server Type** explorer and select **Tomcat v6.0 Server** or **Tomcat v7.0 Server**.
5. Click **Next**.
6. You might be prompted to install Apache. If you are using Microsoft Windows, you can click the **Download and Install** button to install the software. After you have installed the software, click **Finish**.

   You will now see your application appear on the localhost.

After you have tested your application, you can now deploy to AWS Elastic Beanstalk.

Deploy to AWS Elastic Beanstalk

The Elastic Beanstalk server is an Eclipse WTP server with added functionality that enables you to restart, publish, and terminate your Java web application. The Elastic Beanstalk server in Eclipse represents an AWS Elastic Beanstalk environment. An *environment* is a running instance of an application on the AWS platform.

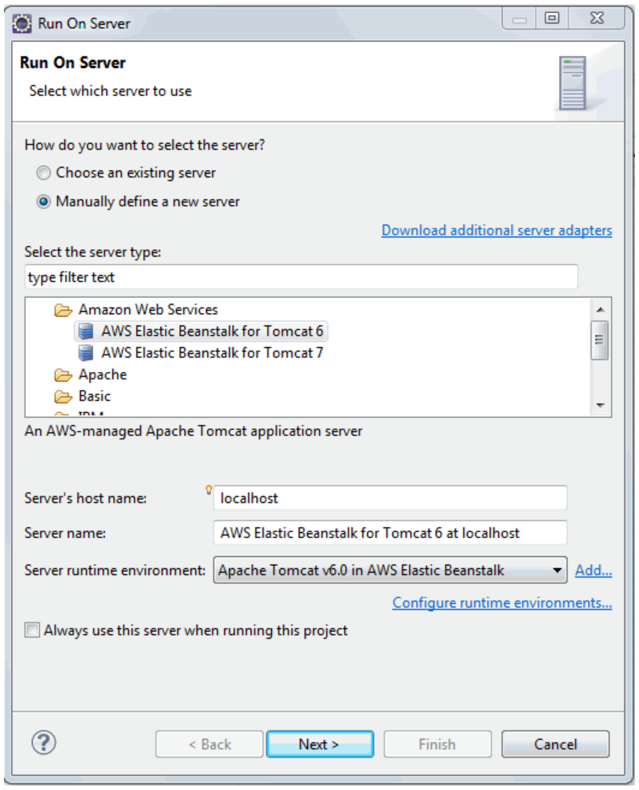
Define Your Server

Before deploying your application, you must define your server and configure your application.

To define your Java web application server

1. Right-click your Java web project in the **Project Explorer** view.

2. Select **Amazon Web Services**, and click **Deploy to AWS Elastic Beanstalk**. The **Run On Server** wizard appears.


Run On Server
Run On Server
Select which server to use

How do you want to select the server?
Choose an existing server
Manually define a new server
Download additional server adapters
Select the server type:
type filter text
Amazon Web Services
AWS Elastic Beanstalk for Tomcat 6
AWS Elastic Beanstalk for Tomcat 7
Apache
Basic
An AWS-managed Apache Tomcat application server

Server's host name: localhost
Server name: AWS Elastic Beanstalk for Tomcat 6 at localhost
Server runtime environment: Apache Tomcat v6.0 in AWS Elastic Beanstalk
Add...
Configure runtime environments...
Always use this server when running this project

3. Select **Manually define a new server**.
4. Expand the **Amazon Web Services** item in the **Server Type** explorer and select **AWS Elastic Beanstalk for Tomcat 6** or **AWS Elastic Beanstalk for Tomcat 7**.
5. Enter the server's host name and server name in the provided fields. Click **Next**. The **Configure Application and Environment** options appear.

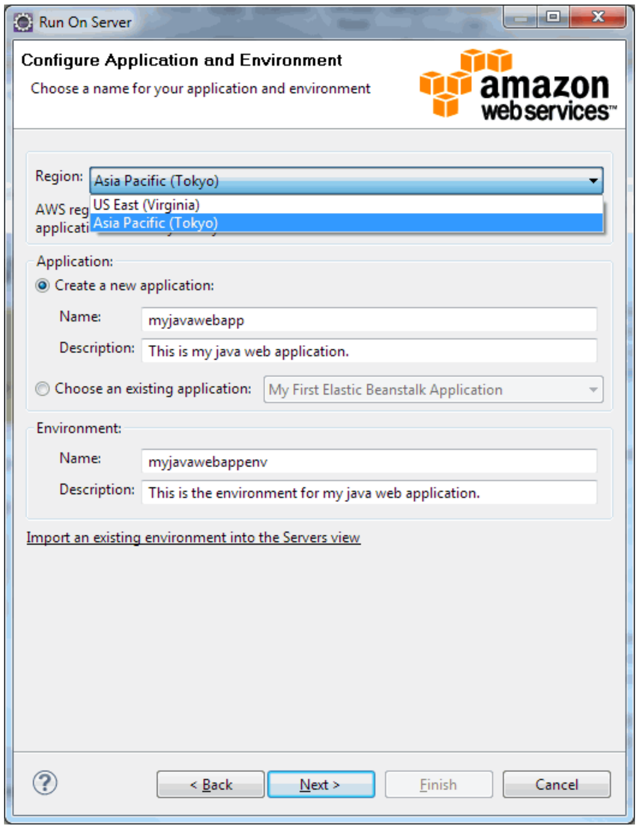
You have defined your AWS Java web application server. You must still configure your AWS Java web application before deployment.

Configure

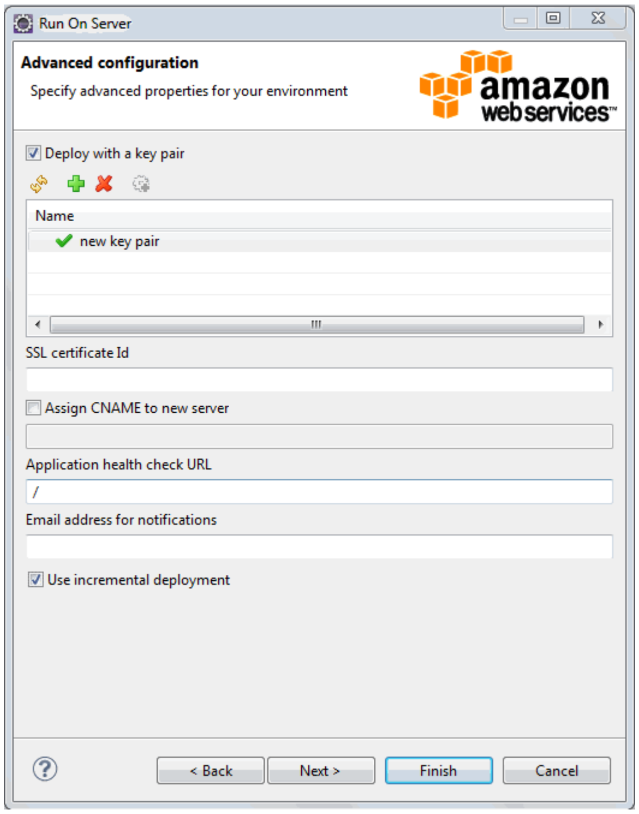
Each application has a configuration and a version. A specific application instance is called an *environment*. An environment can have only one version at a time, but you can have multiple simultaneous environments running the same or different versions. AWS resources created for an environment include one elastic load balancer, an Auto Scaling group, and one or more Amazon EC2 instances.

To configure your Amazon web application

1. Select an Amazon region from the **Region** drop-down menu.


Run On Server
Configure Application and Environment
Choose a name for your application and environment
amazon web services
Region:
Asia Pacific (Tokyo)
US East (Virginia)
Asia Pacific (Tokyo)
AWS reg
applicati
Application:
Create a new application:
Name:
myjavawebapp
Description:
This is my java web application.
Choose an existing application:
My First Elastic Beanstalk Application
Environment:
Name:
myjavawebappenv
Description:
This is the environment for my java web application.
Import an existing environment into the Servers view
< Back
Next >
Finish
Cancel

2. Enter an application name in the provided field, and optionally provide an application description.
3. Enter an environment name in the provided field, and optionally provide an environment description. The environment name must be unique within your AWS Account. Click **Next**.
4. Click **Deploy with a key pair**.



Advanced configuration

Specify advanced properties for your environment

[✓] Deploy with a key pair

Name
✓ new key pair

SSL certificate Id

[ ] Assign CNAME to new server

Application health check URL: /

Email address for notifications

[✓] Use incremental deployment

5. Right-click anywhere inside the key pair list menu and select **New Key Pair**. The **Create New Key Pair** dialog box appears.
6. Enter a key pair name in the **Key Pair Name** text field. Enter a private key directory in the **Private Key Directory** text field, or click **Browse** and select a directory.
7. Select the **Use incremental deployment** check box to deploy only the changed files. An incremental deployment is faster because you are updating only the files that changed instead of all the files.
8. Click **OK** and select the newly created key pair in the key pair list menu.
9. Click **Finish**.

After you finish the wizard, you will be prompted to enter a new version label for your application version, and then a new server will appear in the Server view. Your Java web project will be exported as a WAR file, uploaded to Amazon S3, and registered as a new application version with Elastic Beanstalk. The Elastic Beanstalk deployment feature will monitor your environment until it becomes available with the newly deployed code, and it will open your application in a web browser when it's ready.

Test Remotely

To test your application remotely, you can run your application in debug mode. For more information, see Remote Debugging (p. 47).

View Logs

You can configure your environment so that AWS Elastic Beanstalk copies the logs from the Amazon EC2 instances running your applications to the Amazon S3 bucket associated with your application. For more information on viewing these logs from the AWS Management Console, see Working with

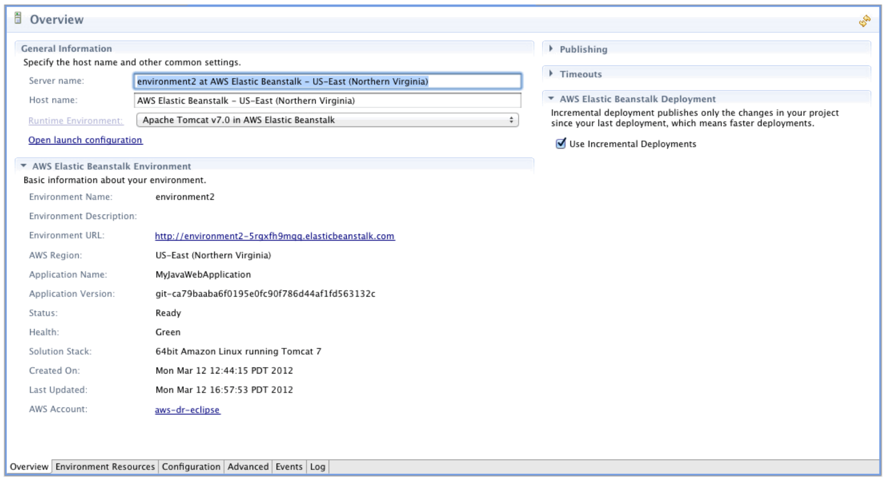
Logs (p. 199). For information on how to enable hourly rotation of your logs to Amazon S3, see Amazon S3 Log Rotation (p. 47).

Edit the Application and Redeploy

Now that you have tested your application, it is easy to edit and redeploy it, and to see the results in moments. Typically, when you redeploy your application, all of the files will get updated. However, if you choose to do an incremental deployment, the toolkit will update the modified files, making your deployment faster.

To choose to use incremental deployments

1. In the AWS Toolkit for Eclipse, expand the AWS Elastic Beanstalk node and your application node.
2. In **AWS Explorer**, double-click your AWS Elastic Beanstalk environment.



General Information
Specify the host name and other common settings.

Server name:	environment2 at AWS Elastic Beanstalk - US-East (Northern Virginia)
Host name:	AWS Elastic Beanstalk - US-East (Northern Virginia)
Runtime Environment:	Apache Tomcat v7.0 in AWS Elastic Beanstalk

Open launch configuration

AWS Elastic Beanstalk Environment
Basic information about your environment.

Environment Name:	environment2
Environment Description:	
Environment URL:	http://environment2-5rqxfh9mqq.elasticbeanstalk.com
AWS Region:	US-East (Northern Virginia)
Application Name:	MyJavaWebApplication
Application Version:	git-ca79baaba6f0195e0fc90f786d44af1fd563132c
Status:	Ready
Health:	Green
Solution Stack:	64bit Amazon Linux running Tomcat 7
Created On:	Mon Mar 12 12:44:15 PDT 2012
Last Updated:	Mon Mar 12 16:57:53 PDT 2012
AWS Account:	aws-dr-eclipse

AWS Elastic Beanstalk Deployment
Incremental deployment publishes only the changes in your project since your last deployment, which means faster deployments.

[✓] Use Incremental Deployments

3. In the **Overview** pane, expand the **AWS Elastic Beanstalk Deployment** tab, and then select **Use Incremental Deployments**.

To edit and redeploy your Java web application

1. In Eclipse, locate and open the index.jsp file in the **Project Explorer** or **Package Explorer** view. Edit the source contained in the file.
2. Right-click your Java web project in the **Package Explorer** or **Package Explorer** view.
3. Select **Amazon Web Services**, and click **Deploy to AWS Elastic Beanstalk**. The **Run on Server** wizard appears.
4. Select **Choose an existing server**, and click **Finish**.

> **Note**
>
> If you launched a new environment using the AWS Management Console, you will need to import your existing environments into Eclipse. For more information, see Importing Existing Environments into Eclipse (p. 30).

When the application has deployed successfully, index.jsp is displayed in Eclipse and shows any edits you have made.

Deploy to Production

When you are satisfied with all of the changes you want to make to your application, you can deploy your application to your production environment. To deploy the existing version of the application to production with zero downtime, follow the steps at Deploying Versions with Zero Downtime (p. 143). You can also create a new environment inside Eclipse, but you will need to specify a new version for your application. For instructions on deploying to AWS Elastic Beanstalk inside Eclipse, see Deploy to AWS Elastic Beanstalk (p. 25).

Importing Existing Environments into Eclipse

You can import existing environments that you created in the AWS Management Console into Eclipse. To import existing environments, expand the **AWS Elastic Beanstalk** node and double-click on an environment in the AWS Explorer inside Eclipse. You can now deploy your AWS Elastic Beanstalk applications to this environment.

Using Custom Environment Properties with AWS Elastic Beanstalk

You can use the AWS Management Console or the AWS Toolkit for Eclipse to set environment properties that AWS Elastic Beanstalk passes to your server instances. Environment properties are properties specific to your application environment and are not actual (shell) environment variables. More specifically, PARAM1, PARAM2, etc are system properties passed in to the JVM at startup using the -D flag, and you can use them to pass database connection strings, security credentials, or other information that you don’t want to hardcode into your application. Storing this information in environment properties can help increase the portability and scalability of your application.

Using Custom Environment Properties with AWS Toolkit for Eclipse

The following example sets the `JDBC_CONNECTION_STRING` and `PARAM1` environment properties in the AWS Toolkit for Eclipse. After you set these properties, they become available to your AWS Elastic Beanstalk application as system properties called `JDBC_CONNECTION_STRING` and `PARAM1`, respectively.

The procedure for setting other environment properties is the same. You can use `PARAM1` through `PARAM5` for any purpose you choose, but you cannot rename the properties.

To set environment properties for your AWS Elastic Beanstalk application

1. In the AWS Toolkit for Eclipse, expand the AWS Elastic Beanstalk node and your application node, and then double-click your AWS Elastic Beanstalk environment in the AWS Explorer.
2. Click the **Configuration** tab.
3. Under **Container**, navigate to the **Environment Properties**.
4. In the **JDBC_CONNECTION_STRING** text box, type a connection string. For example, the following JDBC connection string would connect to a MySQL database instance on port 3306 of localhost, with a username of "me" and a password of "mypassword":

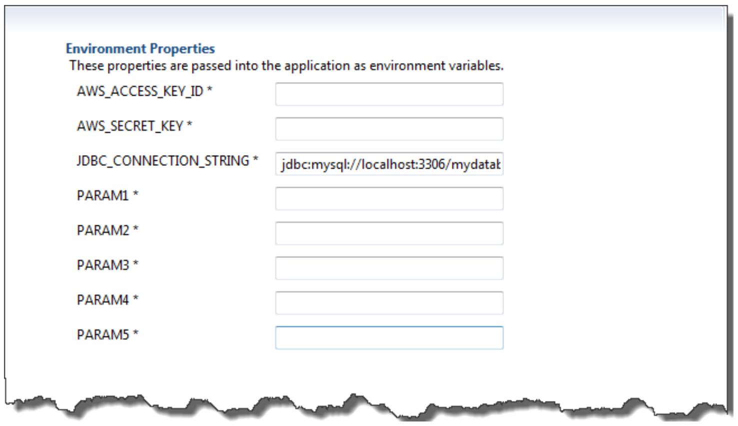
   ```
   jdbc:mysql://localhost:3306/mydatabase?user=me&password=mypassword
   ```

   This will be accessible to your AWS Elastic Beanstalk application via a system property called `JDBC_CONNECTION_STRING`.

5. In the **PARAM1** text box, enter a string. For example:

   `My test parameter.`

   This will be accessible to your AWS Elastic Beanstalk application via an system property called `PARAM1`.



Environment Properties
These properties are passed into the application as environment variables.

AWS_ACCESS_KEY_ID *	
AWS_SECRET_KEY *	
JDBC_CONNECTION_STRING *	jdbc:mysql://localhost:3306/mydatat
PARAM1 *	
PARAM2 *	
PARAM3 *	
PARAM4 *	
PARAM5 *	

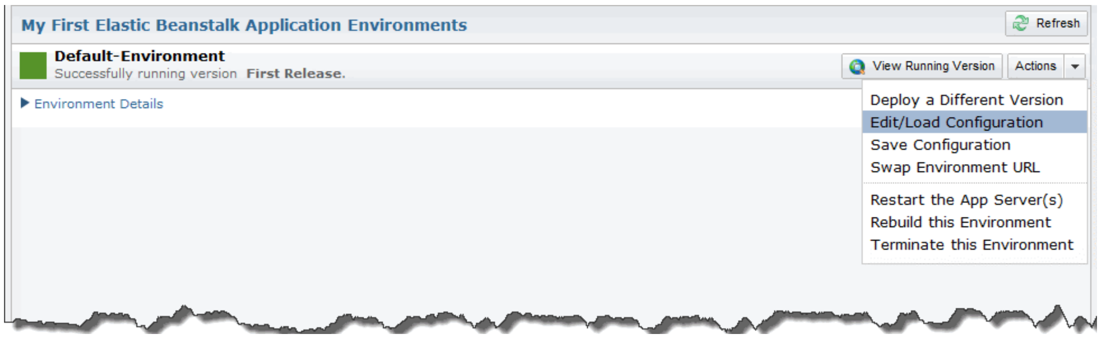
Using Custom Environment Properties with AWS Management Console

The following example sets the `JDBC_CONNECTION_STRING` and `PARAM1` environment properties in the AWS Toolkit for Eclipse. After you set these properties, they become available to your AWS Elastic Beanstalk application as system properties called `JDBC_CONNECTION_STRING` and `PARAM1`, respectively.

The procedure for setting other environment properties is the same. You can use `PARAM1` through `PARAM5` for any purpose you choose, but you cannot rename the properties.

To set environment properties for your AWS Elastic Beanstalk application

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Select your application from the drop-down list at the top of the AWS Elastic Beanstalk page.
3. On the AWS Elastic Beanstalk console, in the **Environments** list for your application, click the **Actions** arrow, and then click **Edit/Load Configuration**.



My First Elastic Beanstalk Application Environments

Default-Environment
Successfully running version First Release.

4. In the **Edit Configuration** window, click the **Container** tab.
5. Scroll down to the **Environment Properties** section.

6. In the **JDBC_CONNECTION_STRING** text box, type a connection string. For example, the following JDBC connection string would connect to a MySQL database instance on port 3306 of localhost, with a username of "me" and a password of "mypassword":

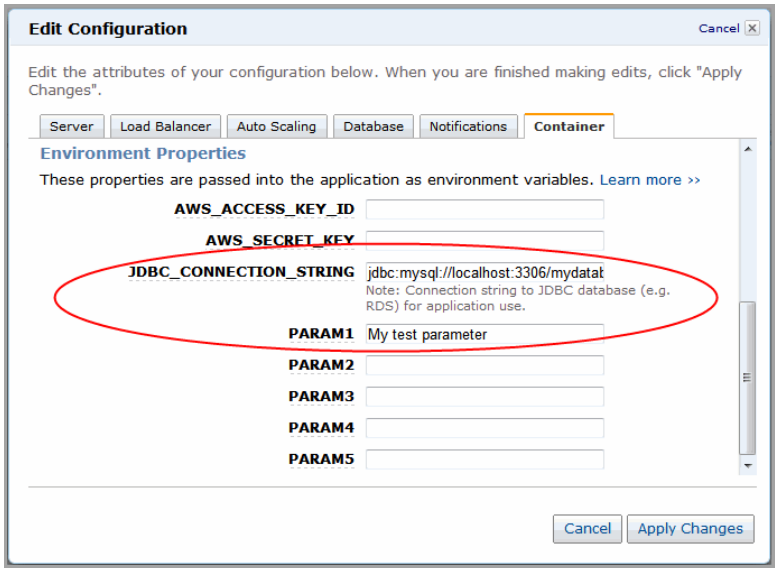
   `jdbc:mysql://localhost:3306/mydatabase?user=me&password=mypassword`

   This will be accessible to your AWS Elastic Beanstalk application via a system property called `JDBC_CONNECTION_STRING`.

7. In the **PARAM1** text box, enter a string. For example:

   `My test parameter.`

   This will be accessible to your AWS Elastic Beanstalk application via an system property called `PARAM1`.



Edit the attributes of your configuration below. When you are finished making edits, click "Apply Changes".

These properties are passed into the application as environment variables. Learn more ››

AWS_ACCESS_KEY_ID	
AWS_SECRET_KEY	
JDBC_CONNECTION_STRING	jdbc:mysql://localhost:3306/mydatat Note: Connection string to JDBC database (e.g. RDS) for application use.
PARAM1	My test parameter
PARAM2	
PARAM3	
PARAM4	
PARAM5	

8. Click **Apply Changes**.

   AWS Elastic Beanstalk will update your environment. This might take several minutes.

Accessing Custom Environment Properties

After you set your environment properties for your AWS Elastic Beanstalk application, you can access the environment properties from your code. For example, the following code snippet shows how to access the AWS Elastic Beanstalk environment properties using JavaScript in a JavaServer Page (JSP):

```

<p>
    The JDBC_CONNECTION_STRING environment variable is:
    <%= System.getProperty("JDBC_CONNECTION_STRING") %>
</p>

<p>
    The PARAM1 environment variable is:
    <%= System.getProperty("PARAM1") %>
    </p>
```

Using Amazon RDS and MySQL Connector/J

With Amazon Relational Database Service (Amazon RDS), you can quickly and easily provision and maintain a MySQL, Oracle, or Microsoft SQL Server instance in the cloud. For more information about Amazon RDS, go to http://aws.amazon.com/rds/.

This topic explains how you can use Amazon RDS and MySQL Connector/J with your AWS Elastic Beanstalk Java application.

Note

The instructions in this topic are for non-legacy container types. If you have deployed an AWS Elastic Beanstalk application using a legacy container type, we recommend that you migrate to a non-legacy container type to gain access to new features. For instructions on how to check the container type and migrate your application, see Migrating Your Application from a Legacy Container Type (p. 204). For instructions on using MySQL Connector/J with applications running on legacy container types, see Using Amazon RDS and MySQL Connector/J (Legacy Container Types) (p. 349).

To use Amazon RDS from your AWS Elastic Beanstalk application, you need to do the following:

1. Create an Amazon RDS DB Instance.
2. Download and install MySQL Connector/J.
3. Establish a database connection in your Java code by using the connectivity information for your Amazon RDS DB Instance.
4. Deploy your application to AWS Elastic Beanstalk.

This topic walks you through the following:

- Using a new Amazon RDS DB Instance with Java
- Using an existing Amazon RDS DB Instance with Java

Using a New Amazon RDS DB Instance with Java

This topic walks you through creating a new Amazon RDS DB Instance and using it with your Java application.

To use a new Amazon RDS DB Instance and Java from your AWS Elastic Beanstalk application

1. Create an Amazon RDS DB Instance. You can create an RDS DB Instance in one of the following ways:

   - Create an RDS DB Instance when you create an application. For instructions using the AWS Elastic Beanstalk console, see Creating New Applications (p. 124).
   - Create an RDS DB Instance when you launch a new environment. For instructions using the AWS Elastic Beanstalk console, see Launching New Environments (p. 135).
   - If you already deployed an application to AWS Elastic Beanstalk, you can create an RDS DB Instance and attach it to an existing environment. For instructions using the AWS Elastic Beanstalk console, see Configuring Databases with AWS Elastic Beanstalk (p. 177).

2. Download and install MySQL Connector/J for your development environment. For download and installation instructions, go to http://dev.mysql.com/downloads/connector/j.
3. Establish a database connection in your Java code using your Amazon RDS DB Instance's connectivity information. You can access your connectivity information using environment variables. The following is an example code snippet.

   ```
   String dbName = System.getProperty("RDS_DB_NAME");
   String userName = System.getProperty("RDS_USER");
   String password = System.getProperty("RDS_PASSWORD");
   String hostname = System.getProperty("RDS_HOSTNAME");
   String port = System.getProperty("RDS_PORT");
   ```

   Build your JDBC connection string:

   ```
   String jdbcUrl = "jdbc:mysql://" + hostname + ":" + port + "/" + dbName +
   "?user=" + userName + "&password=" + password;
   ```

   Your connection string should look similar to this:

   ```
   jdbc:mysql://mydbinstance.abcdefghijkl.us-east-1.rds.amazonaws.com:3306/em
   ployees?user=sa&password=mypassword
   ```

   For more information on getting started using the MySQL Connector/J to access your MySQL database, go to http://dev.mysql.com/doc/refman/5.0/en/connector-j.html.
4. Copy the MySQL Connector/J JAR file into your AWS Elastic Beanstalk application's WEB-INF/lib directory.
5. Deploy your application to AWS Elastic Beanstalk. For information on how to deploy your application using AWS Elastic Beanstalk and the AWS Management Console, see Getting Started Using AWS Elastic Beanstalk (p. 5). For information on how to deploy your application using Eclipse, see Creating and Deploying AWS Elastic Beanstalk Applications in Java Using AWS Toolkit for Eclipse (p. 22).

Using an Existing Amazon RDS DB Instance with Java

Amazon Relational Database Service (Amazon RDS) lets you quickly and easily provision and maintain a MySQL Server instance in the cloud. This topic discusses how you can use Amazon RDS and the MySQL Connector/J with your AWS Elastic Beanstalk application.

To use Amazon RDS from your AWS Elastic Beanstalk application, you need to do the following:

- Create an Amazon RDS DB Instance.
- Configure your Amazon RDS DB security group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application.
- Create a JDBC connection string using your Amazon RDS DB Instance's public DNS name and configure your AWS Elastic Beanstalk environment to pass the string to your AWS Elastic Beanstalk application as an environment variable.
- Download and install MySQL Connector/J.
- Retrieve the JDBC connection string from the environment property passed to your server instance from AWS Elastic Beanstalk and use MySQL Connector/J to access your Amazon RDS database.

To use Amazon RDS with MySQL Connector/J from your AWS Elastic Beanstalk application

1. Create an Amazon RDS DB Instance. For instructions on how to do this, go to the Amazon Relational Database Service Getting Started Guide.
2. Configure your Amazon RDS DB security group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application. For instructions on how to find the name of your EC2 security group using the AWS Management Console, see Amazon EC2 Security Groups (p. 161). For more information, go to the "Authorizing Network Access to an Amazon EC2 Security Group" section of Working with DB Security Groups in the *Amazon Relational Database Service User Guide*.
3. Download and install MySQL Connector/J for your development environment. For download and installation instructions, go to http://dev.mysql.com/downloads/connector/j.
4. Create a JDBC connection string using your Amazon RDS DB Instance's public DNS name, port number, and (optionally) database name and login credentials. The following example shows a JDBC connection string that would connect to the employees database on an RDS instance at mydbinstance.abcdefghijkl.us-east-1.rds.amazonaws.com using port 3306, with the user name "sa" and the password "mypassword".

```
jdbc:mysql://mydbinstance.abcdefghijkl.us-east-1.rds.amazonaws.com:3306/em
ployees?user=sa&password=mypassword
```

5. Configure your AWS Elastic Beanstalk environment to pass the string to your AWS Elastic Beanstalk application as an environment property. For instructions on how to do this, go to Using Custom Environment Properties with AWS Elastic Beanstalk (p. 30).
6. Retrieve the JDBC connection string from the environment property passed to your server instance from AWS Elastic Beanstalk and use MySQL Connector/J to access your Amazon RDS database. The following code example shows how to retrieve the JDBC_CONNECTION_STRING custom environment property from a Java Server Page (JSP).

```
<p>
    The JDBC_CONNECTION_STRING environment variable is:
    <%= System.getProperty("JDBC_CONNECTION_STRING") %>
</p>
```

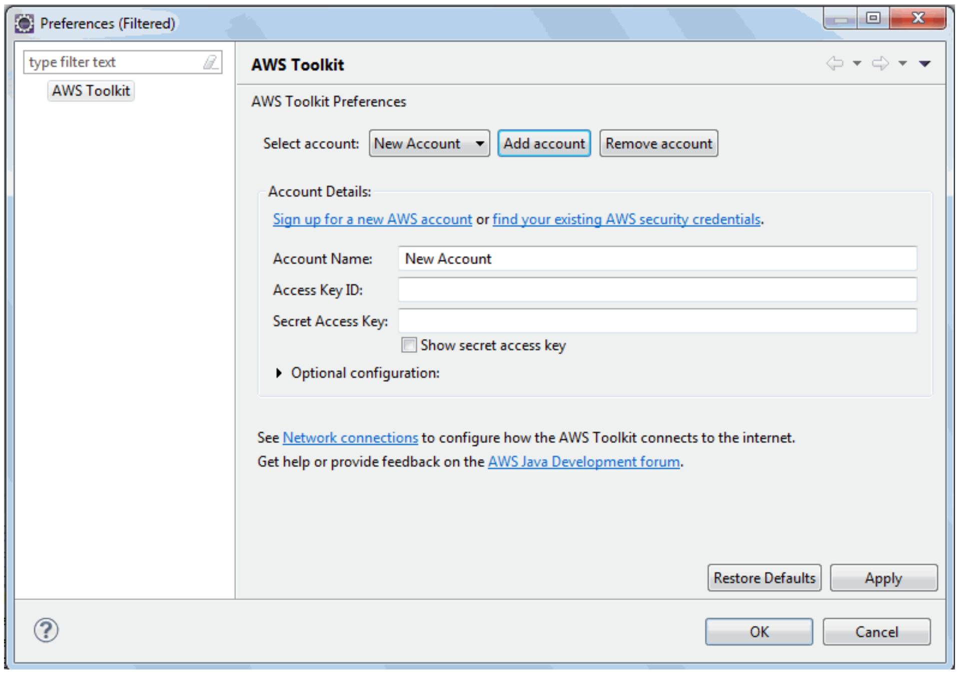
   For more information on getting started using the MySQL Connector/J to access your MySQL database, go to http://dev.mysql.com/doc/refman/5.0/en/connector-j.html.
7. Copy the MySQL Connector/J JAR file into your AWS Elastic Beanstalk application's WEB-INF/lib directory.
8. Deploy your application to AWS Elastic Beanstalk. For information on how to deploy your application using AWS Elastic Beanstalk and the AWS Management Console, see Getting Started Using AWS Elastic Beanstalk (p. 5). For information on how to deploy your application using Eclipse, go to Getting Started with AWS Elastic Beanstalk Deployment in Eclipse.

Managing Multiple AWS Accounts

You might want to set up different AWS accounts to perform different tasks, such as testing, staging, and production. You can use the AWS Toolkit for Eclipse to add, edit, and delete accounts easily.

To add an AWS account with the AWS Toolkit for Eclipse

1. On the AWS toolbar in Eclipse, click **Preferences**.
2. Click **Add account**.



3. In the **Account Name** text box, type the display name for the account.
4. In the **Access Key ID** text box, type your AWS access key ID.
5. In the **Secret Access Key** text box, type your AWS secret key.
6. Click **OK**.

To use a different account to deploy an application to AWS Elastic Beanstalk

1. On the AWS toolbar in Eclipse, click **Preferences**.
2. Click the **Select** account arrow, and then select the account you want to use to deploy applications to AWS Elastic Beanstalk.
3. Click **OK**.
4. In **Project Explorer**, right-click the application you want to deploy, and then select **Deploy to AWS Elastic Beanstalk**.

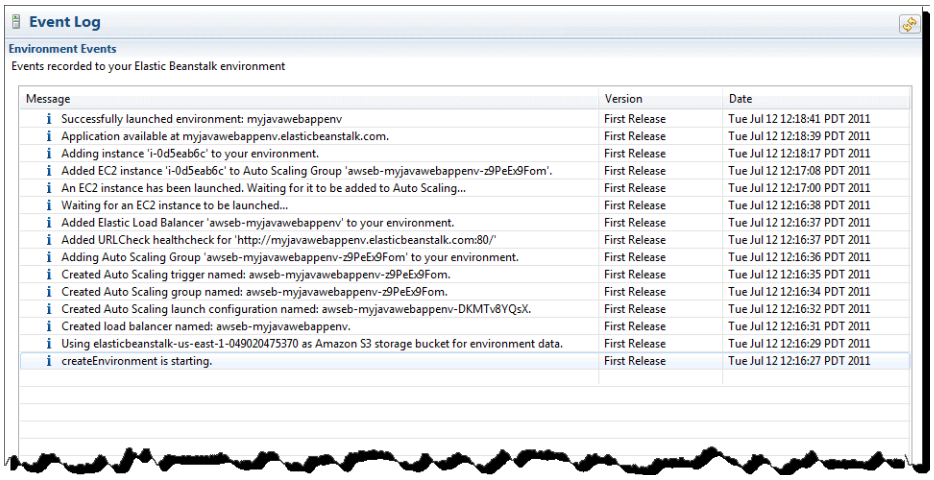
Viewing Events

You can use the AWS Toolkit for Eclipse to access events and notifications associated with your application. For more details on the most common events, see Understanding Environment Launch Events (p. 216).

To view application events

1. In the AWS Toolkit for Eclipse, expand the AWS Elastic Beanstalk node and your application node.

2. In the AWS Explorer, double-click your AWS Elastic Beanstalk environment.
3. At the bottom of the **Overview** pane, click the **Events** tab.
   A list of the events for all environments for your application is displayed.



Event Log

Environment Events

Events recorded to your Elastic Beanstalk environment

Message	Version	Date
Successfully launched environment: myjavawebappenv	First Release	Tue Jul 12 12:18:41 PDT 2011
Application available at myjavawebappenv.elasticbeanstalk.com.	First Release	Tue Jul 12 12:18:39 PDT 2011
Adding instance 'i-0d5eab6c' to your environment.	First Release	Tue Jul 12 12:18:17 PDT 2011
Added EC2 instance 'i-0d5eab6c' to Auto Scaling Group 'awseb-myjavawebappenv-z9PeEx9Fom'.	First Release	Tue Jul 12 12:17:08 PDT 2011
An EC2 instance has been launched. Waiting for it to be added to Auto Scaling...	First Release	Tue Jul 12 12:17:00 PDT 2011
Waiting for an EC2 instance to be launched...	First Release	Tue Jul 12 12:16:38 PDT 2011
Added Elastic Load Balancer 'awseb-myjavawebappenv' to your environment.	First Release	Tue Jul 12 12:16:37 PDT 2011
Added URLCheck healthcheck for 'http://myjavawebappenv.elasticbeanstalk.com:80/'	First Release	Tue Jul 12 12:16:37 PDT 2011
Adding Auto Scaling Group 'awseb-myjavawebappenv-z9PeEx9Fom' to your environment.	First Release	Tue Jul 12 12:16:36 PDT 2011
Created Auto Scaling trigger named: awseb-myjavawebappenv-z9PeEx9Fom.	First Release	Tue Jul 12 12:16:35 PDT 2011
Created Auto Scaling group named: awseb-myjavawebappenv-z9PeEx9Fom.	First Release	Tue Jul 12 12:16:34 PDT 2011
Created Auto Scaling launch configuration named: awseb-myjavawebappenv-DKMTv8YQsX.	First Release	Tue Jul 12 12:16:32 PDT 2011
Created load balancer named: awseb-myjavawebappenv.	First Release	Tue Jul 12 12:16:31 PDT 2011
Using elasticbeanstalk-us-east-1-049020475370 as Amazon S3 storage bucket for environment data.	First Release	Tue Jul 12 12:16:29 PDT 2011
createEnvironment is starting.	First Release	Tue Jul 12 12:16:27 PDT 2011

Managing AWS Elastic Beanstalk Application Environments

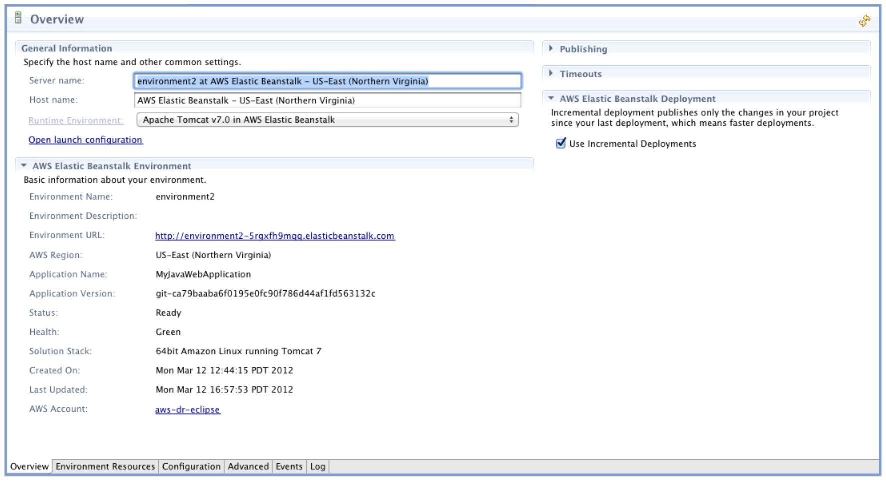
With the AWS Toolkit for Eclipse and the AWS Management Console, you can change the provisioning and configuration of the AWS resources that are used by your application environments. For information on how to manage your application environments using the AWS Management Console, see Managing Environments (p. 157). This section discusses the specific service settings you can edit in the AWS Toolkit for Eclipse as part of your application environment configuration.

Changing Environment Configuration Settings

When you deploy your application, AWS Elastic Beanstalk configures a number of AWS cloud computing services. You can control how these individual services are configured using the AWS Toolkit for Eclipse.

To edit an application's environment settings

1. In the AWS Toolkit for Eclipse, expand the AWS Elastic Beanstalk node and your application node.
2. In the AWS Explorer, double-click your AWS Elastic Beanstalk environment.



General Information
Specify the host name and other common settings.

Server name:	environment2 at AWS Elastic Beanstalk - US-East (Northern Virginia)
Host name:	AWS Elastic Beanstalk - US-East (Northern Virginia)
Runtime Environment:	Apache Tomcat v7.0 in AWS Elastic Beanstalk

Open launch configuration

AWS Elastic Beanstalk Environment
Basic information about your environment.

Environment Name:	environment2
Environment Description:	
Environment URL:	http://environment2-5rqxfh9mqq.elasticbeanstalk.com
AWS Region:	US-East (Northern Virginia)
Application Name:	MyJavaWebApplication
Application Version:	git-ca79baaba6f0195e0fc90f786d44af1fd563132c
Status:	Ready
Health:	Green
Solution Stack:	64bit Amazon Linux running Tomcat 7
Created On:	Mon Mar 12 12:44:15 PDT 2012
Last Updated:	Mon Mar 12 16:57:53 PDT 2012
AWS Account:	aws-dr-eclipse

AWS Elastic Beanstalk Deployment
Incremental deployment publishes only the changes in your project since your last deployment, which means faster deployments.

[✓] Use Incremental Deployments

3. At the bottom of the **Overview** pane, click the **Configuration** tab.

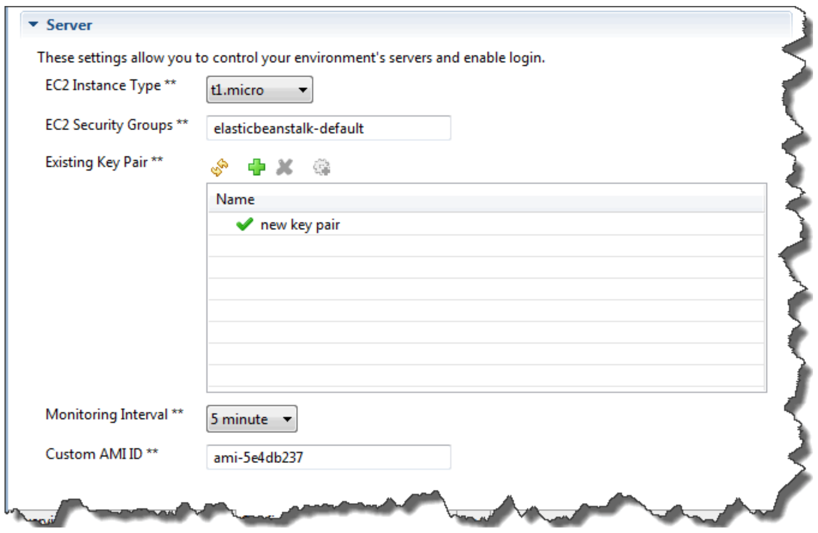
   You can now configure settings for the following:

   - EC2 Server Instances
   - Load balancer
   - Auto Scaling
   - Notifications
   - Environment properties

Configuring EC2 Server Instances Using AWS Toolkit for Eclipse

Amazon Elastic Compute Cloud (EC2) is a web service that you use to launch and manage server instances in Amazon's data centers. You can use Amazon EC2 server instances at any time, for as long as you need, and for any legal purpose. Instances are available in different sizes and configurations. For more information, go to the Amazon EC2 product page.

Under **Server**, on the **Configuration** tab inside the Toolkit for Eclipse, you can edit the AWS Elastic Beanstalk environment's Amazon EC2 instance configuration.



These settings allow you to control your environment's servers and enable login.

EC2 Instance Type ** t1.micro

EC2 Security Groups ** elasticbeanstalk-default

Existing Key Pair **

Name
✓ new key pair

Monitoring Interval ** 5 minute

Custom AMI ID ** ami-5e4db237

Amazon EC2 Instance Types

The **EC2 Instance Type** drop-down list box displays the instance types available to your AWS Elastic Beanstalk application. Changing the instance type enables you to select a server with the characteristics (including memory size and CPU power) that are most appropriate to your application. For instance, applications with intensive and long-running operations may require more CPU or memory. AWS Elastic Beanstalk regularly checks your running instances to ensure they are healthy. If your application consumes 95 percent or greater of the CPU, AWS Elastic Beanstalk will fire an event. For more information about this event, see CPU Utilization Greater Than 95.00% (p. 216).

Note

You cannot change between 32-bit and 64-bit instance types. For example, if your application is built on a 32-bit platform, only 32-bit instance types appear in the list.

For more information about the Amazon EC2 instance types available for your AWS Elastic Beanstalk application, go to Instance Families and Types in the *Amazon Elastic Compute Cloud User Guide*.

Amazon EC2 Security Groups

You can control access to your AWS Elastic Beanstalk application using an *Amazon EC2 Security Group*. A security group defines firewall rules for your instances. These rules specify which ingress (i.e., incoming) network traffic should be delivered to your instance. All other ingress traffic will be discarded. You can modify rules for a group at any time. The new rules are automatically enforced for all running instances and instances launched in the future.

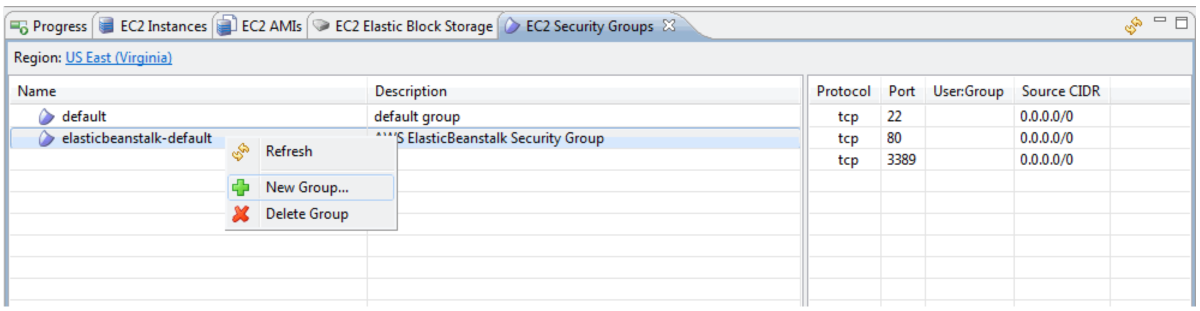
You can set up your Amazon EC2 security groups using the AWS Management Console or by using the AWS Toolkit for Eclipse. You can specify which Amazon EC2 Security Groups control access to your AWS Elastic Beanstalk application by entering the names of one or more Amazon EC2 security group names (delimited by commas) into the **EC2 Security Groups** text box.

Note

If you are running your application using a legacy container type, make sure port 80 (HTTP) is accessible from 0.0.0.0/0 as the source CIDR range if you want to enable health checks for your application. For more information about health checks, see Health Checks (p. 43). To check if you are using a legacy container type, see Why are some container types marked legacy? (p. 204).

To create a security group using the AWS Toolkit for Eclipse

1. In the AWS Toolkit for Eclipse, click the **AWS Explorer** tab. Expand the **Amazon EC2** node, and then double-click **Security Groups**.
2. Right-click anywhere in the left table, and then click**New Group**.



Region: US East (Virginia)

Name	Description
default	default group
elasticbeanstalk-default	AWS ElasticBeanstalk Security Group

Protocol	Port	User:Group	Source CIDR
tcp	22		0.0.0.0/0
tcp	80		0.0.0.0/0
tcp	3389		0.0.0.0/0

3. In the **Security Group** dialog box, type the security group name and description and then click **OK**.

For more information on Amazon EC2 Security Groups, see Using Security Groups in the *Amazon Elastic Compute Cloud User Guide*.

Amazon EC2 Key Pairs

You can securely log in to the Amazon EC2 instances provisioned for your AWS Elastic Beanstalk application with an Amazon EC2 key pair.

Important

You must create an Amazon EC2 key pair and configure your AWS Elastic Beanstalk-provisioned Amazon EC2 instances to use the Amazon EC2 key pair before you can access your AWS Elastic Beanstalk-provisioned Amazon EC2 instances. You can create your key pair using the **Publish to Beanstalk Wizard** inside AWS Toolkit for Eclipse when you deploy your application to AWS Elastic Beanstalk. Alternatively, you can set up your Amazon EC2 key pairs using the AWS Management Console. For instructions on creating a key pair for Amazon EC2, see the Amazon Elastic Compute Cloud Getting Started Guide.

For more information on Amazon EC2 key pairs, go to Using Amazon EC2 Credentials in the *Amazon Elastic Compute Cloud User Guide*. For more information on connecting to Amazon EC2 instances, go to Connecting to Instances and Connecting to a Linux/UNIX Instance from Windows using PuTTY in the *Amazon Elastic Compute Cloud User Guide*.

By default, only basic Amazon CloudWatch metrics are enabled; they return data in five-minute periods. You can enable more granular one-minute CloudWatch metrics by selecting **`1 minute`** in the **Monitoring Interval** drop-down list box.

Note

Amazon CloudWatch service charges can apply for one-minute interval metrics. See the Amazon CloudWatch product page for more information.

Custom AMI ID

You can override the default AMI used for your Amazon EC2 instances with your own custom AMI by entering the identifier of your custom AMI into the **Custom AMI ID** text box.

Important

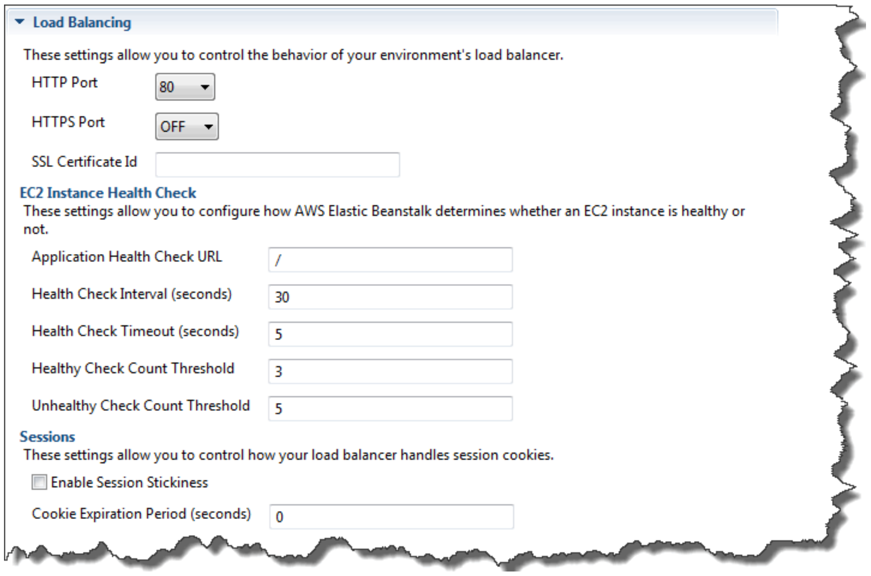
Using your own AMI is an advanced use case and should be done with care. If you need a custom AMI, we recommend you start with the default AWS Elastic Beanstalk AMI, and then modify it. To be considered healthy, AWS Elastic Beanstalk expects Amazon EC2 instances to meet a set of requirements, including having a running host manager. If these requirements are not met, your environment might not work properly.

Configuring Elastic Load Balancing Using AWS Toolkit for Eclipse

Elastic Load Balancing is an Amazon web service that improves the availability and scalability of your application. With Elastic Load Balancing, you can distribute application loads between two or more Amazon EC2 instances. Elastic Load Balancing improves availability through redundancy, and it supports traffic growth for your application.

Elastic Load Balancing automatically distributes and balances incoming application traffic among all the EC2 server instances you are running. The service also makes it easy to add new instances when you need to increase the capacity of your application.

AWS Elastic Beanstalk automatically provisions Elastic Load Balancing when you deploy an application. Under **Load Balacing**, on the **Configuration** tab inside the Toolkit for Eclipse, you can edit the AWS Elastic Beanstalk environment's load balancing configuration.


Load Balancing

These settings allow you to control the behavior of your environment's load balancer.

HTTP Port	80
HTTPS Port	OFF
SSL Certificate Id	

EC2 Instance Health Check

These settings allow you to configure how AWS Elastic Beanstalk determines whether an EC2 instance is healthy or not.

Application Health Check URL	/
Health Check Interval (seconds)	30
Health Check Timeout (seconds)	5
Healthy Check Count Threshold	3
Unhealthy Check Count Threshold	5

Sessions

These settings allow you to control how your load balancer handles session cookies.

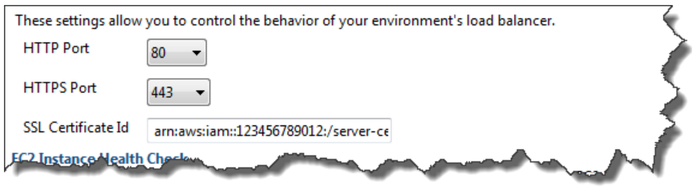
☐ Enable Session Stickiness

Cookie Expiration Period (seconds)	0

The following sections describe the Elastic Load Balancing parameters you can configure for your application.

Ports

The load balancer provisioned to handle requests for your AWS Elastic Beanstalk application sends requests to the Amazon EC2 instances that are running your application. The provisioned load balancer can listen for requests on HTTP and HTTPS ports and route requests to the Amazon EC2 instances in your AWS Elastic Beanstalk application. By default, the load balancer handles requests on the HTTP port. At least one of the ports (either HTTP or HTTPS) must be turned on.


These settings allow you to control the behavior of your environment's load balancer.

HTTP Port: 80

HTTPS Port: 443

SSL Certificate Id: arn:aws:iam::123456789012:/server-ce

EC2 Instance Health Check

Important

Make sure that the port you specified is not locked down; otherwise, users will not be able to connect to your AWS Elastic Beanstalk application.

Controlling the HTTP port

To turn off the HTTP port, you select OFF from the **HTTP Listener Port** drop-down list. To turn on the HTTP port, you select an HTTP port (for example, 80) from the drop-down list.

Note

If you want to access your environment using a different port other than the default port 80 (e.g., port 8080), you can add a listener to the existing load balancer and configure the new listener to listen on that port. For example, using the Elastic Load Balancing command line tools, type the following command replacing `<yourloadbalancername>` with the name of your load balancer for Elastic Beanstalk.

```
elb-create-lb-listeners --lb <yourloadbalancername> --listener "pro
tocol=http, lb-port=8080, instance-port=80"
```

If you want Elastic Beanstalk to monitor your environment, do not remove the listener on port 80.

Controlling the HTTPS port

Elastic Load Balancing supports the HTTPS/TLS protocol to enable traffic encryption for client connections to the load balancer. Connections from the load balancer to the EC2 instances are done using plaintext. By default, the HTTPS port is turned off.

To turn on the HTTPS port

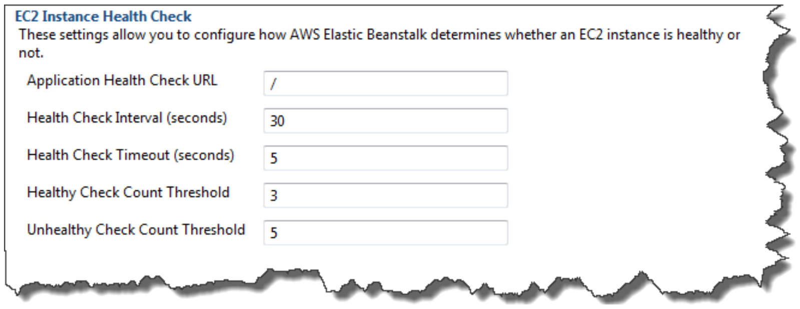
1. Create and upload a certificate and key to the AWS Identity and Access (IAM) service. The IAM service will store the certificate and provide an Amazon Resource Name (ARN) for the SSL certificate you've uploaded. For more information on creating and uploading certificates, see the Managing Server Certificates section of *Using AWS Identity and Access Management*.
2. Specify the HTTPS port by selecting a port from the **HTTPS Listener Port** drop-down list.
3. In the **SSL Certificate ID** text box, enter the Amazon Resources Name (ARN) of your SSL certificate (e.g., `arn:aws:iam::123456789012:server-certificate/abc/certs/build`). Use the SSL certificate that you created and uploaded in step 1. For information on viewing the certificate's ARN, see Verify the Certificate Object topic in the *Creating and Uploading Server Certificates* section of the *Using IAM Guide*.

To turn off the HTTPS port, select OFF from the **HTTPS Listener Port** drop-down list.

Health Checks

Elastic Load Balancing uses a *health check* to determine whether the Amazon EC2 instances running your application are healthy. The health check determines an instance's health status by probing a specified URL at a set interval; if the URL returns an error message, or fails to return within a specified timeout period, the health check fails. AWS Elastic Beanstalk uses Elastic Load Balancing health checks to set the status of your application.

You can control the settings for the health check using the **EC2 Instance Health Check** section of the **Load Balancing** panel.



EC2 Instance Health Check
These settings allow you to configure how AWS Elastic Beanstalk determines whether an EC2 instance is healthy or not.

Application Health Check URL	/
Health Check Interval (seconds)	30
Health Check Timeout (seconds)	5
Healthy Check Count Threshold	3
Unhealthy Check Count Threshold	5

The health check definition includes a URL to be queried for instance health. Override the default URL to match an existing resource in your application (e.g. '/myapp/index.jsp') by entering it in the **Application Health Check URL** text box.

The following list describes the health check parameters you can set for your application.

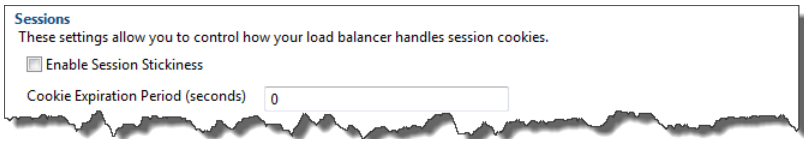
- In the **Health Check Interval (seconds)** text box, enter the number of seconds that define the interval at which Elastic Load Balancing will check the health of your application's Amazon EC2 instances.
- Specify the number of seconds for Elastic Load Balancing to wait for a response before it considers the instance non-responsive in the **Health Check Timeout (seconds)** text box.
- Specify the number of consecutive successful or unsuccessful URL probes before Elastic Load Balancing changes the instance health status in the **Healthy Check Count Threshold** and **Unhealthy Check Count Threshold** text boxes. For example, specifying 5 in the **Unhealthy Check Count Threshold** text box means that the URL would have to return an error message or timeout five consecutive times before Elastic Load Balancing considers the health check "failed."

Sessions

By default, a load balancer routes each request independently to the server instance with the smallest load. By comparison, a sticky session binds a user's session to a specific server instance so that all requests coming from the user during the session are sent to the same server instance.

AWS Elastic Beanstalk uses load balancer-generated HTTP cookies when sticky sessions are enabled for an application. The load balancer uses a special load balancer-generated cookie to track the application instance for each request. When the load balancer receives a request, it first checks to see if this cookie is present in the request. If so, the request is sent to the application instance specified in the cookie. If there is no cookie, the load balancer chooses an application instance based on the existing load balancing algorithm. A cookie is inserted into the response for binding subsequent requests from the same user to that application instance. The policy configuration defines a cookie expiry, which establishes the duration of validity for each cookie.

Under **Load Balancer**, in the **Sessions** section specify whether or not the load balancer for your application allows session stickiness.



Sessions

These settings allow you to control how your load balancer handles session cookies.

Enable Session Stickiness [ ]

Cookie Expiration Period (seconds)	0

For more information on Elastic Load Balancing, go to the Elastic Load Balancing Developer Guide.

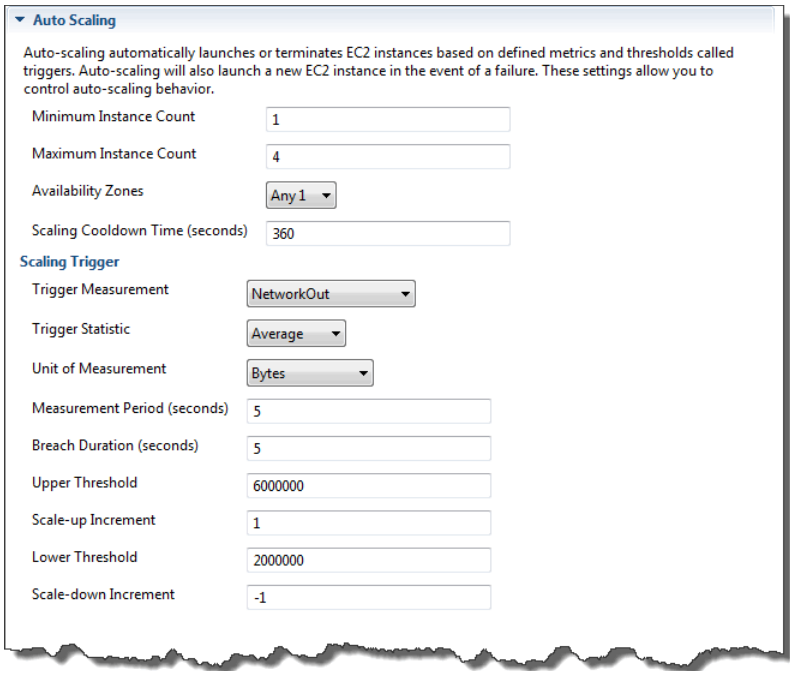
Configuring Auto Scaling Using AWS Toolkit for Eclipse

Auto Scaling is an Amazon web service designed to automatically launch or terminate Amazon EC2 instances based on user-defined triggers. Users can set up *Auto Scaling groups* and associate *triggers* with these groups to automatically scale computing resources based on metrics such as bandwidth usage or CPU utilization. Auto Scaling works with Amazon CloudWatch to retrieve metrics for the server instances running your application.

Auto Scaling lets you take a group of Amazon EC2 instances and set various parameters to have this group automatically increase or decrease in number. Auto Scaling can add or remove Amazon EC2 instances from that group to help you seamlessly deal with traffic changes to your application.

Auto Scaling also monitors the health of each Amazon EC2 instance that it launches. If any instance terminates unexpectedly, Auto Scaling detects the termination and launches a replacement instance. This capability enables you to maintain a fixed, desired number of Amazon EC2 instances automatically.

AWS Elastic Beanstalk provisions Auto Scaling for your application. Under **Auto Scaling**, on the **Configuration** tab inside the Toolkit for Eclipse, you can edit the AWS Elastic Beanstalk environment's Auto Scaling configuration.



Auto Scaling

Auto-scaling automatically launches or terminates EC2 instances based on defined metrics and thresholds called triggers. Auto-scaling will also launch a new EC2 instance in the event of a failure. These settings allow you to control auto-scaling behavior.

Minimum Instance Count	1
Maximum Instance Count	4
Availability Zones	Any 1
Scaling Cooldown Time (seconds)	360

Scaling Trigger

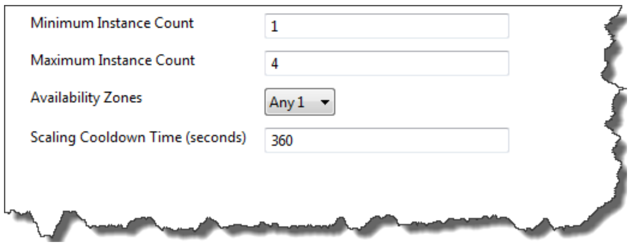
Trigger Measurement	NetworkOut
Trigger Statistic	Average
Unit of Measurement	Bytes
Measurement Period (seconds)	5
Breach Duration (seconds)	5
Upper Threshold	6000000
Scale-up Increment	1
Lower Threshold	2000000
Scale-down Increment	-1

The following section discusses how to configure Auto Scaling parameters for your application.

Launch Configuration

You can edit the launch configuration to control how your AWS Elastic Beanstalk application provisions Auto Scaling resources.

The **Minimum Instance Count** and **Maximum Instance Count** text boxes let you specify the minimum and maximum size of the Auto Scaling group that your AWS Elastic Beanstalk application uses.



Minimum Instance Count	1
Maximum Instance Count	4
Availability Zones	Any 1
Scaling Cooldown Time (seconds)	360

Note

To maintain a fixed number of Amazon EC2 instances, set the **Minimum Instance Count** and **Maximum Instance Count** text boxes to the same value.

The **Availability Zones** text box lets you specify the number of Availability Zones you want your Amazon EC2 instances to be in. It is important to set this number if you want to build fault-tolerant applications. If one Availability Zone goes down, your instances will still be running in your other Availability Zones.

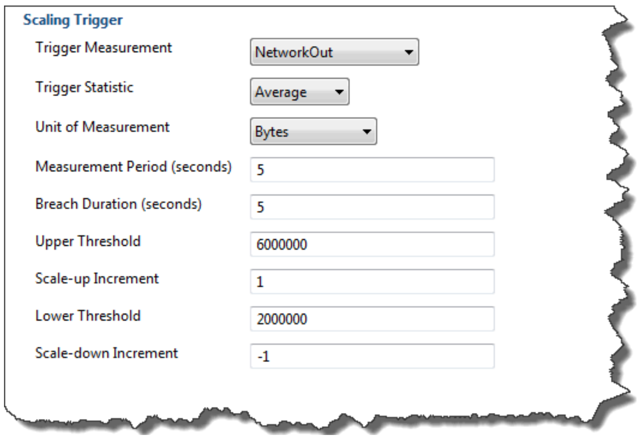
Note

Currently it is not possible to specify which Availability Zone your instance will be in.

Triggers

A *trigger* is an Auto Scaling mechanism that you set to tell the system when you want to increase (*scale out*) the number of instances, and when you want to decrease (*scale in*) the number of instances. You can configure triggers to *fire* on any metric published to Amazon CloudWatch, such as CPU utilization, and determine if the conditions you specified have been met. When the upper or lower thresholds of the conditions you have specified for the metric have been breached for the specified period of time, the trigger launches a long-running process called a *Scaling Activity*.

You can define a scaling trigger for your AWS Elastic Beanstalk application using the AWS Toolkit for Eclipse.



Scaling Trigger

Trigger Measurement	NetworkOut
Trigger Statistic	Average
Unit of Measurement	Bytes
Measurement Period (seconds)	5
Breach Duration (seconds)	5
Upper Threshold	6000000
Scale-up Increment	1
Lower Threshold	2000000
Scale-down Increment	-1

Auto Scaling triggers work by watching a specific Amazon CloudWatch metric for an instance. Triggers include CPU utilization, network traffic, and disk activity. Use the **Trigger Measurement** drop-down list box to select a metric for your trigger.

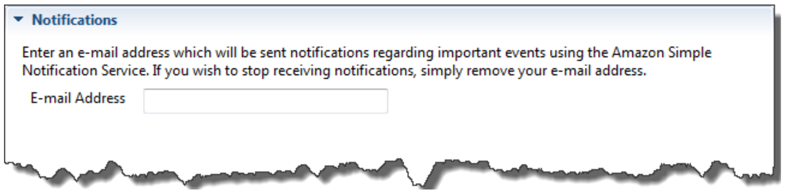
The following list describes the trigger parameters you can configure using the AWS Management Console.

- You can specify which statistic the trigger should use. You can select **`Minimum`**, **`Maximum`**, **`Sum`**, or **`Average`** using the **Trigger Statistic** drop-down list.
- Specify the unit for the trigger measurement using the **Unit of Measurement** drop-down list.
- The value in the **Measurement Period** text box specifies how frequently Amazon CloudWatch measures the metrics for your trigger. The **Breach Duration** is the amount of time a metric can be beyond its defined limit (as specified in the **Upper Threshold** and **Lower Threshold** text boxes) before the trigger fires.
- The **Upper Breach Scale Increment** and **Lower Breach Scale Increment** text boxes specify how many Amazon EC2 instances to add or remove when performing a scaling activity.

For more information on Auto Scaling, go to the Auto Scaling documentation.

Configuring Notifications Using AWS Toolkit for Eclipse

AWS Elastic Beanstalk uses the Amazon Simple Notification Service (Amazon SNS) to notify you of important events affecting your application. To enable Amazon SNS notifications, simply enter your email address in the **Email Address** text box under **Notifications** on the **Configuration** tab inside the Toolkit for Eclipse. To disable Amazon SNS notifications, remove your email address from the text box.


Notifications

Enter an e-mail address which will be sent notifications regarding important events using the Amazon Simple Notification Service. If you wish to stop receiving notifications, simply remove your e-mail address.

E-mail Address

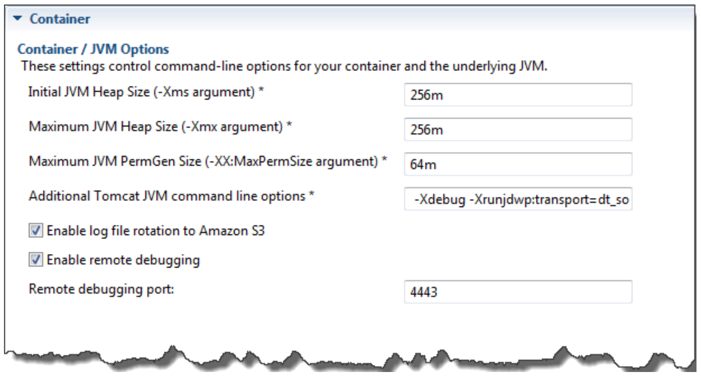
Configuring Java Containers Using AWS Toolkit for Eclipse

The **Container/JVM Options** panel lets you fine-tune the behavior of the Java Virtual Machine on your Amazon EC2 instances and enable or disable Amazon S3 log rotation. You can use the AWS Toolkit for Eclipse to configure your container information.

> **Note**
>
> You can modify your configuration settings with zero downtime by swapping the CNAME for your environments. For more information, see Deploying Versions with Zero Downtime (p. 143).

To access the Container/JVM Options panel for your AWS Elastic Beanstalk application

1. In the AWS Toolkit for Eclipse, expand the AWS Elastic Beanstalk node and your application node.
2. In the AWS Explorer, double-click your AWS Elastic Beanstalk environment.
3. At the bottom of the **Overview** pane, click the **Configuration** tab.
4. Under **Container**, you can configure container options.



These settings control command-line options for your container and the underlying JVM.

Initial JVM Heap Size (-Xms argument) *	256m
Maximum JVM Heap Size (-Xmx argument) *	256m
Maximum JVM PermGen Size (-XX:MaxPermSize argument) *	64m
Additional Tomcat JVM command line options *	-Xdebug -Xrunjdwp:transport=dt_so
✓ Enable log file rotation to Amazon S3	
✓ Enable remote debugging	
Remote debugging port:	4443

JVM Options

The heap size in the Java Virtual Machine affects how many objects can be created in memory before *garbage collection*—a process of managing your application's memory—occurs. You can specify an *initial heap size* and a *maximum heap size*. A larger initial heap size allows more objects to be created before garbage collection occurs, but it also means that the garbage collector will take longer to compact the heap. The maximum heap size specifies the maximum amount of memory the JVM can allocate when expanding the heap during heavy activity.

You can set the initial and the maximum JVM heap sizes using the **Initial JVM Heap Size (MB)** and **Maximum JVM Heap Size (MB)** text boxes. The available memory depends on the Amazon EC2 instance type. For more information about the Amazon EC2 instance types available for your AWS Elastic Beanstalk environment, see Instance Families and Types in the *Amazon EC2 User Guide*.

The *permanent generation* is a section of the JVM heap that is used to store class definitions and associated metadata. To modify the size of the permanent generation, enter the new size in the **Maximum JVM Permanent Generation Size (MB)** text box.

Full documentation of JVM is beyond the scope of this guide; for more information on JVM garbage collection, go to Tuning Garbage Collection with the 1.4.2 Java Virtual Machine at the Sun website.

Amazon S3 Log Rotation

AWS Elastic Beanstalk can copy the log files on an hourly basis for your application's Amazon EC2 instances to the Amazon S3 bucket associated with your application. To enable this feature, select **Enable log file rotation to Amazon S3**.

Remote Debugging

To test your application remotely, you can run your application in debug mode.

To enable this remote debugging

1. Select **Enable remote debugging**.
2. In the **Remote debugging** port text box, specify the port number that you will use for remote debugging.

   The **Additional Tomcat JVM command line options** text box will be automatically populated.

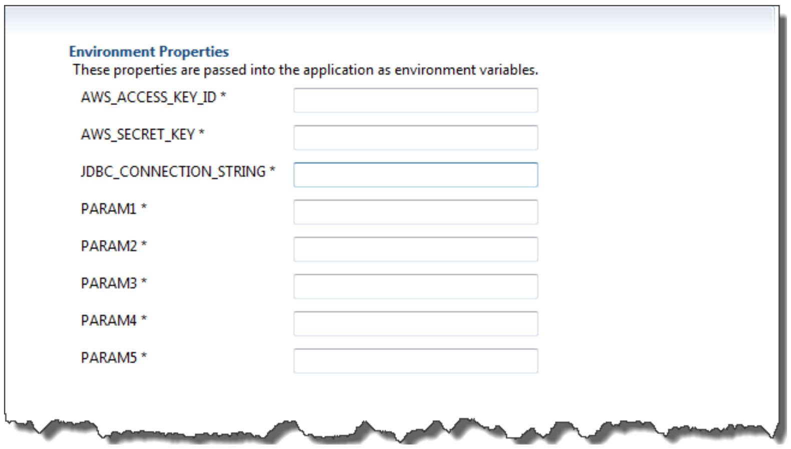
To start remote debugging

1. In the AWS Toolkit for Eclipse, on the **Window** menu, point to **Show View**, and then select **Other**.

2. Expand the **Server** folder and then click **Servers**.
3. Right-click the server your application is running on, and then click **Restart in Debug**.

Environment Properties

The **Environment Properties** section of the **Container** tab on the **Edit Configuration** window lets you specify environment variables (such as connection strings and database options) on the Amazon EC2 instances that are running your application. This setting enables greater portability by eliminating the need to recompile your source code as you move between environments. Environment properties are properties specific to your application environment and are not actual (shell) environment variables. More specifically, PARAM1, PARAM2, etc are system properties passed in to the JVM at startup using the -D flag. You can acquire them with System.getProperty(name). For more information on using and accessing custom environment properties, see Using Custom Environment Properties with AWS Elastic Beanstalk (p. 30)


Environment Properties
These properties are passed into the application as environment variables.

AWS_ACCESS_KEY_ID *	
AWS_SECRET_KEY *	
JDBC_CONNECTION_STRING *	
PARAM1 *	
PARAM2 *	
PARAM3 *	
PARAM4 *	
PARAM5 *	

You can use this section to configure the following environment properties:

- Specify AWS credentials using the **AWS_ACCESS_KEY_ID** and **AWS_SECRET_KEY** text boxes.
- Specify a connection string to an external database (such as Amazon RDS) by entering it in the **JDBC_CONNECTION_STRING** text box. For more information on how to set your JDBC_CONNECTION_STRING, see Using Custom Environment Properties with AWS Elastic Beanstalk (p. 30).
- Specify up to five additional environment variables by entering them in the **PARAM** text boxes.

Note

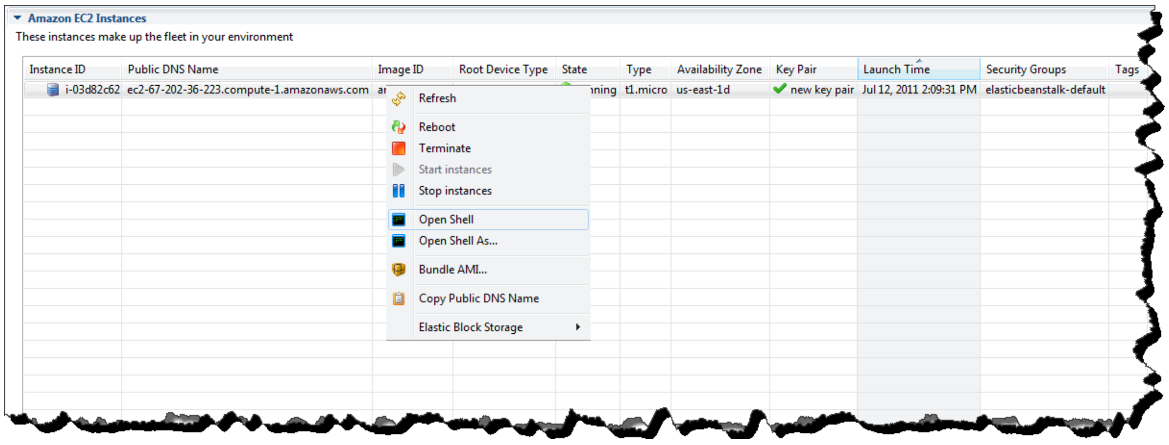
Environment properties can contain any printable ASCII character except the grave accent (`, ASCII 96) and cannot exceed 200 characters in length.

Listing and Connecting to Server Instances

You can view a list of Amazon EC2 instances running your AWS Elastic Beanstalk application environment through the AWS Toolkit for Eclipse or from the AWS Management Console. You can connect to these instances using Secure Shell (SSH). For information about listing and connecting to your server instances using the AWS Management Console, see Listing and Connecting to Server Instances (p. 197). The following section steps you through viewing and connecting you to your server instances using the AWS Toolkit for Eclipse.

To view and connect to Amazon EC2 instances for an environment

1. In the AWS Toolkit for Eclipse, click **AWS Explorer**. Expand the Amazon EC2 node, and then double-click **Instances**.
2. In the Amazon EC2 Instances window, in the **Instance ID** column, right-click the **Instance ID** for the Amazon EC2 instance running in your application's load balancer, and then click **Open Shell**.



Amazon EC2 Instances

These instances make up the fleet in your environment

Instance ID	Public DNS Name	Image ID	Root Device Type	State	Type	Availability Zone	Key Pair	Launch Time	Security Groups	Tags
i-03d82c62	ec2-67-202-36-223.compute-1.amazonaws.com	ar		nning	t1.micro	us-east-1d	✓ new key pair	Jul 12, 2011 2:09:31 PM	elasticbeanstalk-default	

Eclipse automatically opens the SSH client and makes the connection to the EC2 instance.

For more information on connecting to an Amazon EC2 instance, see the Amazon Elastic Compute Cloud Getting Started Guide.

Terminating an Environment

To avoid incurring charges for unused AWS resources, you can use the AWS Toolkit for Eclipse to terminate a running environment.

Note

You can always launch a new environment using the same version later.

To terminate an environment

1. In the AWS Toolkit for Eclipse, click the **AWS Explorer** tab. Expand the **AWS Elastic Beanstalk** node.
2. Expand the AWS Elastic Beanstalk application and right-click on the AWS Elastic Beanstalk environment.
3. Click **Terminate**. It will take a few minutes for AWS Elastic Beanstalk to terminate the AWS resources running in the environment.

Tools

Topics

- AWS Toolkit for Eclipse (p. 50)

The AWS SDK for Java provides a Java API for AWS infrastructure services, making it even easier for developers to build applications that tap into the cost-effective, scalable, and reliable AWS cloud. Using the SDK, developers can build solutions for Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), Amazon SimpleDB, and more. You can use the AWS Toolkit for Eclipse to add the AWS Java SDK to an existing project, or create a new Java project based on the AWS Java SDK.

AWS SDK for Java

With the AWS SDK for Java, you can get started in minutes with a single, downloadable package that includes the AWS Java library, code samples, and documentation. You can build Java applications on top of APIs to take the complexity out of coding directly against a web service interface. The library provides APIs that hide much of the lower-level plumbing, including authentication, request retries, and error handling. Practical examples are provided for how to use the library to build applications. For more information about the AWS SDK for Java, go to http://aws.amazon.com/sdkforjava/.

AWS Toolkit for Eclipse

The AWS Toolkit for Eclipse is an open source plug-in for the Eclipse Java IDE that makes it easier to develop and debug Java applications using Amazon Web Services, and now includes the AWS Elastic Beanstalk deployment feature. With the Elastic Beanstalk deployment feature, developers can use Eclipse to deploy Java web applications to AWS Elastic Beanstalk and within minutes access their applications running on AWS infrastructure services. The toolkit provides a Travel Log sample web application template that demonstrates the use of Amazon S3, Amazon SimpleDB, and Amazon SNS. For more information about the AWS Toolkit for Eclipse, go to http://aws.amazon.com/eclipse.

Resources

There are several places you can go to get additional help when developing your Java applications.

Resource	Description
The AWS Java Development Forum	Post your questions and get feedback.
Java Developer Center	One-stop shop for sample code, documentation, tools, and additional resources.
AWS SDK for Java FAQs	Get answers to commonly asked questions.

Creating and Deploying AWS Elastic Beanstalk Applications in .NET Using AWS Toolkit for Visual Studio

Topics

AWS Elastic Beanstalk for .NET makes it easier to deploy, manage, and scale your ASP.NET web applications that use Amazon Web Services. AWS Elastic Beanstalk for .NET is available to anyone who is developing or hosting a web application that uses IIS. The first part of this topic will give you step-by-step instructions for creating, testing, deploying, and redeploying your ASP.NET web application to AWS Elastic Beanstalk using the AWS Toolkit for Visual Studio. The second part explains how to manage and configure your applications and environments using the AWS Toolkit for Visual Studio. For more information on prerequisites, installation instructions and running code samples, go to http://aws.amazon.com/visualstudio/. This topic also provides useful information covering tools, how-to topics, and additional resources for ASP.NET developers. If you are not a developer, but want to easily manage and configure your AWS Elastic Beanstalk applications and environments, you can use the AWS Management Console. For information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Get Started

This topic walks you through how to quickly get started deploying an open source ASP.NET application, NerdDinner, using the AWS Toolkit for Visual Studio with Amazon Relational Database Service (Amazon RDS). NerdDinner is an application that helps people host lunches, dinners, and informal get-togethers. To deploy NerdDinner, you will do the following steps:

1. Step 1: Set Up the NerdDinner Application (p. 52)
2. Step 2: Launch an Amazon RDS DB Instance (p. 52)
3. Step 3: Set Up the NerdDinner Database (p. 54)
4. Step 4: Deploy to AWS Elastic Beanstalk (p. 56)

Video available

Step 1: Set Up the NerdDinner Application

In this step, you'll download and open NerdDinner inside Visual Studio, and then script the NerdDinner database to later recreate the database for Amazon RDS. For this example, we use NerdDinner 2.0.

To open NerdDinner project in Microsoft Visual Studio

1. Download NerdDinner from http://nerddinner.codeplex.com/, and extract the files to your local computer.
2. In Visual Studio, on the **File** menu, click **Open -> Project**. Navigate to the NerdDinner project, and click **Open**.

To script the NerdDinner database

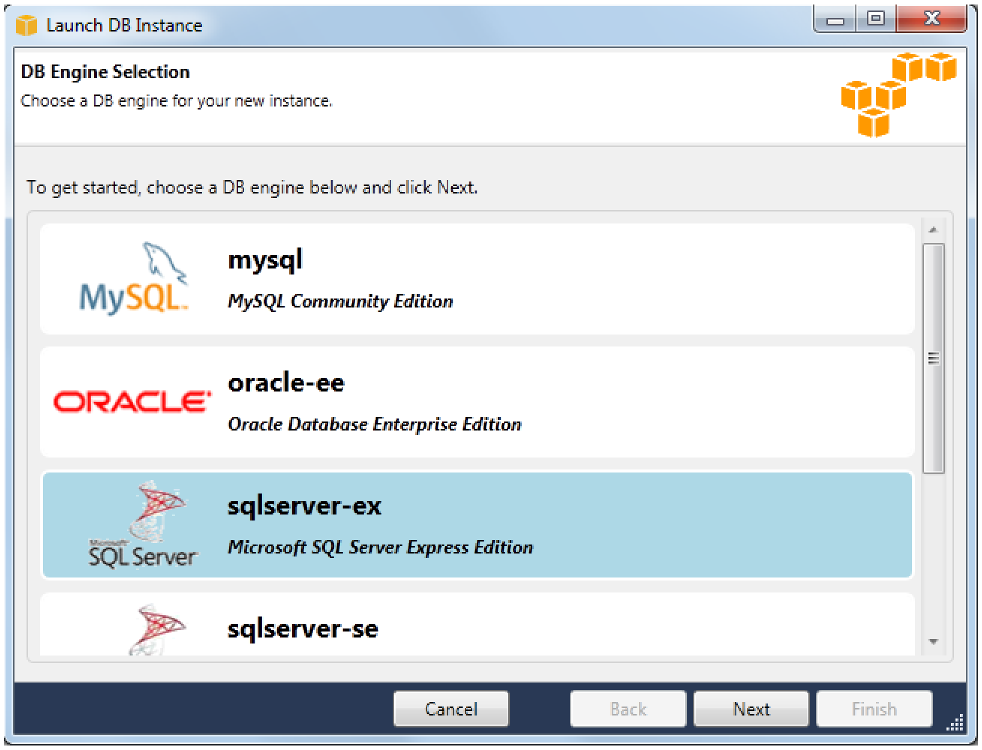
1. In **Server Explorer**, right-click **Data Connections**, and then click **Add Connection**.
2. In the **Add Connection** dialog box, click **Change**, and then click **Microsoft SQL Server Database File**. Click **OK**.
3. Click **Browse** next to the **Database file name** box, and select the NerdDinner.mdf file located in the NerdDinner/App_Data directory. Click **Open**.
4. In **Server Explorer**, right-click **NerdDinner.mdf**, and click **Publish to provider**.
5. In the **Database Publishing** wizard, click **Next** twice. On the **Select an Output Location** page, note the path and file name in the **File name** box. You will need this for later.
6. Click **Finish** twice to accept all defaults in the wizard.

Step 2: Launch an Amazon RDS DB Instance

Next, you'll want to launch your Amazon RDS database instance because it will take a few minutes to set up.

To launch an Amazon RDS database

1. In Visual Studio on the **View** menu, click **AWS Explorer**.
2. Expand **Amazon RDS**. Right-click **DB Instances**, and then click **Launch DB Instance**. The Launch DB Instance wizard opens.



Launch DB Instance

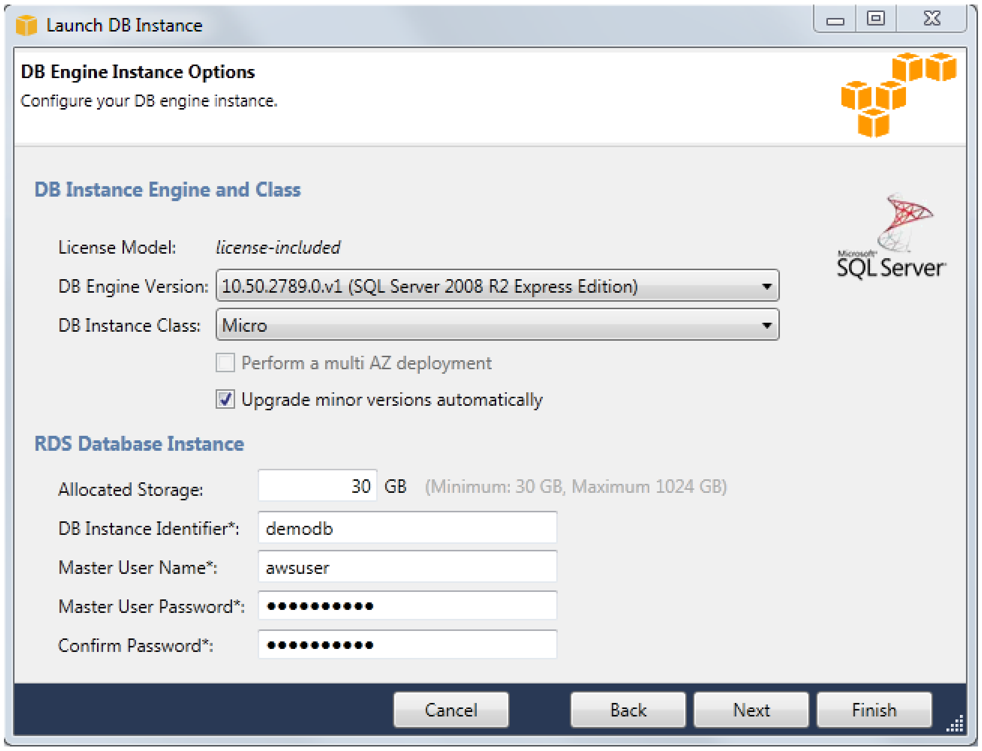
DB Engine Selection

Choose a DB engine for your new instance.

To get started, choose a DB engine below and click Next.

- **mysql** — *MySQL Community Edition*
- **oracle-ee** — *Oracle Database Enterprise Edition*
- **sqlserver-ex** — *Microsoft SQL Server Express Edition*
- **sqlserver-se**

3. On the **DB Engine Selection** page in the Launch DB Instance wizard, select a database engine. Microsoft SQL Server Express Edition is eligible for the Amazon RDS for SQL Server – Free Usage Tier. For more information, go to http://aws.amazon.com/rds/sqlserver/free. For this example, select **sqlserver-ex**, and then click **Next**. The **DB Engine Instance Options** page appears.



Launch DB Instance

DB Engine Instance Options

Configure your DB engine instance.

DB Instance Engine and Class

License Model: *license-included*

DB Engine Version: 10.50.2789.0.v1 (SQL Server 2008 R2 Express Edition)

DB Instance Class: Micro

☐ Perform a multi AZ deployment

☑ Upgrade minor versions automatically

RDS Database Instance

Allocated Storage: 30 GB (Minimum: 30 GB, Maximum 1024 GB)

DB Instance Identifier*: demodb

Master User Name*: awsuser

Master User Password*: ●●●●●●●●●●

Confirm Password*: ●●●●●●●●●●

4. On the **DB Engine Instance Options** page, do the following, and then click **Finish** to accept the default values in the rest of the wizard. Your database instance will appear under the DB Instances node in Visual Studio.

   - In the **DB Instance Class** list, click the type of database you want. The db.t1.micro is eligible for the Amazon RDS for SQL Server – Free Usage Tier when used with Microsoft SQL Server Express Edition. For more information, go to http://aws.amazon.com/rds/sqlserver/free.
   - Type **demodb** in the **DB Instance Identifier** text box. The DB Instance is a name for your DB Instance that is unique for your account in a Region.
   - Type a name for your master user in the **Master Username** text box. You use the master user name to log on to your DB Instance with all database privileges.
   - Type a password for your master user in the **Master User Password** text box.

Step 3: Set Up the NerdDinner Database

To set up the NerdDinner database, you'll need to do the following steps:

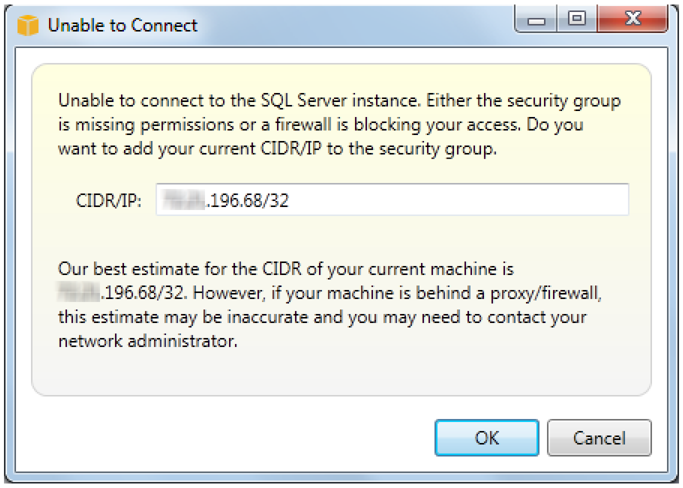
1. Create a SQL DB
2. Update your DB connection string
3. Connect to the database
4. Close the connection to the NerdDinner database

To create a SQL DB

1. When the Amazon RDS database is in the available state, right-click the DB instance, and then click **Create SQL DB**.

   Note

   To check if your database is in the available state, right-click the DB instance, and then click **View**. Check that the **Status** column says **available**.
2. The toolkit needs to add permissions to the Amazon RDS security group so that your computer can connect to the SQL database. In the **Unable to Connect** dialog, click **OK**.



Unable to Connect

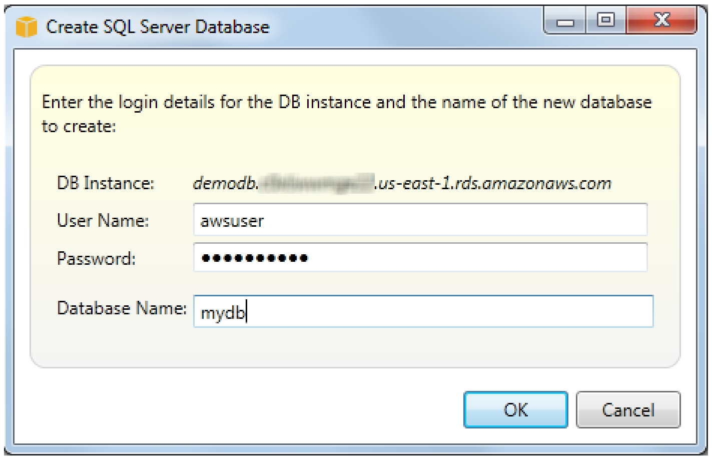
Unable to connect to the SQL Server instance. Either the security group is missing permissions or a firewall is blocking your access. Do you want to add your current CIDR/IP to the security group.

CIDR/IP: [redacted].196.68/32

Our best estimate for the CIDR of your current machine is [redacted].196.68/32. However, if your machine is behind a proxy/firewall, this estimate may be inaccurate and you may need to contact your network administrator.

OK Cancel

3. In the **Create SQL Server Database** dialog box, type the user name, password, and name for your database, and then click **OK**. The database now appears in Server Explorer.



Enter the login details for the DB instance and the name of the new database to create:

DB Instance: *demodb.[redacted].us-east-1.rds.amazonaws.com*

User Name: awsuser

Password: ●●●●●●●●●●●

Database Name: mydb

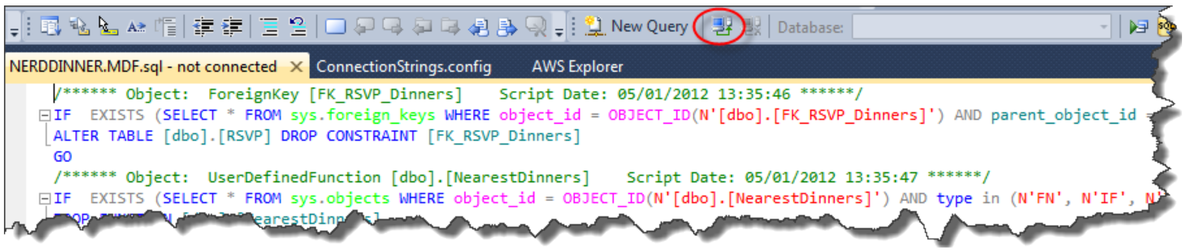
To update your connection string

1. In **Server Explorer**, click the Amazon RDS database, and copy the connection string from the **Properties** window.
2. In **Solution Explorer**, open the ConnectionStrings.config file.
3. For NerdDinnerEntities, update the embeded connection string by replacing the data source in the provider connection string with your Amazon RDS connection string as in the following example.

```
Data Source=<Your RDS endpoint>,1433;Initial Catalog=mydb;User
ID=awsuser;Password=***********
```

To connect to your database

1. In Visual Studio, open the file NerdDinner.mdf.mysql. Navigate to the path and file name you wrote down when you scripted the NerdDinner database in Step 1: Set Up the NerdDinner Application (p. 52).
2. On the Toolbar, click the **Connect** icon.

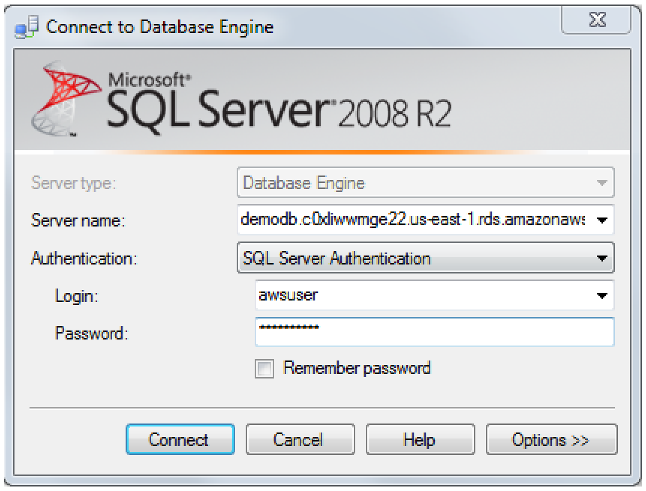


```
/****** Object:  ForeignKey [FK_RSVP_Dinners]    Script Date: 05/01/2012 13:35:46 ******/
IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = OBJECT_ID(N'[dbo].[FK_RSVP_Dinners]') AND parent_object_id =
ALTER TABLE [dbo].[RSVP] DROP CONSTRAINT [FK_RSVP_Dinners]
GO
/****** Object:  UserDefinedFunction [dbo].[NearestDinners]    Script Date: 05/01/2012 13:35:47 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[NearestDinners]') AND type in (N'FN', N'IF', N
```

3. In the **Connect to Database Engine** dialog box, in the **Server name** box, type or paste the Amazon RDS endpoint.

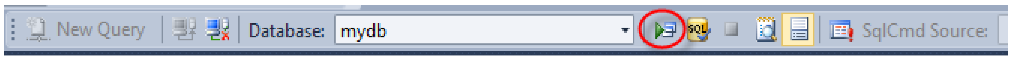
   Note

   To get the Amazon RDS endpoint, click the Amazon RDS DB instance, and copy the endpoint from the **Properties** window.



Server type: Database Engine
Server name: demodb.c0xliwwmge22.us-east-1.rds.amazonaws
Authentication: SQL Server Authentication
Login: awsuser
Password: ***********
Remember password

4. Click **Options**, and in the **Database name** box, type the name of your database. In this example, we used **mydb**. The name of the NerdDinner.MDF.sql tab will now contain the name of the Amazon RDS endpoint.
5. On the Toolbar, click the **Execute SQL** icon. You will see a message say "Command(s) completed successfully under the **Messages** tab.



6. To verify everything worked correctly, in **Server Explorer**, expand the Amazon RDS database, and then expand the **Tables** node. You should now see Dinners and RSVP.

Before you deploy to AWS Elastic Beanstalk, close the connection to the NerdDinner database.

To close the connection to the NerdDinner database

1. In Visual Studio, in Server Explorer, expand **Data Connections**. Right-click **NerdDinner.mdf**.
2. Click **Close Connection**.

Step 4: Deploy to AWS Elastic Beanstalk

After configuring your NerdDinner application to connect to your Amazon RDS database, you are ready to deploy your application to AWS Elastic Beanstalk.

To deploy your application to AWS Elastic Beanstalk using the AWS Toolkit for Visual Studio

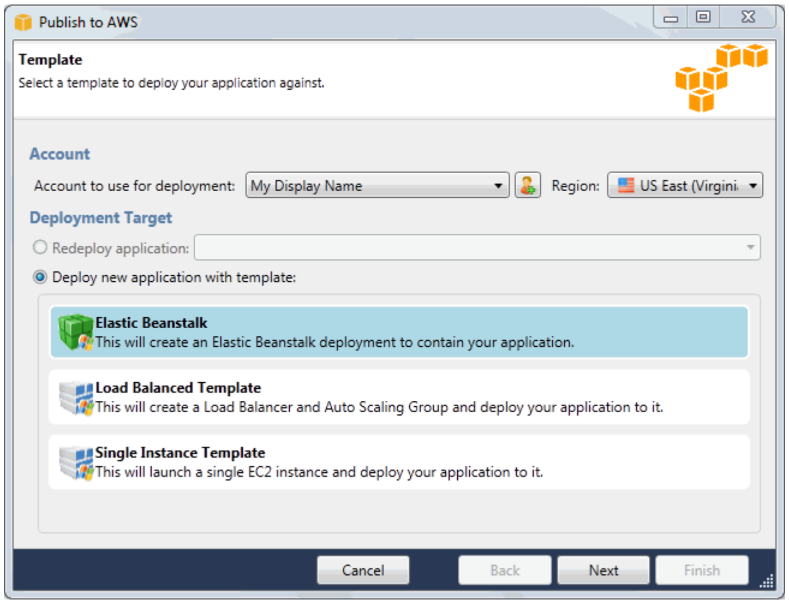
1. In **Solution Explorer**, right-click your application and then select **Publish to AWS**.
2. Select a template.

   a. In the **AWS account to use for deployment** list, click your account, or click the **Add another account** icon to enter new account information.

   b. In the **Region** list, click the region where you want to deploy your application. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference. If you select a region that is not supported by AWS Elastic Beanstalk, then the option to deploy to AWS Elastic Beanstalk will become unavailable.

   c. Select **Deploy new application with template** text box and click **Elastic Beanstalk**.

d. Click **Next**.



Template

Select a template to deploy your application against.

Account

Account to use for deployment: My Display Name

Region: US East (Virgini

Deployment Target

○ Redeploy application:

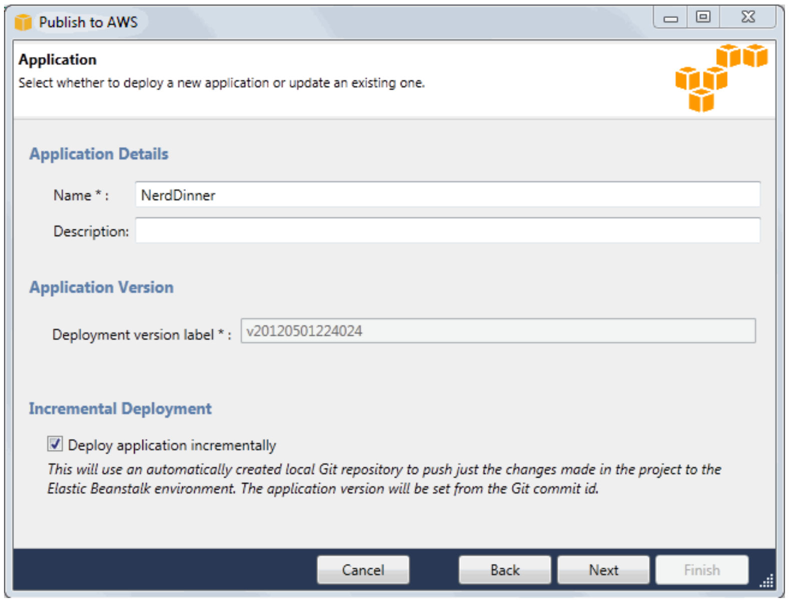
◉ Deploy new application with template:

Elastic Beanstalk
This will create an Elastic Beanstalk deployment to contain your application.

Load Balanced Template
This will create a Load Balancer and Auto Scaling Group and deploy your application to it.

Single Instance Template
This will launch a single EC2 instance and deploy your application to it.

3. In the **Name** box, type the name of the application, and then click **Next**.



Application

Select whether to deploy a new application or update an existing one.

Application Details

Name * : NerdDinner

Description:

Application Version

Deployment version label * : v20120501224024

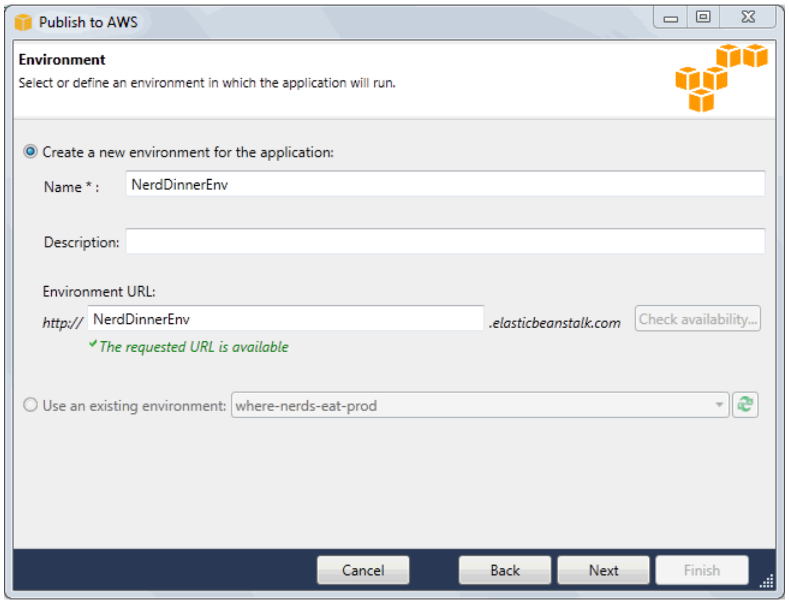
Incremental Deployment

☑ Deploy application incrementally

This will use an automatically created local Git repository to push just the changes made in the project to the Elastic Beanstalk environment. The application version will be set from the Git commit id.

4. On the **Environment** page, do the following:

   a. In the **Name** box, type a name for your environment.
   b. Click the **Check availability** button to make sure the environment URL is available.
   c. Click **Next** twice to accept the default settings and skip to the **Amazon RDS Database Security Group** page.



Environment
Select or define an environment in which the application will run.

Create a new environment for the application: [✓]

Name * : NerdDinnerEnv

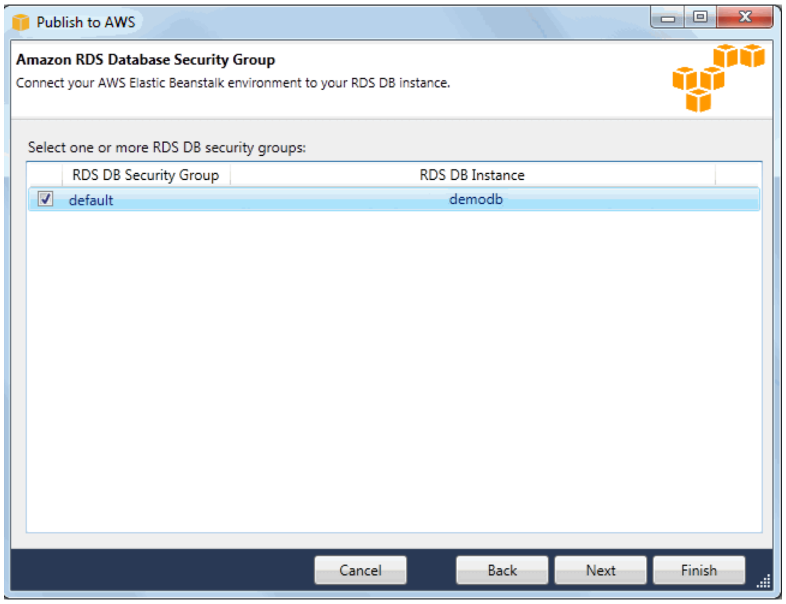
Description:

Environment URL:
http:// NerdDinnerEnv .elasticbeanstalk.com

✓ The requested URL is available

Use an existing environment: where-nerds-eat-prod

5. If you want to connect your AWS Elastic Beanstalk environment to your Amazon RDS database instance, select the RDS Database security group associated with your database instance. Click **Finish**, and then click **Deploy**. It will take a few minutes for Elastic Beanstalk to deploy your application.

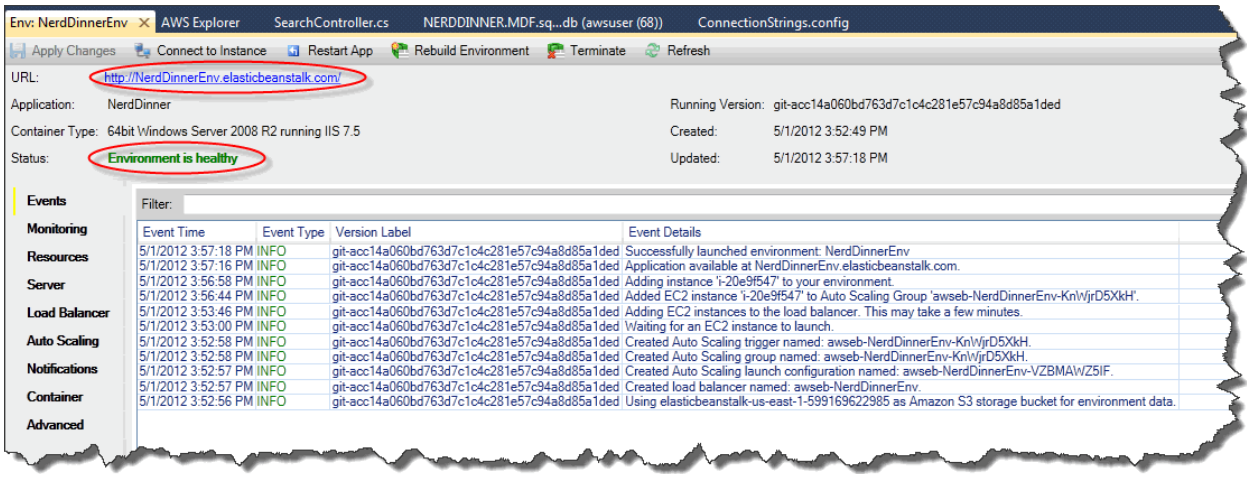


Amazon RDS Database Security Group
Connect your AWS Elastic Beanstalk environment to your RDS DB instance.

Select one or more RDS DB security groups:

	RDS DB Security Group	RDS DB Instance
✓	default	demodb

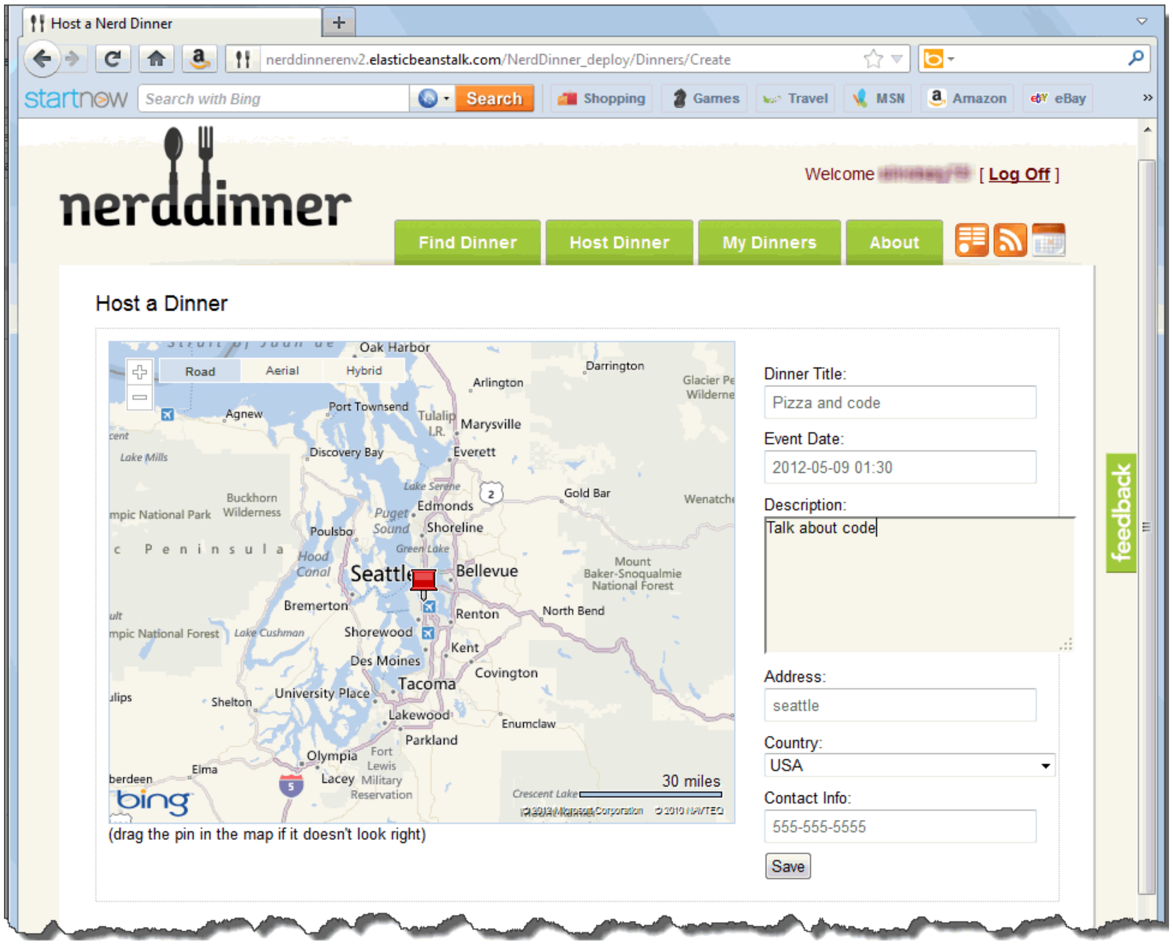
6. To check the status of your environment, in **AWS Explorer**, right-click your Elastic Beanstalk environment, and then click **View Status**. When your status says **Environment is healthy**, you can view your application.



URL: http://NerdDinnerEnv.elasticbeanstalk.com/
Application: NerdDinner
Container Type: 64bit Windows Server 2008 R2 running IIS 7.5
Status: Environment is healthy
Running Version: git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded
Created: 5/1/2012 3:52:49 PM
Updated: 5/1/2012 3:57:18 PM

Event Time	Event Type	Version Label	Event Details
5/1/2012 3:57:18 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Successfully launched environment: NerdDinnerEnv
5/1/2012 3:57:16 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Application available at NerdDinnerEnv.elasticbeanstalk.com.
5/1/2012 3:56:58 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Adding instance 'i-20e9f547' to your environment.
5/1/2012 3:56:44 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Added EC2 instance 'i-20e9f547' to Auto Scaling Group 'awseb-NerdDinnerEnv-KnWjrD5XkH'.
5/1/2012 3:53:46 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Adding EC2 instances to the load balancer. This may take a few minutes.
5/1/2012 3:53:00 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Waiting for an EC2 instance to launch.
5/1/2012 3:52:58 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Created Auto Scaling trigger named: awseb-NerdDinnerEnv-KnWjrD5XkH.
5/1/2012 3:52:58 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Created Auto Scaling group named: awseb-NerdDinnerEnv-KnWjrD5XkH.
5/1/2012 3:52:57 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Created Auto Scaling launch configuration named: awseb-NerdDinnerEnv-VZBMAWZ5IF.
5/1/2012 3:52:57 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Created load balancer named: awseb-NerdDinnerEnv.
5/1/2012 3:52:56 PM	INFO	git-acc14a060bd763d7c1c4c281e57c94a8d85a1ded	Using elasticbeanstalk-us-east-1-599169622985 as Amazon S3 storage bucket for environment data.

7. To view your application, in the Elastic Beanstalk environment tab, click the URL as shown in the previous diagram. Log in and host a dinner!



Welcome [redacted] [Log Off]

Host a Dinner

(drag the pin in the map if it doesn't look right)

Dinner Title: Pizza and code
Event Date: 2012-05-09 01:30
Description: Talk about code
Address: seattle
Country: USA
Contact Info: 555-555-5555
Save

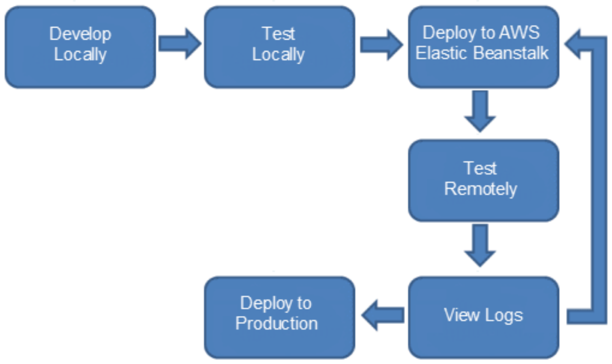
Note

By default, at least one Amazon EC2 instance is launched as part of your Auto Scaling group. If more than one instance is launched (e.g., due to an increase in traffic load), we recommend enabling session stickiness when deploying NerdDinner. Your load balancer makes a user's session "sticky" by binding it to a specific server instance so that all requests coming from the user during the session will be sent to the same server instance. For information about how to enable session stickiness, see Sessions (p. 167).

Develop, Test, and Deploy

Topics

The following diagram of a typical software development life cycle includes deploying your application to AWS Elastic Beanstalk.



After developing and testing your application locally, you will typically deploy your application to AWS Elastic Beanstalk. After deployment, your application will be live at a URL such as http://myexampleapp-wpams3yrvj.elasticbeanstalk.com. Because your application is live, you should consider setting up multiple environments, such as a testing environment and a production environment. You can use the AWS Toolkit for Visual Studio if you want to set up different AWS accounts for testing, staging, and production. For more information about managing multiple accounts, see Managing Multiple Accounts (p. 72).

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can point your domain name to the Amazon Route 53 CNAME <yourappname>.elasticbeanstalk.com. Contact your DNS provider to set this up. For information about how to map your root domain to your Elastic Load Balancer, see Using AWS Elastic Beanstalk with Amazon Route 53 to Map Your Root Domain to Your Load Balancer (p. 221).

After you remotely test and debug your AWS Elastic Beanstalk application, you can make any updates and then redeploy to AWS Elastic Beanstalk. After you are satisfied with all of your changes, you can upload the latest version to your production environment. The following sections explain each stage of the software development life cycle.

Create Project

Visual Studio provides templates for different programming languages and application types. You can start with any of these templates. The AWS Toolkit for Visual Studio also provides three project templates that bootstrap development of your application: AWS Console Project, AWS Web Project, and AWS Empty Project. For this example, you'll create a new ASP.NET Web Application.

To create a new ASP.NET Web Application project

1. In Visual Studio, on the **File** menu, click **New -> Project**.
2. In the **New Project** dialog box, click **Installed Templates**, click **Visual C#**, and then click **Web**. Click **ASP.NET Web Application**, type a project name, and then click **OK**.

To run a project

- Do one of the following:
  - Press **F5**.
  - Select **Start Debugging** from the **Debug** menu.

Test Locally

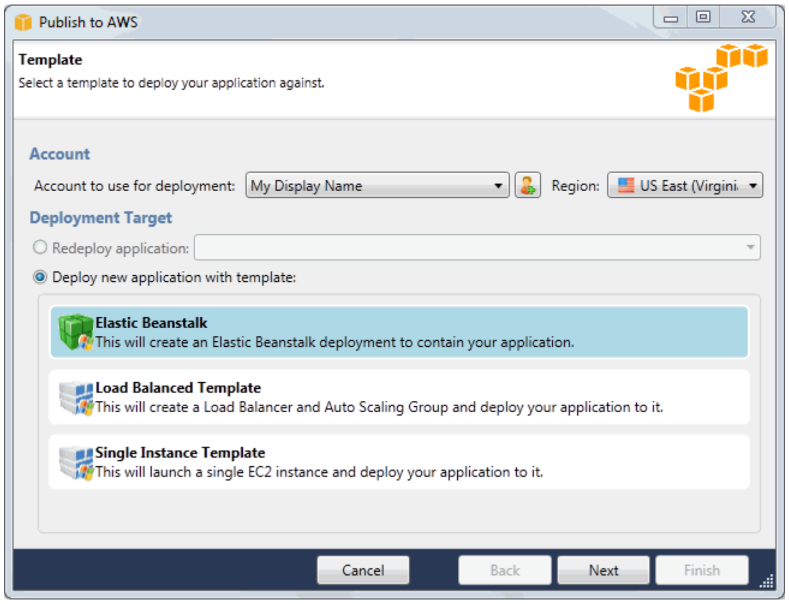
Visual Studio makes it easy for you to test your application locally. To test or run ASP.NET web applications, you need a web server. Visual Studio offers several options, such as Internet Information Services (IIS), IIS Express, or the built-in Visual Studio Development Server. To learn about each of these options and to decide which one is best for you, go to Web Servers in Visual Studio for ASP.NET Web Projects .

Deploy to AWS Elastic Beanstalk

After testing your application, you are ready to deploy it to AWS Elastic Beanstalk.

To deploy your application to AWS Elastic Beanstalk using the AWS Toolkit for Visual Studio

1. In **Solution Explorer**, right-click your application and then select **Publish to AWS**. The Publish to AWS wizard opens.



Publish to AWS

Template
Select a template to deploy your application against.

Account
Account to use for deployment: My Display Name
Region: US East (Virginia)

Deployment Target
( ) Redeploy application:
(●) Deploy new application with template:

- **Elastic Beanstalk** — This will create an Elastic Beanstalk deployment to contain your application.
- **Load Balanced Template** — This will create a Load Balancer and Auto Scaling Group and deploy your application to it.
- **Single Instance Template** — This will launch a single EC2 instance and deploy your application to it.

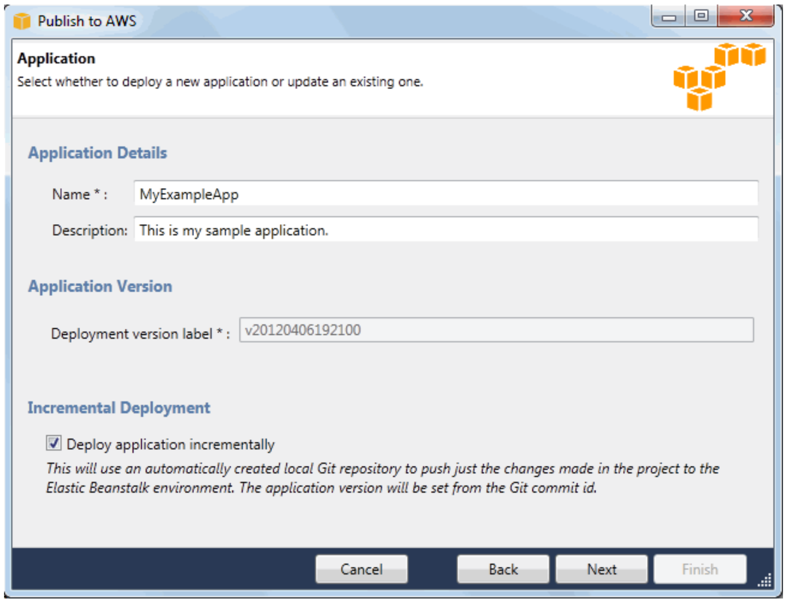
2. Enter your account information.

   a. In the **AWS account to use for deployment** list, click your account, or click **Other** to enter new account information.

   b. In the **Region** list, click the region where you want to deploy your application. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference. If you select a region that is not supported by AWS Elastic Beanstalk, then the option to deploy to AWS Elastic Beanstalk will become unavailable.

   c. Select **Deploy new application with template** text box and click **Elastic Beanstalk**.

   d. Click **Next**. The Application page appears.



Publish to AWS

Application
Select whether to deploy a new application or update an existing one.

Application Details
Name * : MyExampleApp
Description: This is my sample application.

Application Version
Deployment version label * : v20120406192100

Incremental Deployment
[✓] Deploy application incrementally
This will use an automatically created local Git repository to push just the changes made in the project to the Elastic Beanstalk environment. The application version will be set from the Git commit id.

3. Enter your application details.

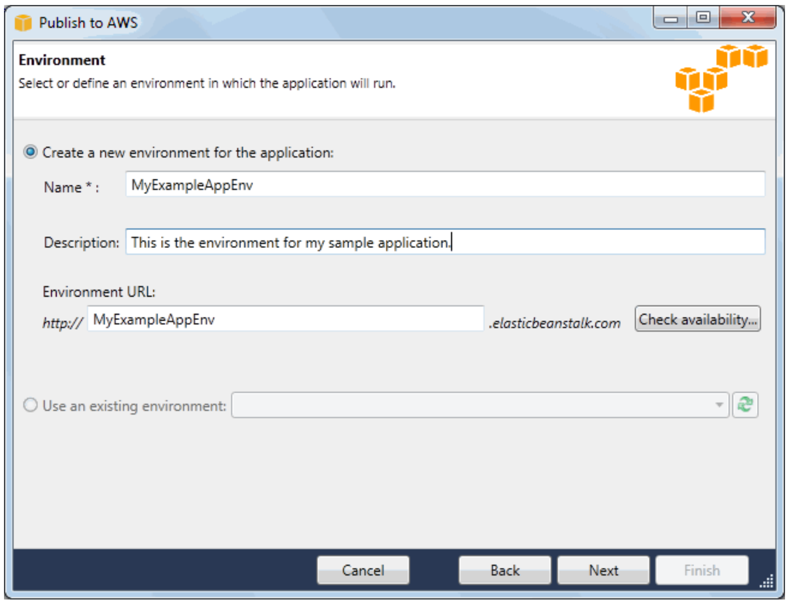
a. In the **Name** box, type the name of the application.

b. In the **Description** box, type a description of the application. This step is optional.

c. The version label of the application is automatically populated in the **Deployment version label** text box.

d. Select **Deploy application incrementally** to deploy only the changed files. An incremental deployment is faster because you are updating only the files that changed instead of all the files. If you choose this option, an application version will be set from the Git commit id. If you choose to not deploy your application incrementally, then you can update the version label in the **Deployment version label** box.

e. Click **Next**. The Environment page appears.


Publish to AWS

Environment

Select or define an environment in which the application will run.

Create a new environment for the application:

Name * : MyExampleAppEnv

Description: This is the environment for my sample application.

Environment URL:

http:// MyExampleAppEnv .elasticbeanstalk.com

4. Describe your environment details.

   a. Select the **Create a new environment for this application** button.

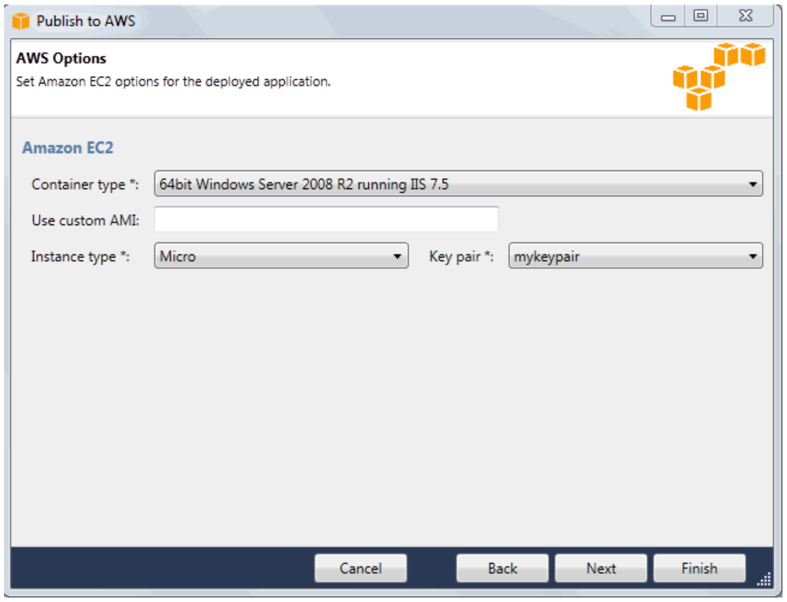
   b. In the **Name** box, type a name for your environment.

   c. In the **Description** box, type a description for your environment. This step is optional.

   d. The environment URL is automatically populated in the **Environment URL** text box.

   e. Click the **Check availability** button to make sure the environment URL is available.

   f. Click **Next**. The AWS Options page appears.



AWS Options
Set Amazon EC2 options for the deployed application.

Amazon EC2

Container type *: 64bit Windows Server 2008 R2 running IIS 7.5
Use custom AMI:
Instance type *: Micro
Key pair *: mykeypair

5. Configure additional options and security information for your deployment.

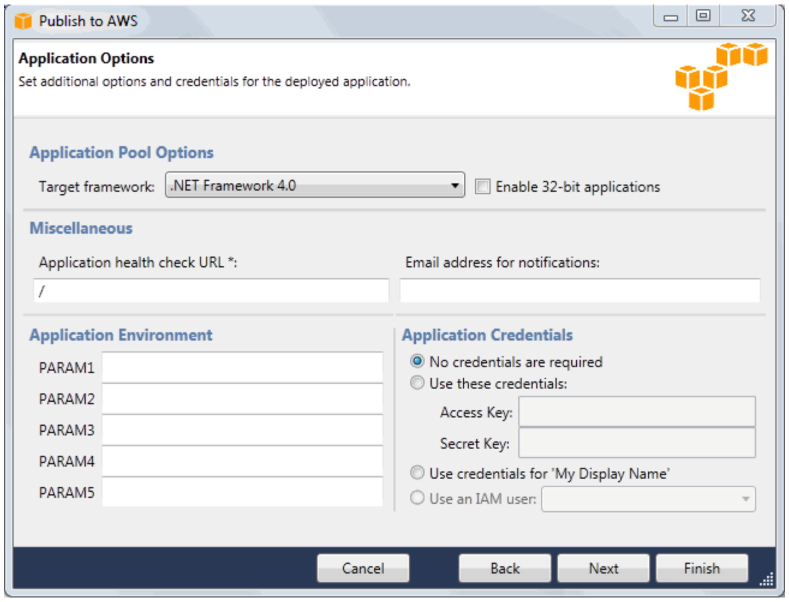
   a. In the **Container Type** list, click **64bit Windows running IIS 7.5**.

   b. In the **Instance Type** list, click **Micro**.

   c. In the **Key pair** list, click **Create new key pair**. A key pair enables you to remote desktop into your Amazon EC2 instances. For more information on Amazon EC2 key pairs, see Using Credentials in the *Amazon Elastic Compute Cloud User Guide*.

   d. Type a name for the new key pair—in this example, we use **my example key pair**—and then click **OK**.

   e. Click **Next**. The Application Options page appears.



Application Options
Set additional options and credentials for the deployed application.

Application Pool Options

Target framework: .NET Framework 4.0
Enable 32-bit applications

Miscellaneous

Application health check URL *: /
Email address for notifications:

Application Environment

PARAM1
PARAM2
PARAM3
PARAM4
PARAM5

Application Credentials

[✓ No credentials are required]
Use these credentials:
Access Key:
Secret Key:
Use credentials for 'My Display Name'
Use an IAM user:

6. Configure your application options.

   a. Select **.NET Runtime 4.0** from the **Target runtime** pull-down menu.

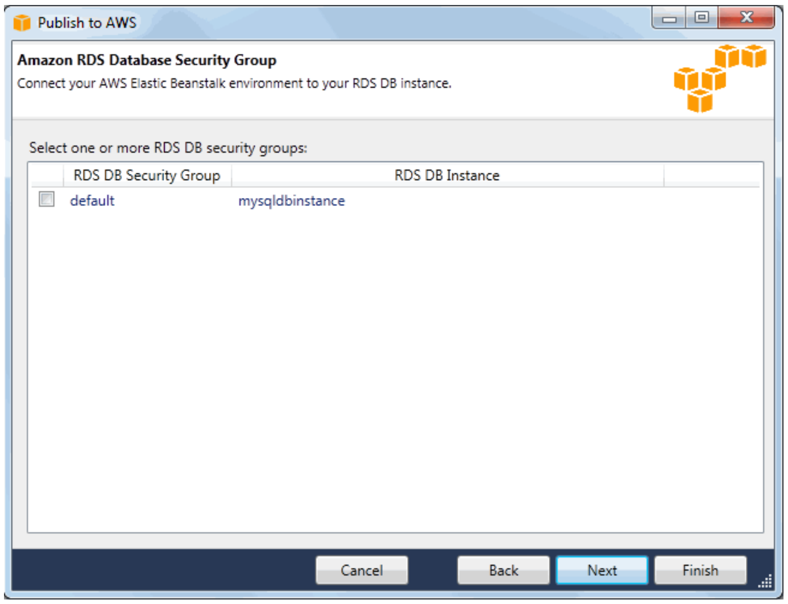
   b. Elastic Load Balancing uses a health check to determine whether the Amazon EC2 instances running your application are healthy. The health check determines an instance's health status by probing a specified URL at a set interval. You can override the default URL to match an existing resource in your application (e.g. '/myapp/index.jsp') by entering it in the **Application health check URL** box. For more information about application health checks, see Health Checks (p. 167).

   c. Type an email address if you want to recieve Amazon Simple Notification Service (Amazon SNS) notifications of important events affecting your application.

   d. The **Application Environment** section lets you specify environment variables on the Amazon EC2 instances that are running your application. This setting enables greater portability by eliminating the need to recompile your source code as you move between environments.

   e. Click the application credentials option you want to use to deploy your application.

   f. Click **Next**. If you have previously set up an Amazon RDS database, the Amazon RDS DB Security Group page appears. Otherwise, skip the next step to review your deployment information.



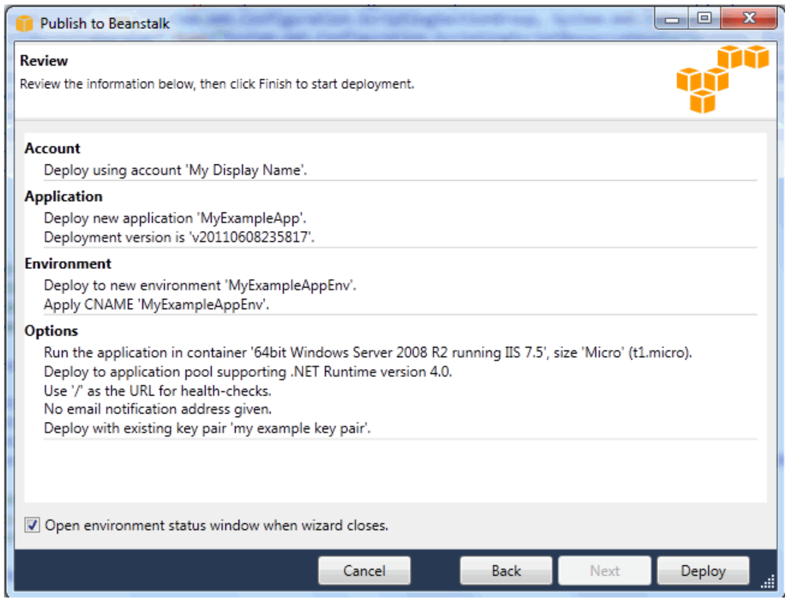
Amazon RDS Database Security Group
Connect your AWS Elastic Beanstalk environment to your RDS DB instance.

Select one or more RDS DB security groups:

	RDS DB Security Group	RDS DB Instance
☐	default	mysqldbinstance

7. Configure the Amazon RDS Database security group.

   - If you want to connect your Elastic Beanstalk environment to your Amazon RDS DB Instance, then select one or more security groups. Click **Next**. The Review page appears.



Review

Review the information below, then click Finish to start deployment.

Account

Deploy using account 'My Display Name'.

Application

Deploy new application 'MyExampleApp'.
Deployment version is 'v20110608235817'.

Environment

Deploy to new environment 'MyExampleAppEnv'.
Apply CNAME 'MyExampleAppEnv'.

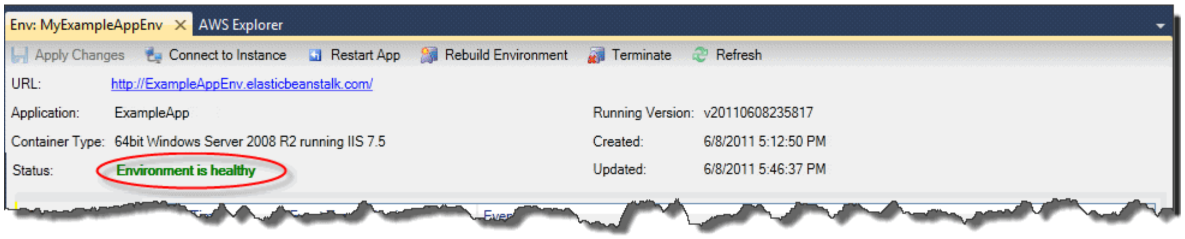
Options

Run the application in container '64bit Windows Server 2008 R2 running IIS 7.5', size 'Micro' (t1.micro).
Deploy to application pool supporting .NET Runtime version 4.0.
Use '/' as the URL for health-checks.
No email notification address given.
Deploy with existing key pair 'my example key pair'.

[✓] Open environment status window when wizard closes.

8. Review your deployment options. If everything is as you want, click **Deploy**.

   Your ASP.NET project will be exported as a web deploy file, uploaded to Amazon S3, and registered as a new application version with Elastic Beanstalk. The Elastic Beanstalk deployment feature will monitor your environment until it becomes available with the newly deployed code. On the env:<environment name> tab, you will see status for your environment.



URL: http://ExampleAppEnv.elasticbeanstalk.com/
Application: ExampleApp
Container Type: 64bit Windows Server 2008 R2 running IIS 7.5
Status: Environment is healthy
Running Version: v20110608235817
Created: 6/8/2011 5:12:50 PM
Updated: 6/8/2011 5:46:37 PM

Test Remotely

To test your application remotely, use the URL link on the env:<environment name> tab. To investigate any issues you can connect to your Amazon EC2 instance by using a Remote Desktop connection. For instructions on how to connect to your instance using a Remote Desktop connection, see Listing and Connecting to Server Instances (p. 86).You can also view logs to help you with debugging. For more information about viewing logs, see Debug/View Logs (p. 66).

Debug/View Logs

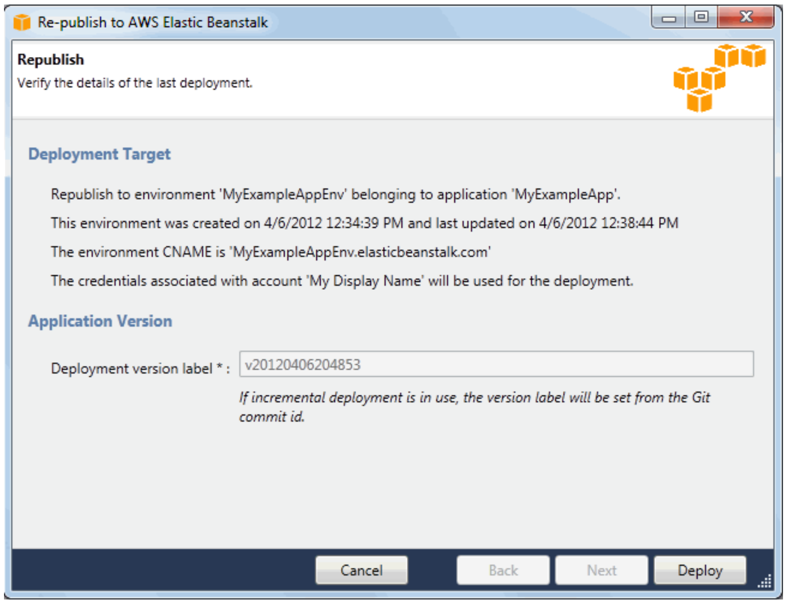
You can configure your environment so that the logs from the Amazon EC2 instances running your applications are copied by AWS Elastic Beanstalk to the Amazon S3 bucket associated with your application. For instructions on how to view these logs from the AWS Management Console, see Working with Logs (p. 199). For instructions on how to enable hourly rotation of your logs to Amazon S3, see Amazon S3 Log Rotation (p. 192).

Edit the Application and Redeploy

Now that you have tested your application, it is easy to edit and redeploy your application and see the results in moments.

To edit and redeploy your ASP.NET web application

1. In **Solution Explorer**, right-click your application, and then click **Republish to Environment <your environment name>**. The Re-publish to AWS Elastic Beanstalk wizard opens.


Re-publish to AWS Elastic Beanstalk
Republish
Verify the details of the last deployment.
Deployment Target
Application Version
Cancel
Back
Next
Deploy

Republish to environment 'MyExampleAppEnv' belonging to application 'MyExampleApp'.

This environment was created on 4/6/2012 12:34:39 PM and last updated on 4/6/2012 12:38:44 PM

The environment CNAME is 'MyExampleAppEnv.elasticbeanstalk.com'

The credentials associated with account 'My Display Name' will be used for the deployment.

Deployment version label * : v20120406204853

If incremental deployment is in use, the version label will be set from the Git commit id.

2. Review your deployment details and click **Deploy**.

   Note

   If you want to change any of your settings, you can click **Cancel** and use the Publish to AWS wizard instead. For instructions, see Deploy to AWS Elastic Beanstalk (p. 61).

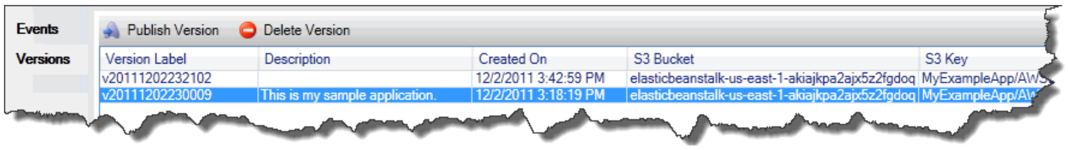
   Your updated ASP.NET web project will be exported as a web deploy file with the new version label, uploaded to Amazon S3, and registered as a new application version with Elastic Beanstalk. The Elastic Beanstalk deployment feature will monitor your existing environment until it becomes available with the newly deployed code. On the env:<environment name> tab, you will see the status of your environment.

Deploy to Production

When you are satisfied with all of the changes you want to make to your application, you can deploy your application version to your production environment.

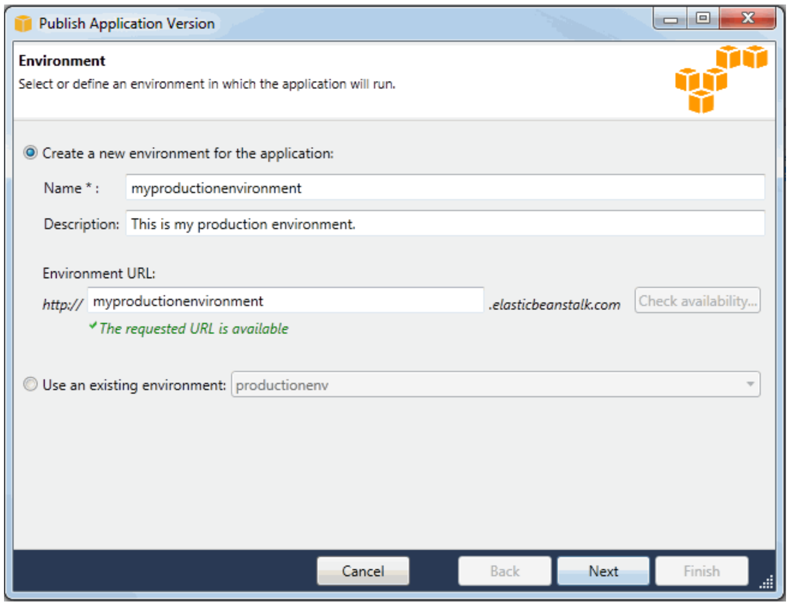
To deploy an application version to production

1. Right-click your AWS Elastic Beanstalk application by expanding the AWS Elastic Beanstalk node in **AWS Explorer**. Select **View Status**.
2. In the **App: <application name>** tab, click **Versions**.



Version Label	Description	Created On	S3 Bucket	S3 Key
v20111202232102		12/2/2011 3:42:59 PM	elasticbeanstalk-us-east-1-akiajkpa2ajx5z2fgdoq	MyExampleApp/AWS
v20111202230009	This is my sample application.	12/2/2011 3:18:19 PM	elasticbeanstalk-us-east-1-akiajkpa2ajx5z2fgdoq	MyExampleApp/AW

3. Click the application version you want to deploy and click **Publish Version**. The Publish Application Version wizard appears.



Publish Application Version

Environment

Select or define an environment in which the application will run.

(●) Create a new environment for the application:

Name * : myproductionenvironment

Description: This is my production environment.

Environment URL:

http:// myproductionenvironment .elasticbeanstalk.com

✓ The requested URL is available

( ) Use an existing environment: productionenv

4. Describe your environment details.

   a. Select the **Create a new environment for this application** button.

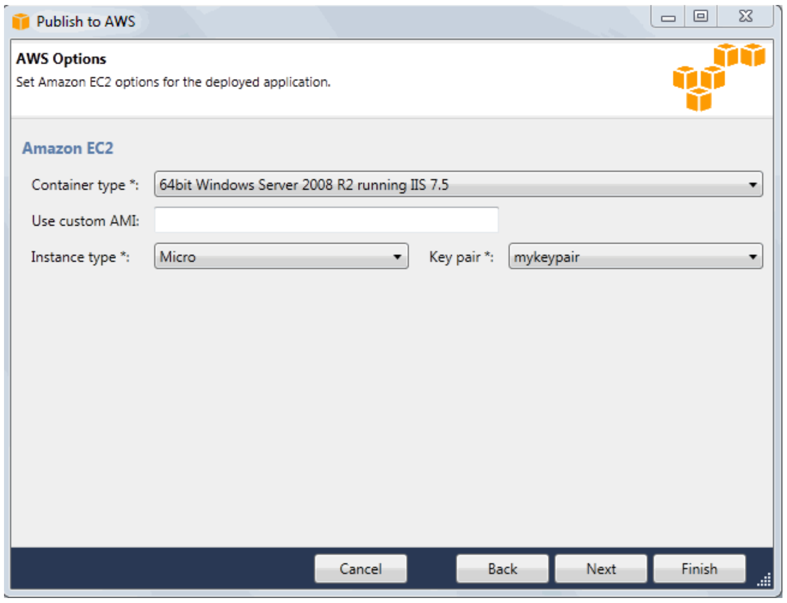
   b. In the **Name** box, type a name for your environment name.

   c. In the **Description** box, type a description for your environment. This step is optional.

   d. The environment URL is auto-populated in the **Environment URL** box.

   e. Click **Check availability** to make sure the environment URL is available.

   f. Click **Next**. The Options page appears.



Container type *: 64bit Windows Server 2008 R2 running IIS 7.5

Use custom AMI:

Instance type *: Micro

Key pair *: mykeypair

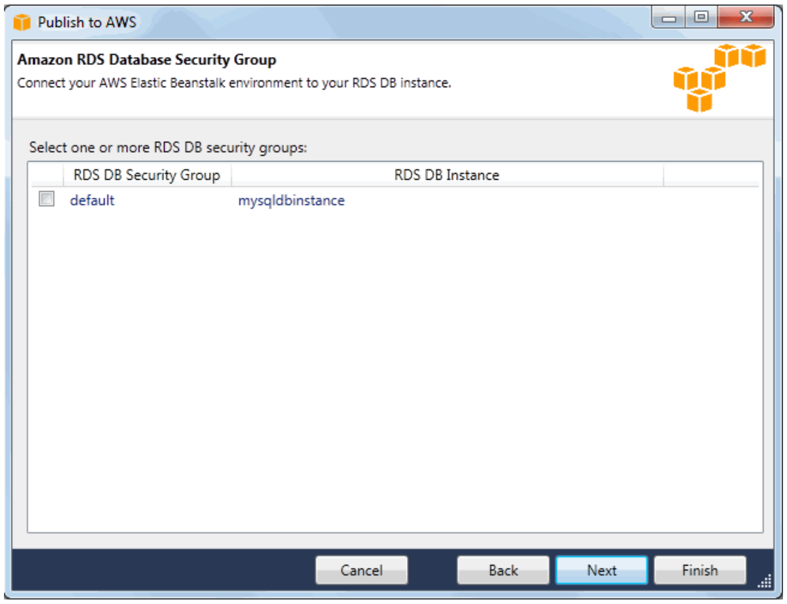
5. Configure your security information for your deployment.

   a. Configure your Amazon EC2 options.

      i. Select **64bit Windows running IIS 7.5** from the **Container Type** list.
      ii. Select **Micro** from the **Instance Type** list.
      iii. Select your key pair from the **Key pair** list.

   b. Configure your application pool options and application health check URL.

      i. Select **.NET Runtime 4.0** from the **Target runtime** list.
      ii. Elastic Load Balancing uses a health check to determine whether the Amazon EC2 instances running your application are healthy. The health check determines an instance's health status by probing a specified URL at a set interval. You can override the default URL to match an existing resource in your application (e.g. '/myapp/index.jsp') by entering it in the **Application health check URL** box. For more information about application health checks, see Health Checks (p. 167).
      iii. Click **Next**. If you have previously set up an Amazon RDS database, the Amazon RDS DB Security Group page appears. Otherwise, skip the next step to review your deployment information.



Amazon RDS Database Security Group

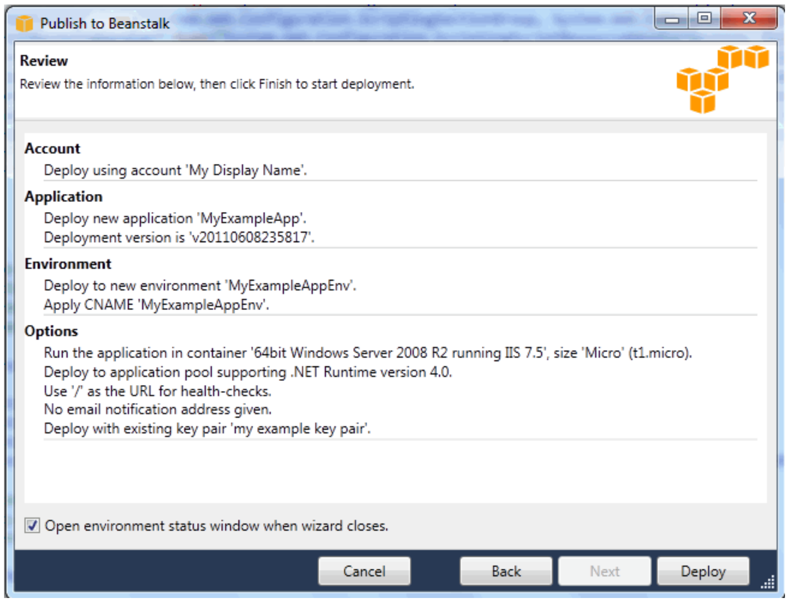
Connect your AWS Elastic Beanstalk environment to your RDS DB instance.

Select one or more RDS DB security groups:

RDS DB Security Group	RDS DB Instance
☐ default	mysqldbinstance

6. Configure the Amazon RDS Database security group.

   - If you want to connect your Elastic Beanstalk environment to your Amazon RDS DB Instance, select one or more security groups. Click **Next**. The Review page appears.



Review

Review the information below, then click Finish to start deployment.

Account
Deploy using account 'My Display Name'.

Application
Deploy new application 'MyExampleApp'.
Deployment version is 'v20110608235817'.

Environment
Deploy to new environment 'MyExampleAppEnv'.
Apply CNAME 'MyExampleAppEnv'.

Options
Run the application in container '64bit Windows Server 2008 R2 running IIS 7.5', size 'Micro' (t1.micro).
Deploy to application pool supporting .NET Runtime version 4.0.
Use '/' as the URL for health-checks.
No email notification address given.
Deploy with existing key pair 'my example key pair'.

☑ Open environment status window when wizard closes.

7. Review your deployment options, and click **Deploy**.

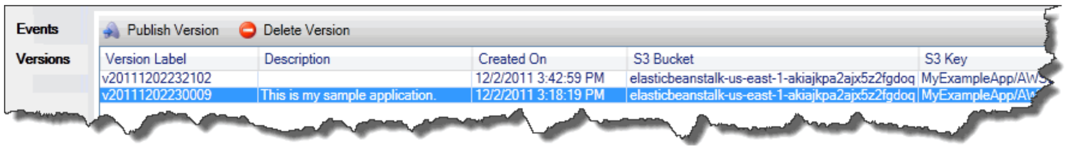
   Your ASP.NET project will be exported as a web deploy file and uploaded to Amazon S3. The Elastic Beanstalk deployment feature will monitor your environment until it becomes available with the newly deployed code. On the env:<environment name> tab, you will see status for your environment.

Deploy an Existing Application Version to an Existing Environment

You can deploy an existing application to an existing environment if, for instance, you need to roll back to a previous application version.

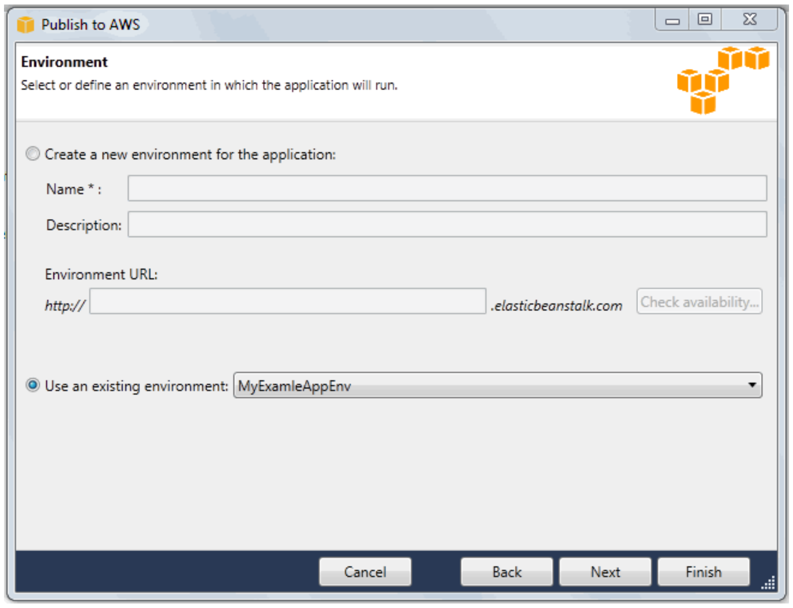
To deploy an application version to an existing environment

1. Right-click your AWS Elastic Beanstalk application by expanding the AWS Elastic Beanstalk node in **AWS Explorer**. Select **View Status**.
2. In the **App: <application name>** tab, click **Versions**.



Version Label	Description	Created On	S3 Bucket	S3 Key
v20111202232102		12/2/2011 3:42:59 PM	elasticbeanstalk-us-east-1-akiajkpa2ajx5z2fgdoq	MyExampleApp/AWS
v20111202230009	This is my sample application.	12/2/2011 3:18:19 PM	elasticbeanstalk-us-east-1-akiajkpa2ajx5z2fgdoq	MyExampleApp/AW

3. Click the application version you want to deploy and click **Publish Version**. The **Publish Application Version** wizard appears.

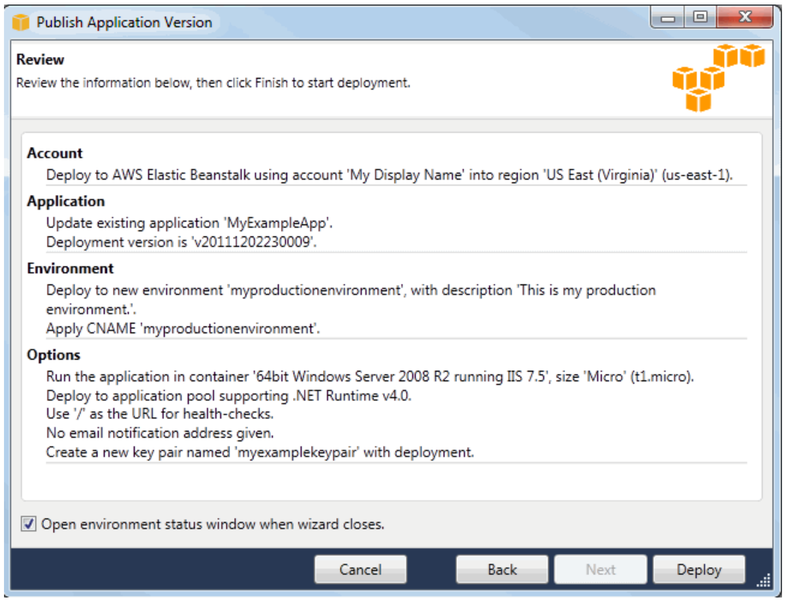


Environment
Select or define an environment in which the application will run.

Create a new environment for the application:
Name * :
Description:
Environment URL:
http:// .elasticbeanstalk.com

Use an existing environment: MyExamleAppEnv

4. Click **Next**. The Review page appears.



Review

Review the information below, then click Finish to start deployment.

Account

Deploy to AWS Elastic Beanstalk using account 'My Display Name' into region 'US East (Virginia)' (us-east-1).

Application

Update existing application 'MyExampleApp'.
Deployment version is 'v20111202230009'.

Environment

Deploy to new environment 'myproductionenvironment', with description 'This is my production environment.'.
Apply CNAME 'myproductionenvironment'.

Options

Run the application in container '64bit Windows Server 2008 R2 running IIS 7.5', size 'Micro' (t1.micro).
Deploy to application pool supporting .NET Runtime v4.0.
Use '/' as the URL for health-checks.
No email notification address given.
Create a new key pair named 'myexamplekeypair' with deployment.

☑ Open environment status window when wizard closes.

5. Review your deployment options, and click **Deploy**.

   Your ASP.NET project will be exported as a web deploy file and uploaded to Amazon S3. The Elastic Beanstalk deployment feature will monitor your environment until it becomes available with the newly deployed code. On the env:<environment name> tab, you will see status for your environment.

Using Amazon RDS and MySQL .NET Connector with AWS Elastic Beanstalk

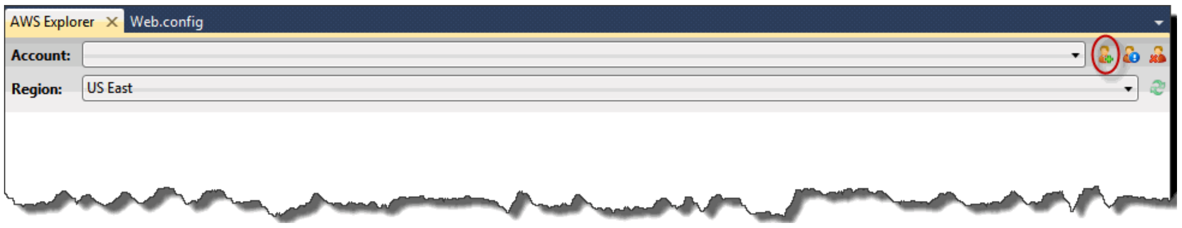
With the Amazon Relational Database Service (Amazon RDS), you can quickly and easily provision and maintain a MySQL Server or a SQL Server instance in the cloud. For MySQL Server, you can use Amazon RDS and the MySQL .NET Connector with your AWS Elastic Beanstalk application. For instructions on how to use RDS in your .NET application, go to Amazon RDS for C# Developers. For instructions on how to use SQL Server with your AWS Elastic Beanstalk application, see Get Started (p. 52). For more information on AWS storage options, go to Storage Options in the AWS Cloud.

Managing Multiple Accounts

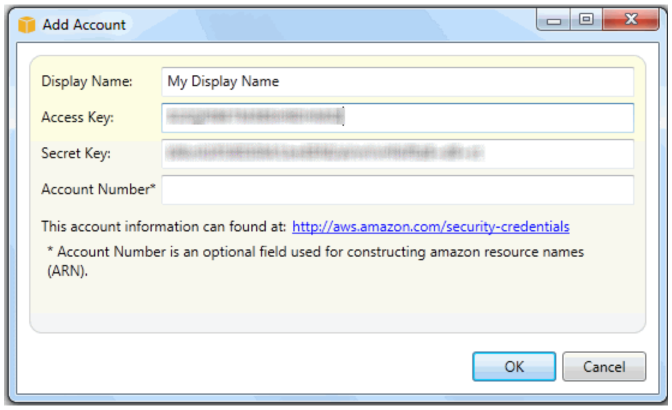
If you want to set up different AWS accounts to perform different tasks, such as testing, staging, and production, you can add, edit, and delete accounts using the AWS Toolkit for Visual Studio.

To manage multiple accounts

1. In Visual Studio, on the **View** menu, click **AWS Explorer**.
2. Beside the **Account** list, click the **Add Account** button.



The **Add Account** dialog box appears.



Add Account

Display Name: My Display Name

Access Key:

Secret Key:

Account Number*

This account information can found at: http://aws.amazon.com/security-credentials

* Account Number is an optional field used for constructing amazon resource names (ARN).

3. In the **Display Name** box, type the display name.
4. In the **Access Key** box, type your AWS access ID.
5. In the **Secret Key** box, type your AWS secret key.

   Your account information now appears on the **AWS Explorer** tab. When you publish to AWS Elastic Beanstalk, you can select which account you would like to use.

Monitoring Application Health

When you are running a production website, it is important to know that your application is available and responding to requests. To assist with monitoring your application's responsiveness, AWS Elastic Beanstalk performs a periodic health check on running applications and can be configured to notify you by email of potential problems.

Understanding Environment Health

To check application health status, every minute or two Elastic Beanstalk sends an HTTP HEAD request to the application health check URL. By default, the application health check URL is the application root (e.g., http://myapp.elasticbeanstalk.com/), but you can change this value using the AWS Toolkit for Visual Studio or the AWS Management Console. For instructions on modifying your health check URL using the AWS Toolkit for Visual Studio, see Health Checks (p. 81). For instructions on modifying your health check URL using the AWS Management Console, see Health Checks (p. 167). AWS Elastic Beanstalk expects a response of `200 OK` for the application to be considered healthy.

AWS Elastic Beanstalk will change the environment health status to one of four color values depending on how the application responds to the health check. The following table describes the color codes.

Color	Status
Green	Your application responded to the application health check URL within the last minute.
Yellow	Your application hasn't responded to the application health check URL within the last five minutes.
Red	One of the following: • Your application hasn't responded to the application health check URL for more than five minutes. • An environment is also considered red if AWS Elastic Beanstalk detects other problems with the environment that are known to make the application unavailable (e.g., the load balancer was deleted).
Gray	Your application's health status is unknown because status is reported when the application is not in the *Ready* state.

Whenever the application health check URL fails to return a `200 OK` response, AWS Elastic Beanstalk performs a series of additional checks to try to determine the cause of the failure. These additional checks include verifying the following:

- The load balancer still exists
- The Auto Scaling group still exists
- There is at least one Amazon EC2 instance “In-Service” behind the load balancer
- The Amazon EC2 security group is configured to allow ingress on port 80
- The environment CNAME exists and is pointing to the right load balancer
- All Amazon EC2 instances are communicating

In addition to environment-level health checking, AWS Elastic Beanstalk also communicates with every Amazon EC2 instance running as part of your Elastic Beanstalk application. If any Amazon EC2 instance fails to respond to ten consecutive health checks, Elastic Beanstalk will terminate the instance, and Auto Scaling will start a new instance.

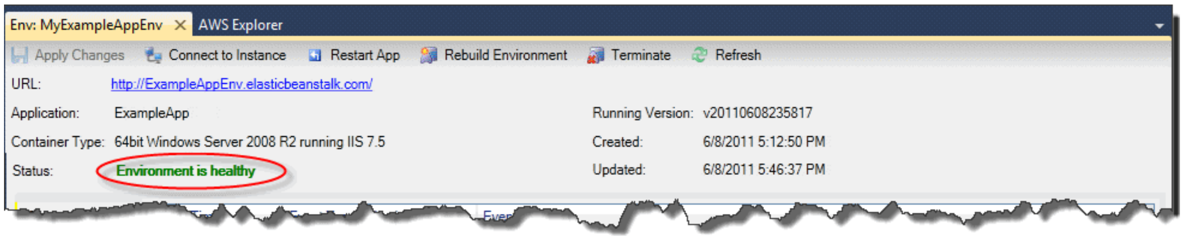
If the status of your application health changes to red, you can take several corrective actions:

- Look at environment events. You might find more information about the problem here.
- If you recently deployed a new version of the application, try rolling back to an older version that is known to work.
- If you recently made configuration changes, try reverting to the former settings.
- If the problem appears to be with the environment, try rebuilding the environment. In the AWS Toolkit for Visual Studio, on the **AWS Explorer** tab, right-click your application environment, and then click **Rebuild Environment**.
- Try using Snapshot logs to view recent log file entries or log in to the Amazon EC2 instance and troubleshoot directly.

Viewing Application Health and Environment Status

You can access operational information about your application by using either the AWS Toolkit for Visual Studio or the AWS Management Console.

The toolkit displays your environment's status and application health at a glance in the Status field.



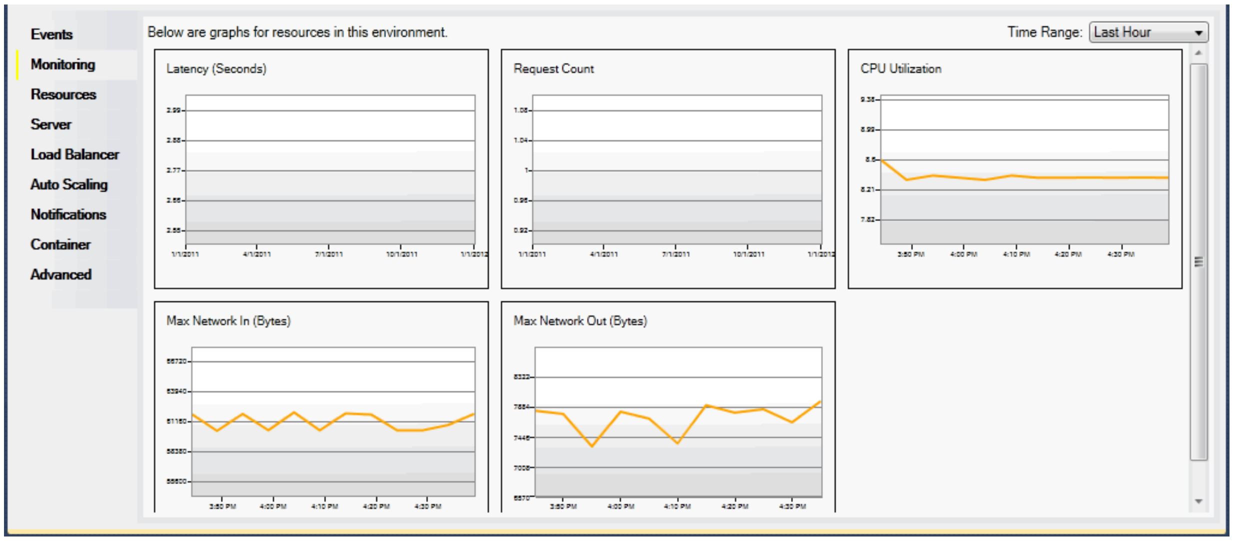
URL: http://ExampleAppEnv.elasticbeanstalk.com/
Application: ExampleApp
Container Type: 64bit Windows Server 2008 R2 running IIS 7.5
Status: Environment is healthy
Running Version: v20110608235817
Created: 6/8/2011 5:12:50 PM
Updated: 6/8/2011 5:46:37 PM

For information on how to view application health by using the AWS Management Console, see Viewing Application Health and Environment Status (p. 153).

To monitor application health

1. In the AWS Toolkit for Visual Studio, in **AWS Explorer**, expand the AWS Elastic Beanstalk node, and then expand your application node.
2. Right-click your AWS Elastic Beanstalk environment, and then click **View Status**.
3. On your application environment tab, click **Monitoring**.

   The **Monitoring** panel appears.



Below are graphs for resources in this environment.

A set of graphs showing resource usage for your particular application environment is displayed.

Note

By default, the time range is set to the last hour. To modify this setting, in the **Time Range** list, click a different time range.

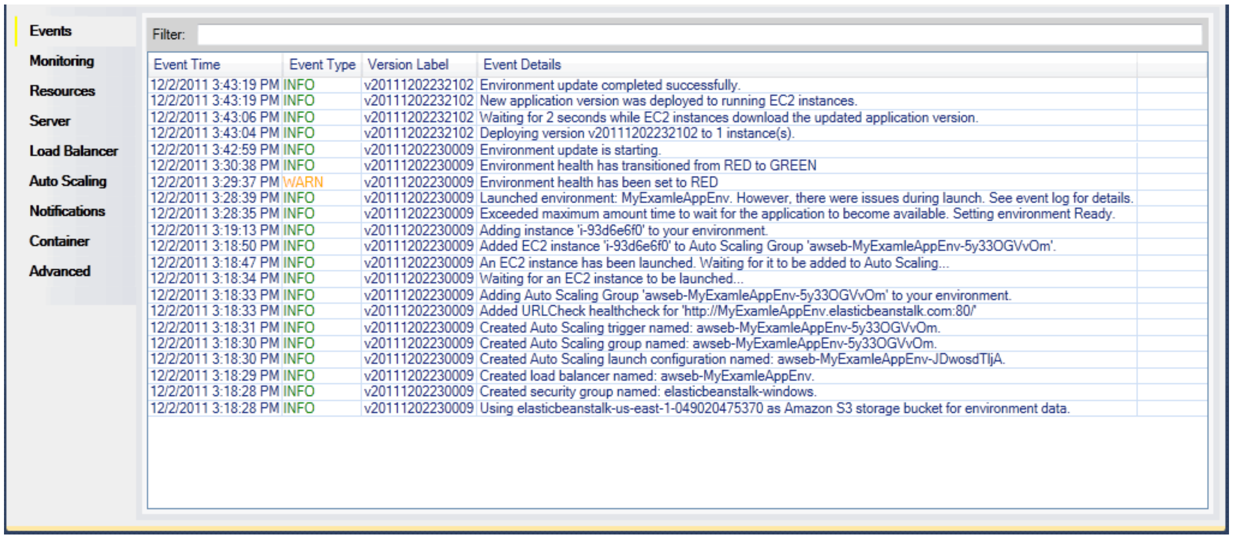
Viewing Events

You can use the AWS Toolkit for Visual Studio or the AWS Management Console to view events associated with your application. For information on the most common events, see Understanding Environment Launch Events (p. 216). For instructions on how to use the AWS Management Console to view events, see Viewing Events (p. 155). This section steps you through viewing your events using the AWS Toolkit for Visual Studio.

To view application events

1. In the AWS Toolkit for Visual Studio, in **AWS Explorer**, expand the AWS Elastic Beanstalk node and your application node.
2. Right-click your AWS Elastic Beanstalk environment in **AWS Explorer**, and then click **View Status**.
3. In your application environment tab, click **Events**.

   The **Events** panel appears.



Event Time	Event Type	Version Label	Event Details
12/2/2011 3:43:19 PM	INFO	v20111202232102	Environment update completed successfully.
12/2/2011 3:43:19 PM	INFO	v20111202232102	New application version was deployed to running EC2 instances.
12/2/2011 3:43:06 PM	INFO	v20111202232102	Waiting for 2 seconds while EC2 instances download the updated application version.
12/2/2011 3:43:04 PM	INFO	v20111202232102	Deploying version v20111202232102 to 1 instance(s).
12/2/2011 3:42:59 PM	INFO	v20111202230009	Environment update is starting.
12/2/2011 3:30:38 PM	INFO	v20111202230009	Environment health has transitioned from RED to GREEN
12/2/2011 3:29:37 PM	WARN	v20111202230009	Environment health has been set to RED
12/2/2011 3:28:39 PM	INFO	v20111202230009	Launched environment: MyExamleAppEnv. However, there were issues during launch. See event log for details.
12/2/2011 3:28:35 PM	INFO	v20111202230009	Exceeded maximum amount time to wait for the application to become available. Setting environment Ready.
12/2/2011 3:19:13 PM	INFO	v20111202230009	Adding instance 'i-93d6e6f0' to your environment.
12/2/2011 3:18:50 PM	INFO	v20111202230009	Added EC2 instance 'i-93d6e6f0' to Auto Scaling Group 'awseb-MyExamleAppEnv-5y33OGVvOm'.
12/2/2011 3:18:47 PM	INFO	v20111202230009	An EC2 instance has been launched. Waiting for it to be added to Auto Scaling...
12/2/2011 3:18:34 PM	INFO	v20111202230009	Waiting for an EC2 instance to be launched...
12/2/2011 3:18:33 PM	INFO	v20111202230009	Adding Auto Scaling Group 'awseb-MyExamleAppEnv-5y33OGVvOm' to your environment.
12/2/2011 3:18:33 PM	INFO	v20111202230009	Added URLCheck healthcheck for 'http://MyExamleAppEnv.elasticbeanstalk.com:80/'
12/2/2011 3:18:31 PM	INFO	v20111202230009	Created Auto Scaling trigger named: awseb-MyExamleAppEnv-5y33OGVvOm.
12/2/2011 3:18:30 PM	INFO	v20111202230009	Created Auto Scaling group named: awseb-MyExamleAppEnv-5y33OGVvOm.
12/2/2011 3:18:30 PM	INFO	v20111202230009	Created Auto Scaling launch configuration named: awseb-MyExamleAppEnv-JDwosdTljA.
12/2/2011 3:18:29 PM	INFO	v20111202230009	Created load balancer named: awseb-MyExamleAppEnv.
12/2/2011 3:18:28 PM	INFO	v20111202230009	Created security group named: elasticbeanstalk-windows.
12/2/2011 3:18:28 PM	INFO	v20111202230009	Using elasticbeanstalk-us-east-1-049020475370 as Amazon S3 storage bucket for environment data.

Managing Your AWS Elastic Beanstalk Application Environments

With the AWS Toolkit for Visual Studio and the AWS Management Console, you can change the provisioning and configuration of the AWS resources used by your application environments. For information on how to manage your application environments using the AWS Management Console, see Managing Environments (p. 157). This section discusses the specific service settings you can edit in the AWS Toolkit for Visual Studio as part of your application environment configuration.

Changing Environment Configurations Settings

When you deploy your application, AWS Elastic Beanstalk configures a number of AWS cloud computing services. You can control how these individual services are configured using the AWS Toolkit for Visual Studio.

To edit an application's environment settings

- Expand the AWS Elastic Beanstalk node and your application node, and then right-click your AWS Elastic Beanstalk environment in **AWS Explorer**. Select **View Status**.

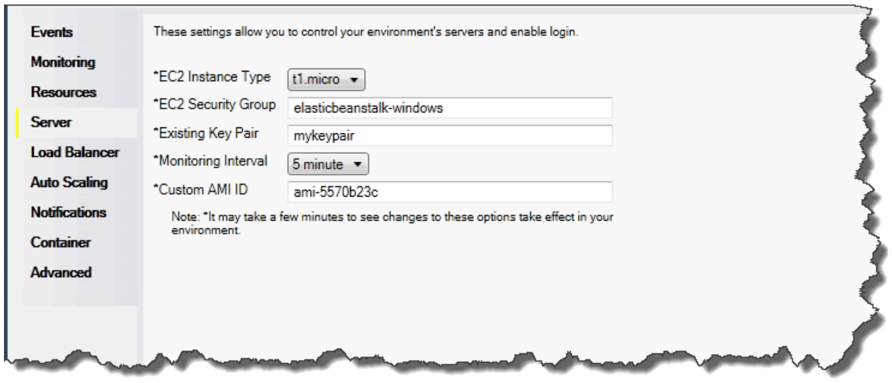
  You can now configure settings for the following:

  - Server
  - Load balancer
  - Auto Scaling
  - Notifications
  - Environment properties

Configuring EC2 Server Instances Using AWS Toolkit for Visual Studio

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that you use to launch and manage server instances in Amazon's data centers. You can use Amazon EC2 server instances at any time, for as long as you need, and for any legal purpose. Instances are available in different sizes and configurations. For more information, go to the Amazon EC2 product page).

You can edit the AWS Elastic Beanstalk environment's Amazon EC2 instance configuration with the **Server** tab inside your application environment tab in the AWS Toolkit for Visual Studio.



These settings allow you to control your environment's servers and enable login.

*EC2 Instance Type	t1.micro
*EC2 Security Group	elasticbeanstalk-windows
*Existing Key Pair	mykeypair
*Monitoring Interval	5 minute
*Custom AMI ID	ami-5570b23c

Note: *It may take a few minutes to see changes to these options take effect in your environment.

Amazon EC2 Instance Types

The **EC2 Instance Type** drop-down list box displays the instance types available to your AWS Elastic Beanstalk application. Changing the instance type enables you to select a server with the characteristics (including memory size and CPU power) that are most appropriate to your application. For instance, applications with intensive and long-running operations may require more CPU or memory. AWS Elastic Beanstalk regularly checks your running instances to ensure they are healthy. If your application consumes

95 percent or greater of the CPU, AWS Elastic Beanstalk will fire an event. For more information about this event, see CPU Utilization Greater Than 95.00% (p. 216).

> **Note**
>
> You cannot change between 32-bit and 64-bit instance types. For example, if your application is built on a 32-bit platform, only 32-bit instance types appear in the list.

For more information about the Amazon EC2 instance types available for your AWS Elastic Beanstalk application, go to Instance Families and Types in the *Amazon Elastic Compute Cloud User Guide*.

Amazon EC2 Security Groups

You can control access to your AWS Elastic Beanstalk application using an *Amazon EC2 Security Group*. A security group defines firewall rules for your instances. These rules specify which ingress (i.e., incoming) network traffic should be delivered to your instance. All other ingress traffic will be discarded. You can modify rules for a group at any time. The new rules are automatically enforced for all running instances and instances launched in the future.

You can set up your Amazon EC2 security groups using the AWS Management Console or by using the AWS Toolkit for Visual Studio. You can specify which Amazon EC2 Security Groups control access to your AWS Elastic Beanstalk application by entering the names of one or more Amazon EC2 security group names (delimited by commas) into the **EC2 Security Groups** text box.

> **Note**
>
> Make sure port 80 (HTTP) is accessible from 0.0.0.0/0 as the source CIDR range if you want to enable health checks for your application. For more information about health checks, see Health Checks (p. 81).

To create a security group using the AWS Toolkit for Visual Studio

1. In Visual Studio, in **AWS Explorer**, expand the **Amazon EC2** node, and then double-click **Security Groups**.
2. Click **Create Security Group**, and enter a name and description for your security group.
3. Click **OK**.

For more information on Amazon EC2 Security Groups, see Using Security Groups in the *Amazon Elastic Compute Cloud User Guide*.

Amazon EC2 Key Pairs

You can securely log in to the Amazon EC2 instances provisioned for your AWS Elastic Beanstalk application with an Amazon EC2 key pair.

> **Important**
>
> You must create an Amazon EC2 key pair and configure your AWS Elastic Beanstalk-provisioned Amazon EC2 instances to use the Amazon EC2 key pair before you can access your AWS Elastic Beanstalk-provisioned Amazon EC2 instances. You can create your key pair using the Publish to AWS Wizard inside AWS Toolkit for Visual Studio when you deploy your application to AWS Elastic Beanstalk. If you want to create additional key pairs using the AWS Toolkit for Visual Studio, follow the steps below. Alternatively, you can set up your Amazon EC2 key pairs using the AWS Management Console. For instructions on creating a key pair for Amazon EC2, see the Amazon Elastic Compute Cloud Getting Started Guide.

The **Existing Key Pair** text box lets you specify the name of an Amazon EC2 key pair you can use to securely log in to the Amazon EC2 instances running your AWS Elastic Beanstalk application.
To specify the name of an Amazon EC2 key pair

1. Expand the **Amazon EC2** node and double-click **Key Pairs**.
2. Click **Create Key Pair** and enter the key pair name.
3. Click **OK**.

For more information on Amazon EC2 key pairs, go to Using Amazon EC2 Credentials in the *Amazon Elastic Compute Cloud User Guide*. For more information on connecting to Amazon EC2 instances, see Listing and Connecting to Server Instances (p. 86).

Monitoring Interval

By default, only basic Amazon CloudWatch metrics are enabled; they return data in five-minute periods. You can enable more granular one-minute CloudWatch metrics by selecting `1 minute` in the **Monitoring Interval** drop-down list box.

Note

Amazon CloudWatch service charges can apply for one-minute interval metrics. See the Amazon CloudWatch product page for more information.

Custom AMI ID

You can override the default AMI used for your Amazon EC2 instances with your own custom AMI by entering the identifier of your custom AMI into the **Custom AMI ID** text box.

Important

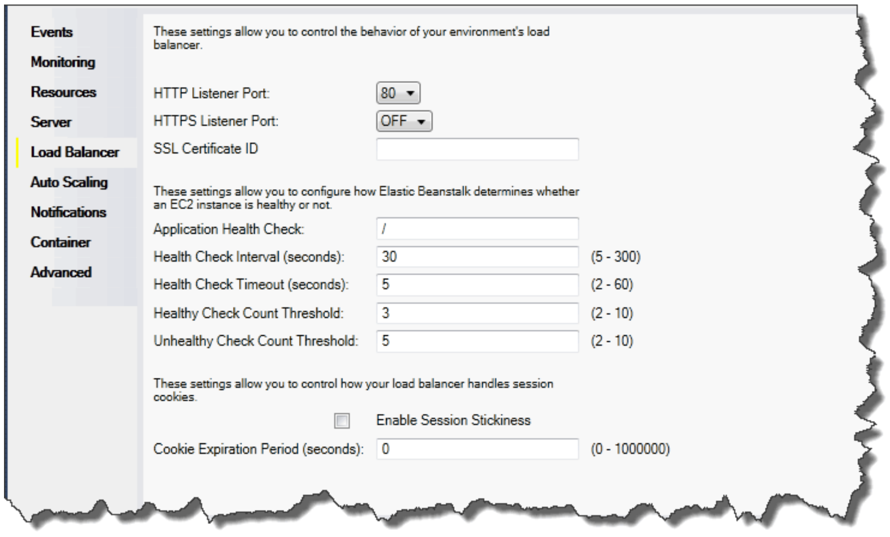
Using your own AMI is an advanced use case and should be done with care. If you need a custom AMI, we recommend you start with the default AWS Elastic Beanstalk AMI, and then modify it. To be considered healthy, AWS Elastic Beanstalk expects Amazon EC2 instances to meet a set of requirements, including having a running host manager. If these requirements are not met, your environment might not work properly.

Configuring Elastic Load Balancing Using AWS Toolkit for Visual Studio

Elastic Load Balancing is an Amazon web service that helps you improve the availability and scalability of your application. This service makes it easy for you to distribute application loads between two or more Amazon EC2 instances. Elastic Load Balancing enables availability through redundancy and supports traffic growth for your application.

Elastic Load Balancing lets you automatically distribute and balance the incoming application traffic among all the instances you are running. The service also makes it easy to add new instances when you need to increase the capacity of your application.

AWS Elastic Beanstalk automatically provisions Elastic Load Balancing when you deploy an application. You can edit the AWS Elastic Beanstalk environment's Amazon EC2 instance configuration with the **Load Balancer** tab inside your application environment tab in AWS Toolkit for Visual Studio.



These settings allow you to control the behavior of your environment's load balancer.

HTTP Listener Port:	80	
HTTPS Listener Port:	OFF	
SSL Certificate ID		

These settings allow you to configure how Elastic Beanstalk determines whether an EC2 instance is healthy or not.

Application Health Check:	/	
Health Check Interval (seconds):	30	(5 - 300)
Health Check Timeout (seconds):	5	(2 - 60)
Healthy Check Count Threshold:	3	(2 - 10)
Unhealthy Check Count Threshold:	5	(2 - 10)

These settings allow you to control how your load balancer handles session cookies.

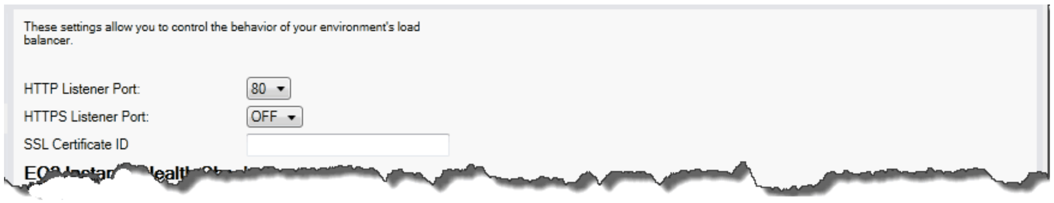
☐ Enable Session Stickiness

Cookie Expiration Period (seconds):	0	(0 - 1000000)

The following sections describe the Elastic Load Balancing parameters you can configure for your application.

Ports

The load balancer provisioned to handle requests for your AWS Elastic Beanstalk application sends requests to the Amazon EC2 instances that are running your application. The provisioned load balancer can listen for requests on HTTP and HTTPS ports and route requests to the Amazon EC2 instances in your AWS Elastic Beanstalk application. By default, the load balancer handles requests on the HTTP port. At least one of the ports (either HTTP or HTTPS) must be turned on.



These settings allow you to control the behavior of your environment's load balancer.

HTTP Listener Port:	80
HTTPS Listener Port:	OFF
SSL Certificate ID	

Important

Make sure that the port you specified is not locked down, otherwise users will not be able to connect to your AWS Elastic Beanstalk application.

Controlling the HTTP port

To turn off the HTTP port, you select OFF from the **HTTP Listener Port** drop-down list. To turn on the HTTP port, you select an HTTP port (for example, 80) from the drop-down list.

Note

If you want to access your environment using a different port other than the default port 80 (e.g., port 8080), you can add a listener to the existing load balancer and configure the new listener to listen on that port. For example, using the Elastic Load Balancing command line tools, type the following command replacing `<yourloadbalancername>` with the name of your load balancer for Elastic Beanstalk.

```
elb-create-lb-listeners --lb <yourloadbalancername> --listener "pro
tocol=http, lb-port=8080, instance-port=80"
```

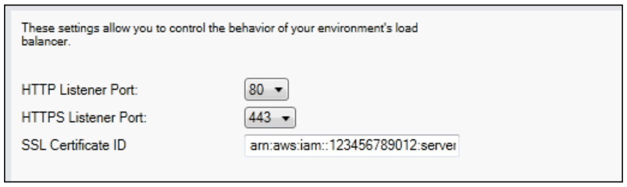
If you want Elastic Beanstalk to monitor your environment, do not remove the listener on port 80.

Controlling the HTTPS port

Elastic Load Balancing supports the HTTPS/TLS protocol to enable traffic encryption for client connections to the load balancer. Connections from the load balancer to the EC2 instances are done using plaintext. By default, the HTTPS port is turned off.

To turn on the HTTPS port

1. Create and upload a certificate and key to the AWS Access and Identity Management (AWS IAM) service. The IAM service will store the certificate and provide an Amazon Resource Name (ARN) for the SSL certificate you've uploaded. For more information creating and uploading certificates, see the Managing Server Certificates section of *Using AWS Identity and Access Management*.
2. Specify the HTTPS port by selecting a port from the **HTTPS Listener Port** drop-down list.



These settings allow you to control the behavior of your environment's load balancer.

HTTP Listener Port:	80
HTTPS Listener Port:	443
SSL Certificate ID	arn:aws:iam::123456789012:server

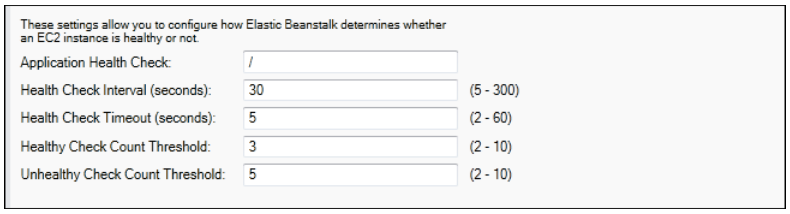
3. In the **SSL Certificate ID** text box, enter the Amazon Resources Name (ARN) of your SSL certificate (e.g., `arn:aws:iam::123456789012:server-certificate/abc/certs/build`). Use the SSL certificate that you created and uploaded in step 1. For information on viewing the certificate's ARN, see Verify the Certificate Object topic in the *Creating and Uploading Server Certificates* section of the *Using IAM Guide*.

To turn off the HTTPS port, select OFF from the **HTTPS Listener Port** drop-down list.

Health Checks

Elastic Load Balancing uses a *health check* to determine whether the Amazon EC2 instances running your application are healthy. The health check determines an instance's health status by probing a specified URL at a set interval; if the URL returns an error message, or fails to return within a specified timeout period, the health check fails. AWS Elastic Beanstalk uses Elastic Load Balancing health checks to set the status of your application.

You can control the settings for the health check using the **EC2 Instance Health Check** section of the **Load Balancing** panel.



These settings allow you to configure how Elastic Beanstalk determines whether an EC2 instance is healthy or not.

Application Health Check:	/	
Health Check Interval (seconds):	30	(5 - 300)
Health Check Timeout (seconds):	5	(2 - 60)
Healthy Check Count Threshold:	3	(2 - 10)
Unhealthy Check Count Threshold:	5	(2 - 10)

The health check definition includes a URL to be queried for instance health. Override the default URL to match an existing resource in your application (e.g. '/myapp/index.jsp') by entering it in the **Application Health Check URL** text box.

The following list describes the health check parameters you can set for your application.

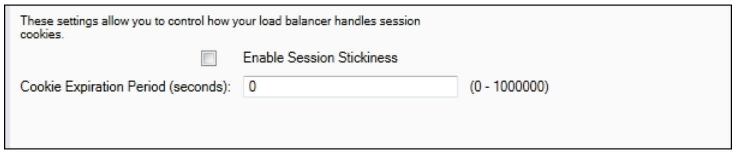
- Enter the number of seconds in the **Health Check Interval (seconds)** text box to define the interval at which Elastic Load Balancing will check the health of your application's Amazon EC2 instances.
- Specify the number of seconds Elastic Load Balancing will wait for a response before it considers the instance non-responsive in the **Health Check Timeout (seconds)** text box.
- Specify the number of consecutive successful or unsuccessful URL probes before Elastic Load Balancing changes the instance health status in the **Healthy Check Count Threshold** and **Unhealthy Check Count Threshold** text boxes. For example, specifying 5 in the **Unhealthy Check Count Threshold** text box means that the URL would have to return an error message or timeout five consecutive times before Elastic Load Balancing considers the health check "failed."

Sessions

By default a load balancer routes each request independently to the server instance with the smallest load. By comparison, a sticky session binds a user's session to a specific server instance so that all requests coming from the user during the session will be sent to the same server instance.

AWS Elastic Beanstalk uses load balancer-generated HTTP cookies when sticky sessions are enabled for an application. The load balancer uses a special load balancer-generated cookie to track the application instance for each request. When the load balancer receives a request, it first checks to see if this cookie is present in the request. If so, the request is sent to the application instance specified in the cookie. If there is no cookie, the load balancer chooses an application instance based on the existing load balancing algorithm. A cookie is inserted into the response for binding subsequent requests from the same user to that application instance. The policy configuration defines a cookie expiry, which establishes the duration of validity for each cookie.

You can use the **Sessions** section on the **Load Balancer** tab to specify whether or not the load balancer for your application allows session stickiness.



These settings allow you to control how your load balancer handles session cookies.

☐ Enable Session Stickiness

Cookie Expiration Period (seconds): 0 (0 - 1000000)

For more information on Elastic Load Balancing, go to the Elastic Load Balancing Developer Guide.

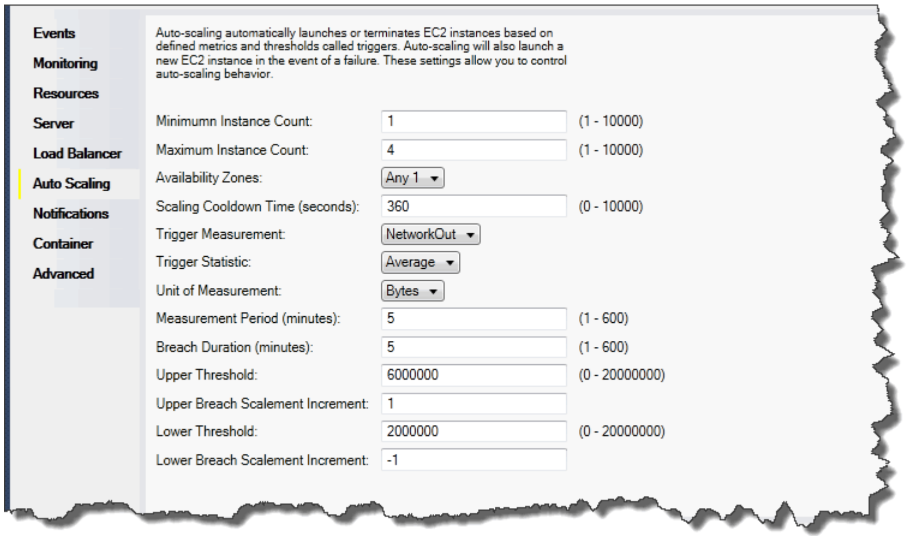
Configuring Auto Scaling Using AWS Toolkit for Visual Studio

Auto Scaling is an Amazon web service designed to automatically launch or terminate Amazon EC2 instances based on user-defined triggers. Users can set up *Auto Scaling groups* and associate *triggers* with these groups to automatically scale computing resources based on metrics such as bandwidth usage or CPU utilization. Auto Scaling works with Amazon CloudWatch to retrieve metrics for the server instances running your application.

Auto Scaling lets you take a group of Amazon EC2 instances and set various parameters to have this group automatically increase or decrease in number. Auto Scaling can add or remove Amazon EC2 instances from that group to help you seamlessly deal with traffic changes to your application.

Auto Scaling also monitors the health of each Amazon EC2 instance that it launches. If any instance terminates unexpectedly, Auto Scaling detects the termination and launches a replacement instance. This capability enables you to maintain a fixed, desired number of Amazon EC2 instances automatically.

AWS Elastic Beanstalk provisions Auto Scaling for your application. You can edit the AWS Elastic Beanstalk environment's Amazon EC2 instance configuration with the **Auto Scaling** tab inside your application environment tab in the AWS Toolkit for Visual Studio.



Auto-scaling automatically launches or terminates EC2 instances based on defined metrics and thresholds called triggers. Auto-scaling will also launch a new EC2 instance in the event of a failure. These settings allow you to control auto-scaling behavior.

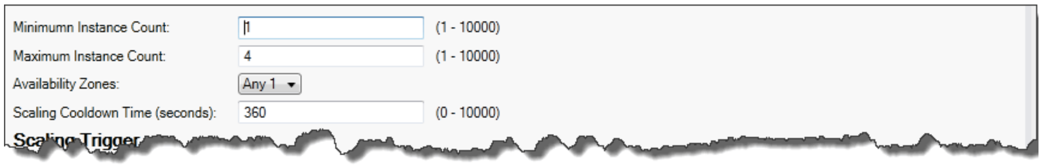
Minimumn Instance Count:	1	(1 - 10000)
Maximum Instance Count:	4	(1 - 10000)
Availability Zones:	Any 1	
Scaling Cooldown Time (seconds):	360	(0 - 10000)
Trigger Measurement:	NetworkOut	
Trigger Statistic:	Average	
Unit of Measurement:	Bytes	
Measurement Period (minutes):	5	(1 - 600)
Breach Duration (minutes):	5	(1 - 600)
Upper Threshold:	6000000	(0 - 20000000)
Upper Breach Scalement Increment:	1	
Lower Threshold:	2000000	(0 - 20000000)
Lower Breach Scalement Increment:	-1	

The following section discusses how to configure Auto Scaling parameters for your application.

Launch Configuration

You can edit the launch configuration to control how your AWS Elastic Beanstalk application provisions Auto Scaling resources.

The **Minimum Instance Count** and **Maximum Instance Count** text boxes let you specify the minimum and maximum size of the Auto Scaling group that your AWS Elastic Beanstalk application uses.



Minimumn Instance Count:	1	(1 - 10000)
Maximum Instance Count:	4	(1 - 10000)
Availability Zones:	Any 1	
Scaling Cooldown Time (seconds):	360	(0 - 10000)

Note

To maintain a fixed number of Amazon EC2 instances, set the **Minimum Instance Count** and **Maximum Instance Count** text boxes to the same value.

The **Availability Zones** text box lets you specify the number of Availability Zones you want your Amazon EC2 instances to be in. It is important to set this number if you want to build fault-tolerant applications. If one Availability Zone goes down, your instances will still be running in your other Availability Zones.

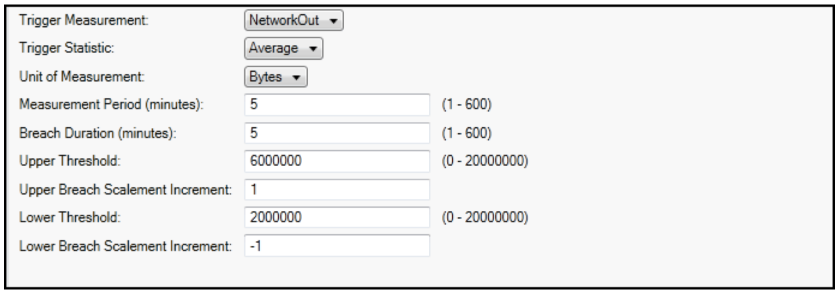
Note

Currently it is not possible to specify which Availability Zone your instance will be in.

Triggers

A *trigger* is an Auto Scaling mechanism that you set to tell the system when you want to increase (*scale out*) the number of instances, and when you want to decrease (*scale in*) the number of instances. You can configure triggers to *fire* on any metric published to Amazon CloudWatch, such as CPU utilization, and determine if the conditions you specified have been met. When the upper or lower thresholds of the conditions you have specified for the metric have been breached for the specified period of time, the trigger launches a long-running process called a *Scaling Activity*.

You can define a scaling trigger for your AWS Elastic Beanstalk application using AWS Toolkit for Visual Studio.



Trigger Measurement:	NetworkOut	
Trigger Statistic:	Average	
Unit of Measurement:	Bytes	
Measurement Period (minutes):	5	(1 - 600)
Breach Duration (minutes):	5	(1 - 600)
Upper Threshold:	6000000	(0 - 20000000)
Upper Breach Scalement Increment:	1	
Lower Threshold:	2000000	(0 - 20000000)
Lower Breach Scalement Increment:	-1	

Auto Scaling triggers work by watching a specific Amazon CloudWatch metric for an instance. Triggers include CPU utilization, network traffic, and disk activity. Use the **Trigger Measurement** drop-down list box to select a metric for your trigger.

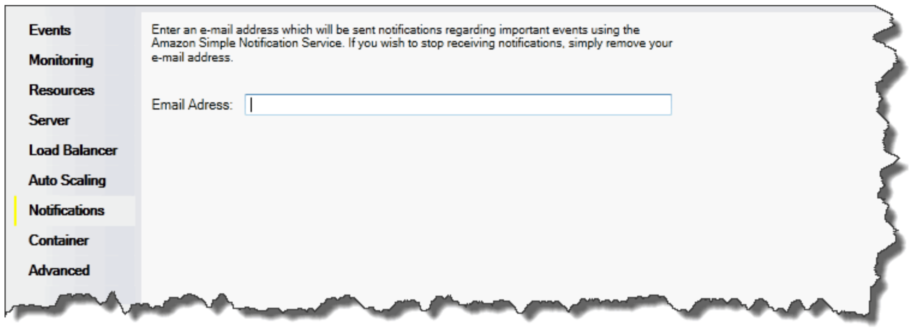
The following list describes the trigger parameters you can configure using the AWS Management Console.

- You can specify which statistic the trigger should use. You can select **`Minimum`**, **`Maximum`**, **`Sum`**, or **`Average`** using the **Trigger Statistic** drop-down list.
- Specify the unit for the trigger measurement using the **Unit of Measurement** drop-down list.
- The value in the **Measurement Period** text box specifies how frequently Amazon CloudWatch measures the metrics for your trigger. The **Breach Duration** is the amount of time a metric can be beyond its defined limit (as specified in the **Upper Threshold** and **Lower Threshold** text boxes) before the trigger fires.
- The **Upper Breach Scale Increment** and **Lower Breach Scale Increment** text boxes specify how many Amazon EC2 instances to add or remove when performing a scaling activity.

For more information on Auto Scaling, go to the Auto Scaling documentation.

Configuring Notifications Using AWS Toolkit for Visual Studio

AWS Elastic Beanstalk uses the Amazon Simple Notification Service (Amazon SNS) to notify you of important events affecting your application. To enable Amazon SNS notifications, simply enter your email address in the **Email Address** text box. To disable Amazon SNS notifications, remove your email address from the text box.



Enter an e-mail address which will be sent notifications regarding important events using the Amazon Simple Notification Service. If you wish to stop receiving notifications, simply remove your e-mail address.

Email Adress:

Configuring .NET Containers Using AWS Toolkit for Visual Studio

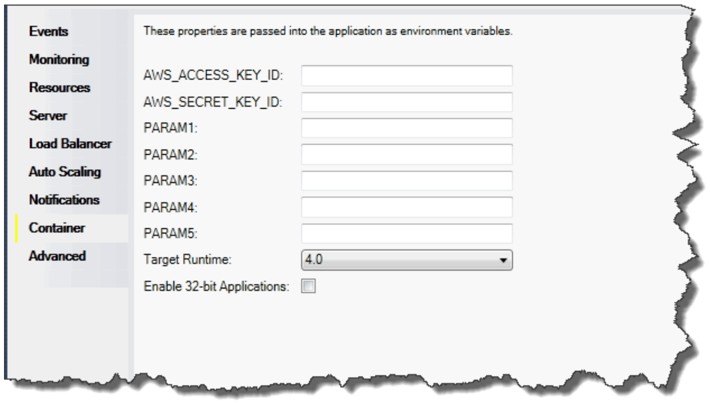
The **Container/.NET Options** panel lets you fine-tune the behavior of your Amazon EC2 instances and enable or disable Amazon S3 log rotation. You can use the AWS Toolkit for Visual Studio to configure your container information.

Note

You can modify your configuration settings with zero downtime by swapping the CNAME for your environments. For more information, see Deploying Versions with Zero Downtime (p. 143).

To access the Container/.NET Options panel for your AWS Elastic Beanstalk application

1. In AWS Toolkit for Visual Studio, expand the AWS Elastic Beanstalk node and your application node.
2. In **AWS Explorer**, double-click your AWS Elastic Beanstalk environment.
3. At the bottom of the **Overview** pane, click the **Configuration** tab.
4. Under **Container**, you can configure container options.



These properties are passed into the application as environment variables.

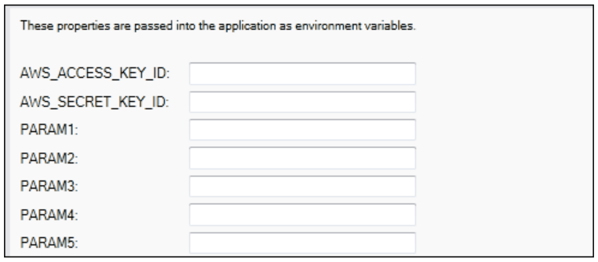
AWS_ACCESS_KEY_ID:
AWS_SECRET_KEY_ID:
PARAM1:
PARAM2:
PARAM3:
PARAM4:
PARAM5:
Target Runtime: 4.0
Enable 32-bit Applications: ☐

.NET Container Options

You can choose the version of .NET Framework for your application. You can choose either 2.0 or 4.0 in the **Target runtime** drop-down menu. Select **Enable 32-bit Applications** if you want to enable 32-bit applications.

Environment Properties

The **Environment Properties** section of the **Container** panel lets you specify environment variables on the Amazon EC2 instances that are running your application. This setting enables greater portability by eliminating the need to recompile your source code as you move between environments.


These properties are passed into the application as environment variables.

AWS_ACCESS_KEY_ID:	
AWS_SECRET_KEY_ID:	
PARAM1:	
PARAM2:	
PARAM3:	
PARAM4:	
PARAM5:	

You can use this panel to configure the following environment properties:

- Specify AWS credentials using the **AWS_ACCESS_KEY_ID** and **AWS_SECRET_KEY** text boxes.
- Specify up to five additional environment variables by entering them in the **PARAM** text boxes.

  You might have a code snippet that looks similar to the following to access the keys and parameters:

```
NameValueCollection appConfig = ConfigurationManager.AppSettings;

string accessKey = appConfig["AWSAccessKey"];
string secretKey = appConfig["AWSSecretKey"];
string param1 = appConfig["PARAM1"];
```

Note

Environment variables can contain any printable ASCII character except the grave accent (`, ASCII 96) and cannot exceed 200 characters in length.

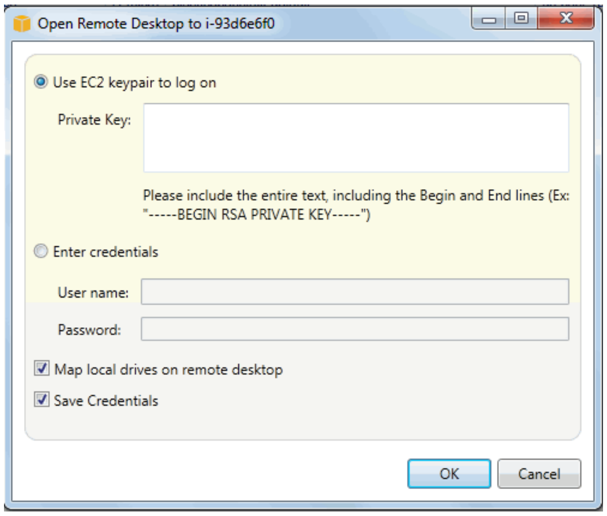
Listing and Connecting to Server Instances

You can view a list of Amazon EC2 instances running your AWS Elastic Beanstalk application environment through the AWS Toolkit for Visual Studio or from the AWS Management Console. You can connect to these instances using Remote Desktop Connection. For information about listing and connecting to your server instances using the AWS Management Console, see Listing and Connecting to Server Instances (p. 197). The following section steps you through viewing and connecting you to your server instances using the AWS Toolkit for Visual Studio.

To view and connect to Amazon EC2 instances for an environment

1. In Visual Studio, in **AWS Explorer**, expand the **Amazon EC2** node and double-click **Instances**.
2. Right-click the instance ID for the Amazon EC2 instance running in your application's load balancer in the **Instance** column and select **Open Remote Desktop** from the context menu.

   The Open Remote Desktop dialog box appears.


Open Remote Desktop to i-93d6e6f0
Use EC2 keypair to log on
Private Key:
Please include the entire text, including the Begin and End lines (Ex: "-----BEGIN RSA PRIVATE KEY-----")
Enter credentials
User name:
Password:
Map local drives on remote desktop
Save Credentials
OK
Cancel

3. Select **Use EC2 keypair to log on** and paste the contents of your private key file that you used to deploy your application in the **Private key** box. Alternatively, enter your user name and password in the **User name** and **Password** text boxes.

   Note

   If the key pair is stored inside the Toolkit, the text box does not appear.

4. Click **OK**.

Terminating an Environment

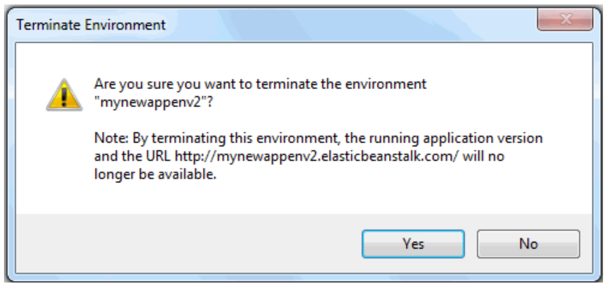
To avoid incurring charges for unused AWS resources, you can terminate a running environment using AWS Toolkit for Visual Studio.

Note

You can always launch a new environment using the same version later.

To terminate an environment

1. Expand the AWS Elastic Beanstalk node and the application node in AWS Explorer. Right-click your application environment, and select **Terminate Environment**.
2. In the dialog box that appears, click **Yes** to confirm that you want to terminate the environment. It will take a few minutes for AWS Elastic Beanstalk to terminate the AWS resources running in the environment.



Are you sure you want to terminate the environment "mynewappenv2"?

Note: By terminating this environment, the running application version and the URL http://mynewappenv2.elasticbeanstalk.com/ will no longer be available.

Note

When you terminate your environment, the CNAME associated with the terminated environment becomes available for anyone to use.

Tools

Topics

The AWS SDK for .NET makes it even easier for Windows developers to build .NET applications that tap into the cost-effective, scalable, and reliable AWS cloud. Using the SDK, developers will be able to build solutions for AWS infrastructure services, including Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), and Amazon SimpleDB. You can use the AWS Toolkit for Visual Studio to add the AWS .NET SDK to an existing project, or create a new .NET project based on the AWS .NET SDK.

AWS SDK for .NET

With the AWS SDK for .NET, you can get started in minutes with a single, downloadable package complete with Visual Studio project templates, the AWS .NET library, C# code samples, and documentation. You can build .NET applications on top of APIs that take the complexity out of coding directly against web services interfaces. The all-in-one library provides .NET developer-friendly APIs that hide much of the lower-level plumbing associated with programming for the AWS cloud, including authentication, request retries, and error handling. Practical examples are provided in C# for how to use the libraries to build applications. Online video tutorials and reference documentation are provided to help you learn how to use the libraries and code samples. For more information about the AWS SDK for .NET, go to http://aws.amazon.com/sdkfornet/.

AWS Toolkit for Visual Studio

The AWS Toolkit for .NET is a plug-in for Visual Studio that makes it easier to develop and debug .NET applications using Amazon Web Services. You can get started quickly with building solutions for the AWS cloud using Visual Studio Project Templates. For more information on prerequisites, installation instructions and running code samples, go to http://aws.amazon.com/visualstudio/.

Deploying AWS Elastic Beanstalk Applications in .NET Using the Deployment Tool

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

The deployment tool supports both initial deployments and redeployments. If you previously deployed your application using the deployment tool, you can redeploy using the deployment wizard within Visual Studio. Similarly, if you have deployed using the wizard, you can redeploy using the deployment tool.

This chapter walks you through deploying a sample .NET application to AWS Elastic Beanstalk using the deployment tool, and then redeploying the application using an incremental deployment. For a more in-depth discussion about the deployment tool, including the parameter options, go to Deployment Tool.

Prerequisites

In order to deploy your web application using the deployment tool, you need to package it as a .zip file. For more information about how to package your application for deployment, go to How to: Deploy a Web Application Project Using a Web Deployment Package at MSDN.

To use the deployment tool, you need to install the AWS Toolkit for Visual Studio. For information on prerequisites and installation instructions, go to http://aws.amazon.com/visualstudio/.

The deployment tool is typically installed in one of the following directories on Windows:

32-bit	64-bit
C:\Program Files\AWS Tools\Deployment Tool\awsdeploy.exe	C:\Program Files (x86)\AWS Tools\Deployment Tool\awsdeploy.exe

Deploy to AWS Elastic Beanstalk

To deploy the sample application to AWS Elastic Beanstalk using the deployment tool, you first need to modify the **ElasticBeanstalkDeploymentSample.txt** configuration file, which is provided in the **Samples** directory. This configuration file contains the information necessary to deploy your application, including the application name, application version, environment name, and your AWS Access Credentials. After modifying the configuration file, you then use the command line to deploy the sample application. Your web deploy file is uploaded to Amazon S3 and registered as a new application version with Elastic Beanstalk. It will take a few minutes to deploy your application. Once the environment is healthy, the deployment tool outputs a URL for the running application.

To deploy a .NET application to AWS Elastic Beanstalk

1. From the **Samples** subdirectory where the deployment tool is installed, open **ElasticBeanstalkDeploymentSample.txt** and enter your **AWS Access Key** and **AWS Secret Key** as in the following example. You can find your AWS Access Key and AWS Secret Key at Access Credentials.

```
### AWS Access Key and Secret Key used to create and deploy the application
 instance
AWSAccessKey = AKIAIOSFODNN7EXAMPLE
AWSSecretKey = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

2. At the command line prompt, type the following:

```
C:\Program Files (x86)\AWS Tools\Deployment Tool>awsdeploy.exe /w
Samples\ElasticBeanstalkDeploymentSample.txt
```

   It takes a few minutes to deploy your application. If the deployment succeeds, you will see the message, **Application deployment completed; environment health is Green**.

   Note

   If you receive the following error, the CNAME already exists.

```
[Error]: Deployment to AWS Elastic Beanstalk failed with exception:
DNS name (MyAppEnv.elasticbeanstalk.com) is not available.
```

Because a CNAME must be unique, you need to change **Environment.CNAME** in **ElasticBeanstalkDeploymentSample.txt**.

3. In your web browser, navigate to the URL of your running application. The URL will be in the form <CNAME.elasticbeanstalk.com> (i.e., MyAppEnv.elasticbeanstalk.com).

Redeploy to AWS Elastic Beanstalk

You can redeploy your application using an incremental deployment. Incremental deployments are faster because you are updating only the files that have changed instead of all the files. This section walks you through redeploying the sample application you deployed in Deploy to AWS Elastic Beanstalk (p. 89).

To edit and redeploy a .NET application to AWS Elastic Beanstalk

1. Extract **AWSDeploymentSampleApp.zip** from the Samples directory to a location on your computer such as **c:\mydeploymentarchive\AWSDeploymentSampleApp**.
2. Modify one of the files in the AWSDeploymentSampleApp directory. For example, you can modify the title in default.aspx.
3. In the **ElasticBeanstalkDeploymentSample.txt** configuration file, do the following:

   - Specify the location where you extracted the files. This means modifying the value for the **DeploymentPackage** key in the **Incremental Deployment Settings** section in **ElasticBeanstalkDeploymentSample.txt**. For example:

   ```
   C:\mydeploymentarchive\AWSDeploymentSampleApp
   ```

   - Remove the **#** in front of **IncrementalPushRepository** and **DeploymentPackage**.
   - Add a **#** in front of **DeploymentPackage** in the **Non-Incremental Deployment Settings**.

4. At the command line, type the following:

   ```
   C:\Program Files (x86)\AWS Tools\Deployment Tool>awsdeploy.exe /r
   Samples\ElasticBeanstalkDeploymentSample.txt
   ```

   If this command succeeds, you should see something similar to the following:

   ```
   ...environment 'MyAppEnvironment' found and available for redeployment
   (configuration parameters not required for redeployment will be ignored)
   ...starting redeployment to AWS Elastic Beanstalk environment 'MyAppEnviron
   ment'
   ...starting incremental deployment to environment 'MyAppEnvironment'
   ...finished incremental deployment in 9199.9199 ms
   ```

5. In your web browser, navigate to the same URL as in Deploy to AWS Elastic Beanstalk (p. 89). You should see your updated application.

Resources

There are several places you can go to get additional help when developing your .NET applications:

Resource	Description
.NET Development Forum	Post your questions and get feedback.
.NET Developer Center	One-stop shop for sample code, documentation, tools, and additional resources.
AWS SDK for .NET FAQs	Get answers to commonly asked questions.

Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git

Topics

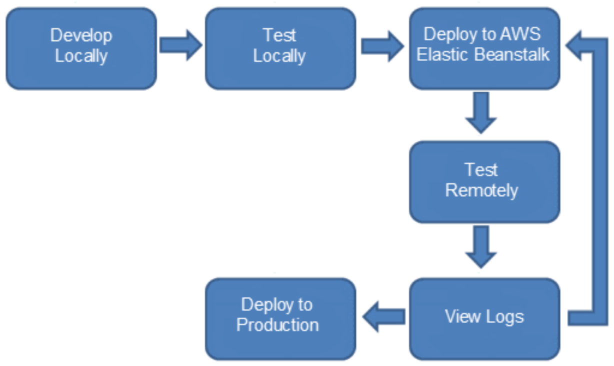
AWS Elastic Beanstalk for PHP makes it easy to deploy, manage, and scale your PHP web applications using Amazon Web Services. AWS Elastic Beanstalk for PHP is available to anyone developing or hosting a web application using PHP. This section provides step-by-step instructions for deploying your PHP web application to AWS Elastic Beanstalk using eb (an updated command line interface) and Git. To complete this walkthrough, you will need to download the command line tools at the AWS Sample Code & Libraries website. For instructions on how to get set up, see Get Set Up (p. 93). You can also use the AWS Management Console, CLIs, or APIs to upload your PHP files using a ZIP file. For more information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

After you deploy your AWS Elastic Beanstalk application, you can use the AWS Management Console, CLIs, or the APIs to manage your AWS Elastic Beanstalk environment. For more information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Develop, Test, and Deploy

Topics

The following diagram illustrates a typical software development life cycle including deploying your application to AWS Elastic Beanstalk.



Typically, after developing and testing your application locally, you will deploy your application to AWS Elastic Beanstalk. At this point, your application will be live at a URL such as http://myexampleapp-wpams3yrvj.elasticbeanstalk.com. Because your application will be live, you should consider setting up multiple environments, such as a testing environment and a production environment. You can point your domain name to the Amazon Route 53 (a highly available and scalable Domain Name System (DNS) web service) CNAME <*yourappname*>.elasticbeanstalk.com. Contact your DNS provider to set this up. For information about how to map your root domain to your Elastic Load Balancer, see Using AWS Elastic Beanstalk with Amazon Route 53 to Map Your Root Domain to Your Load Balancer (p. 221). After you remotely test and debug your AWS Elastic Beanstalk application, you can then make any updates and redeploy to AWS Elastic Beanstalk. After you are satisfied with all of your changes, you can upload your latest version to your production environment. The following sections provide more details explaining each stage of the software development life cycle.

Get Set Up

A Git client extension, AWS DevTools, is available as part of the CLI package. It enables you to deploy applications to AWS Elastic Beanstalk quickly. Before you can use the command line interface, you will need to install the following prerequisite software, and then intialize your Git repository.

To install the prerequisite software and initialize your Git repository

1. Install the following software onto your local computer:
   - Git. To download Git, go to http://git-scm.com/. Make sure you have at least version 1.6.6 or later.
   - For Linux, download Ruby version 1.8.7 or later. To view and download Ruby clients, go to http://www.ruby-lang.org/en/.
   - For Windows, download PowerShell 2.0. Windows 7 and Windows Server 2008 R2 come with PowerShell 2.0. For earlier versions of Windows, you can download PowerShell 2.0. Visit http://technet.microsoft.com/en-us/scriptcenter/dd742419.aspx for more details.

2. Initialize your Git repository. In this example, we use the following command from the directory where we installed the command line interface:

```
git init .
```

Develop Locally

After installing eb on your local computer, you use the Git command line as you normally would to create your local repository and add and commit changes. You create your PHP application as you normally would with your favorite editor. If you don't already have a PHP application ready, you can use a simple "Hello World" application. Type the following program into your favorite editor, and save it as a PHP file.

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

Next, create a new local repository, add your new program, and commit your change.

```
git add index.php
git commit -m "initial check-in"
```

Note

For information about Git commands, go to Git - Fast Version Control System.

Test Locally

Normally, at this point you would test your application locally before deploying to AWS Elastic Beanstalk. Suppose you find a few issues you would like to fix. Using the above "Hello World" application, add a "!" after "Hello World" and check in your changes. Update your index.php file, and then type the following commands to check in your updated file.

```
git add index.php
git commit -m "my second check-in"
```

After you commit your changes, you should see a response similar to the following:

```
[master 0535814] my second check-in
1 files changed, 1 insertions(+), 1 deletions(-)
```

Note the commit ID that is generated. This ID is used to generate a version label for your application.

Deploy to AWS Elastic Beanstalk

After testing your application, you are ready to deploy it to AWS Elastic Beanstalk. Deploying requires the following steps:

- Configure AWS Elastic Beanstalk.

- Deploy a sample application.
- Update the sample application with your application.

When you update the sample application with your application, AWS Elastic Beanstalk replaces the existing sample application version with your new application version in the existing environment.

Use the `init` command, and AWS Elastic Beanstalk will prompt you to enter this information. If a default value is available, and you want to use it, press **Enter**.

To configure AWS Elastic Beanstalk

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb init
   ```

2. When you are prompted for the AWS access key, type your access key. To get your access key information, go to Access Credentials.

   ```
   Enter your AWS Access Key ID (current value is "AKIAIOSFODNN7EXAMPLE"):
   ```

3. When you are prompted for the AWS secret key, type your secret key. To get your secret key information, go to Access Credentials.

   ```
   Enter your AWS Secret Access Key (current value is "wJalrXUtnFEMI/K7MDENG/bPxR
   fiCYEXAMPLEKEY"):
   ```

4. When you are prompted for the AWS Elastic Beanstalk region, type the number of the region. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference.

   ```
   Select (1 to 5): 1
   ```

5. When you are prompted for the AWS Elastic Beanstalk application name, type the name of the application. AWS Elastic Beanstalk auto-generates an application name based on the current directory name if an application name has not been previously configured. In this example, we use HelloWorld.

   ```
   Enter an AWS Elastic Beanstalk application name (auto-generated value is
   "windows"): HelloWorld
   ```

   Note

   If you have a space in your application name, make sure you do not use quotes.

6. When you are prompted for the AWS Elastic Beanstalk environment name, type the name of the environment. AWS Elastic Beanstalk automatically creates an environment name based on your application name. If you want to accept the default, press **Enter**.

   ```
   Enter an AWS Elastic Beanstalk environment name (current value is "HelloWorld-
   env"):
   ```

> **Note**
>
> If you have a space in your application name, make sure you do not have a space in your environment name.

7. When you are prompted for the solution stack, type the number of the solution stack you want. For this example, we'll use **64bit Amazon Linux running PHP 5.3**.

```
Select (1 to 7): 6
```

If you want to update your AWS Elastic Beanstalk configuration, you can use the **`init`** command again. When prompted, you can update your configuration options. If you want to keep any previous settings, press the **Enter** key.

After configuring AWS Elastic Beanstalk, you are ready to deploy a sample application.

To deploy a sample application

- From your directory where you installed the command line interface, type the following command.

```
eb start
```

  This process may take several minutes to complete. AWS Elastic Beanstalk will provide status updates during the process. If at any time you want to stop polling for status updates, press **Ctrl+C**. Once the environment status is Green, AWS Elastic Beanstalk will output a URL for the application. You can copy and paste the URL into your web browser to view the application.

To update the sample application with your local application

1. Type the following command.

```
git aws.push
```

2. If everything worked as expected, you should see something similar to the following:

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects:100% (2/2), done.
Writing objects: 100% (3/3), 298 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To https://<some long string>@git.elasticbeanstalk.us-east-
1.amazon.com/helloworld/helloworldenv
 44c7066..b1f11a1 master -> master
```

3. Verify that your application has been updated by refreshing your web browser.

> **Note**
>
> The running version is updated and begins with the commit ID from your last commit.

Test Remotely

To investigate any issues, you can connect to your Amazon EC2 instance. For instructions on how to connect to your instance, see Listing and Connecting to Server Instances (p. 197). You can also view logs to help you with debugging. For more information about viewing logs, see Debug/View Logs (p. 97).

Debug/View Logs

You can configure your environment so that the logs from the Amazon EC2 instances running your applications are copied by AWS Elastic Beanstalk to the Amazon S3 bucket associated with your application. For instructions on how to view these logs from the AWS Management Console, see Working with Logs (p. 199).

Edit the Application and Redeploy

Now that you have tested your application, it is easy to edit your application, redeploy, and see the results in moments. First, make changes to your application and commit your changes. Then deploy a new application version to your existing AWS Elastic Beanstalk environment.

```
git add index.php
git commit -m "my third check-in"
git aws.push
```

A new application version will be uploaded to your AWS Elastic Beanstalk environment.

You can use the AWS Management Console, CLIs, or APIs to manage your AWS Elastic Beanstalk environment. For more information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Deploy to Production

When you are satisfied with all of the changes you want to make to your application, you can deploy it to your production environment. First, you'll commit and push your changes, and then you can update your application in your production environment using eb. When you update your application using eb, AWS Elastic Beanstalk will create a new application version. For information on how to deploy an already existing application version to a new environment, see Launching New Environments (p. 135). The following steps walk you through committing your new changes and then updating your environment with a new application version using eb.

To deploy to production using eb

1. Check in final changes.

   ```
   git add index.php
   git commit -m "final check-in"
   ```

2. Deploy your application to AWS Elastic Beanstalk.

   ```
   git aws.push
   ```

Note

Make sure when you call `git aws.push` that your directory is not empty. You cannot push empty directories.

3. To launch a new environment, type the following command:

   ```
   eb start -e helloworldprod
   ```

Deploy an Existing Application Version to an Existing Environment

If you need to deploy an existing application to an existing environment, you can do so using the AWS Management Console, CLI, or APIs. You may want to do this if, for instance, you need to roll back to a previous application version. For instructions on how to deploy an existing application version to an existing environment, see Deploying Versions to Existing Environments (p. 141).

Configuring PHP Containers Using Eb

You can fine-tune the behavior of your Amazon EC2 instances and enable or disable Amazon S3 log rotation. This topic walks you through how to update your environment with your container settings using eb. You can also use the AWS Management Console, CLI, or the API to configure these options. For instructions, see Configuring PHP Containers with AWS Elastic Beanstalk (p. 186).

Configuring Containers Using Eb

To update your AWS Elastic Beanstalk environment

1. Open the **OPTIONSETTINGS** file in the `.elasticbeanstalk` directory.
2. Type value for the option name of the namespace. For example, type **2** for the **maxsize** option name of the **aws:autoscaling:asg** namespace.
3. Type the following command:

   ```
   eb update
   ```

   AWS Elastic Beanstalk displays a message once it has successfully updated the environment.

Container Options

You can specify the following settings in the OPTIONSETTINGS file.

- **Document Root** — Enables you to specify the child directory of your project that is treated as the public-facing web root. If your root document is stored in your project directory, leave this set to "/". If your root document is inside a child directory (e.g., <*project*>/public), set this value to match the child directory. Values should begin with a "/" character, and may NOT begin with a "." character. (This value is written to the http-vhosts.conf file.)
- **Memory Limit** — Specifies the amount of memory allocated to the PHP environment. (This value is written to the php.ini file.)

- **Zlib Output Compression** — Specifies whether PHP should use compression for output. (This value is written to the php.ini file.)
- **Enable log file rotation to Amazon S3** — Specifies whether log files for your application's Amazon EC2 instances should be copied to your Amazon S3 bucket associated with your application on an hourly basis.
- **Allow URL Fopen** — Specifies whether the PHP's file functions are allowed to retrieve data from remote locations, such as websites or FTP servers. (This value is written to the php.ini file.)
- **Display Errors** —Specifies whether error messages should be part of the output. (This value is written to the php.ini file.)
- **Max Execution Time** — Sets the maximum time, in seconds, a script is allowed to run before it is terminated by the environment. This helps prevent poorly written scripts from tying up the server.

For a list of namespaces, option names, and default values for these settings, see PHP Container Options (p. 284).

Environment Properties

Environment properties lets you specify environment configuration settings on the Amazon EC2 instances that are running your application. Environment properties are passed in as key-value pairs to the application.

You can configure the following environment configuration settings as part of the `aws:elasticbeanstalk:application:environment` namespace:

- Specify AWS credentials: **AWS_ACCESS_KEY_ID** and **AWS_SECRET_KEY**.
- Specify up to five additional environment configuration settings, for example VAR1.

Note

Environment configuration settings can contain any printable ASCII character except the grave accent (`, ASCII 96) and cannot exceed 200 characters in length.

Accessing Environment Properties

Inside the PHP environment running in AWS Elastic Beanstalk, these values are written to /etc/php.d/environment.ini and are accessible using PHP's `get_cfg_var()` function.

Note

There is a subtle difference in behavior between `get_cfg_var()` and `ini_get()`. The function, `ini_get()`, will only return values that are defined by PHP core or a PHP extension, and will not return custom values. The function, `get_cfg_var()`, however, will successfully return the values of custom settings.

You might have a code snippet that looks similar to the following to access the keys and parameters:

```
echo get_cfg_var('aws.access_key');
echo get_cfg_var('aws.secret_key');
echo get_cfg_var('aws.var1');
```

Using Amazon RDS with PHP

Amazon Relational Database Service (Amazon RDS) lets you quickly and easily provision and maintain a MySQL Server instance in the cloud. This topic discusses how you can use Amazon RDS and PHP with your AWS Elastic Beanstalk application.

Note

For more information on AWS storage options, go to Storage Options in the AWS Cloud.

To use Amazon RDS from your AWS Elastic Beanstalk application, you need to do the following:

- Create an Amazon RDS DB Instance.
- Configure your Amazon RDS DB Security Group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application.
- If you plan to use PDO, install the PDO drivers.
- Establish a database connection in your PHP code using your Amazon RDS DB Instance's public DNS name.

To use Amazon RDS and PHP from your AWS Elastic Beanstalk application

1. Create an Amazon RDS DB Instance. For instructions on how to do this, go to the Amazon RDS Getting Started Guide.
2. Configure your Amazon RDS DB Security Group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application. For instructions on how to find the name of your EC2 security group using the AWS Management Console, see Amazon EC2 Security Groups (p. 161). For more information, go to the "Authorizing Network Access to an Amazon EC2 Security Group" section of Working with DB Security Groups in the *Amazon Relational Database Service User Guide*.
3. If you plan to use PDO, install the PDO drivers. For more information, go to http://www.php.net/manual/pdo.installation.php.
4. Establish a database connection in your PHP code using your Amazon RDS DB Instance's public DNS name, port number, and (optionally) database name and login credentials. The following examples show examples that would connect to the employee database on an RDS instance at mydbinstance.abcdefghijkl.us-east-1.rds.amazonaws.com using port 3306, with the user name "sa" and the password "mypassword".

   Example 1. Example using PDO to connect to an RDS database

   ```
   <?php
   $dsn = 'mysql:host=mydbinstance.abcdefghijkl.us-east-1.rds.amazon
   aws.com;port=3306;dbname=mydb';
   $username = 'sa';
   $password = 'mypassword';

   $dbh = new PDO($dsn, $username, $password);
   ?>
   ```

   For more information about constructing a connection string using PDO, go to http://www.phpbuilder.com/manual/pdo.construct.php.

Example 2. Example using mysql_connect() to connect to an RDS database

```
<?php
$dbhost = 'mydbinstance.abcdefghijkl.us-east-1.rds.amazonaws.com:3306';
$username = 'sa';
$password = 'mypassword';
$dbname = 'mydb';

$link = mysql_connect($dbhost, $username, $password, $dbname);
mysql_select_db($dbname);
?>
```

For more information on using `mysql_connect()`, go to
http://www.phpbuilder.com/manual/function.mysql-connect.php.

Example 3. Example using mysqli_connect() to connect to an RDS database

```
$link = mysqli_connect('mydbinstance.abcdefghijkl.us-east-1.rds.amazon
aws.com', 'sa', 'mypassword', 'mydb', 3306);
```

For more information on using `mysqli_connect()`, go to
http://www.phpbuilder.com/manual/function.mysqli-connect.php.

5. Deploy your application to AWS Elastic Beanstalk. For information on how to deploy your application using AWS Elastic Beanstalk and the AWS Management Console, see Getting Started Using AWS Elastic Beanstalk (p. 5). For information on how to deploy your application using AWS DevTools, see Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git (p. 92).

Tools

Topics

The AWS SDK for PHP makes it even easier for developers to build PHP applications that tap into the cost-effective, scalable, and reliable AWS cloud. Using the SDK, developers will be able to build solutions for AWS infrastructure services, including Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), and Amazon SimpleDB. You can create a new project based on the AWS SDK for PHP.

If you are familiar with the Git command line, you can use AWS DevTools to help you deploy your applications to AWS Elastic Beanstalk.

AWS SDK for PHP

With the AWS SDK for PHP, you can get started in minutes with a single, downloadable package complete with the AWS PHP library, code samples, and documentation. You can build PHP applications on top of APIs that take the complexity out of coding directly against web services interfaces. The all-in-one library provides PHP developer-friendly APIs that hide much of the lower-level tasks associated with programming for the AWS cloud, including authentication, request retries, and error handling. Practical examples are provided in PHP for how to use the libraries to build applications. Online video tutorials and reference documentation are provided to help you learn how to use the libraries and code samples. For more information about the AWS SDK for PHP, go to http://aws.amazon.com/sdkforphp/. For instructions on installing the AWS SDK for PHP, go to Installation.

Git Deployment Via Eb

Eb is an updated command line interface for AWS Elastic Beanstalk that enables you to deploy applications quickly and more easily. To learn how to get started deploying a PHP application to AWS Elastic Beanstalk using eb and Git, see Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git (p. 92).

Resources

There are several places you can go to get additional help when developing your PHP applications:

Resource	Description
GitHub	Install the AWS SDK for PHP using GitHub.
PHP Development Forum	Post your questions and get feedback.
PHP Developer Center	One-stop shop for sample code, documentation, tools, and additional resources.
AWS SDK for PHP FAQs	Get answers to commonly asked questions.

Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git

Topics

AWS Elastic Beanstalk for Python makes it easy to deploy, manage, and scale your Python web applications using Amazon Web Services. AWS Elastic Beanstalk is available to anyone developing or hosting a web application using Python. This section provides step-by-step instructions for deploying a sample application to AWS Elastic Beanstalk using eb (an updated command line interface) and Git, and then updating the application to use the Django and Flask web application frameworks. To complete this walkthrough, you will need to download the command line tools at the AWS Sample Code & Libraries website, and optionally you can set up a Python development environment. For information on setting up a Python development environment, go to virtualenv.

After you deploy your AWS Elastic Beanstalk application, you can use the AWS Management Console, CLIs, or the APIs to manage your AWS Elastic Beanstalk environment. For more information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Customizing and Configuring a Python Container

When deploying your Python application, you may want to customize and configure the behavior of your EC2 instances. You can easily customize your instances at the same time that you deploy your application version by including a configuration file with your source bundle. This section walks you through the process of creating a configuration file and bundling it with your source.

To customize and configure your Python container

1. Create a configuration file with the extension **.config** and place it in an **.ebextensions** top-level directory of your source bundle. You can have multiple configuration files in your **.ebextensions** directory. The configuration file specifies the following information:

   - **packages** — You can use the packages key to download and install pre-packaged applications and components. Supported packages include apt, yum, rubygems, python and rpm.
   - **container_commands** — - You can use the container_commands key to execute commands for your container. Container_commands are processed in alphabetical order by name. They run after the application and web server have been setup and the application version file has been extracted, but before the application version is deployed. They also have access to environment variables. Additionally, you can use **leader_only**. One instance is chosen to be the leader in an Auto Scaling group. If the **leader_only** value is set to true, then the command only run on the instance that is marked as the leader. In the following example, the syncdb/migrate commands run once, instead of once per instance.
   - **option_settings** — You can use the option_settings key to define container settings. The option settings support the following namespaces:
     - aws:elasticbeanstalk:container:python:environment
     - aws:elasticbeanstalk:container:python
     - aws:elasticbeanstalk:container:python:staticfiles

     For more information about the Python container options, see Python Container Options (p. 285). The **option_settings** key supports two values, a list or a mapping. In the following example, a mapping is shown. The mapping syntax has the namespace as the key, and the value is another mapping where the key is the option name and the value is the value of the option name.

   The following is an example snippet of a configuration file.

```
packages:
  yum:
    libmemcached-devel: '0.31'

container_commands:
  collectstatic:
    command: "django-admin.py collectstatic --noinput"
  01syncdb:
    command: "django-admin.py syncdb --noinput"
    leader_only: true
  02migrate:
    command: "django-admin.py migrate"
    leader_only: true
  99customize:
    command: "scripts/customize.sh"

# You can specify any key-value pairs in the aws:elasticbeanstalk:contain
er:python:environment namespace and it will be
# passed in as environment variables on your EC2 instances
option_settings:
  "aws:elasticbeanstalk:container:python:environment":
    DJANGO_SETTINGS_MODULE: "djproject.settings"
    "application_stage": "staging"
  "aws:elasticbeanstalk:container:python":
    WSGIPath: djproject/wsgi.py
    NumProcesses: 3
    NumThreads: 20
  "aws:elasticbeanstalk:container:python:staticfiles":
    "/static/": "static/"
```

The following is an example of specifying the **option_settings** in a list format:

```
option_settings:
  - namespace: "aws:elasticbeanstalk:container:python"
    option_name: WSGIPath
    value: djproject/wsgi.py
```

Note

You can specify any key-value pairs in the `aws:elasticbeanstalk:container:python:environment` namespace and it will be passed in as environment variables on your EC2 instances.

2. Create a **requirements.txt** file and place it in the top-level directory of your source bundle. A typical python application will have dependencies on other third-party Python packages. In Python, pip is the standard way of installing packages. Pip has a feature that allows you to specify all the packages you need (as well as their versions) in a single requirements file. For more information about the requirements file, go to Requirements files. The following is an example requirements.txt file for Django.

```
Django==1.4.1
MySQL-python==1.2.3
```

From your working environment, you can also type the following command to generate the requirements file.

```
pip install django
pip install MySQL-python
pip freeze > requirements.txt
```

3. Deploy your application version.

For an example walkthrough of deploying a Django application using an instance configuration file, see Deploying a Django Application to AWS Elastic Beanstalk (p. 106). For an example walkthrough of deploying a Flask application, see Deploying a Flask Application to AWS Elastic Beanstalk (p. 115).

Accessing Environment Variables

Inside the Python environment running in AWS Elastic Beanstalk, these values are accessible using Python's `os.environ` function. For more information, go to http://docs.python.org/library/os.html.

You might have a code snippet that looks similar to the following to access the keys and parameters:

```
import os

os.environ['AWS_ACCESS_KEY'];
os.environ['AWS_SECRET_KEY'];
os.environ['DJANGO_SETTINGS_MODULE'];
```

Deploying a Django Application to AWS Elastic Beanstalk

This section walks you through deploying a sample application to AWS Elastic Beanstalk using eb (an updated command line interface) and Git, and then updating the application to use the Django framework. This example uses a configuration file to customize and configure the Python container. For more information about the configuration file, see Customizing and Configuring a Python Container (p. 103). The following frameworks have been tested with AWS Elastic Beanstalk:

- Django 1.4.1
- Django 1.3.3

Step 1: Set Up Your Git Repository

A Git client extension, AWS DevTools, is available as part of the CLI package. It enables you to deploy applications to AWS Elastic Beanstalk quickly. Before you can use the command line interface, you will need to install the following prerequisite software, and then intialize your Git repository.

To install the prerequisite software for your Git repository

- Install the following software onto your local computer:
  - Git. To download Git, go to http://git-scm.com/. Make sure you have at least version 1.6.6 or later.
  - For Linux, download Ruby version 1.8.7 or later. To view and download Ruby clients, go to http://www.ruby-lang.org/en/.
  - For Windows, download PowerShell 2.0. Windows 7 and Windows Server 2008 R2 come with PowerShell 2.0. For earlier versions of Windows, you can download PowerShell 2.0. Visit http://technet.microsoft.com/en-us/scriptcenter/dd742419.aspx for more details.

Step 2: Set Up Your Python Development Environment

Set up your Python development environment and then create the basic structure for your Python application. For more information, go to virtualenv. The following walks you through an example using an Amazon EC2 instance running Linux.

To set up your Python development environment

- Install the necessary packages. Python 2.7 or 3.0 is required for eb to run on the Linux and Mac operating systems. It is best practice to develop your application using the same python environment that your application will be running in production. Currently, AWS Elastic Beanstalk supports Python

2.6 running on the Amazon Linux 2012.03 AMI. In this example, we will set up two different environments: one for application development and one for deployment.

```
$ sudo su -
$ yum install -y python27 python27-devel
$ yum install -y gcc mysql mysql-devel
$ yum install -y python-devel
$ yum install -y git
$ wget https://s3.amazonaws.com/elasticbeanstalk/cli/AWS-ElasticBeanstalk-
CLI-2.1.zip
$ unzip AWS-ElasticBeanstalk-CLI-2.1.zip
$ easy_install pip
$ pip install -d . virtualenv
$ tar xvfz virtualenv-1.8.2.tar.gz
$ cd virtualenv-1.8.2
$ python2.7 setup.py install
```

Create a virtual environment for django development.

```
$ virtualenv –p python2.6 /tmp/djangodev
```

Activate virtualenv.

```
$ . /tmp/djangodev/bin/activate
```

Install django and mysql-python.

```
(djangodev)# pip install django
(djangodev)# pip install mysql-python
```

Create the django project structure.

```
(djangodev)# django-admin.py startproject mysite
(djangodev)# cd mysite
```

Freeze the requirements.txt file.

```
(djangodev)# pip freeze > requirements.txt
```

Note

Make sure your requirements.txt file contains the following:

```
Django==1.4.1
MySQL-python==1.2.3
```

Note

You can verify that the development server is working by typing the following command:

```
(djangodev)# python manage.py runserver
```

At this point, you should see the introductory Django page if you start up the dev server.

Be sure to stop the dev server before continuing.

Step 3: Configure AWS Elastic Beanstalk

Use the `init` command, and AWS Elastic Beanstalk will prompt you to enter this information. If a default value is available, and you want to use it, press **Enter**.

To configure AWS Elastic Beanstalk

1. Initialize your Git repository.

```
git init .
```

2. Create an alias to use eb with Python2.7. Specify the path where you installed eb. On a Linux operating system, it would look similar to the following example.

```
alias eb="python2.7 ../../AWS-ElasticBeanstalk-CLI-2.1/eb/linux/python2.7/eb"
```

3. From your directory where you installed the command line interface, type the following command.

```
eb init
```

4. When you are prompted for the AWS access key, type your access key. To get your access key information, go to Access Credentials.

```
Enter your AWS Access Key ID (current value is "AKIAIOSFODNN7EXAMPLE"):
```

5. When you are prompted for the AWS secret key, type your secret key. To get your secret key information, go to Access Credentials.

```
Enter your AWS Secret Access Key (current value is "wJalrXUtnFEMI/K7MDENG/bPxR
fiCYEXAMPLEKEY"):
```

6. When you are prompted for the AWS Elastic Beanstalk region, type the number of the region. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference. For this example, we'll use **US East (Virginia)**.
7. When you are prompted for the AWS Elastic Beanstalk application name, type the name of the application. AWS Elastic Beanstalk auto-generates an application name based on the current directory name if an application name has not been previously configured. In this example, we use HelloWorld.

```
Enter an AWS Elastic Beanstalk application name (auto-generated value is
"windows"): djangoapp
```

Note

If you have a space in your application name, make sure you do not use quotes.

8. When you are prompted for the AWS Elastic Beanstalk environment name, type the name of the environment. AWS Elastic Beanstalk automatically creates an environment name based on your application name. If you want to accept the default, press **Enter**.

   ```
   Enter an AWS Elastic Beanstalk environment name (current value is "djangoapp-
   env"):
   ```

   Note

   If you have a space in your application name, make sure you do not have a space in your environment name.
9. When you are prompted for the solution stack, type the number of the solution stack you want. For this example, we'll use **32bit Amazon Linux running Python**.
10. When you are prompted to create an Amazon RDS database, type **y** or **n**. For this example, we'll type **y**.

    ```
    Create an RDS DB Instance? [y/n]:
    ```

11. When you are prompted to create the database from scratch or a snapshot, type your selection. For this example, we'll use **No snapshot**.
12. When you are prompted to enter your RDS user master password, type your password containing from 8 to 16 printable ASCII characters (excluding /,", and @).

    ```
    Enter an RDS DB master password:
    ```

    ```
    Retype password to confirm:
    ```

13. When you are prompted to create a snapshot if you delete the Amazon RDS DB Instance, **y** or **n**. For this example, we'll type **n**. If you select **n**, then your RDS DB Instance will be deleted and your data will be lost if you terminate your environment.

    You should see a confirmation that your AWS Credential file was successfully updated.

    By default, eb sets the following default values for Amazon RDS.

    - **Database engine** — MySQL
    - **Default version:** — 5.5
    - **Database name:** — ebdb
    - **Allocated storage** — 5GB
    - **Instance class** — db.t1.micro
    - **Deletion policy** — delete
    - **Master username** — ebroot

After configuring AWS Elastic Beanstalk, you are ready to deploy a sample application.

If you want to update your AWS Elastic Beanstalk configuration, you can use the `init` command again. When prompted, you can update your configuration options. If you want to keep any previous settings, press the **Enter** key. If you want to update your Amazon RDS DB configuration settings, you can update your **optionsettings** file in the **.elasticbeanstalk** directory, and then use the `eb update` command to update your AWS Elastic Beanstalk environment.

Step 4: Create an Application

Next, you will create and deploy a sample application. For this step, you use a sample application that is already prepared. AWS Elastic Beanstalk uses the configuration information you specified in the previous step to do the following:

- Creates an application using the application name you specified.
- Launches an environment using the environment name you specified that provisions the AWS resources to host the application.
- Deploys the application into the newly created environment.

Use the `start` command to create and deploy a sample application.

To create an application

- From your directory where you installed the command line interface, type the following command.

  ```
  eb start
  ```

This process may take several minutes to complete. AWS Elastic Beanstalk will provide status updates during the process. If at any time you want to stop polling for status updates, press **Ctrl+C**. Once the environment status is Green, AWS Elastic Beanstalk will output a URL for the application.

Step 5: View Application

In the previous step, you created an application and deployed it to AWS Elastic Beanstalk. After the environment is ready and its status is Green, AWS Elastic Beanstalk provides a URL to view the application. In this step, you can check the status of the environment to make sure it is set to Green and then copy and paste the URL to view the application.

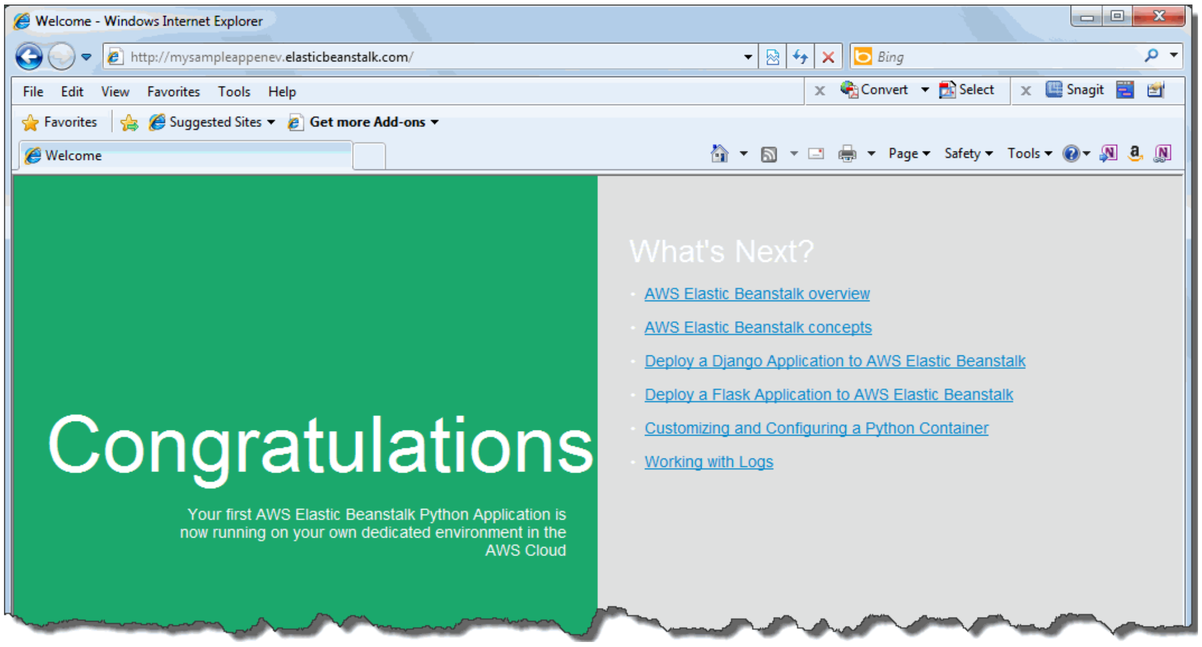
Use the `status` command to check the environment status, and then use the URL to view the application. Use `--verbose` to get detailed status information.

To view an application

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb status --verbose
   ```

   AWS Elastic Beanstalk displays the environment status. If the environment is set to Green, then AWS Elastic Beanstalk displays the URL for the application. If you attached an RDS DB Instance to your environment, your RDS DB information is displayed.
2. Copy and paste the URL into your web browser to view your application. You should see the following welcome page.


Welcome - Windows Internet Explorer
http://mysampleappenev.elasticbeanstalk.com/
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Get more Add-ons
Welcome
Page Safety Tools

Congratulations

Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud

What's Next?

- AWS Elastic Beanstalk overview
- AWS Elastic Beanstalk concepts
- Deploy a Django Application to AWS Elastic Beanstalk
- Deploy a Flask Application to AWS Elastic Beanstalk
- Customizing and Configuring a Python Container
- Working with Logs

Step 6: Update Application

After you have deployed a sample application, you can update it with your own application. In this step, we'll update the sample application to use the Django framework.

To update the sample application

1. Create an **.ebextensions** directory.

```
mkdir .ebextensions
```

2. Create a **python.config** file and place it in the **.ebextensions** directory. For more information about the configuration file, see Customizing and Configuring a Python Container (p. 103).

```

container_commands:
  01_syncdb:
    command: "django-admin.py syncdb --noinput"
    leader_only: true

option_settings:
  "aws:elasticbeanstalk:container:python:environment":
    DJANGO_SETTINGS_MODULE: "mysite.settings"
  "aws:elasticbeanstalk:container:python":
    WSGIPath: "mysite/wsgi.py"
```

In this example, the **syncdb** command is run on the command header during deployment. The environment variable, **DJANGO_SETTINGS_MODULE** is set to **mysite.settings**, and is available to the commands specified in the `commands` section. **WSGIPath** for mod_wsgi points to **mysite/wsgi.py**, which is one of the autogenerated files from the django-admin.py startproject command.

3. Edit **mysite/settings.py** to use Amazon RDS. AWS Elastic Beanstalk has built-in support for Amazon RDS, so when you launch a Python container, you can associate an Amazon RDS DB Instance. The values of the Amazon RDS parameters are available through environment variables.

```
import os

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.environ['RDS_DB_NAME'],
        'USER': os.environ['RDS_USERNAME'],
        'PASSWORD': os.environ['RDS_PASSWORD'],
        'HOST': os.environ['RDS_HOSTNAME'],
        'PORT': os.environ['RDS_PORT'],
    }
}
```

4. Add your files to your local Git repository, and then commit your change.

```
git add .
git commit -m "eb configuration"
```

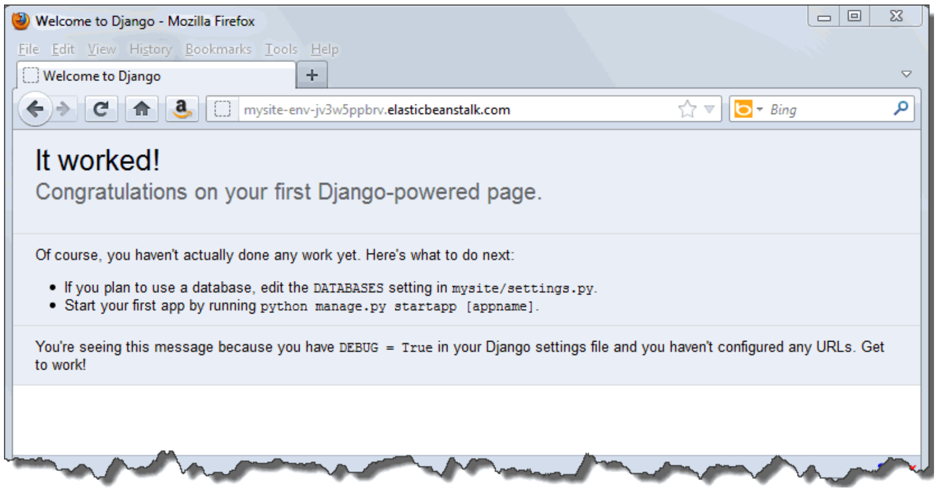
Note

For information about Git commands, go to Git - Fast Version Control System.

5. Deploy to AWS Elastic Beanstalk.

```
git aws.push
```

6. Use the `eb status --verbose` command to check your environment status. When your environment is green and ready, refresh your web browser to view your updated application. You should see the following.


Welcome to Django - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Welcome to Django
mysite-env-jv3w5ppbrv.elasticbeanstalk.com
Bing

It worked!
Congratulations on your first Django-powered page.

Of course, you haven't actually done any work yet. Here's what to do next:

- If you plan to use a database, edit the `DATABASES` setting in `mysite/settings.py`.
- Start your first app by running `python manage.py startapp [appname]`.

You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

Step 7: Configure the Django Admin Site (Optional)

Now that you have your Django application up and running, you can **optionally** configure the admin interface. For more information, go to The Django admin site.

To configure the Django admin site

1. Edit the **mysite/settings.py** file with the following:

   - Uncomment the following line in INSTALLED_APPS:

     ```
     'django.contrib.admin'
     ```

     Your snippet should look similar to the following:

     ```
     INSTALLED_APPS = (
         'django.contrib.auth',
         'django.contrib.contenttypes',
         'django.contrib.sessions',
         'django.contrib.sites',
         'django.contrib.messages',
         'django.contrib.staticfiles',
         # Uncomment the next line to enable the admin:
         'django.contrib.admin',
         # Uncomment the next line to enable admin documentation:
         # 'django.contrib.admindocs',
     )
     ```

   - Change the value of **STATIC_ROOT** to create a static/ directory in the top level of your source bundle that will contain static content:

     ```
     STATIC_ROOT = os.path.join(os.path.dirname(os.path.dirname(os.path.ab
     spath(__file__))), 'static')
     ```

2. Edit the **mysite/urls.py** to uncomment out four lines. The file should look like the following:

   ```
   from django.conf.urls import patterns, include, url

   # Uncomment the next two lines to enable the admin:
   from django.contrib import admin
   admin.autodiscover()

   urlpatterns = patterns('',
       # Examples:
       url(r'^$', 'mysite.views.home', name='home'),
       # url(r'^mysite/', include('mysite.foo.urls')),

       # Uncomment the admin/doc line below to enable admin documentation:
       # url(r'^admin/doc/', include('django.contrib.admindocs.urls')),

       # Uncomment the next line to enable the admin:
   ```

```
    url(r'^admin/', include(admin.site.urls)),
)
```

3. Create a file called **views.py** in the **/mysite** directory.

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello from django, try out <a href=”/admin/”>/ad
min/</a>\n")
```

4. Create a file called **createadmin.py** and place it in a directory called **scripts** in your top level directory. Make the script executable.

```
mkdir scripts
```

```
#!/usr/bin/env python

from django.contrib.auth.models import User
if User.objects.count() == 0:
    admin = User.objects.create(username='admin')
    admin.set_password('admin')
    admin.is_superuser = true
    admin.is_staff = true
    admin.save()
```

```
chmod +x scripts/createadmin.py
```

This creates a user with username/password of admin/admin if there are no users in the database yet.

5. Update **.ebextensions/python.config**.

```
container_commands:
  01_syncdb:
    command: "django-admin.py syncdb --noinput"
    leader_only: true
  02_createadmin:
    command: "scripts/createadmin.py"
    leader_only: true
  03_collectstatic:
    command: "django-admin.py collectstatic --noinput"
option_settings:
  "aws:elasticbeanstalk:container:python:environment":
    DJANGO_SETTINGS_MODULE: "mysite.settings"
  "aws:elasticbeanstalk:container:python":
    WSGIPath: "mysite/wsgi.py"
  "aws:elasticbeanstalk:container:python:staticfiles":
    "/static/": "static/"
```

The 3 commands will create the initial database tables, create a user with username "admin" password "admin", and collect all the static files used by the admin interface and bundle them in the "static/" directory of your app directory.

6. Add your files to your local Git repository, commit your change, and redeploy.

```
git add .
git commit -m "configure admin interface"
git aws.push
```

Step 8: Clean Up

If you no longer want to run your application, you can clean up by terminating your environment and deleting your application.

Use the `stop` command to terminate your environment and the `delete` command to delete your application.

To delete the application

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb stop
   ```

   This process may take a few minutes. AWS Elastic Beanstalk will display a message once the environment has been successfully terminated.

   Note

   If you attached an RDS DB Instance to your environment, then your RDS DB will be deleted, and you will lose your data. To save your data, create a snapshot before you delete the application. For instructions on how to create a snapshot, go to Creating a DB Snapshot in the *Amazon Relational Database Service User Guide*.

2. From your directory where you installed the command line interface, type the following command.

   ```
   eb delete
   ```

   AWS Elastic Beanstalk will display a message once it has successfully deleted the application.

Deploying a Flask Application to AWS Elastic Beanstalk

This section walks you through deploying a sample application to AWS Elastic Beanstalk using eb (an updated command line interface) and Git, and then updating the application to use the Flask framework. The following frameworks have been tested with AWS Elastic Beanstalk:

- Flask 0.9
- Flask 0.8

Step 1: Initialize Your Git Repository

A Git client extension, AWS DevTools, is available as part of the CLI package. It enables you to deploy applications to AWS Elastic Beanstalk quickly. Before you can use the command line interface, you will need to install the following prerequisite software, and then intialize your Git repository.

To install the prerequisite software and initialize your Git repository

1. Install the following software onto your local computer:
   - Git. To download Git, go to http://git-scm.com/. Make sure you have at least version 1.6.6 or later.
   - For Linux, download Ruby version 1.8.7 or later. To view and download Ruby clients, go to http://www.ruby-lang.org/en/.
   - For Windows, download PowerShell 2.0. Windows 7 and Windows Server 2008 R2 come with PowerShell 2.0. For earlier versions of Windows, you can download PowerShell 2.0. Visit http://technet.microsoft.com/en-us/scriptcenter/dd742419.aspx for more details.

2. Initialize your Git repository. In this example, we use the following command from the directory where we installed the command line interface:

```
git init .
```

Step 2: Configure AWS Elastic Beanstalk

Use the `init` command, and AWS Elastic Beanstalk will prompt you to enter this information. If a default value is available, and you want to use it, press **Enter**.

To configure AWS Elastic Beanstalk

1. From your directory where you installed the command line interface, type the following command.

```
eb init
```

2. When you are prompted for the AWS access key, type your access key. To get your access key information, go to Access Credentials.

```
Enter your AWS Access Key ID (current value is "AKIAIOSFODNN7EXAMPLE"):
```

3. When you are prompted for the AWS secret key, type your secret key. To get your secret key information, go to Access Credentials.

```
Enter your AWS Secret Access Key (current value is "wJalrXUtnFEMI/K7MDENG/bPxR
fiCYEXAMPLEKEY"):
```

4. When you are prompted for the AWS Elastic Beanstalk region, type the number of the region. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference. For this example, we'll use **US East (Virginia)**.

5. When you are prompted for the AWS Elastic Beanstalk application name, type the name of the application. AWS Elastic Beanstalk auto-generates an application name based on the current directory name if an application name has not been previously configured. In this example, we use HelloWorld.

   ```
   Enter an AWS Elastic Beanstalk application name (auto-generated value is
   "windows"): flaskapp
   ```

   Note

   If you have a space in your application name, make sure you do not use quotes.
6. When you are prompted for the AWS Elastic Beanstalk environment name, type the name of the environment. AWS Elastic Beanstalk automatically creates an environment name based on your application name. If you want to accept the default, press **Enter**.

   ```
   Enter an AWS Elastic Beanstalk environment name (current value is "flaskapp-
   env"):
   ```

   Note

   If you have a space in your application name, make sure you do not have a space in your environment name.
7. When you are prompted for the solution stack, type the number of the solution stack you want. For this example, we'll use **32bit Amazon Linux running Python**.
8. When you are prompted to create an Amazon RDS database, type **`Y`** or **`N`**. For this example, we'll type **N**.

   ```
   Create RDS instance? [y/n]: n
   ```

After configuring AWS Elastic Beanstalk, you are ready to deploy a sample application.

If you want to update your AWS Elastic Beanstalk configuration, you can use the **`init`** command again. When prompted, you can update your configuration options. If you want to keep any previous settings, press the **Enter** key.

Step 3: Create Application

Next, you will create and deploy a sample application. For this step, you use a sample application that is already prepared. AWS Elastic Beanstalk uses the configuration information you specified in the previous step to do the following:

- Creates an application using the application name you specified.
- Launches an environment using the environment name you specified that provisions the AWS resources to host the application.
- Deploys the application into the newly created environment.

Use the `start` command to create and deploy a sample application.

To create an application

- From your directory where you installed the command line interface, type the following command.

```
eb start
```

This process may take several minutes to complete. AWS Elastic Beanstalk will provide status updates during the process. If at any time you want to stop polling for status updates, press **Ctrl+C**. Once the environment status is Green, AWS Elastic Beanstalk will output a URL for the application.

Step 4: View Application

In the previous step, you created an application and deployed it to AWS Elastic Beanstalk. After the environment is ready and its status is Green, AWS Elastic Beanstalk provides a URL to view the application. In this step, you can check the status of the environment to make sure it is set to Green and then copy and paste the URL to view the application.

Use the `status` command to check the environment status, and then use the URL to view the application.

To view an application

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb status
   ```

   AWS Elastic Beanstalk displays the environment status. If the environment is set to Green, then AWS Elastic Beanstalk displays the URL for the application. If you attached an RDS DB Instance to your environment, your RDS DB information is displayed.
2. Copy and paste the URL into your web browser to view your application.

Step 5: Update Application

After you have deployed a sample application, you can update the sample application with your own application. In this step, we'll update the sample application with a simple "Hello World" Flask application.

To update the sample application

1. Create a requirements.txt file.

   ```
   Flask==0.9
   ```

   Note

   For more information about the requirements file, go to Requirements files.
2. Create an application.py file.

   ```
   import flask

   application = flask.Flask(__name__)

   #Set application.debug=true to enable tracebacks on Beanstalk log output.
   #Make sure to remove this line before deploying to production.
   application.debug=true
   ```

```
@application.route('/')
def hello_world():
    return "Hello world!"

if __name__ == '__main__':
    application.run(host='0.0.0.0', debug=true)
```

3. Add your two files to your local Git repository, and then commit your change.

   ```
   git add .
   git commit -m "update app"
   ```

   Note

   For information about Git commands, go to Git - Fast Version Control System.
4. Deploy to AWS Elastic Beanstalk.

   ```
   git aws.push
   ```

5. Refresh your web browser to view your updated application when your environment is ready.

Step6: Clean Up

If you no longer want to run your application, you can clean up by terminating your environment and deleting your application.

Use the `stop` command to terminate your environment and the `delete` command to delete your application.

To delete the application

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb stop
   ```

   This process may take a few minutes. AWS Elastic Beanstalk will display a message once the environment has been successfully terminated.

   Note

   If you attached an RDS DB Instance to your environment, then your RDS DB will be deleted, and you will lose your data. To save your data, create a snapshot before you delete the application. For instructions on how to create a snapshot, go to Creating a DB Snapshot in the *Amazon Relational Database Service User Guide*.
2. From your directory where you installed the command line interface, type the following command.

   ```
   eb delete
   ```

   AWS Elastic Beanstalk will display a message once it has successfully deleted the application.

Using Amazon RDS with Python

With Amazon Relational Database Service (Amazon RDS), you can quickly and easily provision and maintain a MySQL, Oracle, or Microsoft SQL Server instance in the cloud. This topic explains how you can use Amazon RDS and Python with your AWS Elastic Beanstalk application. For more information about Amazon RDS, go to http://aws.amazon.com/rds/.

To use Amazon RDS from your AWS Elastic Beanstalk application, you need to do the following:

1. Create an Amazon RDS DB Instance.
2. Establish a database connection in your Python code by using the connectivity information for your Amazon RDS DB Instance.
3. Update your **requirements.txt** file.
4. Deploy your application to AWS Elastic Beanstalk.

This topic walks you through the following:

- Using a new Amazon RDS DB Instance with Python
- Using an existing Amazon RDS DB Instance with Python

Using a New Amazon RDS DB Instance with Python

This topic walks you through creating a new Amazon RDS DB Instance and using it with your Python application.

To use a new Amazon RDS DB Instance and Python from your AWS Elastic Beanstalk application

1. Create an Amazon RDS DB Instance. You can create an RDS DB Instance in one of the following ways:

   - Create an RDS DB Instance when you create an application. For instructions using the AWS Elastic Beanstalk console, see Creating New Applications (p. 124). For a sample walkthrough of a Django application deployment with Amazon RDS using eb, see Deploying a Django Application to AWS Elastic Beanstalk (p. 106).
   - Create an RDS DB Instance when you launch a new environment. For instructions using the AWS Elastic Beanstalk console, see Launching New Environments (p. 135). If you use eb, use the `eb init` command to specify a new environment name and RDS configuration settings. For a sample walkthrough of a Django application deployment with Amazon RDS using eb, see Deploying a Django Application to AWS Elastic Beanstalk (p. 106).
   - If you already deployed an application to AWS Elastic Beanstalk, you can create an RDS DB Instance and attach it to an existing environment. For instructions using the AWS Elastic Beanstalk console, see Configuring Databases with AWS Elastic Beanstalk (p. 177). If you use eb, use the `eb init` command to specify RDS configuration settings, and then use the `eb update` command to update your environment.

2. Establish a database connection in your Python code using your Amazon RDS DB Instance's connectivity information. You can access your connectivity information using environment variables. The following is an example Django code snippet.

```
import os

if 'RDS_HOSTNAME' in os.environ:
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': os.environ['RDS_DB_NAME'],
            'USER': os.environ['RDS_USER'],
            'PASSWORD': os.environ['RDS_PASSWORD'],
            'HOST': os.environ['RDS_HOSTNAME'],
            'PORT': os.environ['RDS_PORT'],
        }
    }
```

3. Create a **requirements.txt** file and place it in the top-level directory of your source bundle. A typical python application will have dependencies on other third-party Python packages. In Python, pip is the standard way of installing packages. Pip has a feature that allows you to specify all the packages you need (as well as their versions) in a single requirements file. For more information about the requirements file, go to Requirements files. The following is an example requirements.txt file for Django.

```
Django==1.4.1
MySQL-python==1.2.3
```

4. Deploy your application to AWS Elastic Beanstalk. For information on how to deploy your application using AWS Elastic Beanstalk and the AWS Management Console, see Getting Started Using AWS Elastic Beanstalk (p. 5). For information on how to deploy your application using eb, see Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git (p. 103).

Using an Existing Amazon RDS DB Instance with Python

You can update your Python application to use an Amazon RDS DB Instance that you have previously created. This topic walks you through how to update your Python application using an existing Amazon RDS DB Instance and deploy your application to AWS Elastic Beanstalk.

To use an existing Amazon RDS DB Instance and Python from your AWS Elastic Beanstalk application

1. Create an AWS Elastic Beanstalk environment in one of the following ways:

   - Create a new application with a new environment. For instructions using the AWS Elastic Beanstalk console, see Creating New Applications (p. 124). When using the AWS Elastic Beanstalk console, you do not need to click **Create an RDS DB Instance with this environment** in the wizard because you already have an existing RDS DB Instance. For a sample walkthrough of a Django application deployment with Amazon RDS using eb, see Deploying a Django Application to AWS Elastic Beanstalk (p. 106).
   - Launch a new environment. For instructions using the AWS Elastic Beanstalk console, see Launching New Environments (p. 135). If you use eb, use the `eb init` command to specify a new environment name. You do not need to create a new RDS DB Instance when using the AWS Elastic Beanstalk console or eb because you already have an existing RDS DB Instance. For a sample walkthrough of a Django application deployment with Amazon RDS using eb, see Deploying a Django Application to AWS Elastic Beanstalk (p. 106).

2. Configure your Amazon RDS DB Security Group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application. For instructions on how to find the name of your EC2 security group using the AWS Management Console, see Amazon EC2 Security Groups (p. 161). For more information, go to the "Authorizing Network Access to an Amazon EC2 Security Group" section of Working with DB Security Groups in the *Amazon Relational Database Service User Guide*.
3. Establish a database connection in your Python code using your Amazon RDS DB Instance's connectivity information. The following is an example Django code snippet.

```
DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': 'exampledb',
            'USER': 'root',
            'PASSWORD': 'mypwd',
            'HOST': 'example.rds.amazonaws.com',
            'PORT': '3306',
        }
    }
```

4. Create a **requirements.txt** file, add the package you need to communicate to the database, and place it in the top-level directory of your source bundle. For more information about the requirements file, go to Requirements files. The following is an example requirements.txt file.

```
MySQL-python==1.2.3
```

5. Deploy your application to AWS Elastic Beanstalk. For information on how to deploy your application using AWS Elastic Beanstalk and the AWS Management Console, see Getting Started Using AWS Elastic Beanstalk (p. 5). For information on how to deploy your application using eb, see Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git (p. 103).

Tools

Topics

Boto, an open source AWS SDK for Python, makes it even easier for developers to build Python applications that tap into the cost-effective, scalable, and reliable AWS cloud. Using Boto, developers can build solutions for AWS infrastructure services, including Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), and Amazon DynamoDB.

You can use eb, an updated command line interface for AWS Elastic Beanstalk, to help you deploy your applications to AWS Elastic Beanstalk quickly and more easily.

Boto (open source AWS SDK for Python)

With Boto, you can get started in minutes with a single, downloadable package complete with the AWS Python library, code samples, and documentation. You can build Python applications on top of APIs that take the complexity out of coding directly against web services interfaces. The all-in-one library provides Python developer-friendly APIs that hide much of the lower-level tasks associated with programming for the AWS cloud, including authentication, request retries, and error handling. Practical examples are provided in Python for how to use the libraries to build applications. For information about Boto, sample code, documentation, tools, and additional resources, go to http://aws.amazon.com/python/.

Git Deployment Via Eb

Eb is an updated command line interface for AWS Elastic Beanstalk that enables you to deploy applications quickly and more easily. To learn how to get started deploying a Python application to AWS Elastic Beanstalk using eb and Git, see Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git (p. 103).

Resources

There are several places you can go to get additional help when developing your Python applications:

Resource	Description
Boto (open source AWS SDK for Python)	Install Boto using GitHub.
Python Development Forum	Post your questions and get feedback.
Python Developer Center	One-stop shop for sample code, documentation, tools, and additional resources.

Managing and Configuring Applications and Environments Using the Console, CLI, and APIs

Topics

This topic discusses some of the most important features of AWS Elastic Beanstalk in detail, including usage examples using the AWS Management Console, CLI, and the APIs. For more information about the CLI, see Tools (p. 270). For more information about the API, go to AWS Elastic Beanstalk API Reference.

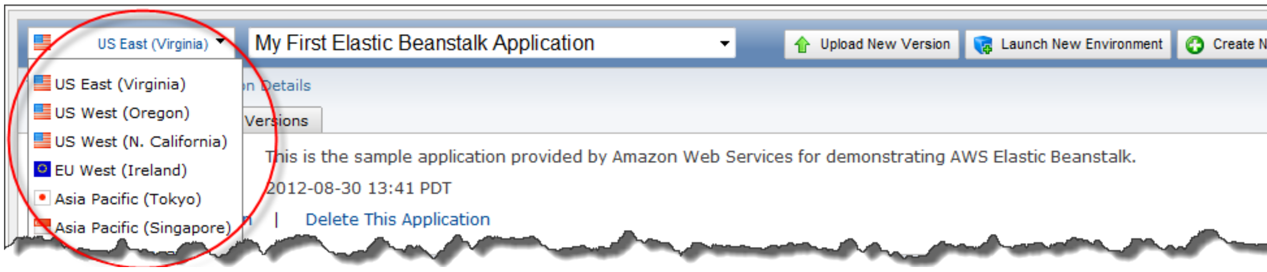
Creating New Applications

You can create new AWS Elastic Beanstalk applications and deploy the application versions to new environments. This section describes how you can use the AWS Management Console, the CLI, or the API to create a new AWS Elastic Beanstalk application and deploy the application version to a new environment.

AWS Management Console

To create a new application

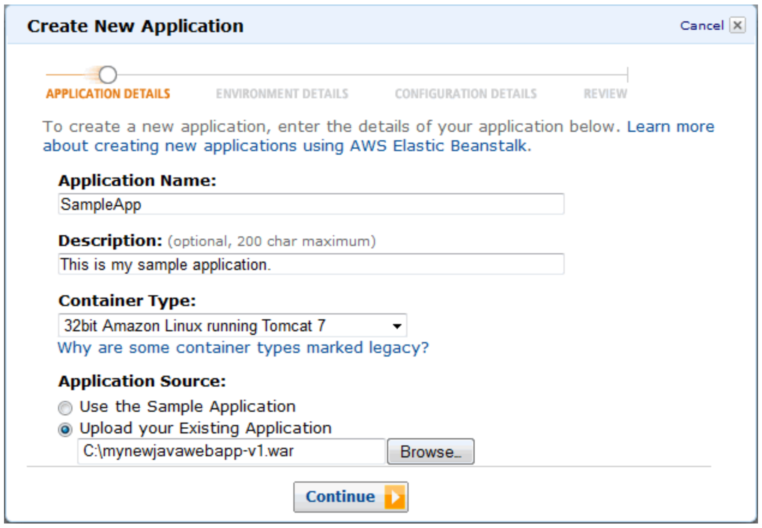
1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.



This is the sample application provided by Amazon Web Services for demonstrating AWS Elastic Beanstalk.
2012-08-30 13:41 PDT

3. Click the **Create New Application** button.

   The **Create New Application Wizard** appears.



To create a new application, enter the details of your application below. Learn more about creating new applications using AWS Elastic Beanstalk.

Application Name: SampleApp

Description: (optional, 200 char maximum) This is my sample application.

Container Type: 32bit Amazon Linux running Tomcat 7

Why are some container types marked legacy?

Application Source:
Use the Sample Application
[✓ Upload your Existing Application]
C:\mynewjavawebapp-v1.war

4. Describe your application details:

   a. Enter a short descriptive name for the application in the **Application Name** box.
   b. Enter a brief description of the application in the **Description** box.
   c. In the **Application Source** section of the **Create New Application** wizard, you can:

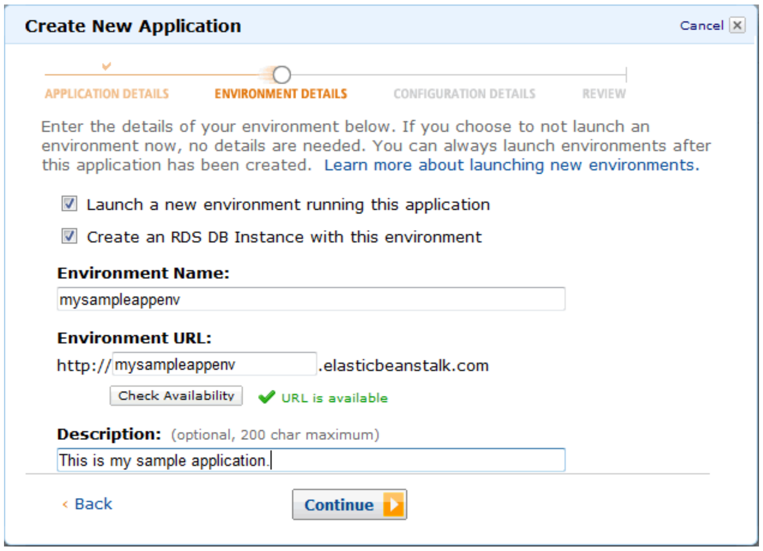
      - Select **Use the sample application**.
      - Select **Upload your existing application**.

   For this example, select **Upload your existing application**.

Note

AWS Elastic Beanstalk supports only a single WAR file for a Java application version and only a single ZIP file for a PHP, .NET, and Python application. The file size limit is 250 MB.

5. Click **Continue** to continue to the **ENVIRONMENT DETAILS** pane.



Enter the details of your environment below. If you choose to not launch an environment now, no details are needed. You can always launch environments after this application has been created. Learn more about launching new environments.

☑ Launch a new environment running this application

☑ Create an RDS DB Instance with this environment

Environment Name: mysampleappenv

Environment URL: http://mysampleappenv.elasticbeanstalk.com

Description: (optional, 200 char maximum) This is my sample application.

Describe your new environment's details:

a. If you want to launch a new environment for your application, click **Launch a new environment running this application**.

b. If you are creating an application using a non-legacy container, then you have the option to associate an Amazon RDS database. If you want to associate an Amazon RDS DB with this application, click **Create an RDS Database with this environment**. For more information about Amazon RDS, go to Amazon Relational Database Service (Amazon RDS). For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).

c. Enter a short descriptive name for the application in the **Environment Name** box.

   This name is then auto-populated in the **Environment URL** box.

d. If you want an environment URL that is different than the name of the environment, enter the URL in the **Environment URL** box.

   AWS Elastic Beanstalk will use this name to create a unique CNAME for the environment.

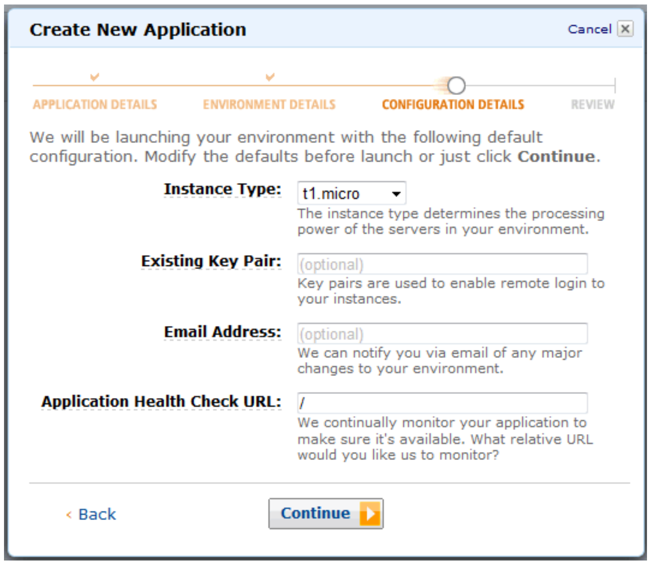
e. Enter a brief description of the environment in the **Description** box.

f. Select a container type from the **Container Type** drop-down list.

   Note

   After you launch a new environment with a container type, you cannot change the container type. If you want to change the container type, you need to launch a new environment.

6. Click **Continue** to continue to the **CONFIGURATION DETAILS** pane.



We will be launching your environment with the following default configuration. Modify the defaults before launch or just click **Continue**.

Instance Type: t1.micro
The instance type determines the processing power of the servers in your environment.

Existing Key Pair: (optional)
Key pairs are used to enable remote login to your instances.

Email Address: (optional)
We can notify you via email of any major changes to your environment.

Application Health Check URL: /
We continually monitor your application to make sure it's available. What relative URL would you like us to monitor?

7. Provide configuration details about your environment.

   a. Select an EC2 instance type.

      The **Instance Type** drop-down list displays the instance types available to your AWS Elastic Beanstalk environment. Select a server with the characteristics (including memory size and CPU power) that are most appropriate to your application.

      Note

      AWS Elastic Beanstalk is free, but the AWS resources that it provisions might not be. For information on Amazon EC2 usage fees, go to Amazon EC2 Pricing.

      For more information about the Amazon EC2 instance types available for your AWS Elastic Beanstalk environment, go to Instance Families and Types in the *Amazon Elastic Compute Cloud User Guide*.

   b. If you need to securely log in to your Amazon EC2 instances, provide an Amazon EC2 key pair.

      You can securely log in to the Amazon EC2 instances provisioned for your AWS Elastic Beanstalk application with an Amazon EC2 key pair.

      Important

      You must create an Amazon EC2 key pair before you can can access your AWS Elastic Beanstalk–provisioned Amazon EC2 instances. You can set up your Amazon EC2 key pairs using the AWS Management Console. For instructions on creating a key pair for Amazon EC2, see the Amazon Elastic Compute Cloud Getting Started Guide.

      The **Existing Key Pair** box lets you specify the name of an Amazon EC2 key pair you use to securely log in to the Amazon EC2 instances running your AWS Elastic Beanstalk application.

      For more information on Amazon EC2 Key Pairs, see Using Credentials in the *Amazon Elastic Compute Cloud User Guide*. For more information on connecting to Amazon EC2 instances, see Connecting to Instances and Connecting to an Instance from Windows using PuTTY in the *Amazon Elastic Compute Cloud User Guide*.

   c. If you want to receive email notifications of important events, provide an email address for Amazon Simple Notification Service (Amazon SNS) notifications.

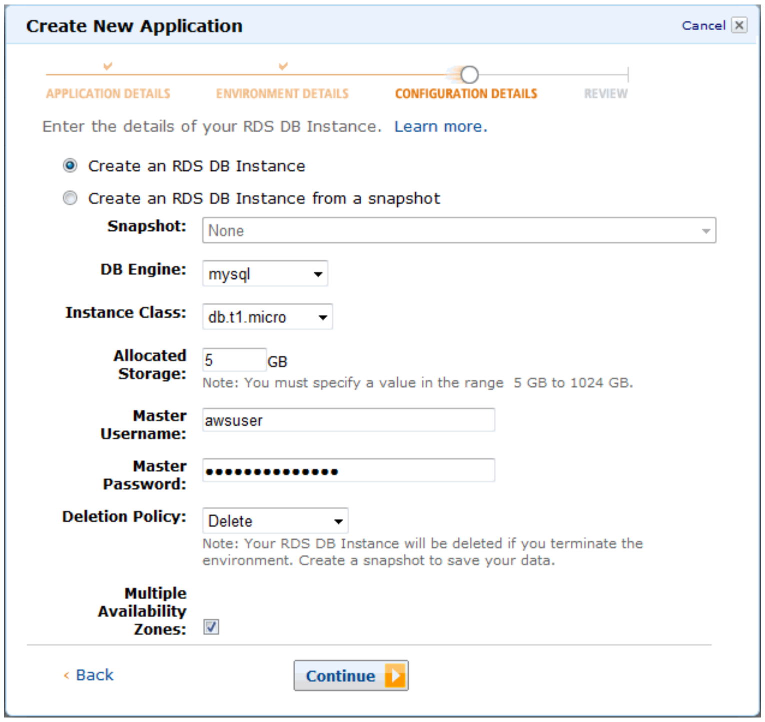
AWS Elastic Beanstalk can notify you of important events regarding your application using the Amazon Simple Notification Service. To enable Amazon SNS notifications, simply enter your email address in the **Email Address** box. You can disable Amazon SNS notifications at a later time by editing the configuration of your running environment.

d. If you want to override the default health check URL, provide a custom health check URL.

   Elastic Load Balancing uses a *health check* to determine whether the Amazon EC2 instances running your application are healthy. The health check determines an instance's health status by probing a specified URL at a set interval; if the URL returns an error message, or fails to respond within a specified timeout period, the health check fails. AWS Elastic Beanstalk uses Elastic Load Balancing health checks to set the status of your application in the AWS Management Console.

   The health check definition includes a URL to be queried for instance health. Override the default URL ("/") by entering the URL you want to check in the **Application Health Check URL** box.

8. Click **Continue**. If you are creating an application using a non-legacy container and chose to associate an Amazon RDS DB, then another **CONFIGURATION** pane appears. Otherwise, skip to the next step in the document to view the **REVIEW** pane details.



Enter the details of your RDS DB Instance. Learn more.

● Create an RDS DB Instance
○ Create an RDS DB Instance from a snapshot

Snapshot: None
DB Engine: mysql
Instance Class: db.t1.micro
Allocated Storage: 5 GB
Note: You must specify a value in the range 5 GB to 1024 GB.
Master Username: awsuser
Master Password: ••••••••••••••
Deletion Policy: Delete
Note: Your RDS DB Instance will be deleted if you terminate the environment. Create a snapshot to save your data.
Multiple Availability Zones: ✓

9. Provide configuration details about your Amazon RDS DB.

   a. Select if you want to create a Amazon RDS DB or create one from a snapshot. If you choose to create a database from a snapshot, then select a snapshot from the **Snapshot** list.
   b. Select a database engine from the **DB Engine** list.
   c. Select a database instance class from the **Instance Class** list. For information about the DB instance classes, go to http://aws.amazon.com/rds/.
   d. Type the allocated storage for your database in the **Allocated Storage** box. In some cases, allocating a higher amount of storage for your DB Instance than the size of your database can improve I/O performance. For information about storage allocation, go to Features.

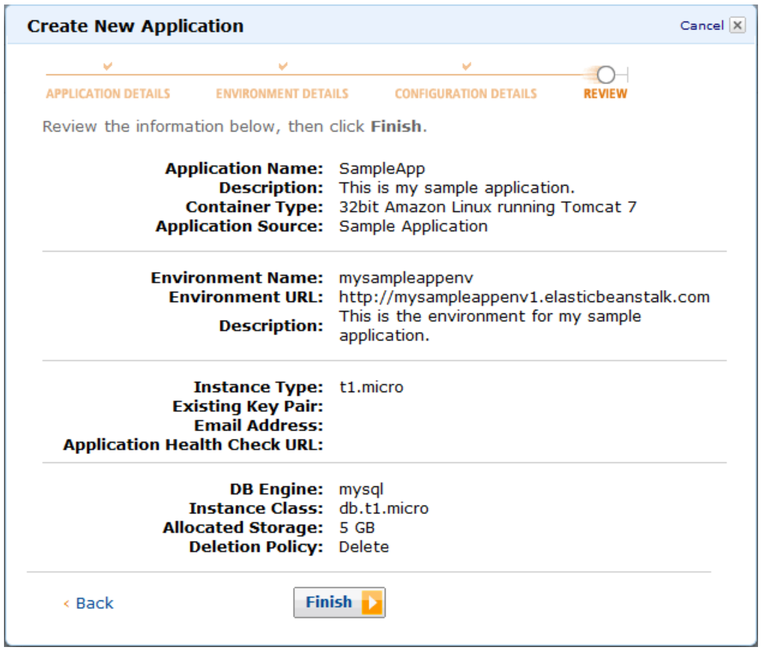
e. Type a name in the **Master Username** box using alphanumeric characters that you will use as the master user name to log on to your DB Instance with all database privileges.

f. Type a password in the **Master Password** box containing from 8 to 16 printable ASCII characters (excluding /,", and @) for your master user password.

g. In the **Deletion Policy** list, select if you want to create a snapshot of your database or delete it if you terminate your AWS Elastic Beanstalk environment. If you select **Delete**, then your RDS DB will be deleted and your data will be lost if you terminate your environment.

h. Click **Multiple Availability Zones** if you want to configure your database across mutliple Availability Zones. Running your database instance in multiple Availability Zones safeguards your data in the unlikely event of a database instance component failure or service health disruption in one Availability Zone.

10. Click **Continue**. The **REVIEW** pane appears.



Review the information below, then click **Finish**.

Application Name:	SampleApp
Description:	This is my sample application.
Container Type:	32bit Amazon Linux running Tomcat 7
Application Source:	Sample Application
Environment Name:	mysampleappenv
Environment URL:	http://mysampleappenv1.elasticbeanstalk.com
Description:	This is the environment for my sample application.
Instance Type:	t1.micro
Existing Key Pair:	
Email Address:	
Application Health Check URL:	
DB Engine:	mysql
Instance Class:	db.t1.micro
Allocated Storage:	5 GB
Deletion Policy:	Delete

11. Review your new environment settings and then click **Finish**.

    The new environment is launched with your new application. Note that it can take several minutes for the new environment to start while AWS Elastic Beanstalk is provisioning AWS resources.

CLI

To create a new application

1. Create a new application.

```
PROMPT> elastic-beanstalk-create-application -a [Application Name] -d
[Description]
```

2. Create a new application version.

```
PROMPT> elastic-beanstalk-create-application-version -a [Application Name]
-l [Version Label] -d [Description] -s [Source Location]
```

Note

If you want to use the sample application, then do not pass the source location parameter.

3. Check DNS availability.

   ```
   PROMPT> elastic-beanstalk-check-dns-availability -c [CNAME Prefix]
   ```
4. Create environment.

   ```
   PROMPT> elastic-beanstalk-create-environment -a [Application Name] -l [Version
   Label] -e [Environment Name] -c [CNAME Prefix] -d [Description] -s [Solution
   Stack Name]
   ```
5. Determine if the new environment is Green and Ready.

   ```
   PROMPT> elastic-beanstalk-describe-environments -e [Environment Name]
   ```

   If the new environment does not come up Green and Ready, you should decide if you want to retry the operation or leave the environment in its current state for investigation. Make sure to terminate the environment after you are finished, and clean up any unused resources.

   Note

   You can adjust the timeout period if the environment doesn't launch in a reasonable time.

API

To create a new application

1. Call `CreateApplication` with the following parameters:

   - *`ApplicationName`* = `SampleApp`
   - *`Description`* = `description`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
   &Description=description
   &Operation=CreateApplicationVersion
   &AuthParams
   ```

2. Call `CreateApplicationVersion` with the following parameters:

   - *`ApplicationName`* = `SampleApp`
   - *`VersionLabel`* = `Version1`
   - *`Description`* = `description`
   - *`SourceBundle.S3Bucket`* = `<your S3 bucket name>`
   - *`SourceBundle.S3Key`* = `mynewjavawebapp-v1.war`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
&VersionLabel=Version1
&Description=description
&SourceBundle.S3Bucket=<your S3 bucket name>
&SourceBundle.S3Key=mynewjavawebapp-v1.war
&Operation=CreateApplicationVersion
&AuthParams
```

3. Call `CheckDNSAvailability` with the following parameters:

   - *`CNAMEPrefix`* = `mysampleapplication`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?CNAMEPrefix=mysampleapplication
&Operation=CheckDNSAvailability
&AuthParams
```

4. Call `CreateEnvironment` with the following parameters:

   - *`ApplicationName`* = `SampleApp`
   - *`VersionLabel`* = `Version1`
   - *`EnvironmentName`* = `mynewappenv`
   - *`SolutionStackName`* = `"32bit Amazon Linux running Tomcat 7"`
   - *`CNAMEPrefix`* = `mysampleapplication`
   - *`Description`* = `description`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
&VersionLabel=Version1
&EnvironmentName=mynewappenv
&SolutionStackName=32bit%20Amazon%20Linux%20running%20Tomcat%207
&CNAMEPrefix=mysampleapplication
&Description=description
&Operation=CreateEnvironment
&AuthParams
```

5. Call `DescribeEnvironments` with the following parameters:

   - *`EnvironmentName`* = `mynewappenv`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=mynewappenv
&Operation=DescribeEnvironments
&AuthParams
```

Creating New Application Versions

You can create different versions for an application. Each application version consists of a unique file (WAR file or ZIP file), as well as contextual information about the version. This topic describes how to create a new version of an existing AWS Elastic Beanstalk application and deploy it to an existing environment. You may want to do this if, for instance, you have updated your application and want to re-deploy it to your testing environment. For information on how to create new application versions using the AWS Toolkit for Eclipse, see Creating and Deploying AWS Elastic Beanstalk Applications in Java Using AWS Toolkit for Eclipse (p. 22). For more information on how to create new application versions for PHP, see Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git (p. 92). For more information on how to create new application versions using the AWS Toolkit for Visual Studio, see Creating and Deploying AWS Elastic Beanstalk Applications in .NET Using AWS Toolkit for Visual Studio (p. 51).

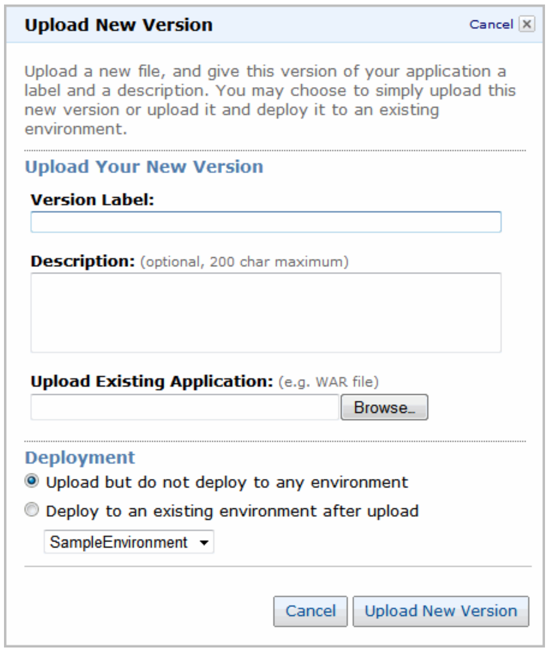
Note

For information on creating a new application, see Creating New Applications (p. 124).

AWS Management Console

To create a new application version

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Select the application to which you want to add a new version.
4. Click **Upload New Version**. The **Upload New Version** window appears.


Upload New Version

Upload a new file, and give this version of your application a label and a description. You may choose to simply upload this new version or upload it and deploy it to an existing environment.

Upload Your New Version

Version Label:

Description: (optional, 200 char maximum)

Upload Existing Application: (e.g. WAR file)

Deployment

(●) Upload but do not deploy to any environment

( ) Deploy to an existing environment after upload

SampleEnvironment

5. Describe your application version details:

   a. Enter a short descriptive name for the application version in the **Version Label** box.

   b. Enter a brief description of the application version in the **Description** box.

c. In the **Upload Existing Application** box, enter the location of the WAR file (Java) or the ZIP file (.NET or PHP) containing the new application version. You can optionally click **Browse** to locate the file in your local file system.

   Note

   AWS Elastic Beanstalk supports only a single WAR file for a Java application version and only a single ZIP file for a .NET, PHP, or Python application. The file size limit is 250 MB.

6. In the **Deployment** section of the **Upload New Version** dialog box, you can:

   - Select **Upload but do not deploy to any environment**.
   - Select **Deploy to an existing environment after upload**, and select the environment from the drop-down list box.

     Note

     For more information about deploying with zero downtime, see Deploying Versions with Zero Downtime (p. 143).

   For this example, select **Deploy to an existing environment after upload**, and then select the **Default Environment** from the drop-down menu.
7. Click **Upload and Deploy New Version**.

   The file you specified is associated with your application, and the application is launched with the selected environment.

CLI

To create a new application version

1. Create a new application version.

   ```
   PROMPT> elastic-beanstalk-create-application-version -a [Application Name]
   -l [Version Label] -d [Description] -s [Source Location]
   ```
2. Update your existing environment.

   ```
   PROMPT> elastic-beanstalk-update-environment -e [Environment Name] -l [Version
   Label] -d [Description]
   ```
3. Determine if the new environment is Green and Ready.

   ```
   PROMPT> elastic-beanstalk-describe-environments -e [Environment Name]
   ```

   If the new environment does not come up Green and Ready, you should decide if you want to retry the operation or leave the environment in its current state for investigation. Make sure to terminate the environment after you are finished, and clean up any unused resources.

   Note

   You can adjust the timeout period if the environment doesn't launch in a reasonable time.

API

To create a new application version

1. Call `CreateApplicationVersion` with the following parameters:
   - *`ApplicationName`* = `SampleApp`
   - *`VersionLabel`* = `Version2`
   - *`Description`* = `description`
   - *`SourceBundle.S3Bucket`* = `<your bucket name>`
   - *`SourceBundle.S3Key`* = `<your application file name>`
   - *`AutoCreateApplication`* = `true`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
   &VersionLabel=Version2
   &Description=description
   &SourceBundle.S3Bucket=amazonaws.com
   &SourceBundle.S3Key=sample.war
   &AutoCreateApplication=true
   &Operation=CreateApplicationVersion
   &AuthParams
   ```

2. Call `UpdateEnvironment` with the following parameters:
   - *`EnvironmentName`* = `SampleAppEnv`
   - *`VersionLabel`* = `Version2`
   - *`Description`* = `description`
   - *`TemplateName`* = `MyConfigTemplate`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=mysampleappenv
   &TemplateName=myconfigtemplate
   &Description=description
   &VersionLabel=Version2
   &Operation=UpdateEnvironment
   &AuthParams
   ```

3. Call `DescribeEnvironments` with the following parameter:
   - *`EnvironmentName`* = `SampleAppEnv`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
   &Operation=DescribeEnvironments
   &AuthParams
   ```

Launching New Environments

You can deploy multiple environments when you need to run multiple versions of an application. For example, you might have development, integration, and production environments. When launched, you can deploy a different version to any environment quickly and easily. For more information about deploying with zero downtime, see Deploying Versions with Zero Downtime (p. 143).

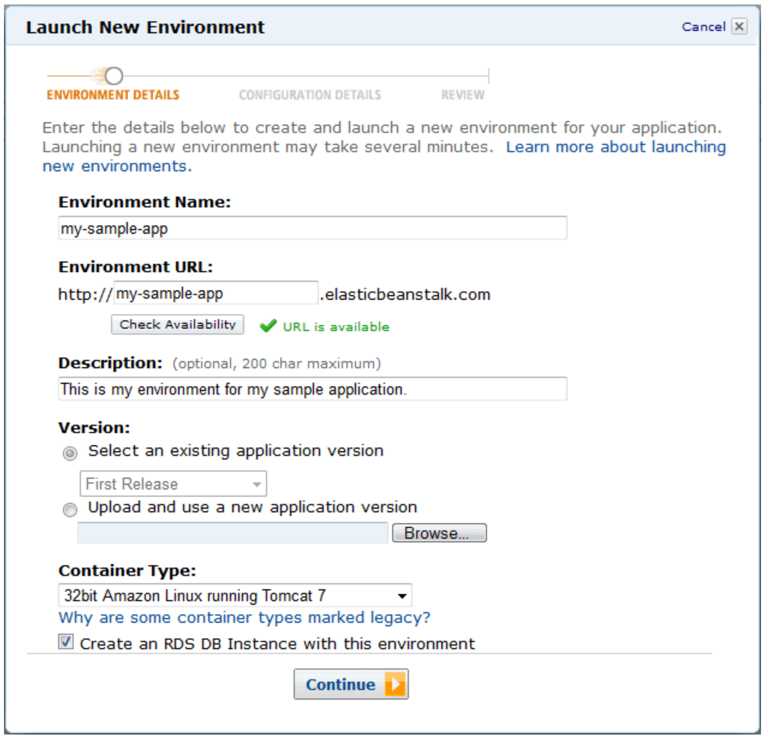
Important

After you create an environment, the environment URL is publicly visible.

AWS Management Console

To launch a new environment

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Click **Launch New Environment**. The **Launch New Environment** wizard appears.



Launch New Environment

Enter the details below to create and launch a new environment for your application. Launching a new environment may take several minutes. Learn more about launching new environments.

Environment Name: my-sample-app

Environment URL: http:// my-sample-app .elasticbeanstalk.com

URL is available

Description: (optional, 200 char maximum) This is my environment for my sample application.

Version:
Select an existing application version: First Release
Upload and use a new application version

Container Type: 32bit Amazon Linux running Tomcat 7

Why are some container types marked legacy?

[✓] Create an RDS DB Instance with this environment

4. Describe your new environment's details:

   a. Enter a short descriptive name for the application version in the **Environment Name** box.

      This name is then auto-populated in the **Environment URL** box.

   b. If you want an environment URL that is different than the name of the environment, enter the URL in the **Environment URL** box.

      AWS Elastic Beanstalk will use this name to create a unique CNAME for the environment.

   c. Enter a brief description of the environment in the **Description** box.

   d. In **Version**, you can choose from two options:

- Click the **Select an existing application version** option, and select the existing application version from the drop-down list.
- Click the **Upload and use a new application version** option, and click **Browse** to choose a file to upload.

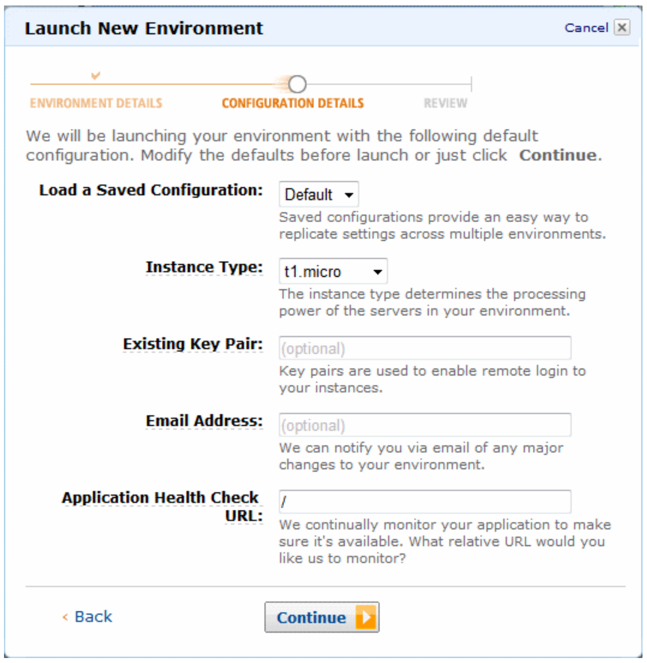
e. Select a container type from the **Container Type** drop-down list.

Note

After you launch a new environment with a container type, you cannot change the container type. If you want to change the container type, you need to launch a new environment.

f. If you are creating an application using a non-legacy container, then you have the option to associate an Amazon RDS database. If you want to associate an Amazon RDS DB with this application, click **Create an RDS Database with this environment**. For more information about Amazon RDS, go to Amazon Relational Database Service (Amazon RDS). For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).

5. Click **Continue** to continue to the **CONFIGURATION DETAILS** pane.



We will be launching your environment with the following default configuration. Modify the defaults before launch or just click **Continue**.

Load a Saved Configuration: Default
Saved configurations provide an easy way to replicate settings across multiple environments.

Instance Type: t1.micro
The instance type determines the processing power of the servers in your environment.

Existing Key Pair: (optional)
Key pairs are used to enable remote login to your instances.

Email Address: (optional)
We can notify you via email of any major changes to your environment.

Application Health Check URL: /
We continually monitor your application to make sure it's available. What relative URL would you like us to monitor?

6. Provide configuration details about your environment.

a. Select a saved configuration.

The **Load a Saved Configuration** drop-down list displays a list of configurations that you have saved for your environments. Select a saved configuration that matches the settings you want for your new environment, or select **Default**.

b. Select an EC2 instance type.

The **Instance Type** drop-down list box displays the instance types available to your AWS Elastic Beanstalk environment. Select a server with the characteristics (including memory size and CPU power) that are most appropriate to your application.

Note

AWS Elastic Beanstalk is free, but the AWS resources that it provisions might not be. For information on Amazon EC2 usage fees, go to Amazon EC2 Pricing.

For more information about the Amazon EC2 instance types available for your AWS Elastic Beanstalk environment, go to Instance Families and Types in the *Amazon Elastic Compute Cloud User Guide*.

c. If you need to securely log in to your Amazon EC2 instances, provide an Amazon EC2 key pair.

   You can securely log in to the Amazon EC2 instances provisioned for your AWS Elastic Beanstalk application with an Amazon EC2 key pair.

   Important

   You must create an Amazon EC2 key pair before you can can access your AWS Elastic Beanstalk-provisioned Amazon EC2 instances. You can set up your Amazon EC2 key pairs using the AWS Management Console. For instructions on creating a key pair for Amazon EC2, see the Amazon Elastic Compute Cloud Getting Started Guide.

   The **Existing Key Pair** text box lets you specify the name of an Amazon EC2 key pair you use to securely log in to the Amazon EC2 instances running your AWS Elastic Beanstalk application.

   For more information on Amazon EC2 Key Pairs, see Using Credentials in the *Amazon Elastic Compute Cloud User Guide*. For more information on connecting to Amazon EC2 instances, see Connecting to Instances and Connecting to an Instance from Windows using PuTTY in the *Amazon Elastic Compute Cloud User Guide*.

d. If you want to receive email notifications of important events, provide an email address for Amazon Simple Notification Service (Amazon SNS) notifications.

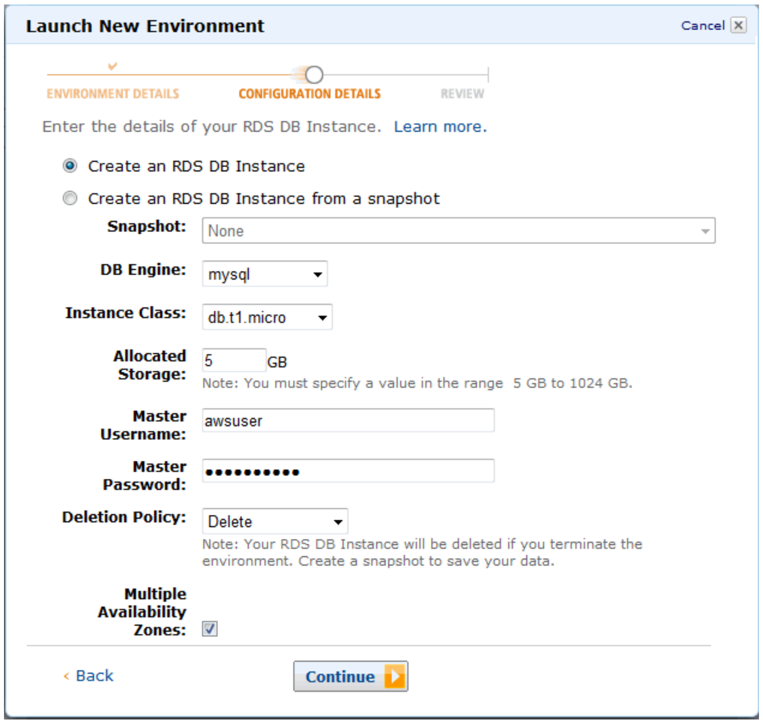
   AWS Elastic Beanstalk can notify you of important events regarding your application using the Amazon Simple Notification Service. To enable Amazon SNS notifications, simply enter your email address in the **Email Address** box. You can disable Amazon SNS notifications at a later time by editing the configuration of your running environment

e. If you want to override the default health check URL, provide a custom health check URL.

   Elastic Load Balancing uses a *health check* to determine whether the Amazon EC2 instances running your application are healthy. The health check determines an instance's health status by probing a specified URL at a set interval; if the URL returns an error message, or fails to respond within a specified timeout period, the health check fails. AWS Elastic Beanstalk uses Elastic Load Balancing health checks to set the status of your application in the AWS Management Console.

   The health check definition includes a URL to be queried for instance health. Override the default URL ("/") by entering the URL you want to check in the **Application Health Check URL** box.

7. Click **Continue**. If you are creating an application using a non-legacy container and chose to associate an Amazon RDS DB, then another **CONFIGURATION** pane appears. Otherwise, skip to the next step in the document to view the **REVIEW** pane details.



Launch New Environment

Cancel

ENVIRONMENT DETAILS CONFIGURATION DETAILS REVIEW

Enter the details of your RDS DB Instance. Learn more.

● Create an RDS DB Instance

○ Create an RDS DB Instance from a snapshot

Snapshot:	None
DB Engine:	mysql
Instance Class:	db.t1.micro
Allocated Storage:	5 GB Note: You must specify a value in the range 5 GB to 1024 GB.
Master Username:	awsuser
Master Password:	••••••••••
Deletion Policy:	Delete Note: Your RDS DB Instance will be deleted if you terminate the environment. Create a snapshot to save your data.
Multiple Availability Zones:	✓

‹ Back Continue

8. Provide configuration details about your Amazon RDS DB.

   a. Select if you want to create a blank Amazon RDS DB or create one from a snapshot. If you choose to create a database from a snapshot, then select a snapshot from the **Snapshot** list.

   b. Select a database engine from the **DB Engine** list.

   c. Select a database instance class from the **Instance Class** list. For information about the DB instance classes, go to http://aws.amazon.com/rds/.

   d. Type the allocated storage for your database in the **Allocated Storage** box. In some cases, allocating a higher amount of storage for your DB Instance than the size of your database can improve I/O performance. For information about storage allocation, go to Features.

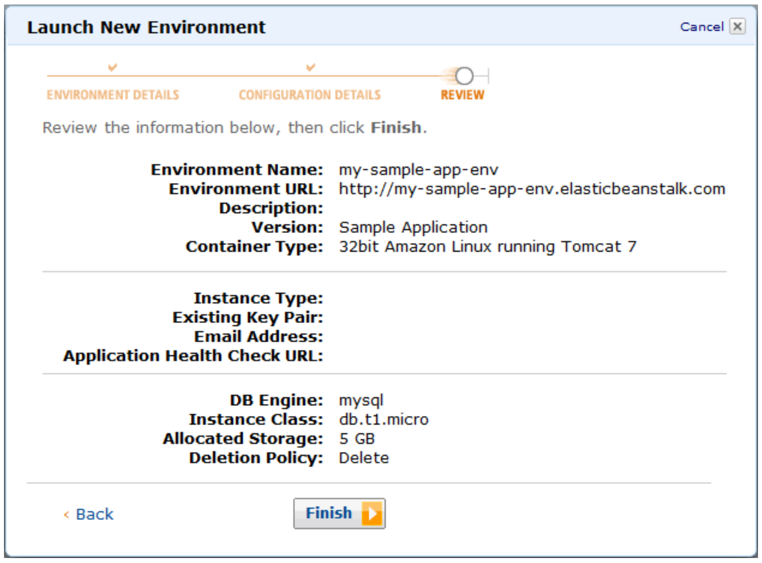
   e. Type a name in the **Master Username** box using alphanumeric characters that you will use as the master user name to log on to your DB Instance with all database privileges.

   f. Type a password in the **Master Password** box containing from 8 to 16 printable ASCII characters (excluding /,", and @) for your master user password.

   g. In the **Deletion Policy** list, select if you want to create a snapshot of your database or delete it if you terminate your AWS Elastic Beanstalk environment.

   h. Click **Multiple Availability Zones** if you want to configure your database across mutliple Availability Zones. Running your database instance in multiple Availability Zones safeguards your data in the unlikely event of a database instance component failure or service health disruption in one Availability Zone.

9. Click **Continue**. The **REVIEW** pane appears.



Review the information below, then click **Finish**.

Environment Name:	my-sample-app-env
Environment URL:	http://my-sample-app-env.elasticbeanstalk.com
Description:	
Version:	Sample Application
Container Type:	32bit Amazon Linux running Tomcat 7
Instance Type:	
Existing Key Pair:	
Email Address:	
Application Health Check URL:	
DB Engine:	mysql
Instance Class:	db.t1.micro
Allocated Storage:	5 GB
Deletion Policy:	Delete

10. Review your new environment settings, and then click **Finish**.

    The new environment is launched. Note that it can take several minutes for the new environment to start while AWS Elastic Beanstalk is provisioning AWS resources.

CLI

To launch a new environment

1. Check if CNAME for the new environment is available.

   ```
   PROMPT> elastic-beanstalk-check-dns-availability -c [CNAME prefix]
   ```

2. Make sure your application version exists.

   ```
   PROMPT> elastic-beanstalk-describe-application-versions -a [Application
   Name] -l [Version Label]
   ```

3. Create a configuration template for an existing application.

   ```
   PROMPT> elastic-beanstalk-create-configuration-template -a [Application
   Name] -t [Template Name]
   ```

4. Create the new environment.

   ```
   PROMPT> elastic-beanstalk-create-environment -a [Application Name] -l [Version
   Label] -e [Environment Name] -t [Template Name] -c [CNAME prefix] -d
   [Description]
   ```

5. Determine if the new environment is Green and Ready.

   ```
   PROMPT> elastic-beanstalk-describe-environments -e [Environment Name]
   ```

   If the new environment does not come up Green and Ready, you should decide if you want to retry the operation or leave the environment in its current state for investigation. Make sure to terminate the environment after you are finished, and clean up any unused resources.

   Note

   You can adjust the timeout period if the environment doesn't launch in a reasonable time.

API

To launch a new environment

1. Call `CheckDNSAvailability` with the following parameter:

   - *`CNAMEPrefix`* = `SampleApp`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?CNAMEPrefix=sampleapplication
   &Operation=CheckDNSAvailability
   &AuthParams
   ```

2. Call `DescribeApplicationVersions` with the following parameters:

   - *`ApplicationName`* = `SampleApp`
   - *`VersionLabel`* = `Version2`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
   &VersionLabel=Version2
   &Operation=DescribeApplicationVersions
   &AuthParams
   ```

3. Call `CreateConfigurationTemplate` with the following parameters:

   - *`ApplicationName`* = `SampleApp`
   - *`TemplateName`* = `MyConfigTemplate`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
   &TemplateName=MyConfigTemplate
   &Operation=CreateConfigurationTemplate
   &AuthParams
   ```

4. Call `CreateEnvironment` with the following parameters:

   - *`EnvironmentName`* = `SampleAppEnv2`
   - *`VersionLabel`* = `Version2`
   - *`Description`* = `description`
   - *`TemplateName`* = `MyConfigTemplate`
   - *`ApplicationName`* = `SampleApp`
   - *`CNAMEPrefix`* = `sampleapplication`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
&VersionLabel=Version2
&EnvironmentName=SampleAppEnv2
&TemplateName=MyConfigTemplate
&CNAMEPrefix=sampleapplication
&Description=description
&Operation=CreateEnvironment
&AuthParams
```

5. Call `DescribeEnvironments` with the following parameter:
   - *`EnvironmentName`* = `SampleAppEnv2`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv2
&Operation=DescribeEnvironments
&AuthParams
```

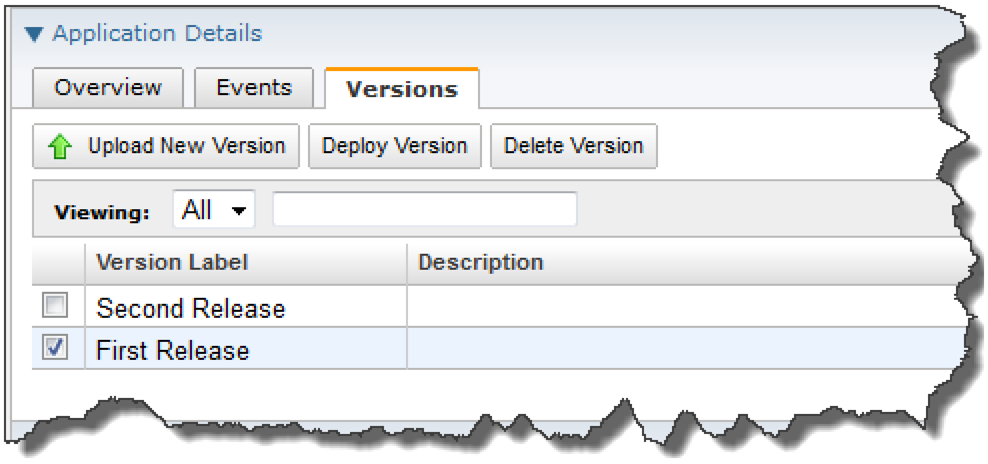
Deploying Versions to Existing Environments

You can deploy existing AWS Elastic Beanstalk application versions to existing environments. You may want to do this if, for instance, you need to roll back to a previous version of your application. This section describes how you can use the AWS Management Console, the CLI, or APIs to deploy an existing AWS Elastic Beanstalk application version to an existing environment.

AWS Management Console

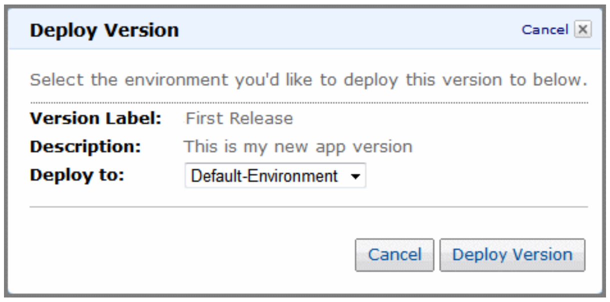
To deploy an existing application version to an existing environment

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Select the application from the drop-down list, and then click the **Versions** tab in the AWS Elastic Beanstalk console.
4. Select the check box next to the version label of the application version you want to deploy.



	Version Label	Description
☐	Second Release	
☑	First Release	

5. Click **Deploy Version**.



Select the environment you'd like to deploy this version to below.

Version Label: First Release
Description: This is my new app version
Deploy to: Default-Environment

6. Click the environment in the **Deploy to** field, and then click **Deploy Version**.
7. A dialog box confirms the update to your environment. Click **Close**.

AWS Elastic Beanstalk now deploys your file to your Amazon EC2 instances. You will see the environment turn gray and the status changed to "Updating." When the deployment is complete, there's an application health check. The environment returns to green when the application responds to the health check.

To view your application version

1. Select your environment in the **Environments** pane.
2. Click the **Events** tab to view current information on the deployment of the new version.
3. Click **View Running Version** in the **Overview** tab to see the new version of your application.

CLI

To deploy an existing application version to an existing environment

1. Make sure your application version exists.

```
PROMPT> elastic-beanstalk-describe-application-versions -a [Application
Name] -l [Version Label]
```

2. Update your environment with your existing application version.

```
PROMPT> elastic-beanstalk-update-environment -e [Environment Name] -l [Version
Label] -d [Description]
```

3. Determine if the environment is Green and Ready.

```
PROMPT> elastic-beanstalk-describe-environments -e [Environment Name]
```

API

To deploy an existing application version to an existing environment

1. Call `UpdateEnvironment` with the following parameters:

   - *`EnvironmentName`* = `SampleAppEnv`
   - *`VersionLabel`* = `FirstRelease`
   - *`Description`* = `description`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=MySampleAppEnv
   
    &Description=description
    &VersionLabel=FirstRelease
    &Operation=UpdateEnvironment
    &AuthParams
   ```

2. Call `DescribeEnvironments` with the following parameter:

   - *`EnvironmentName`* = `SampleAppEnv`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
   
    &Operation=DescribeEnvironments
    &AuthParams
   ```

Deploying Versions with Zero Downtime

Since AWS Elastic Beanstalk performs an in-place update when you update your application versions, you will experience some downtime. However, it is possible to avoid this downtime by swapping the CNAMEs for your environments. This section walks you through how to perform a CNAME swap using the AWS Management Console, the command line interface, or APIs.

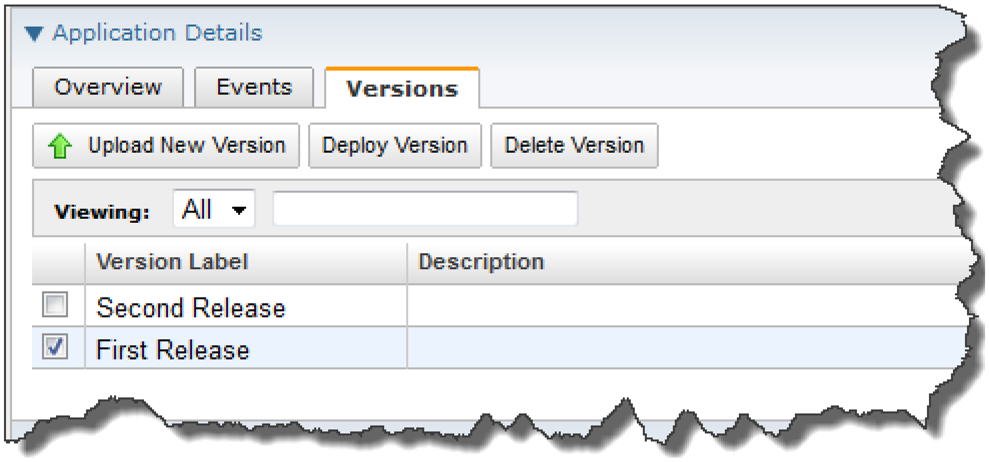
Note

If you have created an Alias record to map your root domain to your Elastic Load Balancer using Amazon Route 53, then after you have created you new environment, you will need to change your resource record set to map your root domain to the Elastic Load Balancer in your new environment. For instructions on how to change your existing resource record set, go to Using AWS Elastic Beanstalk with Amazon Route 53 to Map Your Root Domain to Your Load Balancer (p. 221).

AWS Management Console

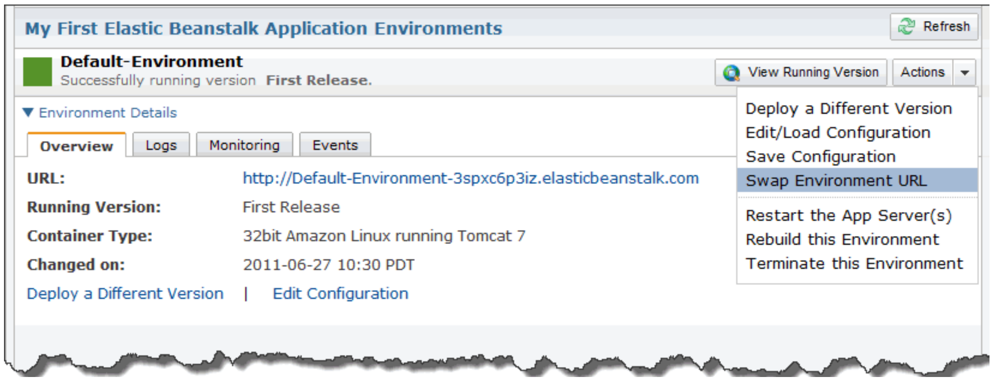
To deploy with zero downtime

1. Check that your current environment is Green and Ready from the application's **Environments** list. For instructions, see Viewing Application Health and Environment Status (p. 153).
2. Verify that your new application version exists.
   a. Select the application from the drop-down list, and then click the **Versions** tab in the AWS Elastic Beanstalk console.
   b. Select the check box next to the version label of the application version you want to deploy.



	Version Label	Description
☐	Second Release	
☑	First Release	

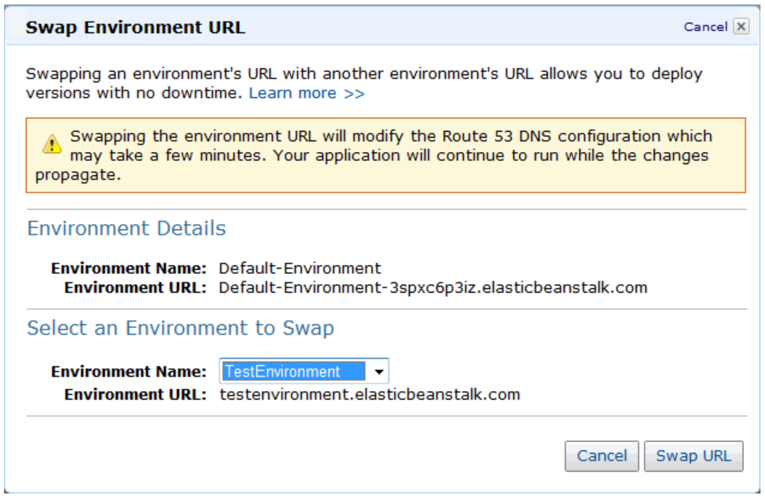
3. Create a configuration template for the current environment. For instructions, see Saving Environment Configuration Settings (p. 157).
4. Launch a new environment with your new application version and saved configuration template. For instructions, see Launching New Environments (p. 135). In the **Launch New Environment** wizard, click your saved configuration template from the **Load a Saved Configuration** list.
5. Make sure that your new environment is in the Green and Ready state. For instructions, see Viewing Application Health and Environment Status (p. 153).
6. Click the **Actions** drop-down list for your application environment and select **Swap Environment URL**.



Successfully running version First Release.

URL:	http://Default-Environment-3spxc6p3iz.elasticbeanstalk.com
Running Version:	First Release
Container Type:	32bit Amazon Linux running Tomcat 7
Changed on:	2011-06-27 10:30 PDT

7. In the **Swap Environment URL** window, select the environment name you want to swap with from the **Environment Name** pull-down menu.



Swap Environment URL

Swapping an environment's URL with another environment's URL allows you to deploy versions with no downtime. Learn more >>

Swapping the environment URL will modify the Route 53 DNS configuration which may take a few minutes. Your application will continue to run while the changes propagate.

Environment Details

Environment Name: Default-Environment
Environment URL: Default-Environment-3spxc6p3iz.elasticbeanstalk.com

Select an Environment to Swap

Environment Name: TestEnvironment
Environment URL: testenvironment.elasticbeanstalk.com

8. Click **Swap**.
9. Delete your old environment after you confirmed the swap operation has completed.

CLI

You can use the command line interface to deploy a new application with zero downtime. The following steps walk you through creating a configuration template, launching a new environment with the new application version using the configuration template, and swapping the environment CNAMEs. You can also use the following steps to perform configuration changes with zero downtime.

To deploy with zero downtime

1. Check your current environment to make sure it is Green and Ready.
   ```
   PROMPT> elastic-beanstalk-describe-environments -E [Environment ID]
   ```
2. Verify that your new application version exists.
   ```
   PROMPT> elastic-beanstalk-describe-application-versions -a [Application
   Name] -l [Version Label]
   ```
3. Create a configuration template from the current environment.
   ```
   PROMPT> elastic-beanstalk-create-configuration-template -E [Environment ID]
   -a [Application Name] -t [Template Name]
   ```
4. Launch a new environment for the new application version and template.
   ```
   PROMPT> elastic-beanstalk-create-environment -e [Environment Name] -t
   [Template Name] -a [Application Name] -l [Version Label]
   ```
   Note

   Environment names must be at least 4 characters, and less than or equal to 23 characters.
5. Determine if the new environment is Green and Ready.
   ```
   PROMPT> elastic-beanstalk-describe-environments -E [Environment ID]
   ```
   If the new environment does not come up Green and Ready, you should decide if you want to retry the operation or leave the environment in its current state for investigation. Make sure to terminate the environment after you are finished, and clean up any unused resources.

Note

You can adjust the timeout period if the environment doesn't launch in a reasonable time.

6. After the new environment is Green and Ready, swap the environment CNAMEs.

   ```
   PROMPT> elastic-beanstalk-swap-environment-cnames -S [Source Environment
   ID] -D [Destination Environment ID]
   ```

7. After you have confirmed that the environment CNAME swap has completed, terminate the old environment.

   ```
   PROMPT> elastic-beanstalk-terminate-environment -E [Environment ID]
   ```

An example script

```
#! /usr/bin/env ruby
require 'optparse'
require 'json'
require 'timeout'

options = {}
optparse = OptionParser.new do |opts|
  opts.banner = "Usage: #{File.basename($0, ".rb")} [options] environment_id"

  options[:verbose] = false
  opts.on('-v', '--verbose', 'Output more information') do
    options[:verbose] = true;
  end

  options[:version_label] = nil
  opts.on('-l LABEL', '--version-label', 'The application version to deploy to
 the new environment') do |label|
    options[:version_label] = label
  end

  options[:application_name] = nil
  opts.on('-a NAME', '--application-name', 'The name of the application associ
ated with the version to deploy to the new environment.',
          'Note if this is different from the application associated with the
environment you are redeploying, the new environment will be associated with
this application.') do |app|
    options[:application_name] = app
  end

  opts.on('-h', '--help', 'Display this information') do
    puts opts
    exit
  end

end
optparse.parse!

verbose = options[:verbose]
puts "Verbose mode enabled" if verbose
environment_id = ARGV[0]
if (environment_id == nil)
  puts optparse.to_s
  exit
```

```
end

puts "Starting zero downtime deployment for environment: #{environment_id}" if
 verbose

## Verify Environment exists and has the right status
puts "Verifying environment exists and has correct status/health" if verbose

results = JSON.parse(%x[elastic-beanstalk-describe-environments -j -E #{envir
onment_id}])
environment_info = results['DescribeEnvironmentsResponse']['DescribeEnvironment
sResult']['Environments'][0] if (results['DescribeEnvironmentsResponse'] &&
                 results['DescribeEnvironmentsResponse']['DescribeEnvironment
sResult'] &&
                 results['DescribeEnvironmentsResponse']['DescribeEnvironment
sResult']['Environments'])

if !environment_info
  puts "No such environment found with environment_id: #{environment_id}"
  exit
end

env_health = environment_info['Health']
env_status = environment_info['Status']
puts "Current Health: #{env_health}   Current Status: #{env_status}" if verbose
if (("Green" != env_health) || ("Ready" != env_status))
  puts "Environment must be Ready and Green to perform a zero downtime deploy
ment. Current Health: #{env_health}  Current Status: #{env_status}"
  exit
end

## Verify version exists
version_label = options[:version_label]
application_name = options[:application_name]

if (! application_name)
  application_name = environment_info['ApplicationName']
end

if (! version_label)
  version_label = environment_info['VersionLabel']
end

puts "Verifying version #{version_label} in application #{application_name}
exists" if verbose
results = JSON.parse(%x[elastic-beanstalk-describe-application-versions -j -a
#{application_name} -l #{version_label}])
version_info = results['DescribeApplicationVersionsResponse']['DescribeApplica
tionVersionsResult']['ApplicationVersions'][0] if (results['DescribeApplication
VersionsResponse'] &&
              results['DescribeApplicationVersionsResponse']['DescribeApplica
tionVersionsResult'] &&
              results['DescribeApplicationVersionsResponse']['DescribeApplica
tionVersionsResult']['ApplicationVersions'])

if (!version_info)
  puts "No such version #{version_label} in application #{application_name}
exists"
```

```
  exit
end

environment_name = environment_info['EnvironmentName']

## New environment name will have 8 additional characters.  This is important 
since the environment
##  name is already limited to 25 characters in length (due to ELB usage).
new_environment_name = environment_name.gsub(/_zdd_\d+$/, "_zdd_#{rand(10)}")

## Create a Configuration Template from the existing environment
template_name = new_environment_name

puts "Creating ConfigurationTemplate named #{template_name}" if verbose
results = JSON.parse(%x[elastic-beanstalk-create-configuration-template -j -E 
#{environment_id} -a #{application_name} -t #{template_name}])
template_info = results['CreateConfigurationTemplateResponse']['CreateConfigur
ationTemplateResult'] if results['CreateConfigurationTemplateResponse']

if (! template_info)
  puts "Error when creating configuration template: #{results}"
  exit
end

## Launch a new environment with the desired version and template
puts "Creating new environment named #{new_environment_name}" if verbose
results = JSON.parse(%x[elastic-beanstalk-create-environment -j -a #{applica
tion_name} -l #{version_label} -e #{new_environment_name} -t #{template_name}}
 ])
new_environment_info = results['CreateEnvironmentResponse']['CreateEnvironmen
tResult'] if results['CreateEnvironmentResponse']

if (! new_environment_info)
  puts "Error when launching new environment: #{results}"
  exit
end

new_environment_id = new_environment_info['EnvironmentId']
puts "New environment: #{new_environment_info}" if verbose

## Wait for the new environment to go ready
## This wait can be customized for a given application to determine when it is
 really ready

max_wait_time_sec = 5 * 60;  ## 5 minutes
begin
  Timeout::timeout(max_wait_time_sec) do
    done = false
    while (! done) do
      puts "Checking if new environment is ready/green" if verbose
      results = JSON.parse(%x[elastic-beanstalk-describe-environments -j -E 
#{new_environment_id}])
      new_environment_info = results['DescribeEnvironmentsResponse']['Descri
beEnvironmentsResult']['Environments'][0] if (results['DescribeEnvironments
Response'] &&
                             results['DescribeEnvironmentsResponse']['Descri
beEnvironmentsResult'] &&
                             results['DescribeEnvironmentsResponse']['Descri
beEnvironmentsResult']['Environments'])

     puts "Current Health: #{new_environment_info['Health']}   Current Status:
 #{new_environment_info['Status']}" if verbose
     done = (new_environment_info &&("Ready" == new_environment_info['Status']))

      if (! done)
        sleep 20  ## seconds
      end
    end
  end
rescue Timeout::Error
  puts "Environment does not seem to be launching in a reasonable time.  Exiting
 after #{max_wait_time_sec} seconds"
  exit
end

results = JSON.parse(%x[elastic-beanstalk-describe-environments -j -E
#{new_environment_id}])
new_environment_info = results['DescribeEnvironmentsResponse']['DescribeEnvir
onmentsResult']['Environments'][0] if (results['DescribeEnvironmentsResponse']
 &&
                       results['DescribeEnvironmentsResponse']['DescribeEnvir
onmentsResult'] &&
                       results['DescribeEnvironmentsResponse']['DescribeEnvir
onmentsResult']['Environments'])

if (! new_environment_info)
  puts "Error waiting for environment to launch: #{results}"
  exit
end

if ("Green" != new_environment_info['Health'])
  puts "New Environment is not healthy, aborting"
  %x[elastic-beanstalk-terminate-environment -E #{new_environment_id}]
  exit
end


## Execute CNAME swap
puts "Executing CNAME Swap between old environment #{environment_id} and new
environment: #{new_environment_id}" if verbose
results = JSON.parse(%x[elastic-beanstalk-swap-environment-cnames j -S #{envir
onment_id} -D #{new_environment_id}])

## Clean up resources

## Once comfortable that the environment swap has completed the old environment
 can be terminated
#results = JSON.parse(%x[elastic-beanstalk-terminate-environment -j -E #{envir
onment_id}])

# Also delete the configuration template
#results = JSON.parse(%x[elastic-beanstalk-delete-configuration-template -j -a
 #{application_name} -t #{template_name}])
```

API

To deploy with zero downtime

1. Check your current environment to make sure it is Green and Ready.
   Call `DescribeEnvironments` with the following parameter:

   - *`EnvironmentID`* = `e-pyumupm7mph`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentID=e-pyumupm7mph
   &Operation=DescribeEnvironments
   &AuthParams
   ```

2. Verify that your new application version exists.
   Call `DescribeApplicationVersions` with the following parameters:

   - *`ApplicationName`* = `SampleApp`
   - *`VersionLabel`* = `Version2`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
   &VersionLabel=Version2
   &Operation=DescribeApplicationVersions
   &AuthParams
   ```

3. Create a configuration template from the current environment.
   Call `CreateConfigurationTemplate` with the following parameters:

   - *`EnvironmentID`* = `e-pyumupm7mph`
   - *`ApplicationName`* = `SampleApp`
   - *`TemplateName`* = `MyConfigTemplate`

   Example

   ```
   https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
   &TemplateName=MyConfigTemplate
   &EnvironmentID=e-pyumupm7mph
   &Operation=CreateConfigurationTemplate
   &AuthParams
   ```

4. Launch a new environment for the new application version and template.
   Call `CreateEnvironment` with the following parameters:

   - *`ApplicationName`* = `SampleApp`
   - *`VersionLabel`* = `Version2`

- *`EnvironmentName`* = `SampleAppEnv2`

  Note

  Environment names must be at least 4 characters, and less than or equal to 23 characters.

- *`TemplateName`* = `MyConfigTemplate`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
&VersionLabel=Version2
&EnvironmentID=SampleAppEnv2
&TemplateName=MyConfigTemplate
&Operation=CreateEnvironment
&AuthParams
```

5. Determine if the new environment is Green and Ready.
   Call `DescribeEnvironments` with the following parameters:

   - *`EnvironmentID`* = `e-eup272zdrw`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentID=e-eup272zdrw
&Operation=DescribeEnvironments
&AuthParams
```

If the new environment does not come up Green and Ready, you should decide if you want to retry the operation or leave the environment in its current state for investigation. Make sure to terminate the environment after you are finished, and clean up any unused resources.

Note

You can adjust the timeout period if the environment doesn't launch in a reasonable time.

6. After the new environment is Green and Ready, swap the environment CNAMEs.
   Call `SwapEnvironmentCNAMEs` with the following parameters:

   - *`SourceEnvironmentID`* = `e-pyumupm7mph`
   - *`DestinationEnvironmentID`* = `e-eup272zdrw`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?SourceEnvironmentID=e-pyu
mupm7mph
&DestinationEnvironmentIDe=e-eup272zdrw
&Operation=SwapEnvironmentCNAMEs
&AuthParams
```

7. After you have confirmed that the environment CNAME swap has completed, terminate the old environment.
   Call `TerminateEnvironment` with the following parameters:

- *EnvironmentID* = `e-pyumupm7mph`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentId=e-pyumupm7mph
&Operation=TerminateEnvironment
&AuthParams
```

Monitoring Application Health

When you are running a production website, it is important to know that your application is available and responding to requests. To assist with monitoring your application's responsiveness, AWS Elastic Beanstalk performs a periodic health check on running applications and can be configured to notify you by email of potential problems.

Understanding Environment Health

To check application health status, every minute or two Elastic Beanstalk sends an HTTP HEAD request to the application health check URL. By default, the application health check URL is the application root (e.g., http://myapp.elasticbeanstalk.com/), but you can change this value using the AWS Management Console. For instructions on modifying your health check URL, see Health Checks (p. 167). AWS Elastic Beanstalk expects a response of `200 OK` for the application to be considered healthy.

AWS Elastic Beanstalk will change the environment health status to one of four color values depending on how the application responds to the health check. The following table describes the color codes.

Color	Status
Green	Your application responded to the application health check URL within the last minute.
Yellow	Your application hasn't responded to the application health check URL within the last five minutes.
Red	One of the following: • Your application hasn't responded to the application health check URL for more than five minutes. • An environment is also considered red if AWS Elastic Beanstalk detects other problems with the environment that are known to make the application unavailable (e.g., the load balancer was deleted).
Gray	Your application's health status is unknown because status is reported when the application is not in the *Ready* state.

Whenever the application health check URL fails to return a 200 OK response, AWS Elastic Beanstalk performs a series of additional checks to try to determine the cause of the failure. These additional checks include verifying the following:

- The load balancer still exists

- The Auto Scaling group still exists
- There is at least one Amazon EC2 instance “In-Service” behind the load balancer
- The Amazon EC2 security group is configured to allow ingress on port 80
- The environment CNAME exists and is pointing to the right load balancer
- All Amazon EC2 instances are communicating

In addition to the environment-level health checking, Elastic Beanstalk also communicates with every Amazon EC2 instance running as part of your Elastic Beanstalk application. Should any Amazon EC2 instance fail to respond to ten consecutive health checks, Elastic Beanstalk will terminate the instance as being unhealthy and Auto Scaling will start a new instance.

You can take several corrective actions if your application health changes to red:

- Look at environment events. You might find more information about the problem here.
- If you recently deployed a new version of the application, try rolling back to an older known working version.
- If you recently made configuration changes, try changing back to the former settings.
- If the problem appears to be with the environment, try the **Rebuild Environment** command from the **Actions** pull-down menu in the **Environment** pane.
- Try using Snapshot logs to view recent log file entries or log in to the Amazon EC2 instance and troubleshoot directly.

Viewing Application Health and Environment Status

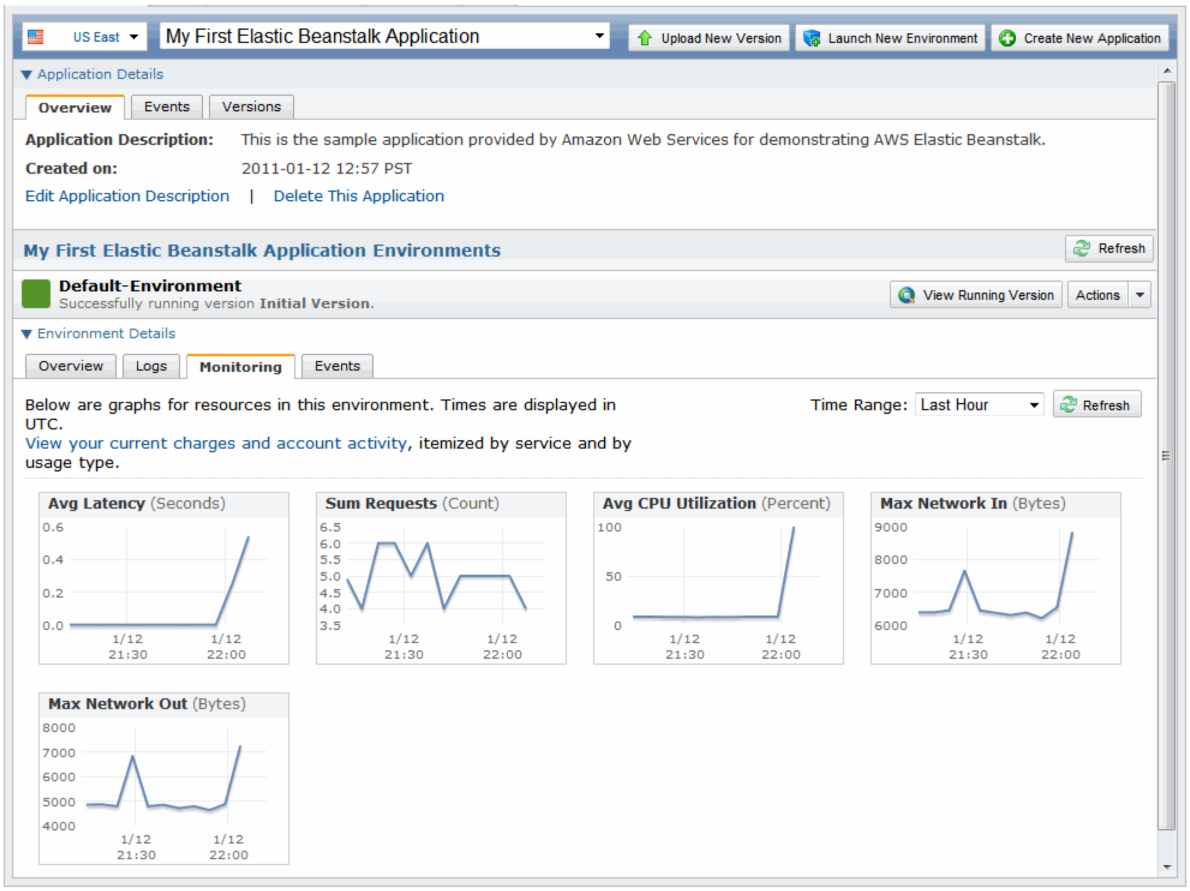
You can access operational information about your application from the AWS Management Console at http://console.aws.amazon.com/elasticbeanstalk.

The AWS Management Console displays your environment's status and application health at a glance. A color-coded box next to your environments's name indicates your application status.

To monitor application health

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Select your application from the drop-down list.
4. In the AWS Elastic Beanstalk console window below the application selection drop-down list, click the **Details** for your application in the application's **Environments** list.
5. Click the **Monitoring** tab.

   The **Monitoring** panel appears.



Application Description: This is the sample application provided by Amazon Web Services for demonstrating AWS Elastic Beanstalk.

Created on: 2011-01-12 12:57 PST

Edit Application Description | Delete This Application

My First Elastic Beanstalk Application Environments

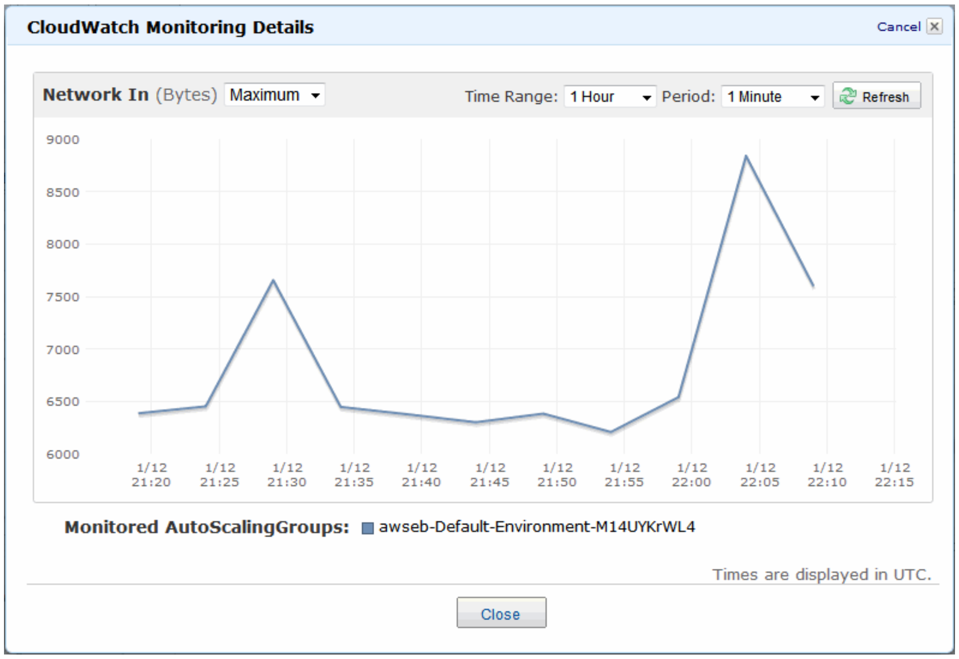
Default-Environment
Successfully running version Initial Version.

Below are graphs for resources in this environment. Times are displayed in UTC.
View your current charges and account activity, itemized by service and by usage type.

A set of graphs showing resource usage for this particular application environment is displayed. You can click any of the graphs to bring up a window that displays more detailed information.

6. Click the **Max Network In** graph.

   A detailed view of the **Max Network In** metric appears.



You can use the drop-down lists at the top of the window to change your view of the selected metric.

Note

By default, only basic Amazon CloudWatch metrics are enabled, which return data in five-minute periods. You can enable more granular one-minute Amazon CloudWatch metrics by editing your environment's configuration settings.

Viewing Events

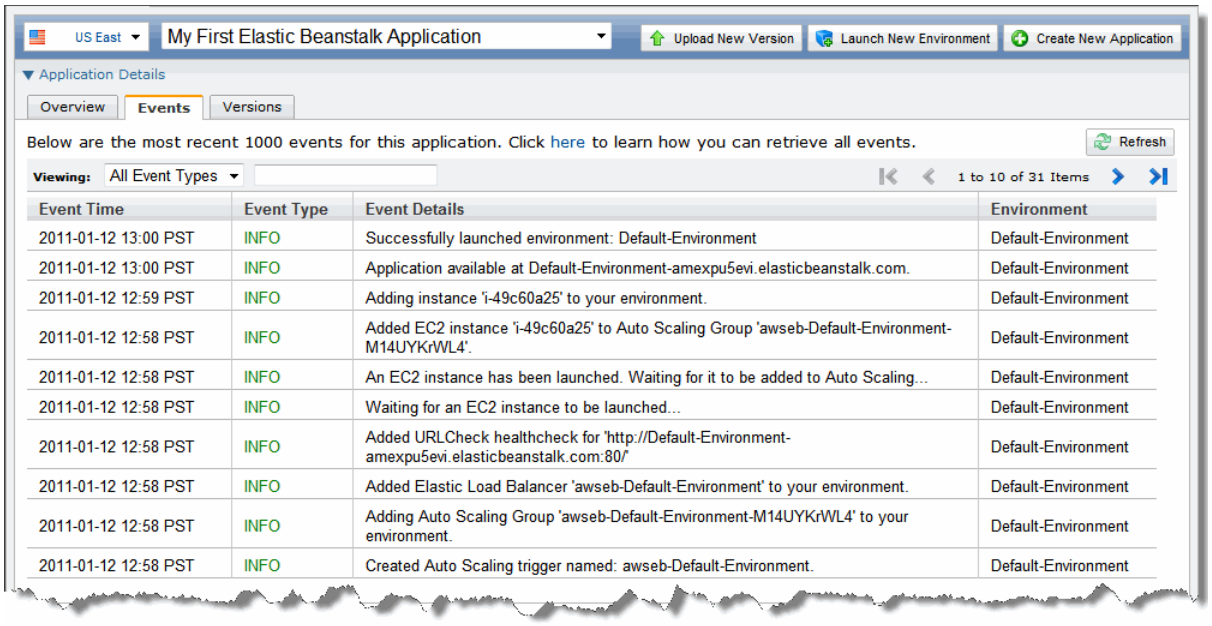
You can use the AWS Management Console to access events and notifications associated with your application. For more details on the most common events, see Understanding Environment Launch Events (p. 216). For information on how to view events using the AWS Toolkit for Eclipse, see Viewing Events (p. 36).

AWS Management Console

To view application events

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Select your application from the drop-down list.
4. In the AWS Management Console window below the application selection drop-down list, click the **Application Details** link.
5. Click the **Events** tab.

   A list of the events for all environments for your application is displayed. You can filter the type of events displayed using the **Viewing** drop-down list at the top of the list.



Below are the most recent 1000 events for this application. Click here to learn how you can retrieve all events.

Event Time	Event Type	Event Details	Environment
2011-01-12 13:00 PST	INFO	Successfully launched environment: Default-Environment	Default-Environment
2011-01-12 13:00 PST	INFO	Application available at Default-Environment-amexpu5evi.elasticbeanstalk.com.	Default-Environment
2011-01-12 12:59 PST	INFO	Adding instance 'i-49c60a25' to your environment.	Default-Environment
2011-01-12 12:58 PST	INFO	Added EC2 instance 'i-49c60a25' to Auto Scaling Group 'awseb-Default-Environment-M14UYKrWL4'.	Default-Environment
2011-01-12 12:58 PST	INFO	An EC2 instance has been launched. Waiting for it to be added to Auto Scaling...	Default-Environment
2011-01-12 12:58 PST	INFO	Waiting for an EC2 instance to be launched...	Default-Environment
2011-01-12 12:58 PST	INFO	Added URLCheck healthcheck for 'http://Default-Environment-amexpu5evi.elasticbeanstalk.com:80/'	Default-Environment
2011-01-12 12:58 PST	INFO	Added Elastic Load Balancer 'awseb-Default-Environment' to your environment.	Default-Environment
2011-01-12 12:58 PST	INFO	Adding Auto Scaling Group 'awseb-Default-Environment-M14UYKrWL4' to your environment.	Default-Environment
2011-01-12 12:58 PST	INFO	Created Auto Scaling trigger named: awseb-Default-Environment.	Default-Environment

AWS Elastic Beanstalk Events Window

CLI

You can use the AWS Elastic Beanstalk command line tools to view all events for your application. In this example, we use the command line interface to get a list of all events for an application named My First Elastic Beanstalk Application.

To view all application events

- Enter the following command at a command prompt:

  ```
  PROMPT> elastic-beanstalk-describe-events -a "My First Elastic Beanstalk
  Application"
  ```

  A list of all of the events for your application is displayed.

You can filter the events displayed using the filter parameters for the `elastic-beanstalk-describe-events` command; enter `elastic-beanstalk-describe-events --help` from a command prompt for a list of available parameters.

API

You can use the AWS Elastic Beanstalk APIs to view all events for your application. In this example, we use the APIs to get a list of all events for an application named My First Elastic Beanstalk Application.

To view all application events

- Call `DescribeEvents` with the following parameter:
  - *`ApplicationName`* = `My First Elastic Beanstalk Application`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?Application
Name=My%20First%20Elastic%20Beanstalk%20Application
&Operation=DescribeEvents
&AuthParams
```

Managing Environments

The AWS Elastic Beanstalk console enables you to save or change how the AWS resources your application environments use are provisioned and configured. This section provides information on the specific service settings you can save or edit as part of your application environment configuration. For information on manaing your application environments using the AWS Toolkit for Eclipse, see Managing AWS Elastic Beanstalk Application Environments (p. 37). For information on launching a new environment with a saved configuration, see Launching New Environments (p. 135).

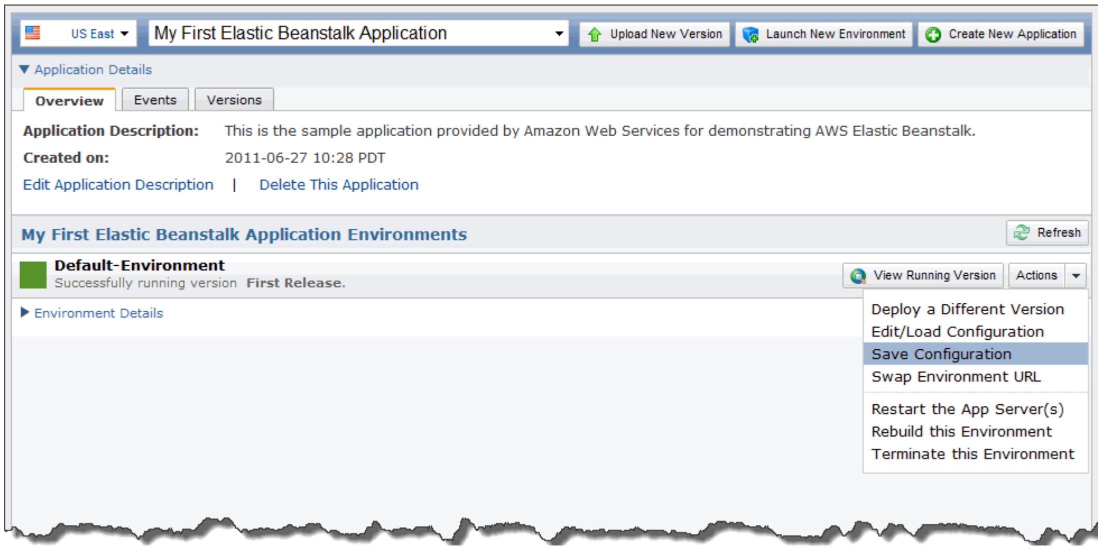
Saving Environment Configuration Settings

AWS Elastic Beanstalk configures a number of AWS cloud computing services when it deploys your application. You can save your preferred environment's configuration settings, which include the settings for these individual services, using the **Save Configuration** wizard. You can easily apply your saved environment's configuration settings to other environments.

AWS Management Console

To save an application's environment settings

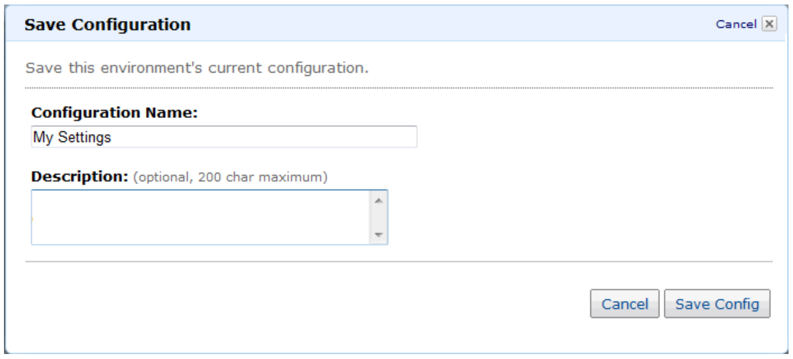
1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Select your application from the drop-down list.
4. Below the application selection drop-down list, click the **Actions** drop-down list for your application environment in the application's **Environments** list and select **Save Configuration**.



Application Description: This is the sample application provided by Amazon Web Services for demonstrating AWS Elastic Beanstalk.
Created on: 2011-06-27 10:28 PDT

Edit Application Description | Delete This Application
My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

The **Save Configuration** window is displayed.


Save Configuration
Cancel
Save this environment's current configuration.
Configuration Name:
My Settings
Description: (optional, 200 char maximum)
Cancel
Save Config

5. Type the name of the configuration you want to save in the **Configuration Name** text box.
6. Type a description for the configuration in the **Description** text box. This step is optional.
7. Click **Save Config**.

CLI

To save an application's environment settings

- Create a configuration template from an existing application.

  `PROMPT> elastic-beanstalk-create-configuration-template -a [Application Name] -t [Template Name]`

API

To save an application's environment settings

- Call `CreateConfigurationTemplate` with the following parameters:
  - *`ApplicationName`* = `SampleApp`
  - *`TemplateName`* = `MyConfigTemplate`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?ApplicationName=SampleApp
&TemplateName=MyConfigTemplate
&Operation=CreateConfigurationTemplate
&AuthParams
```

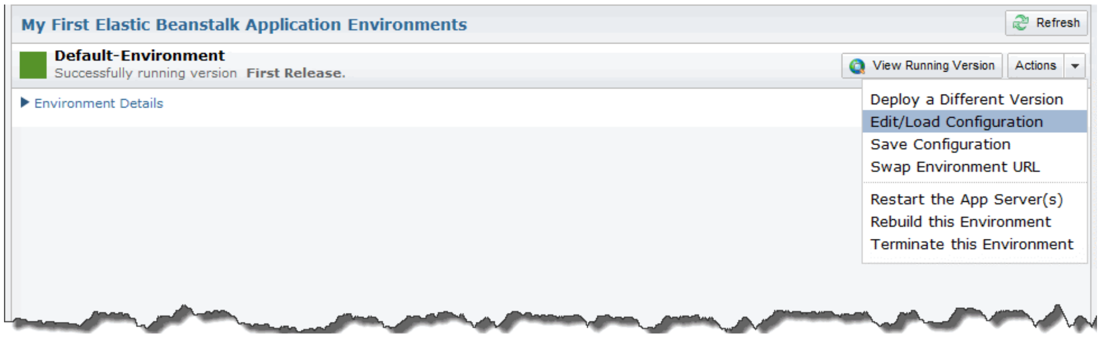
Changing Environment Configurations Settings

AWS Elastic Beanstalk configures a number of AWS cloud computing services when deploying your application. You can edit your environment's configuration settings, which include the settings for these individual services, using the **Edit Configuration** wizard.

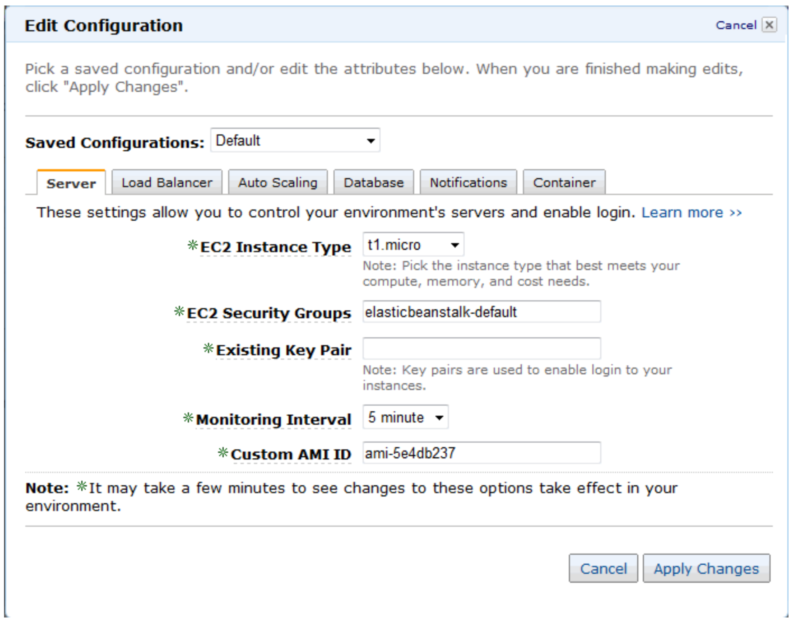
AWS Management Console

To edit an application's environment settings

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Select your application from the drop-down list.
4. Below the application selection drop-down list, click the **Actions** drop-down list for your application environment in the application's **Environments** list and select **Edit/Load Configuration**.


My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

The **Edit Configuration** window is displayed.



Edit Configuration

Cancel

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

Server | Load Balancer | Auto Scaling | Database | Notifications | Container

These settings allow you to control your environment's servers and enable login. Learn more >>

*EC2 Instance Type	t1.micro Note: Pick the instance type that best meets your compute, memory, and cost needs.
*EC2 Security Groups	elasticbeanstalk-default
*Existing Key Pair	 Note: Key pairs are used to enable login to your instances.
*Monitoring Interval	5 minute
*Custom AMI ID	ami-5e4db237

Note: *It may take a few minutes to see changes to these options take effect in your environment.

Cancel | Apply Changes

5. Select the configuration you would like to use for the environment from the **Saved Configurations** drop-down menu. The following sections describe in more detail the configuration options.

CLI

To edit an application's environment settings

- Update an application's environment settings.

```
PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f
"Options.txt"
```

Options.txt

```
[
  {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "LowerThreshold",
   "Value": "1000000"}
]
```

API

To edit an application's environment settings

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.1.OptionName`* = `LowerThreshold`
  - *`OptionSettings.member.1.Value`* = `1000000`

Example

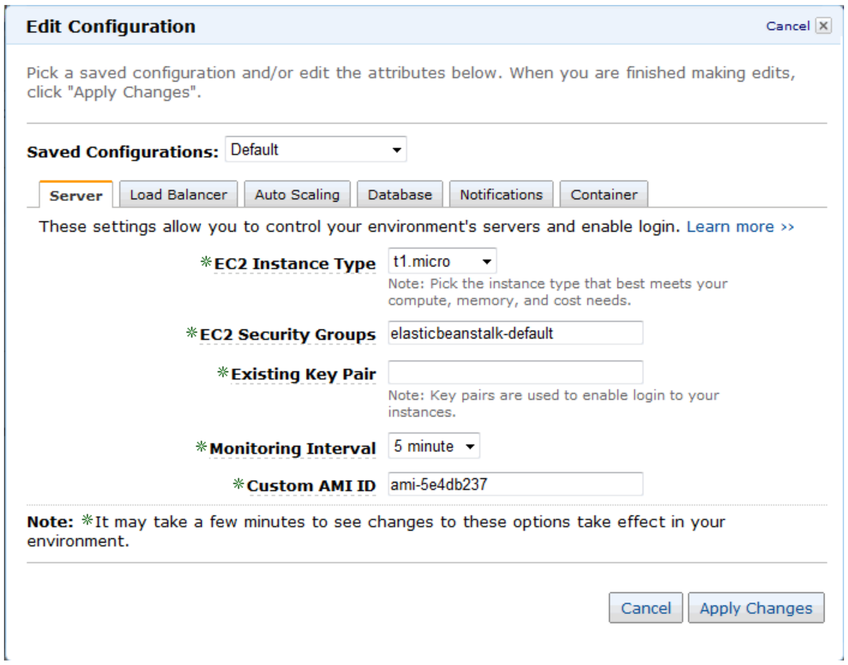
```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.1.OptionName=LowerThreshold
&OptionSettings.member.1.Value=1000000
&Operation=UpdateEnvironment
&AuthParams
```

Configuring EC2 Server Instances with AWS Elastic Beanstalk

Amazon EC2 is a web service that enables you to launch and manage server instances in Amazon's data centers. You can use Amazon EC2 server instances at any time, for as long as you need, and for any legal purpose. Instances are available in different sizes and configurations (for more information, go to the Amazon EC2 product page).

AWS Management Console

You can edit the AWS Elastic Beanstalk environment's Amazon EC2 instance configuration with the **Server** tab on the **Edit Configuration** dialog box in the AWS Management Console.



Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

These settings allow you to control your environment's servers and enable login. Learn more ››

*EC2 Instance Type	t1.micro Note: Pick the instance type that best meets your compute, memory, and cost needs.
*EC2 Security Groups	elasticbeanstalk-default
*Existing Key Pair	 Note: Key pairs are used to enable login to your instances.
*Monitoring Interval	5 minute
*Custom AMI ID	ami-5e4db237

Note: *It may take a few minutes to see changes to these options take effect in your environment.

Amazon EC2 Instance Types

The **EC2 Instance Type** drop-down list box displays the instance types available to your AWS Elastic Beanstalk application. Changing the instance type enables you to select a server with the characteristics (including memory size and CPU power) that are most appropriate to your application. For instance, applications with intensive and long-running operations may require more CPU or memory. AWS Elastic Beanstalk regularly checks your running instances to ensure they are healthy. If your application consumes 95 percent or greater of the CPU, AWS Elastic Beanstalk will fire an event. For more information about this event, see CPU Utilization Greater Than 95.00% (p. 216).

Note

You cannot change between 32-bit and 64-bit instance types. For example, if your application is built on a 32-bit platform, only 32-bit instance types appear in the list.

For more information about the Amazon EC2 instance types available for your AWS Elastic Beanstalk application, go to Instance Families and Types in the *Amazon Elastic Compute Cloud User Guide*.

Amazon EC2 Security Groups

You can control access to your AWS Elastic Beanstalk application using an *Amazon EC2 security group*. A security group defines firewall rules for your instances. These rules specify which ingress (i.e., incoming) network traffic should be delivered to your instance. All other ingress traffic will be discarded. You can modify rules for a group at any time. The new rules are automatically enforced for all running instances and instances launched in the future.

You can set up your Amazon EC2 security groups using the Amazon EC2 console. You can specify which Amazon EC2 security groups control access to your AWS Elastic Beanstalk application by entering one or more Amazon EC2 security group names (delimited by commas) into the **EC2 Security Groups** text box. For more information on Amazon EC2 security groups, see Using Security Groups in the *Amazon Elastic Compute Cloud User Guide*.

Note

If you are running your application using a legacy container type, make sure port 80 (HTTP) is accessible from 0.0.0.0/0 as the source CIDR range if you want to enable health checks for your application. For more information about health checks, see Health Checks (p. 167). To check if you are using a legacy container type, see Why are some container types marked legacy? (p. 204).

To view your Amazon EC2 security group

- In the **Edit Configuration** dialog box, on the **Server** tab, locate the name of your EC2 security group for your AWS Elastic Beanstalk environment.

To modify your Amazon EC2 security group

1. Add a new rule for 80 (HTTP) for your EC2 security group with a new source. For instructions, go to Adding a Security Group Rule in the *Amazon Elastic Compute Cloud User Guide*.
2. Type the public DNS address of your EC2 instance in your web browser to verify you can see your application. For instructions on determining your DNS address, go to Determining Your IP Addresses in the *Amazon Elastic Compute Cloud User Guide*.

Amazon EC2 Key Pairs

You can securely log in to the Amazon EC2 instances provisioned for your AWS Elastic Beanstalk application with an Amazon EC2 key pair.

Important

You must create an Amazon EC2 key pair and configure your AWS Elastic Beanstalk-provisioned Amazon EC2 instances to use the Amazon EC2 key pair before you can access your AWS Elastic Beanstalk-provisioned Amazon EC2 instances. You can set up your Amazon EC2 key pairs using the AWS Management Console. For instructions on creating a key pair for Amazon EC2, see the Amazon Elastic Compute Cloud Getting Started Guide.

The **Existing Key Pair** text box lets you specify the name of an Amazon EC2 key pair you use to securely log in to the Amazon EC2 instances running your AWS Elastic Beanstalk application.

For more information on Amazon EC2 key pairs, see Using Credentials in the *Amazon Elastic Compute Cloud User Guide*. For more information on connecting to Amazon EC2 instances, see Connecting to Instances and Connecting to an Instance from Windows using PuTTY in the *Amazon Elastic Compute Cloud User Guide*.

Monitoring Interval

By default, only basic Amazon CloudWatch metrics are enabled; they return data in five-minute periods. You can enable more granular one-minute CloudWatch metrics by selecting **`1 minute`** in the **Monitoring Interval** drop-down list box.

Note

Amazon CloudWatch service charges can apply for one-minute interval metrics. See the Amazon CloudWatch product page for more information.

Custom AMI ID

You can override the default AMI used for your Amazon EC2 instances with your own custom AMI by entering the identifier of your custom AMI into the **Custom AMI ID** text box.

Important

Using your own AMI is an advanced use case and should be done with care. If you need a custom AMI, we recommend you start with the default AWS Elastic Beanstalk AMI, and then modify it. To be considered healthy, AWS Elastic Beanstalk expects Amazon EC2 instances to meet a set of requirements, including having a running host manager. If these requirements are not met, your environment might not work properly.

CLI

To edit an application's environment settings

- Update an application's environment settings.

  `PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f "Options.txt"`

Options.txt

```
[
  {"Namespace": "aws:autoscaling:launchconfiguration",
   "OptionName": "InstanceType",
   "Value": "t1.micro"},
   {"Namespace": "aws:autoscaling:launchconfiguration",
   "OptionName": "SecurityGroups",
   "Value": "awseb-e-98pjjgr9cs-stack-AWSEBSecurityGroup-D1FOQASTKD12"},
  {"Namespace": "aws:autoscaling:launchconfiguration",
   "OptionName": "EC2KeyName",
   "Value": "mykeypair"},
   {"Namespace": "aws:autoscaling:launchconfiguration",
   "OptionName": "MonitoringInterval",
   "Value": "5 minute"},
   {"Namespace": "aws:autoscaling:launchconfiguration",
   "OptionName": "ImageId",
   "Value": "ami-cbab67a2"}
]
```

API

To edit an application's environment settings

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:autoscaling:launchconfiguration`
  - *`OptionSettings.member.1.OptionName`* = `InstanceType`
  - *`OptionSettings.member.1.Value`* = `m1.small`
  - *`OptionSettings.member.2.Namespace`* = `aws:autoscaling:launchconfiguration`
  - *`OptionSettings.member.2.OptionName`* = `SecurityGroups`
  - *`OptionSettings.member.2.Value`* = `mysecuritygroup`
  - *`OptionSettings.member.3.Namespace`* = `aws:autoscaling:launchconfiguration`
  - *`OptionSettings.member.3.OptionName`* = `EC2KeyName`
  - *`OptionSettings.member.3.Value`* = `mykeypair`

- *OptionSettings.member.4.Namespace* = aws:autoscaling:launchconfiguration
- *OptionSettings.member.4.OptionName* = MonitoringInterval
- *OptionSettings.member.4.Value* = 1 minute
- *OptionSettings.member.5.Namespace* = aws:autoscaling:launchconfiguration
- *OptionSettings.member.5.OptionName* = ImageId
- *OptionSettings.member.5.Value* = ami-cbab67a2

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.1.OptionName=InstanceType
&OptionSettings.member.1.Value=t1.micro
&OptionSettings.member.2.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.2.OptionName=SecurityGroups
&OptionSettings.member.2.Value=awseb-e-98pjjgr9cs-stack-AWSEBSecurityGroup-
D1FOQASTKD12
&OptionSettings.member.3.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.3.OptionName=EC2KeyName
&OptionSettings.member.3.Value=mykeypair
&OptionSettings.member.4.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.4.OptionName=MonitoringInterval
&OptionSettings.member.4.Value=5%20minute
&OptionSettings.member.5.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.5.OptionName=ImageId
&OptionSettings.member.5.Value=ami-cbab67a2
&Operation=UpdateEnvironment
&AuthParams
```

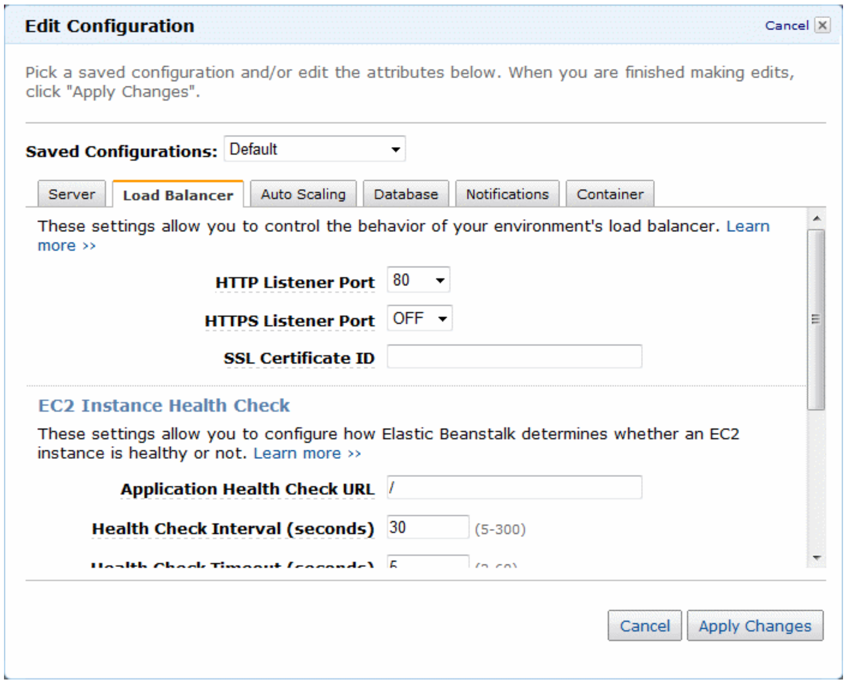
Configuring Elastic Load Balancing with AWS Elastic Beanstalk

Elastic Load Balancing is an Amazon web service that helps you improve the availability and scalability of your application. This service makes it easy for you to distribute application loads between the Amazon EC2 instances. Elastic Load Balancing enables availability through redundancy and supports traffic growth for your application.

Elastic Load Balancing lets you automatically distribute and balance the incoming application traffic among all the instances you are running. The service also makes it easy to add and remove instances when you need to increase or decrease the capacity of your application.

AWS Management Console

AWS Elastic Beanstalk automatically provisions Elastic Load Balancing when you deploy an application. You can modify your environment's Elastic Load Balancing configuration by using the **Load Balancer** tab of the **Edit Configuration** wizard.



Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

These settings allow you to control the behavior of your environment's load balancer. Learn more >>

HTTP Listener Port: 80

HTTPS Listener Port: OFF

SSL Certificate ID:

EC2 Instance Health Check

These settings allow you to configure how Elastic Beanstalk determines whether an EC2 instance is healthy or not. Learn more >>

Application Health Check URL: /

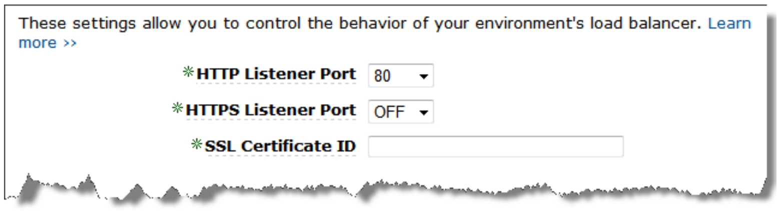
Health Check Interval (seconds): 30 (5-300)

Health Check Timeout (seconds): 5

The following sections describe the Elastic Load Balancing parameters you can configure for your application.

Ports

The load balancer provisioned to handle requests for your AWS Elastic Beanstalk application sends requests to the Amazon EC2 instances that are running your application. The provisioned load balancer can listen for requests on HTTP and HTTPS ports and route requests to the Amazon EC2 instances in your AWS Elastic Beanstalk application. By default, the load balancer handles requests on the HTTP port. At least one of the ports (either HTTP or HTTPS) must be turned on.



These settings allow you to control the behavior of your environment's load balancer. Learn more >>

*HTTP Listener Port: 80

*HTTPS Listener Port: OFF

*SSL Certificate ID:

Important

If you are deploying a an application using a legacy container type, make sure that the port you specified is not locked down; otherwise, users will not be able to connect to your AWS Elastic Beanstalk application. To check if you are using a legacy container type, see Why are some container types marked legacy? (p. 204).

Note

If you are deploying an application using a non-legacy container type, and you want to access your application directly on the EC2 instance using your web browser, modify your HTTP rule in your EC2 security group. For instructions, go to Amazon EC2 Security Groups (p. 161). For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).

Controlling the HTTP port

To turn off the HTTP port, you select OFF from the **HTTP Listener Port** drop-down list. To turn on the HTTP port, you select an HTTP port (for example, 80) from the drop-down list.

Note

If you want to access your environment using a different port other than the default port 80 (e.g., port 8080), you can add a listener to the existing load balancer and configure the new listener to listen on that port. For example, using the Elastic Load Balancing command line tools, type the following command replacing `<yourloadbalancername>` with the name of your load balancer for Elastic Beanstalk.

```
elb-create-lb-listeners --lb <yourloadbalancername> --listener "pro
tocol=http, lb-port=8080, instance-port=80"
```

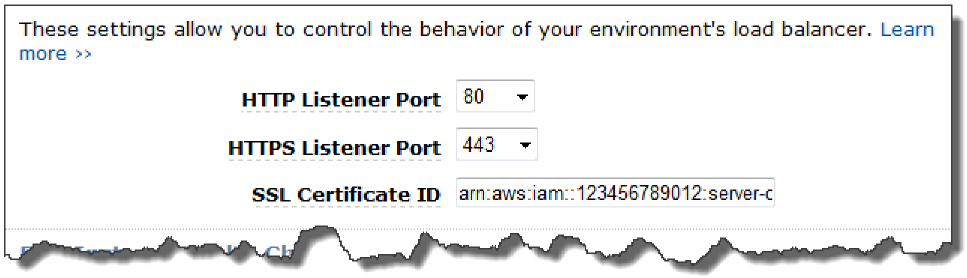
If you want Elastic Beanstalk to monitor your environment, do not remove the listener on port 80.

Controlling the HTTPS port

Elastic Load Balancing supports the HTTPS/TLS protocol to enable traffic encryption for client connections to the load balancer. Connections from the load balancer to the EC2 instances are done using plaintext. By default, the HTTPS port is turned off.

To turn on the HTTPS port

1. Create and upload a certificate and key to the AWS Access and Identity Management (AWS IAM) service. The IAM service will store the certificate and provide an Amazon Resource Name (ARN) for the SSL certificate you've uploaded. For more information on creating and uploading certificates, see the Managing Server Certificates section of *Using AWS Identity and Access Management*.
2. Specify the HTTPS port by selecting a port from the **HTTPS Listener Port** drop-down list.



These settings allow you to control the behavior of your environment's load balancer. Learn more >>

HTTP Listener Port: 80

HTTPS Listener Port: 443

SSL Certificate ID: arn:aws:iam::123456789012:server-c

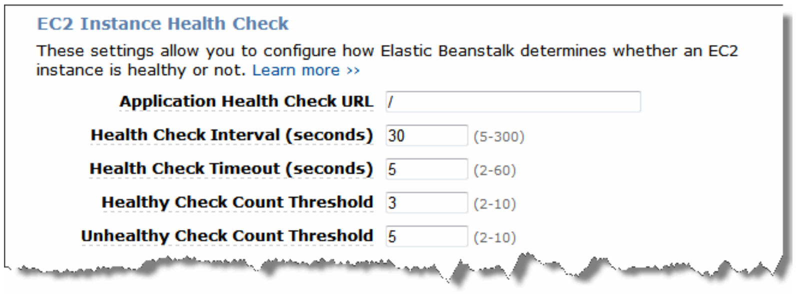
3. In the **SSL Certificate ID** box, enter the Amazon Resources Name (ARN) of your SSL certificate (e.g., `arn:aws:iam::123456789012:server-certificate/abc/certs/build`). Use the SSL certificate that you created and uploaded in step 1. For information on viewing the certificate's ARN, see Verify the Certificate Object topic in the *Creating and Uploading Server Certificates* section of the *Using IAM Guide*.

To turn off the HTTPS port, select OFF from the **HTTPS Listener Port** drop-down list.

Health Checks

Elastic Load Balancing uses a *health check* to determine whether the Amazon EC2 instances running your application are healthy. The health check determines an instance's health status by probing a specified URL at a set interval; if the URL returns an error message, or fails to return within a specified timeout period, the health check fails. AWS Elastic Beanstalk uses Elastic Load Balancing health checks to set the status of your application in the AWS Management Console.

You can control the settings for the health check using the **EC2 Instance Health Check** section of the **Load Balancing** panel.


EC2 Instance Health Check

These settings allow you to configure how Elastic Beanstalk determines whether an EC2 instance is healthy or not. Learn more ››

Application Health Check URL	/	
Health Check Interval (seconds)	30	(5-300)
Health Check Timeout (seconds)	5	(2-60)
Healthy Check Count Threshold	3	(2-10)
Unhealthy Check Count Threshold	5	(2-10)

The health check definition includes a URL to be queried for instance health. Override the default URL to match an existing resource in your application (e.g. '/myapp/index.jsp') by entering it in the **Application Health Check URL** box.

The following list describes the health check parameters you can set for your application.

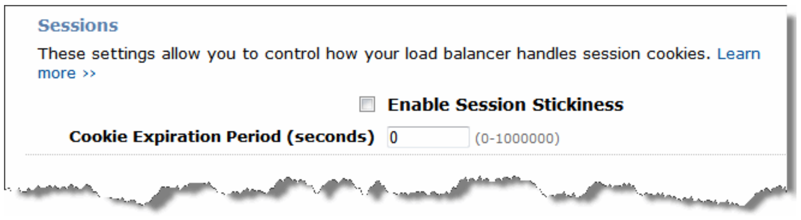
- Enter the number of seconds in the **Health Check Interval (seconds)** box to define the interval at which Elastic Load Balancing will check the health of your application's Amazon EC2 instances.
- Specify the number of seconds Elastic Load Balancing will wait for a response before it considers the instance non-responsive in the **Health Check Timeout (seconds)** box.
- Specify the number of consecutive successful or unsuccessful URL probes before Elastic Load Balancing changes the instance health status in the **Healthy Check Count Threshold** and **Unhealthy Check Count Threshold** boxes. For example, specifying 5 in the **Unhealthy Check Count Threshold** box means that the URL would have to return an error message or timeout five consecutive times before Elastic Load Balancing considers the health check "failed."

Sessions

By default a load balancer routes each request independently to the server instance with the smallest load. By comparison, a sticky session binds a user's session to a specific server instance so that all requests coming from the user during the session will be sent to the same server instance.

AWS Elastic Beanstalk uses load balancer-generated HTTP cookies when sticky sessions are enabled for an application. The load balancer uses a special load balancer-generated cookie to track the application instance for each request. When the load balancer receives a request, it first checks to see if this cookie is present in the request. If so, the request is sent to the application instance specified in the cookie. If there is no cookie, the load balancer chooses an application instance based on the existing load balancing algorithm. A cookie is inserted into the response for binding subsequent requests from the same user to that application instance. The policy configuration defines a cookie expiry, which establishes the duration of validity for each cookie.

You can use the **Sessions** section on the **Load Balancer** tab to specify whether or not the load balancer for your application allows session stickiness.


Sessions

These settings allow you to control how your load balancer handles session cookies. Learn more >>

☐ Enable Session Stickiness

Cookie Expiration Period (seconds) 0 (0-1000000)

For more information on Elastic Load Balancing, go to the Elastic Load Balancing Developer Guide.

CLI

To edit an application's environment settings

- Update an application's environment settings.

  ```
  PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f
  "Options.txt"
  ```

Options.txt

```
[
  {"Namespace": "aws:elb:loadbalancer",
   "OptionName": "LoadBalancerHTTPPort",
   "Value": "80"},
   {"Namespace": "aws:elb:loadbalancer",
   "OptionName": "LoadBalancerHTTPSPort",
   "Value": "443"},
  {"Namespace": "aws:elb:loadbalancer",
   "OptionName": "SSLCertificateId",
   "Value": "arn:aws:iam::123456789012:server-certificate/abc/certs/build"},
   {"Namespace": "aws:elasticbeanstalk:application",
   "OptionName": "Application Healthcheck URL",
   "Value": "/"},
   {"Namespace": "aws:elb:loadbalancer",
   "OptionName": "Interval",
   "Value": "30"},
   {"Namespace": "aws:elb:loadbalancer:healthcheck",
   "OptionName": "Timeout",
   "Value": "5"},
   {"Namespace": "aws:elb:loadbalancer:healthcheck",
   "OptionName": "HealthyThreshold",
   "Value": "3"},
   {"Namespace": "aws:elb:loadbalancer:healthcheck",
   "OptionName": "UnhealthyThreshold",
   "Value": "5"},
   {"Namespace": "aws:elb:policies",
   "OptionName": "Stickiness Policy",
   "Value": "false"},
   {"Namespace": "aws:elb:policies",
   "OptionName": "Stickiness Cookie Expiration",
   "Value": "0"}
]
```

API

To edit an application's environment settings

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:elb:loadbalancer`
  - *`OptionSettings.member.1.OptionName`* = `LoadBalancerHTTPPort`
  - *`OptionSettings.member.1.Value`* = `80`
  - *`OptionSettings.member.2.Namespace`* = `aws:elb:loadbalancer`
  - *`OptionSettings.member.2.OptionName`* = `LoadBalancerHTTPSPort`
  - *`OptionSettings.member.2.Value`* = `443`
  - *`OptionSettings.member.3.Namespace`* = `aws:elb:loadbalancer`
  - *`OptionSettings.member.3.OptionName`* = `SSLCertificateId`
  - *`OptionSettings.member.3.Value`* = `arn:aws:iam::123456789012:server-certificate/abc/certs/build`
  - *`OptionSettings.member.4.Namespace`* = `aws:elasticbeanstalk:application`
  - *`OptionSettings.member.4.OptionName`* = `Application Healthcheck URL`
  - *`OptionSettings.member.4.Value`* = `/`
  - *`OptionSettings.member.5.Namespace`* = `aws:elb:loadbalancer`
  - *`OptionSettings.member.5.OptionName`* = `Interval`
  - *`OptionSettings.member.5.Value`* = `30`
  - *`OptionSettings.member.6.Namespace`* = `aws:elb:loadbalancer:healthcheck`
  - *`OptionSettings.member.6.OptionName`* = `Timeout`
  - *`OptionSettings.member.6.Value`* = `5`
  - *`OptionSettings.member.7.Namespace`* = `aws:elb:loadbalancer:healthcheck`
  - *`OptionSettings.member.7.OptionName`* = `HealthyThreshold`
  - *`OptionSettings.member.7.Value`* = `3`
  - *`OptionSettings.member.8.Namespace`* = `aws:elb:loadbalancer:healthcheck`
  - *`OptionSettings.member.8.OptionName`* = `UnhealthyThreshold`
  - *`OptionSettings.member.8.Value`* = `5`
  - *`OptionSettings.member.9.Namespace`* = `aws:elb:policies`
  - *`OptionSettings.member.9.OptionName`* = `Stickiness Policy`
  - *`OptionSettings.member.9.Value`* = `false`
  - *`OptionSettings.member.10.Namespace`* = `aws:elb:policies`
  - *`OptionSettings.member.10.OptionName`* = `Stickiness Cookie Expiration`
  - *`OptionSettings.member.10.Value`* = `0`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aelb%3Aloadbalancer
&OptionSettings.member.1.OptionName=LoadBalancerHTTPPort
&OptionSettings.member.1.Value=80
&OptionSettings.member.2.Namespace=aws%3Aelb%3Aloadbalancer
&OptionSettings.member.2.OptionName=LoadBalancerHTTPSPort
&OptionSettings.member.2.Value=443
&OptionSettings.member.3.Namespace=aws%3Aelb%3Aloadbalancer
&OptionSettings.member.3.OptionName=SSLCertificateIdSSLCertificateId
&OptionSettings.member.3.Value=arn%3Aaws%3Aiam%3A%3A123456789012%3Aserver-
certificate%2Fabc%2Fcerts%2Fbuild
&OptionSettings.member.4.Namespace=aws%3Aelb%3Aapplication
&OptionSettings.member.4.OptionName=Application%20Healthcheck%20URL
&OptionSettings.member.4.Value=/
&OptionSettings.member.5.Namespace=aws%3Aelb%3Aloadbalancer
&OptionSettings.member.5.OptionName=Interval
&OptionSettings.member.5.Value=30
&OptionSettings.member.6.Namespace=aws%3Aelb%3Ahealthcheck
&OptionSettings.member.6.OptionName=Timeout
&OptionSettings.member.6.Value=5
&OptionSettings.member.7.Namespace=aws%3Aelb%3Ahealthcheck
&OptionSettings.member.7.OptionName=HealthyThreshold
&OptionSettings.member.7.Value=3
&OptionSettings.member.8.Namespace=aws%3Aelb%3Ahealthcheck
&OptionSettings.member.8.OptionName=UnhealthyThreshold
&OptionSettings.member.8.Value=5
&OptionSettings.member.9.Namespace=aws%3Aelb%3Apolicies
&OptionSettings.member.9.OptionName=Stickiness%20Policy
&OptionSettings.member.9.Value=false
&OptionSettings.member.10.Namespace=aws%3Aelb%3Apolicies
&OptionSettings.member.10.OptionName=Stickiness%20Cookie%20Expiration
&OptionSettings.member.10.Value=0
&Operation=UpdateEnvironment
&AuthParams
```

Configuring Auto Scaling with AWS Elastic Beanstalk

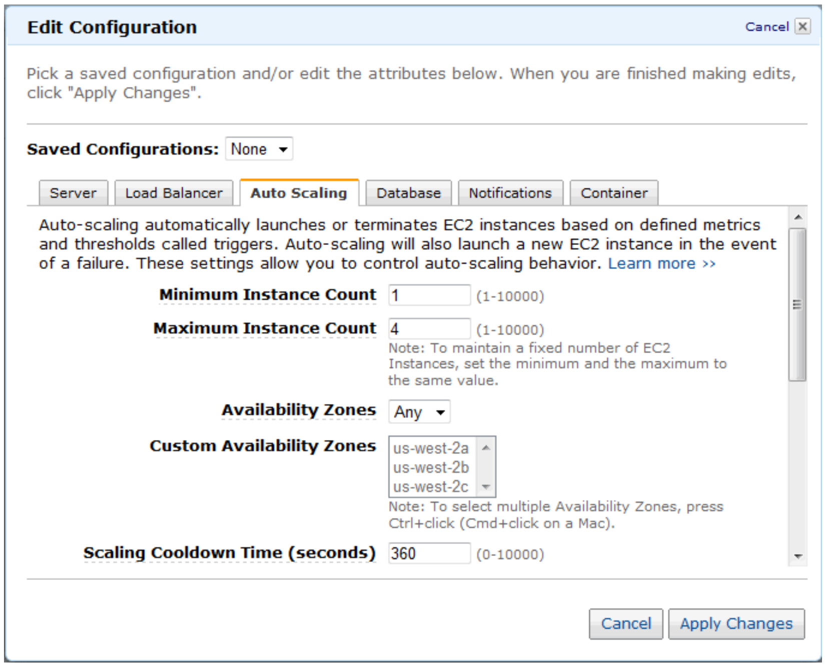
Auto Scaling is a web service designed to automatically launch or terminate Amazon EC2 instances based on user-defined triggers. Users can set up *Auto Scaling groups* and associate *triggers* with these groups to automatically scale computing resources based on metrics such as bandwidth usage or CPU utilization. Auto Scaling works with Amazon CloudWatch to retrieve metrics for the server instances running your application.

Auto Scaling lets you take a group of Amazon EC2 instances and set various parameters to have this group automatically increase or decrease in number. Auto Scaling can add or remove Amazon EC2 instances from that group to help you seamlessly deal with traffic changes to your application.

Auto Scaling also monitors the health of each Amazon EC2 instance that it launches. If any instance terminates unexpectedly, Auto Scaling detects the termination and launches a replacement instance. This capability enables you to maintain a fixed, desired number of Amazon EC2 instances automatically.

AWS Management Console

AWS Elastic Beanstalk provisions Auto Scaling for your application. You can configure how Auto Scaling behaves by clicking the **Auto Scaling** tab in the **Edit Configuration** window.



Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: None

Auto-scaling automatically launches or terminates EC2 instances based on defined metrics and thresholds called triggers. Auto-scaling will also launch a new EC2 instance in the event of a failure. These settings allow you to control auto-scaling behavior. Learn more >>

Minimum Instance Count: 1 (1-10000)

Maximum Instance Count: 4 (1-10000)
Note: To maintain a fixed number of EC2 Instances, set the minimum and the maximum to the same value.

Availability Zones: Any

Custom Availability Zones: us-west-2a, us-west-2b, us-west-2c
Note: To select multiple Availability Zones, press Ctrl+click (Cmd+click on a Mac).

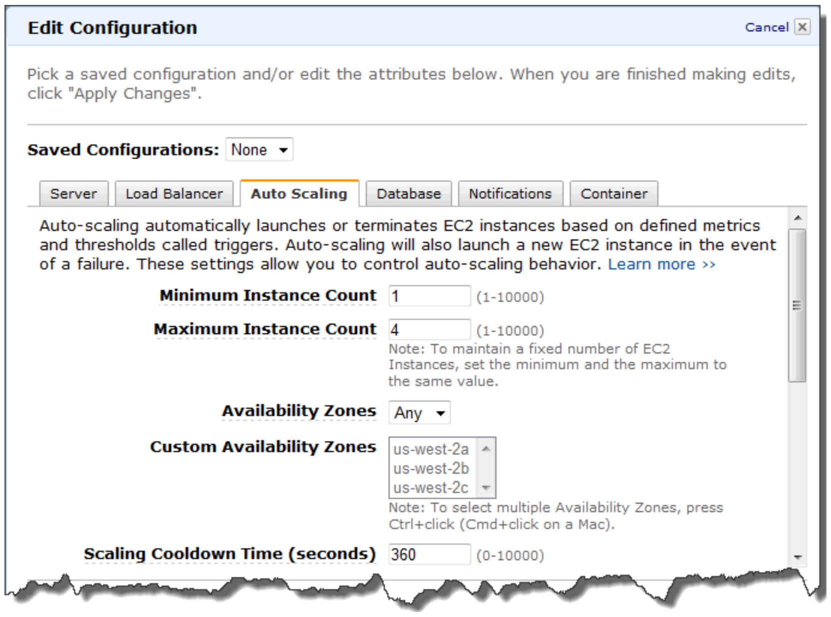
Scaling Cooldown Time (seconds): 360 (0-10000)

The following section discusses how to configure Auto Scaling parameters for your application.

Launch Configuration

You can edit the launch configuration to control how your AWS Elastic Beanstalk application provisions Auto Scaling resources.

The **Minimum Instance Count** and **Maximum Instance Count** boxes let you specify the minimum and maximum size of the Auto Scaling group that your AWS Elastic Beanstalk application uses.



Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: None

Auto-scaling automatically launches or terminates EC2 instances based on defined metrics and thresholds called triggers. Auto-scaling will also launch a new EC2 instance in the event of a failure. These settings allow you to control auto-scaling behavior.

Minimum Instance Count: 1 (1-10000)

Maximum Instance Count: 4 (1-10000)
Note: To maintain a fixed number of EC2 Instances, set the minimum and the maximum to the same value.

Availability Zones: Any

Custom Availability Zones: us-west-2a, us-west-2b, us-west-2c
Note: To select multiple Availability Zones, press Ctrl+click (Cmd+click on a Mac).

Scaling Cooldown Time (seconds): 360 (0-10000)

Note

To maintain a fixed number of Amazon EC2 instances, set the **Minimum Instance Count** and **Maximum Instance Count** boxes to the same value.

The **Availability Zones** box lets you specify if you want custom Availabilty Zones or if you want AWS Elastic Beanstalk to launch instance(s) in any Availability Zone(s) across a region.

Note

The **Availability Zones** box lets you specify the number of Availability Zones you want your Amazon EC2 instances to be in. It is important to set this number if you want to build fault-tolerant applications. If one Availability Zone goes down, your instances will still be running in your other Availability Zones.

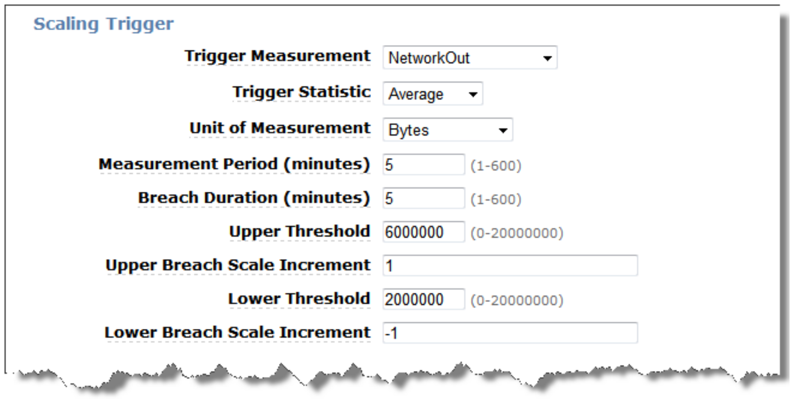
Note

If you purchased Reserved Instances, you need to specify the same Availability Zone(s) that you specified when you purchased your Reserved Instances. Reserved Instances let you make a low, one-time, upfront payment for an instance, reserve it for a one or three year term, and pay a significantly lower hourly rate for that instance. For more information about Reserved Instances Reserved Instances.

Triggers

A *trigger* is an Auto Scaling mechanism that you set to tell the system when you want to increase (*scale out*) the number of instances, and when you want to decrease (*scale in*) the number of instances. You can configure triggers to *fire* on any metric published to Amazon CloudWatch, such as CPU utilization, and determine if the conditions you specified have been met. When the upper or lower thresholds of the conditions you have specified for the metric have been breached for the specified period of time, the trigger launches a long-running process called a *Scaling Activity*.

You can define a scaling trigger for your AWS Elastic Beanstalk application using the AWS Management Console.



Scaling Trigger

Trigger Measurement	NetworkOut
Trigger Statistic	Average
Unit of Measurement	Bytes
Measurement Period (minutes)	5 (1-600)
Breach Duration (minutes)	5 (1-600)
Upper Threshold	6000000 (0-20000000)
Upper Breach Scale Increment	1
Lower Threshold	2000000 (0-20000000)
Lower Breach Scale Increment	-1

Auto Scaling triggers work by watching a specific Amazon CloudWatch metric for an instance. Triggers include CPU utilization, network traffic, and disk activity. Use the **Trigger Measurement** drop-down list to select a metric for your trigger.

The following list describes the trigger parameters you can configure using the AWS Management Console.

- You can use specify which statistic the trigger should use. You can select **`Minimum`**, **`Maximum`**, **`Sum`**, or **`Average`** using the **Trigger Statistic** drop-down list.
- Specify the unit for the trigger measurement using the **Unit of Measurement** drop-down list.
- The value in the **Measurement Period** box specifies how frequently Amazon CloudWatch measures the metrics for your trigger. The **Breach Duration** is the amount of time a metric can be beyond its defined limit (as specified in the **Upper Threshold** and **Lower Threshold** boxes) before the trigger fires.
- The **Upper Breach Scale Increment** and **Lower Breach Scale Increment** boxes specify how many Amazon EC2 instances to add or remove when performing a scaling activity.

For more information on Auto Scaling, go to the Auto Scaling documentation.

CLI

To edit an application's environment settings

- Update an application's environment settings.

  ```
  PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f
  "Options.txt"
  ```

Options.txt

```
[
  {"Namespace": "aws:autoscaling:asg",
   "OptionName": "MinSize",
   "Value": "1"},
   {"Namespace": "aws:autoscaling:asg",
   "OptionName": "MaxSize",
   "Value": "4"},
  {"Namespace": "aws:autoscaling:asg",
   "OptionName": "Availability Zones",
   "Value": "Any 1"},
   {"Namespace": "aws:autoscaling:asg",
```

```
   "OptionName": "Cooldown",
   "Value": "360"},
   {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "MeasureName",
   "Value": "NetworkOut"},
   {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "Statistic",
   "Value": "Average"},
   {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "Unit",
   "Value": "Bytes"},
   {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "Period",
   "Value": "5"},
    {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "BreachDuration",
   "Value": "5"},
    {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "UpperThreshold",
   "Value": "6000000"},
   {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "UpperBreachScaleIncrement",
   "Value": "1"},
   {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "LowerThreshold",
   "Value": "2000000"},
   {"Namespace": "aws:autoscaling:trigger",
   "OptionName": "LowerBreachScaleIncrement",
   "Value": "-1"}
   ]
```

You can also specify custom availability zones using the CLI.

To update an application's environment settings with custom availability zones

- Update an application's environment settings.

  ```
  PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f
  "Options.txt"
  ```

Options.txt

```
[
  {"Namespace": "aws:autoscaling:asg",
   "OptionName": "Custom Availability Zones",
   "Value": "us-east-1b, us-east-1c"}
]
```

> **Note**
>
> You use the command `elastic-beanstalk-update-environment` to specify the same Availability Zone(s) that you specified when you purchased your Reserved Instances. Reserved Instances let you make a low, one-time, upfront payment for an instance, reserve it for a one or three year term, and pay a significantly lower hourly rate for that instance. For more information about Reserved Instances, go to Reserved Instances in the *Amazon Elastic Compute Cloud User Guide*.

API

To edit an application's environment settings

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:autoscaling:asg`
  - *`OptionSettings.member.1.OptionName`* = `MinSize`
  - *`OptionSettings.member.1.Value`* = `1`
  - *`OptionSettings.member.2.Namespace`* = `aws:autoscaling:asg`
  - *`OptionSettings.member.2.OptionName`* = `MaxSize`
  - *`OptionSettings.member.2.Value`* = `4`
  - *`OptionSettings.member.3.Namespace`* = `aws:autoscaling:asg`
  - *`OptionSettings.member.3.OptionName`* = `Availability Zones`
  - *`OptionSettings.member.3.Value`* = `Any 1`
  - *`OptionSettings.member.4.Namespace`* = `aws:autoscaling:asg`
  - *`OptionSettings.member.4.OptionName`* = `Cooldown`
  - *`OptionSettings.member.4.Value`* = `360`
  - *`OptionSettings.member.5.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.5.OptionName`* = `MeasureName`
  - *`OptionSettings.member.5.Value`* = `NetworkOut`
  - *`OptionSettings.member.6.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.6.OptionName`* = `Statistic`
  - *`OptionSettings.member.6.Value`* = `Average`
  - *`OptionSettings.member.7.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.7.OptionName`* = `Unit`
  - *`OptionSettings.member.7.Value`* = `Bytes`
  - *`OptionSettings.member.8.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.8.OptionName`* = `Period`
  - *`OptionSettings.member.8.Value`* = `5`
  - *`OptionSettings.member.9.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.9.OptionName`* = `BreachDuration`
  - *`OptionSettings.member.9.Value`* = `5`
  - *`OptionSettings.member.10.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.10.OptionName`* = `UpperThreshold`
  - *`OptionSettings.member.10.Value`* = `6000000`
  - *`OptionSettings.member.11.Namespace`* = `aws:autoscaling:trigger`
  - *`OptionSettings.member.11.OptionName`* = `UpperBreachScaleIncrement`
  - *`OptionSettings.member.11.Value`* = `1`
  - *`OptionSettings.member.12.Namespace`* = `aws:autoscaling:trigger`

- *OptionSettings.member.12.OptionName* = `LowerThreshold`
- *OptionSettings.member.12.Value* = `2000000`
- *OptionSettings.member.13.Namespace* = `aws:autoscaling:trigger`
- *OptionSettings.member.13.OptionName* = `LowerBreachScaleIncrement`
- *OptionSettings.member.13.Value* = `-1`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Aasg
&OptionSettings.member.1.OptionName=MinSize
&OptionSettings.member.1.Value=1
&OptionSettings.member.2.Namespace=aws%3Aautoscaling%3Aasg
&OptionSettings.member.2.OptionName=MaxSize
&OptionSettings.member.2.Value=4
&OptionSettings.member.3.Namespace=aws%3Aautoscaling%3Aasg
&OptionSettings.member.3.OptionName=Availability%20Zones
&OptionSettings.member.3.Value=Any%201
&OptionSettings.member.4.Namespace=aws%3Aautoscaling%3Aasg
&OptionSettings.member.4.OptionName=Cooldown
&OptionSettings.member.4.Value=360
&OptionSettings.member.5.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.5.OptionName=MeasureName
&OptionSettings.member.5.Value=NetworkOut
&OptionSettings.member.6.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.6.OptionName=Statistic
&OptionSettings.member.6.Value=Average
&OptionSettings.member.7.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.7.OptionName=Unit
&OptionSettings.member.7.Value=Bytes
&OptionSettings.member.8.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.8.OptionName=Period
&OptionSettings.member.8.Value=5
&OptionSettings.member.9.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.9.OptionName=BreachDuration
&OptionSettings.member.9.Value=5
&OptionSettings.member.10.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.10.OptionName=UpperThreshold
&OptionSettings.member.10.Value=6000000
&OptionSettings.member.11.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.11.OptionName=UpperBreachScaleIncrement
&OptionSettings.member.11.Value=1
&OptionSettings.member.12.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.12.OptionName=LowerThreshold
&OptionSettings.member.12.Value=2000000
&OptionSettings.member.13.Namespace=aws%3Aautoscaling%3Atrigger
&OptionSettings.member.13.OptionName=Period
&OptionSettings.member.13.Value=-1
&Operation=UpdateEnvironment
&AuthParams
```

You can also specify custom availability zones using the API.

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:autoscaling:asg`
  - *`OptionSettings.member.1.OptionName`* = `Custom Availability Zones`
  - *`OptionSettings.member.1.Value`* = `us-east-1b, us-east-1c`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Aasg
&OptionSettings.member.1.OptionName=Custom%20Availability%20Zones
&OptionSettings.member.1.Value=us-east-1b%2C%20us-east-1c
&Operation=UpdateEnvironment
&AuthParams
```

Configuring Databases with AWS Elastic Beanstalk

Amazon Web Services offers a number of database options that you can leverage for your application such as Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB, and Amazon ElastiCache.

Amazon RDS is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you to focus on your applications and business.

Depending on the container type you use for your AWS Elastic Beanstalk application, there are different ways to create and access an Amazon RDS DB Instance from your application. If you have deployed an AWS Elastic Beanstalk application using a non-legacy container type, then you can use the **Database** tab to create or view an Amazon RDS DB Instance. If you already created an Amazon RDS DB when you created your application or launched a new environment, then you will see your Amazon RDS DB information on the **Database** tab. Otherwise, you can create a new Amazon RDS DB from inside the **Database** tab and attach it to an existing AWS Elastic Beanstalk environment.

If you have deployed an AWS Elastic Beanstalk using a legacy container type, then you will not see the option to create or view an Amazon RDS DB Instance in the **Database** tab. Depending on your programming language, there are several other alternatives you can use.

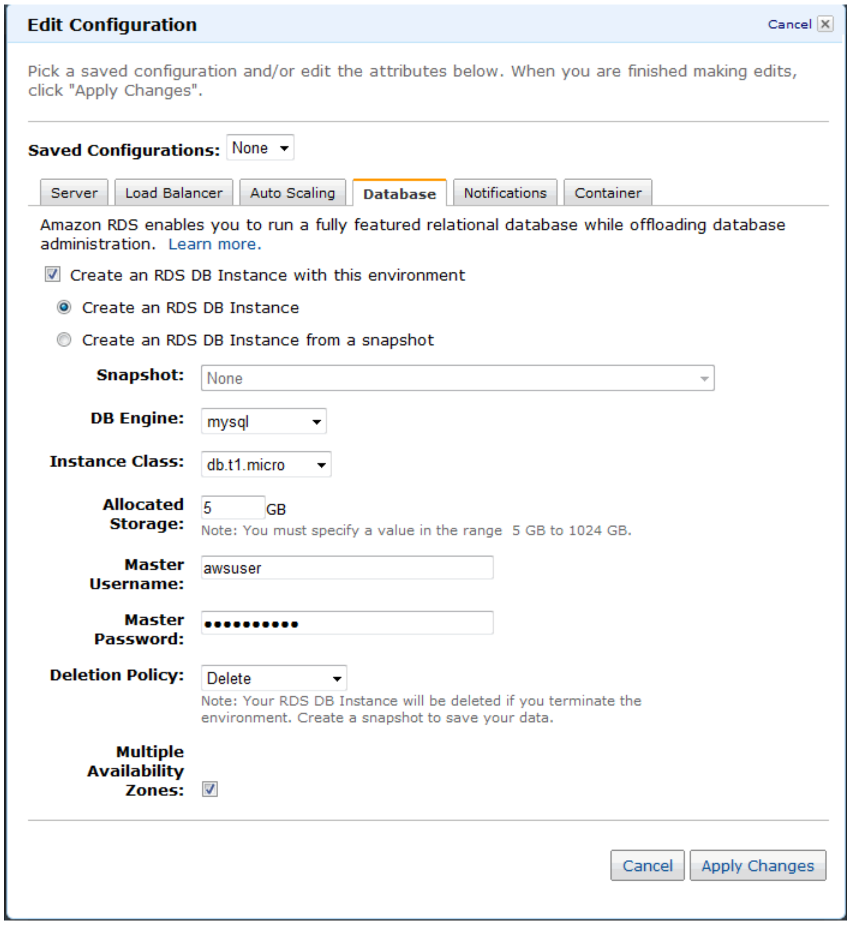
- Java - Using Amazon RDS and MySQL Connector/J (Legacy Container Types) (p. 349)
- PHP - Using Amazon RDS with PHP (p. 100)
- .NET - Amazon RDS for C# Developers

For more information about legacy and non-legacy container types, see Migrating Your Application from a Legacy Container Type (p. 204).

This topic walks you through how to create a new RDS DB Instance using the **Database** tab as well as view the RDS DB Instance information for non-legacy container types.

AWS Management Console

If you have not already associated an Amazon RDS DB with your application, you can do this inside the **Database** tab.



Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: None

Amazon RDS enables you to run a fully featured relational database while offloading database administration. Learn more.

☑ Create an RDS DB Instance with this environment

● Create an RDS DB Instance

○ Create an RDS DB Instance from a snapshot

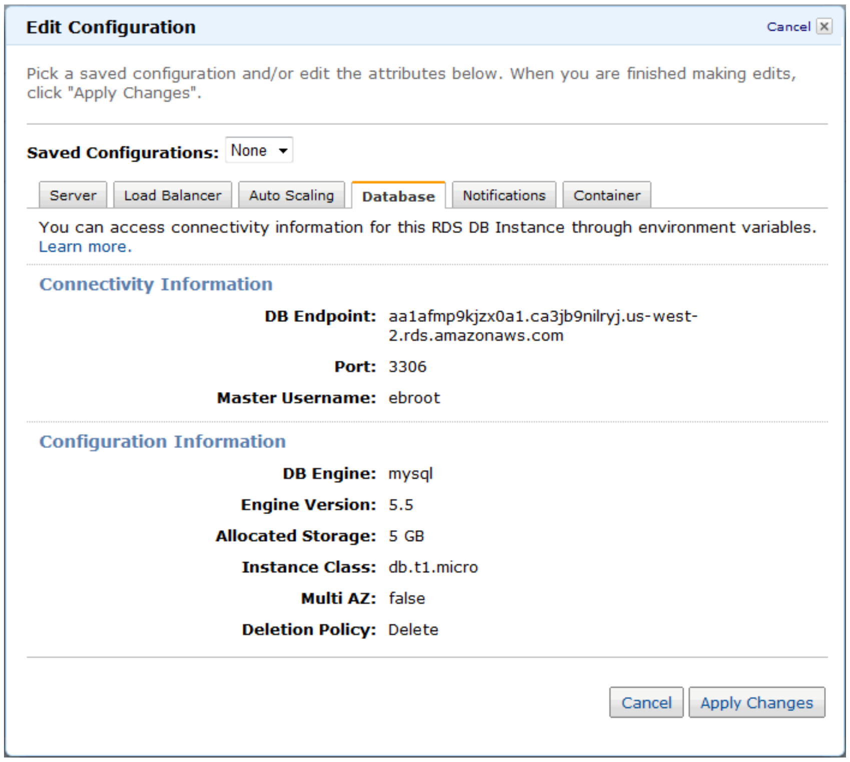
Snapshot:	None
DB Engine:	mysql
Instance Class:	db.t1.micro
Allocated Storage:	5 GB Note: You must specify a value in the range 5 GB to 1024 GB.
Master Username:	awsuser
Master Password:	●●●●●●●●●●●
Deletion Policy:	Delete Note: Your RDS DB Instance will be deleted if you terminate the environment. Create a snapshot to save your data.
Multiple Availability Zones:	☑

To create an Amazon RDS database and associate it with your existing environment

1. Select if you want to create a blank Amazon RDS DB or create one from a snapshot. If you choose to create a database from a snapshot, then select a snapshot from the **Snapshot** list.
2. Select a database engine from the **DB Engine** list.
3. Select a database instance class from the **Instance Class** list. For information about the DB instance classes, go to http://aws.amazon.com/rds/.
4. Type the allocated storage for your database in the **Allocated Storage** box. In some cases, allocating a higher amount of storage for your DB Instance than the size of your database can improve I/O performance. For information about storage allocation, go to Features.
5. Type a name in the **Master Username** box using alphanumeric characters that you will use as the master user name to log on to your DB Instance with all database privileges.
6. Type a password in the **Master Password** box containing from 8 to 16 printable ASCII characters (excluding /,", and @) for your master user password.
7. In the **Deletion Policy** list, select if you want to create a snapshot of your database or delete it if you terminate your AWS Elastic Beanstalk environment. If you select **Delete**, then your RDS DB will be deleted and your data will be lost if you terminate your environment.

8. Click **Multiple Availability Zones** if you want to configure your database across mutliple Availability Zones. Running your database instance in multiple Availability Zones safeguards your data in the unlikely event of a database instance component failure or service health disruption in one Availability Zone.
9. Click **Apply Changes**.

After you have created your DB Instance, you can view your connectivity and configuration information on the tab as shown in the following figure.



Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: None

You can access connectivity information for this RDS DB Instance through environment variables. Learn more.

Connectivity Information

DB Endpoint:	aa1afmp9kjzx0a1.ca3jb9nilryj.us-west-2.rds.amazonaws.com
Port:	3306
Master Username:	ebroot

Configuration Information

DB Engine:	mysql
Engine Version:	5.5
Allocated Storage:	5 GB
Instance Class:	db.t1.micro
Multi AZ:	false
Deletion Policy:	Delete

Use the connectivity information to connect to your DB from inside your application through environment variables. For more information about using Amazon RDS with your applications, see the following topics.

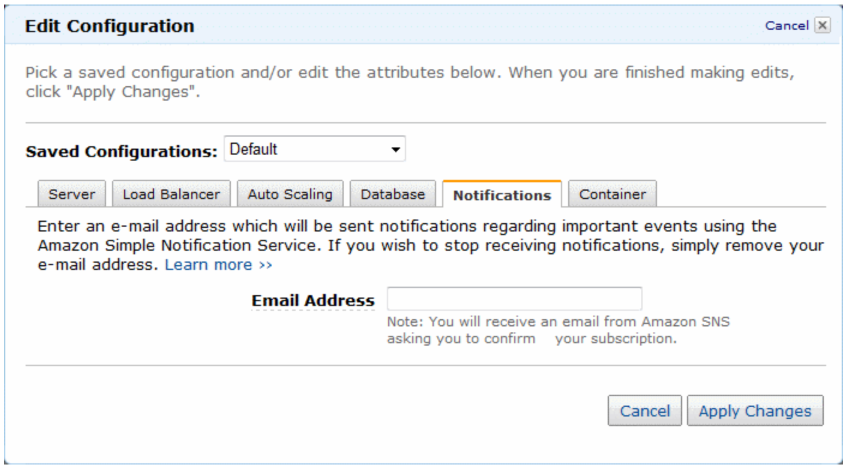
- Python - Using a New Amazon RDS DB Instance with Python (p. 120)
- Java - Accessing Custom Environment Properties (p. 32)

Configuring Notifications with AWS Elastic Beanstalk

AWS Elastic Beanstalk can use the Amazon Simple Notification Service (Amazon SNS) to notify you of important events affecting your application.

AWS Management Console

To enable Amazon SNS notifications, simply enter your email address in the **Email Address** box. To disable Amazon SNS notifications, remove your email address from the box.


Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

Enter an e-mail address which will be sent notifications regarding important events using the Amazon Simple Notification Service. If you wish to stop receiving notifications, simply remove your e-mail address. Learn more »

Email Address

Note: You will receive an email from Amazon SNS asking you to confirm your subscription.

CLI

To edit an application's environment settings

- Update an application's environment settings.

  ```
  PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f
  "Options.txt"
  ```

Options.txt

```
[
  {"Namespace": "aws:elasticbeanstalk:sns:topics",
   "OptionName": "Notification Endpoint",
   "Value": "someone@example.com"}
]
```

API

To edit an application's environment settings

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:elasticbeanstalk:sns:topics`
  - *`OptionSettings.member.1.OptionName`* = `Notification Endpoint`
  - *`OptionSettings.member.1.Value`* = `someone@example.com`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aelasticbeanstalk%3Asns%3Atopics
&OptionSettings.member.1.OptionName=Notification%20Endpoint
&OptionSettings.member.1.Value=janedoe%40example.com
&Operation=UpdateEnvironment
&AuthParams
```

Configuring Containers with AWS Elastic Beanstalk

Topics

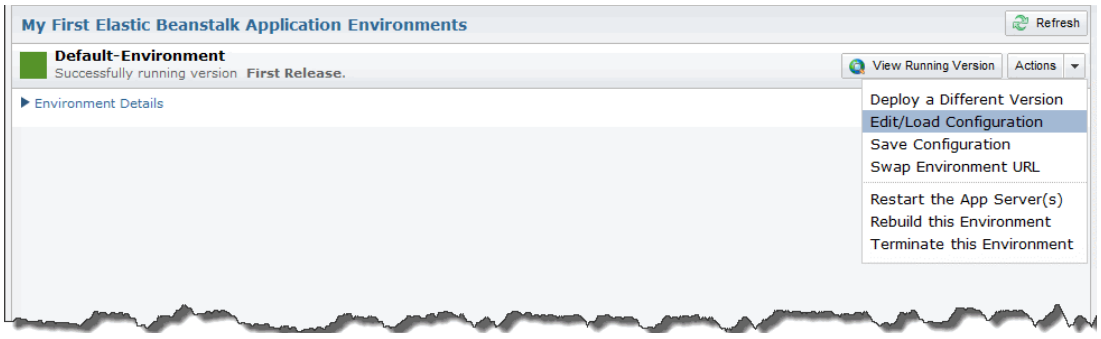
Each container has different options that can be configured and different environment properties that can be set, as well as logging options. For more information, see the relevant sections.

Configuring Java Containers with AWS Elastic Beanstalk

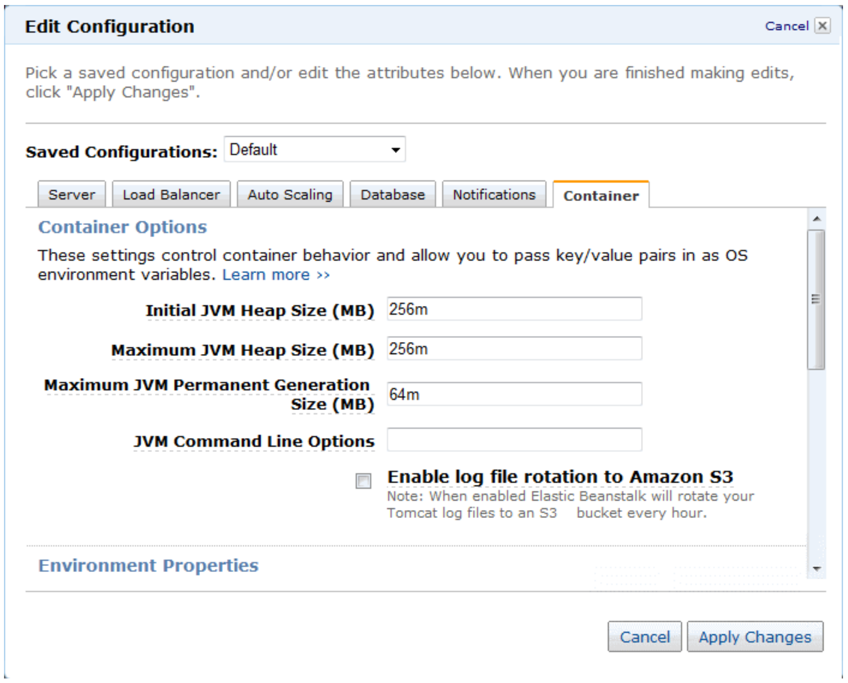
AWS Management Console

To access the Container/JVM Options panel for your AWS Elastic Beanstalk application

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Select your application from the drop-down list.
3. Below the application selection drop-down list, click the **Actions** drop-down list for your application environment in the application's **Environments** list and select **Edit/Load Configuration**.


My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

4. In the **Edit Configuration** window, click the **Container** tab.


Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

Container Options

These settings control container behavior and allow you to pass key/value pairs in as OS environment variables. Learn more ››

Initial JVM Heap Size (MB): 256m

Maximum JVM Heap Size (MB): 256m

Maximum JVM Permanent Generation Size (MB): 64m

JVM Command Line Options:

Enable log file rotation to Amazon S3

Note: When enabled Elastic Beanstalk will rotate your Tomcat log files to an S3 bucket every hour.

Environment Properties

JVM Options

The heap size in the Java Virtual Machine affects how many objects can be created in memory before *garbage collection*—a process of managing your application’s memory—occurs. You can specify an *initial heap size* and a *maximum heap size*. A larger initial heap size allows more objects to be created before garbage collection occurs, but it also means that the garbage collector will take longer to compact the heap. The maximum heap size specifies the maximum amount of memory the JVM can allocate when expanding the heap during heavy activity.

In the **Edit Configuration** window, you can set the initial and the maximum JVM heap sizes using the **Initial JVM Heap Size (MB)** and **Maximum JVM Heap Size (MB)** boxes. The available memory is dependent on the Amazon EC2 instance type. For more information about the Amazon EC2 instance types available for your AWS Elastic Beanstalk environment, go to Instance Families and Types in the *Amazon EC2 User Guide*.

The *permanent generation* is a section of the JVM heap that is used to store class definitions and associated metadata. To modify the size of the permanent generation, type the new size in the **Maximum JVM Permanent Generation Size (MB)** box.

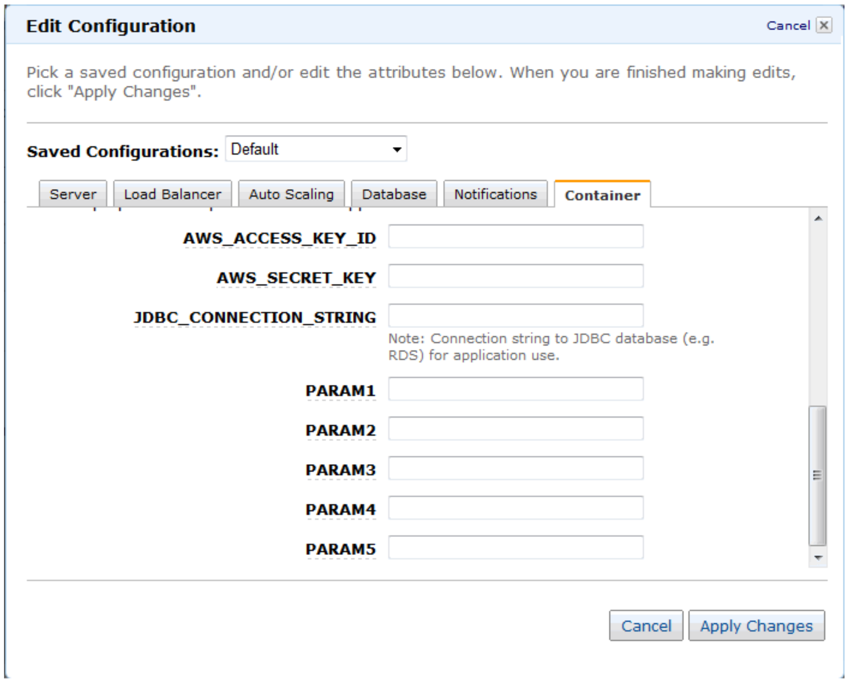
Full documentation of JVM is beyond the scope of this guide; for more information on JVM garbage collection, go to Tuning Garbage Collection with the 1.4.2 Java Virtual Machine.

Amazon S3 Log Rotation

AWS Elastic Beanstalk can copy the log files for your application's Amazon EC2 instances to the Amazon S3 bucket associated with your application on an hourly basis. To enable this feature, select **Enable log file rotation to Amazon S3**.

Environment Properties

The **Environment Properties** section of the **Container** tab on the **Edit Configuration** window lets you specify environment variables (such as connection strings and database options) on the Amazon EC2 instances that are running your application. This setting enables greater portability by eliminating the need to recompile your source code as you move between environments. Environment properties are properties specific to your application environment and are not actual (shell) environment variables. More specifically, PARAM1, PARAM2, etc. are system properties passed in to the JVM at startup using the **-D** flag. You can acquire them with `System.getProperty(name)`. For more information on using and accessing custom environment properties, see Using Custom Environment Properties with AWS Elastic Beanstalk (p. 30).


Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: Default

Note: Connection string to JDBC database (e.g. RDS) for application use.

You can use this window to configure the following environment properties:

- Specify AWS credentials using the **AWS_ACCESS_KEY_ID** and **AWS_SECRET_KEY** boxes.
- Specify a connection string to an external database (such as Amazon RDS) by entering it in the **JDBC_CONNECTION_STRING** box. For more information on how to set your JDBC_CONNECTION_STRING, see Using Custom Environment Properties with AWS Elastic Beanstalk (p. 30).
- Specify up to five additional environment properties by entering them in the **PARAM** boxes.

Note

Environment properties can contain any printable ASCII character except the grave accent (`, ASCII 96) and cannot exceed 200 characters in length.

CLI

To edit an application's environment settings for your container/JVM options

- Update an application's environment settings.

```
PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f
"Options.txt"
```

Options.txt

```
[
  {"Namespace": "aws:elasticbeanstalk:container:tomcat:jvmoptions",
   "OptionName": "Xms",
   "Value": "256m"},
   {"Namespace": "aws:elasticbeanstalk:container:tomcat:jvmoptions",
   "OptionName": "Xmx",
   "Value": "256m"},
   {"Namespace": "aws:elasticbeanstalk:container:tomcat:jvmoptions",
   "OptionName": "XX:MaxPermSize",
   "Value": "64m"},
   {"Namespace": "aws:elasticbeanstalk:container:tomcat:jvmoptions",
   "OptionName": "JVM Options",
   "Value": "somevalue"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "AWS_ACCESS_KEY_ID",
   "Value": "AKIAIOSFODNN7EXAMPLE"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "AWS_SECRET_KEY",
   "Value": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "JDBC_CONNECTION_STRING",
   "Value": "jdbc:mysql://localhost:3306/mydatabase?user=me&password=mypass
word"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "myvar",
   "Value": "somevalue"},
   {"Namespace": "aws:elasticbeanstalk:hostmanager",
   "OptionName": "LogPublicationControl",
   "Value": "false"}
]
```

API

To edit an application's environment settings for your container/JVM options

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:elasticbeanstalk:container:tomcat:jvmoptions`
  - *`OptionSettings.member.1.OptionName`* = `Xms`
  - *`OptionSettings.member.1.Value`* = `256m`
  - *`OptionSettings.member.2.Namespace`* = `aws:elasticbeanstalk:container:tomcat:jvmoptions`
  - *`OptionSettings.member.2.OptionName`* = `Xmx`
  - *`OptionSettings.member.2.Value`* = `256m`
  - *`OptionSettings.member.3.Namespace`* = `aws:elasticbeanstalk:container:tomcat:jvmoptions`
  - *`OptionSettings.member.3.OptionName`* = `XX:MaxPermSize`
  - *`OptionSettings.member.3.Value`* = `64m`
  - *`OptionSettings.member.4.Namespace`* = `aws:elasticbeanstalk:container:tomcat:jvmoptions`

- *OptionSettings.member.4.OptionName* = `JVM Options`
- *OptionSettings.member.4.Value* = `somevalue`
- *OptionSettings.member.5.Namespace* = `aws:elasticbeanstalk:application:environment`
- *OptionSettings.member.5.OptionName* = `AWS_ACCESS_KEY_ID`
- *OptionSettings.member.5.Value* = `AKIAIOSFODNN7EXAMPLE`
- *OptionSettings.member.6.Namespace* = `aws:elasticbeanstalk:application:environment`
- *OptionSettings.member.6.OptionName* = `AWS_SECRET_KEY`
- *OptionSettings.member.6.Value* = `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
- *OptionSettings.member.7.Namespace* = `aws:elasticbeanstalk:application:environment`
- *OptionSettings.member.7.OptionName* = `JDBC_CONNECTION_STRING`
- *OptionSettings.member.7.Value* = `jdbc:mysql://localhost:3306/mydatabase?user=me&password=mypassword`
- *OptionSettings.member.8.Namespace* = `aws:elasticbeanstalk:application:environment`
- *OptionSettings.member.8.OptionName* = `myvar`
- *OptionSettings.member.8.Value* = `somevalue`
- *OptionSettings.member.9.Namespace* = `aws:elasticbeanstalk:hostmanager`
- *OptionSettings.member.9.OptionName* = `LogPublicationControl`
- *OptionSettings.member.9.Value* = `false`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aelasticbeanstalk%3Acontainer%3Atom
cat%3Ajvmoptions
&OptionSettings.member.1.OptionName=Xms
&OptionSettings.member.1.Value=256m
&OptionSettings.member.2.Namespace=aws%3Aelasticbeanstalk%3Acontainer%3Atom
cat%3Ajvmoptions
&OptionSettings.member.2.OptionName=Xmx
&OptionSettings.member.2.Value=256m
&OptionSettings.member.3.Namespace=aws%3Aelasticbeanstalk%3Acontainer%3Atom
cat%3Ajvmoptions
&OptionSettings.member.3.OptionName=XX:MaxPermSize
&OptionSettings.member.3.Value=64m
&OptionSettings.member.4.Namespace=aws%3Aelasticbeanstalk%3Acontainer%3Atom
cat%3Ajvmoptions
&OptionSettings.member.4.OptionName=JVM&20Options
&OptionSettings.member.4.Value=somevalue
&OptionSettings.member.5.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.5.OptionName=AWS_ACCESS_KEY_ID
&OptionSettings.member.5.Value=AKIAIOSFODNN7EXAMPLE
&OptionSettings.member.6.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.6.OptionName=AWS_SECRET_KEY
&OptionSettings.member.6.Value=wJalrXUtnFEMI%2FK7MDENG%2FbPxRfiCYEXAMPLEKEY
&OptionSettings.member.7.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.7.OptionName=JDBC_CONNECTION_STRING
&OptionSettings.member.7.Value=jdbc%3Amysql%3A%2F%2Flocalhost%3A3306%2Fmydata
base%3Fuser%3Dme%26password%3Dmypassword
&OptionSettings.member.8.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.8.OptionName=myvar
&OptionSettings.member.8.Value=somevalue
&OptionSettings.member.9.Namespace=aws%3Aelasticbeanstalk%3Ahostmanager
&OptionSettings.member.9.OptionName=LogPublicationControl
&OptionSettings.member.9.Value=false
&Operation=UpdateEnvironment
&AuthParams
```

Configuring PHP Containers with AWS Elastic Beanstalk

You can fine-tune the behavior of your Amazon EC2 instances and enable or disable Amazon S3 log rotation. You can use the AWS Management Console, CLI, or the API to configure these options.

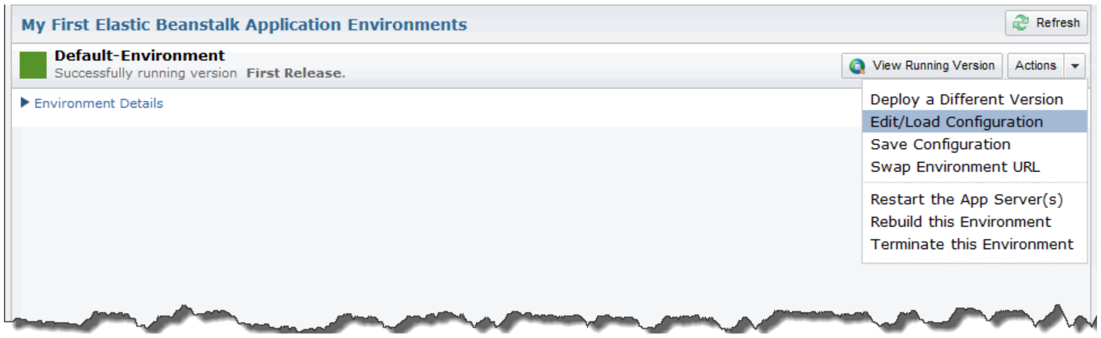
AWS Management Console

The **Container/PHP Options** panel lets you fine-tune the behavior of your Amazon EC2 instances and enable or disable Amazon S3 log rotation. You can edit the AWS Elastic Beanstalk environment's Amazon EC2 instance configuration with the **Container** tab using the AWS Management Console.

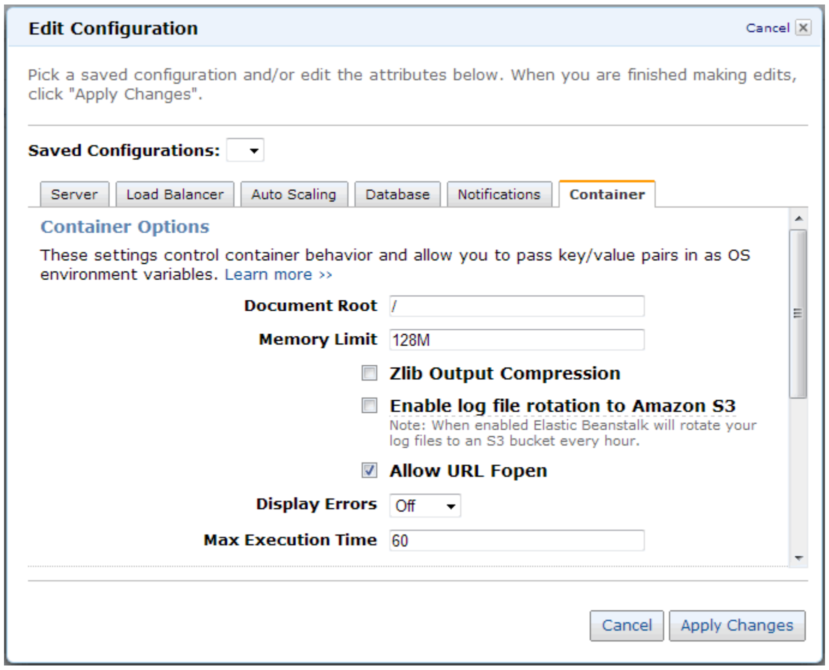
To access the Container panel for your AWS Elastic Beanstalk application

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.

2. Select your application from the drop-down list.
3. Below the application selection drop-down list, click the **Actions** drop-down list for your application environment in the application's **Environments** list and select **Edit/Load Configuration**.


My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

4. In the **Edit Configuration** window, click the **Container** tab.



Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations:

These settings control container behavior and allow you to pass key/value pairs in as OS environment variables.

Document Root: /
Memory Limit: 128M
Zlib Output Compression: unchecked
Enable log file rotation to Amazon S3: unchecked
Note: When enabled Elastic Beanstalk will rotate your log files to an S3 bucket every hour.
Allow URL Fopen: ✓
Display Errors: Off
Max Execution Time: 60

Container Options

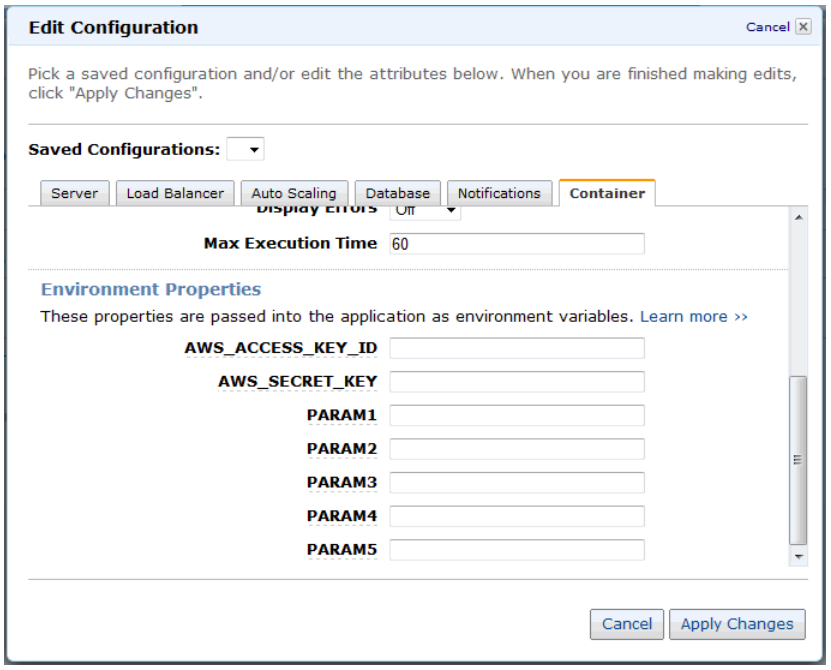
On the **Container** tab, you can specify the following options:

- **Document Root** — Enables you to specify the child directory of your project that is treated as the public-facing web root. If your root document is stored in your project directory, leave this set to "/". If your root document is inside a child directory (e.g., <*project*>/public), set this value to match the child directory. Values should begin with a "/" character, and may NOT begin with a "." character. (This value is written to the http-vhosts.conf file.)
- **Memory Limit** — Specifies the amount of memory allocated to the PHP environment. (This value is written to the php.ini file.)
- **Zlib Output Compression** — Specifies whether PHP should use compression for output. (This value is written to the php.ini file.)

- **Enable log file rotation to Amazon S3** — Specifies whether log files for your application's Amazon EC2 instances should be copied to your Amazon S3 bucket associated with your application on an hourly basis.
- **Allow URL Fopen** — Specifies whether the PHP's file functions are allowed to retrieve data from remote locations, such as websites or FTP servers. (This value is written to the php.ini file.)
- **Display Errors** —Specifies whether error messages should be part of the output. (This value is written to the php.ini file.)
- **Max Execution Time** — Sets the maximum time, in seconds, a script is allowed to run before it is terminated by the environment. This helps prevent poorly written scripts from tying up the server.

Environment Properties

The **Environment Properties** section of the **Container** tab on the **Edit Configuration** window lets you specify environment configuration settings on the Amazon EC2 instances that are running your application. Environment properties are passed in as key-value pairs to the application.


Edit Configuration

Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations:

Max Execution Time 60

Environment Properties

These properties are passed into the application as environment variables. Learn more ››

You can use this window to configure the following environment configuration settings:

- Specify AWS credentials using the **AWS_ACCESS_KEY_ID** and **AWS_SECRET_KEY** boxes.
- Specify up to five additional environment configuration settings by entering them in the **PARAM** boxes.

Note

Environment configuration settings can contain any printable ASCII character except the grave accent (`, ASCII 96) and cannot exceed 200 characters in length.

Accessing Environment Properties

Inside the PHP environment running in AWS Elastic Beanstalk, these values are written to /etc/php.d/environment.ini and are accessible using PHP's `get_cfg_var()` function.

Note

There is a subtle difference in behavior between `get_cfg_var()` and `ini_get()`. The function, `ini_get()`, will only return values that are defined by PHP core or a PHP extension, and will not return custom values. The function, `get_cfg_var()`, however, will successfully return the values of custom settings.

You might have a code snippet that looks similar to the following to access the keys and parameters:

```
echo get_cfg_var('aws.access_key');
echo get_cfg_var('aws.secret_key');
echo get_cfg_var('aws.param1');
echo get_cfg_var('aws.param2');
…
echo get_cfg_var('aws.param5');
```

CLI

To edit an application's environment configuration settings

- Update an application's environment configuration settings.

  `PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f "Options.txt"`

Options.txt

```
[
  {"Namespace": "aws:elasticbeanstalk:container:php:phpini",
   "OptionName": "document_root",
   "Value": "/"},
   {"Namespace": "aws:elasticbeanstalk:container:php:phpini",
   "OptionName": "memory_limit",
   "Value": "128M"},
   {"Namespace": "aws:elasticbeanstalk:container:php:phpini",
   "OptionName": "zlib.output_compression",
   "Value": "false"},
   {"Namespace": "aws:elasticbeanstalk:container:php:phpini",
   "OptionName": "allow_url_fopen",
   "Value": "true"},
   {"Namespace": "aws:elasticbeanstalk:container:php:phpini",
   "OptionName": "display_errors",
   "Value": "Off"},
   {"Namespace": "aws:elasticbeanstalk:container:php:phpini",
   "OptionName": "max_execution_time",
   "Value": "60"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "AWS_ACCESS_KEY_ID",
   "Value": "AKIAIOSFODNN7EXAMPLE"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "AWS_SECRET_KEY",
   "Value": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "myvar",
   "Value": "somevalue"},
   {"Namespace": "aws:elasticbeanstalk:hostmanager",
   "OptionName": "LogPublicationControl",
   "Value": "false"}
]
```

API

To edit an application's environment configuration settings

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`
  - *`OptionSettings.member.1.Namespace`* = `aws:elasticbeanstalk:container:php:phpini`
  - *`OptionSettings.member.1.OptionName`* = `document_root`
  - *`OptionSettings.member.1.Value`* = `/`
  - *`OptionSettings.member.2.Namespace`* = `aws:elasticbeanstalk:container:php:phpini`
  - *`OptionSettings.member.2.OptionName`* = `memory_limit`
  - *`OptionSettings.member.2.Value`* = `128M`
  - *`OptionSettings.member.3.Namespace`* = `aws:elasticbeanstalk:container:php:phpini`
  - *`OptionSettings.member.3.OptionName`* = `zlib.output_compression`
  - *`OptionSettings.member.3.Value`* = `false`
  - *`OptionSettings.member.4.Namespace`* = `aws:elasticbeanstalk:container:php:phpini`
  - *`OptionSettings.member.4.OptionName`* = `allow_url_fopen`
  - *`OptionSettings.member.4.Value`* = `true`
  - *`OptionSettings.member.5.Namespace`* = `aws:elasticbeanstalk:container:php:phpini`
  - *`OptionSettings.member.5.OptionName`* = `display_errors`
  - *`OptionSettings.member.5.Value`* = `Off`
  - *`OptionSettings.member.6.Namespace`* = `aws:elasticbeanstalk:container:php:phpini`
  - *`OptionSettings.member.6.OptionName`* = `max_execution_time`
  - *`OptionSettings.member.6.Value`* = `60`
  - *`OptionSettings.member.7.Namespace`* = `aws:elasticbeanstalk:application:environment`
  - *`OptionSettings.member.7.OptionName`* = `AWS_ACCESS_KEY_ID`
  - *`OptionSettings.member.7.Value`* = `AKIAIOSFODNN7EXAMPLE`
  - *`OptionSettings.member.8.Namespace`* = `aws:elasticbeanstalk:application:environment`
  - *`OptionSettings.member.8.OptionName`* = `AWS_SECRET_KEY`
  - *`OptionSettings.member.8.Value`* = `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
  - *`OptionSettings.member.9.Namespace`* = `aws:elasticbeanstalk:application:environment`
  - *`OptionSettings.member.9.OptionName`* = `myvar`
  - *`OptionSettings.member.9.Value`* = `somevalue`
  - *`OptionSettings.member.10.Namespace`* = `aws:elasticbeanstalk:hostmanager`
  - *`OptionSettings.member.10.OptionName`* = `LogPublicationControl`
  - *`OptionSettings.member.10.Value`* = `false`

Example

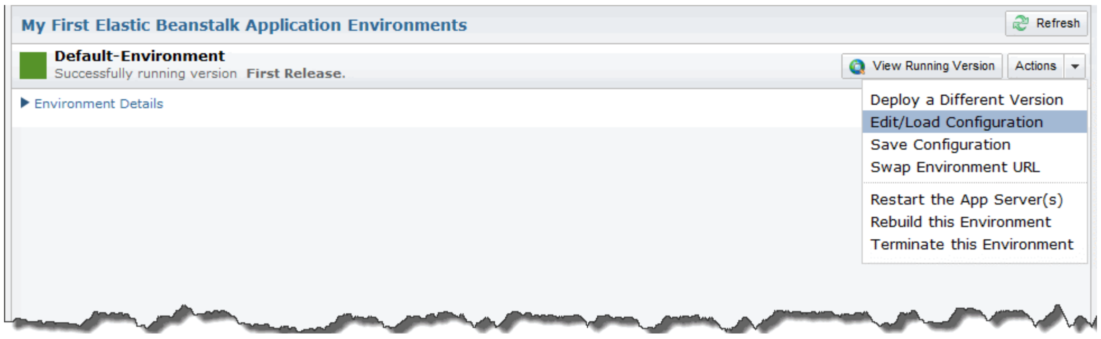
```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aelasticbeanstalk%3Acontain
er%3Aphp%3Aphpini
&OptionSettings.member.1.OptionName=document_root
&OptionSettings.member.1.Value=/
&OptionSettings.member.2.Namespace=aws%3Aelasticbeanstalk%3Acontain
er%3Aphp%3Aphpini
&OptionSettings.member.2.OptionName=memory_limit
&OptionSettings.member.2.Value=128M
&OptionSettings.member.3.Namespace=aws%3Aelasticbeanstalk%3Acontain
er%3Aphp%3Aphpini
&OptionSettings.member.3.OptionName=zlib.output_compression
&OptionSettings.member.3.Value=false
&OptionSettings.member.4.Namespace=aws%3Aelasticbeanstalk%3Acontain
er%3Aphp%3Aphpini
&OptionSettings.member.4.OptionName=allow_url_fopen
&OptionSettings.member.4.Value=true
&OptionSettings.member.5.Namespace=aws%3Aelasticbeanstalk%3Acontain
er%3Aphp%3Aphpini
&OptionSettings.member.5.OptionName=display_errors
&OptionSettings.member.5.Value=Off
&OptionSettings.member.6.Namespace=aws%3Aelasticbeanstalk%3Acontain
er%3Aphp%3Aphpini
&OptionSettings.member.6.OptionName=max_execution_time
&OptionSettings.member.6.Value=60
&OptionSettings.member.7.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.7.OptionName=AWS_ACCESS_KEY_ID
&OptionSettings.member.7.Value=AKIAIOSFODNN7EXAMPLE
&OptionSettings.member.8.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.8.OptionName=AWS_SECRET_KEY
&OptionSettings.member.8.Value=wJalrXUtnFEMI%2FK7MDENG%2FbPxRfiCYEXAMPLEKEY
&OptionSettings.member.9.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.9.OptionName=myvar
&OptionSettings.member.9.Value=somevalue
&OptionSettings.member.10.Namespace=aws%3Aelasticbeanstalk%3Ahostmanager
&OptionSettings.member.10.OptionName=LogPublicationControl
&OptionSettings.member.10.Value=false
&Operation=UpdateEnvironment
&AuthParams
```

Configuring .NET Containers with AWS Elastic Beanstalk

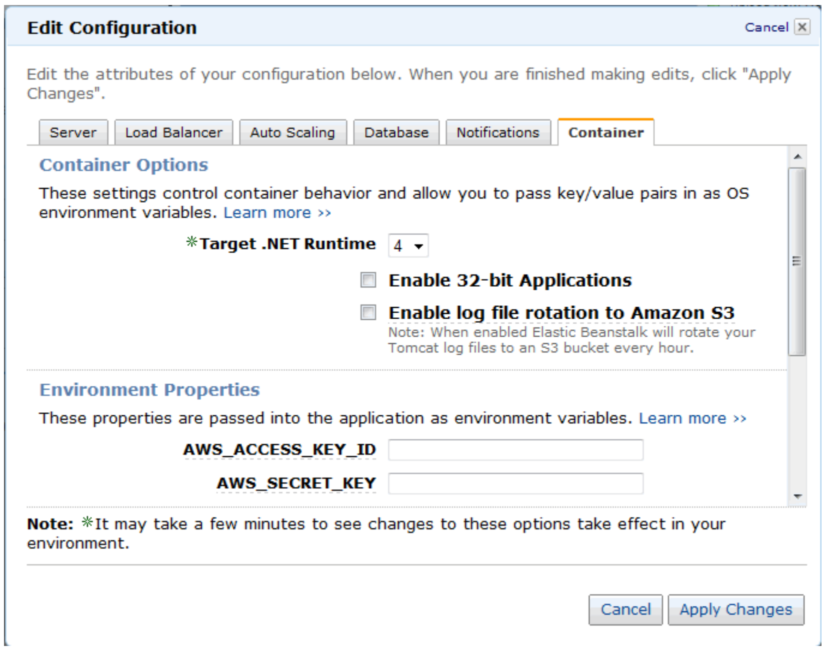
AWS Management Console

To access the Container/.NET Options panel for your AWS Elastic Beanstalk application

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Select your application from the drop-down list.
3. Below the application selection drop-down list, click the **Actions** drop-down list for your application environment in the **Environments** list and select **Edit Configuration**.


My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

4. In the **Edit Configuration** window, click the **Container** tab.



Edit Configuration

Edit the attributes of your configuration below. When you are finished making edits, click "Apply Changes".

Container Options

These settings control container behavior and allow you to pass key/value pairs in as OS environment variables. Learn more ››

*Target .NET Runtime: 4

Enable 32-bit Applications

Enable log file rotation to Amazon S3

Note: When enabled Elastic Beanstalk will rotate your Tomcat log files to an S3 bucket every hour.

Environment Properties

These properties are passed into the application as environment variables. Learn more ››

AWS_ACCESS_KEY_ID

AWS_SECRET_KEY

Note: *It may take a few minutes to see changes to these options take effect in your environment.

.NET Container Options

You can choose the version of .NET Framework for your application. You can choose either 2.0 or 4.0 in the pulldown menu. Select **Enable 32-bit Applications** if you want to enable 32-bit applications.

Amazon S3 Log Rotation

AWS Elastic Beanstalk can copy the log files for your application's Amazon EC2 instances to the Amazon S3 bucket associated with your application on an hourly basis. You can enable this feature using the AWS Management Console.

Environment Properties

The **Environment Properties** section of the **Container** panel lets you specify environment properties on the Amazon EC2 instances that are running your application. This setting enables greater portability by eliminating the need to recompile your source code as you move between environments.

You can use this panel to configure the following environment properties:

- Specify AWS credentials using the **AWS_ACCESS_KEY_ID** and **AWS_SECRET_KEY** text boxes.
- Specify up to five additional environment properties by entering them in the **PARAM** text boxes.

  You might have a code snippet that looks similar to the following to access the keys and parameters:

```
NameValueCollection appConfig = ConfigurationManager.AppSettings;

string accessKey = appConfig["AWSAccessKey"];
string secretKey = appConfig["AWSSecretKey"];
string param1 = appConfig["PARAM1"];
```

Note

Environment properties can contain any printable ASCII character except the grave accent (`, ASCII 96) and cannot exceed 200 characters in length.

CLI

To access the Container/.NET Options panel for your AWS Elastic Beanstalk application

- Update an application's environment settings.

  `PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f "Options.txt"`

Options.txt

```
[
  {"Namespace": "aws:elasticbeanstalk:container:dotnet:apppool",
   "OptionName": "Target Runtime",
   "Value": "4.0"},
   {"Namespace": "aws:elasticbeanstalk:container:dotnet:apppool",
   "OptionName": "Enable 32-bit Applications",
   "Value": "false"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "AWS_ACCESS_KEY_ID",
   "Value": "AKIAIOSFODNN7EXAMPLE"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "AWS_SECRET_KEY",
   "Value": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"},
   {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "myvar",
   "Value": "somevalue"},
   {"Namespace": "aws:elasticbeanstalk:hostmanager",
   "OptionName": "LogPublicationControl",
   "Value": "false"}
]
```

API

To access the Container/.NET Options panel for your AWS Elastic Beanstalk application

- Call `UpdateEnvironment` with the following parameters:
  - *`EnvironmentName`* = `SampleAppEnv`

- *OptionSettings.member.1.Namespace* = `aws:elasticbeanstalk:container:dotnet:apppool`
- *OptionSettings.member.1.OptionName* = `Target Runtime`
- *OptionSettings.member.1.Value* = `4.0`
- *OptionSettings.member.2.Namespace* = `aws:elasticbeanstalk:container:dotnet:apppool`
- *OptionSettings.member.2.OptionName* = `Enable 32-bit Applications`
- *OptionSettings.member.2.Value* = `false`
- *OptionSettings.member.3.Namespace* = `aws:elasticbeanstalk:application:environment`
- *OptionSettings.member.3.OptionName* = `AWS_ACCESS_KEY_ID`
- *OptionSettings.member.3.Value* = `AKIAIOSFODNN7EXAMPLE`
- *OptionSettings.member.4.Namespace* = `aws:elasticbeanstalk:application:environment`
- *OptionSettings.member.4.OptionName* = `AWS_SECRET_KEY`
- *OptionSettings.member.4.Value* = `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
- *OptionSettings.member.5.Namespace* = `aws:elasticbeanstalk:application:environment`
- *OptionSettings.member.5.OptionName* = `myvar`
- *OptionSettings.member.5.Value* = `somevalue`
- *OptionSettings.member.6.Namespace* = `aws:elasticbeanstalk:hostmanager`
- *OptionSettings.member.6.OptionName* = `LogPublicationControl`
- *OptionSettings.member.6.Value* = `false`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=MySampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aelasticbeanstalk%3Acontainer%3Adot
net%3Aapppool
&OptionSettings.member.1.OptionName=Target%20Runtime
&OptionSettings.member.1.Value=4.0
&OptionSettings.member.2.Namespace=aws%3Aelasticbeanstalk%3Acontainer%3Adot
net%3Aapppool
&OptionSettings.member.2.OptionName=Enable%2032-bit%20Applications
&OptionSettings.member.2.Value=false
&OptionSettings.member.3.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.3.OptionName=AWS_ACCESS_KEY_ID
&OptionSettings.member.3.Value=AKIAIOSFODNN7EXAMPLE
&OptionSettings.member.4.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.4.OptionName=AWS_SECRET_KEY
&OptionSettings.member.4.Value=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
&OptionSettings.member.5.Namespace=aws%3Aelasticbeanstalk%3Aapplication%3Aen
vironment
&OptionSettings.member.5.OptionName=myvar
&OptionSettings.member.5.Value=somevalue
&OptionSettings.member.6.Namespace=aws%3Aelasticbeanstalk%3Ahostmanager
&OptionSettings.member.6.OptionName=LogPublicationControl
&OptionSettings.member.6.Value=false
&Operation=UpdateEnvironment
&AuthParams
```

Configuring Python Containers with AWS Elastic Beanstalk

You can fine-tune the behavior of your Amazon EC2 instances using a configuration file to configure your container settings. For instructions on customizing and configuring a Python container, see Customizing and Configuring a Python Container (p. 103). For a list of container options, see Python Container Options (p. 285).

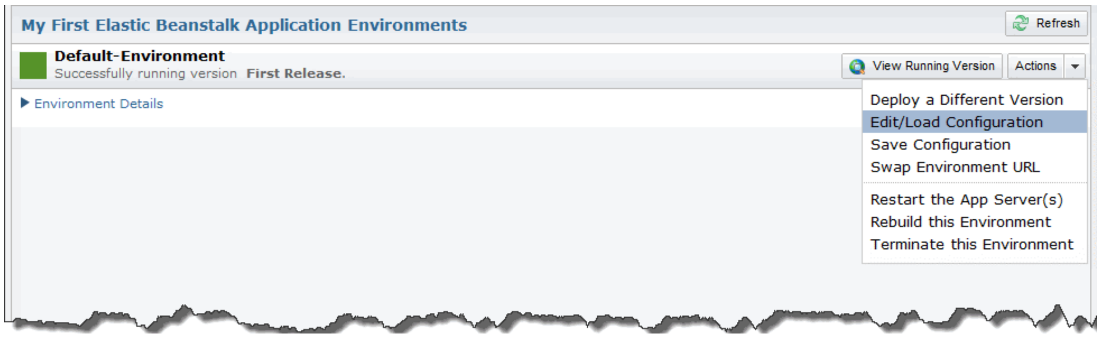
If you want to enable or disable Amazon S3 log rotation, you can use an instance configuration file, or you can use the AWS Management Console, CLI, or the API. The topic explains how to configure this setting using the AWS Management Console, CLI, and the API.

AWS Management Console

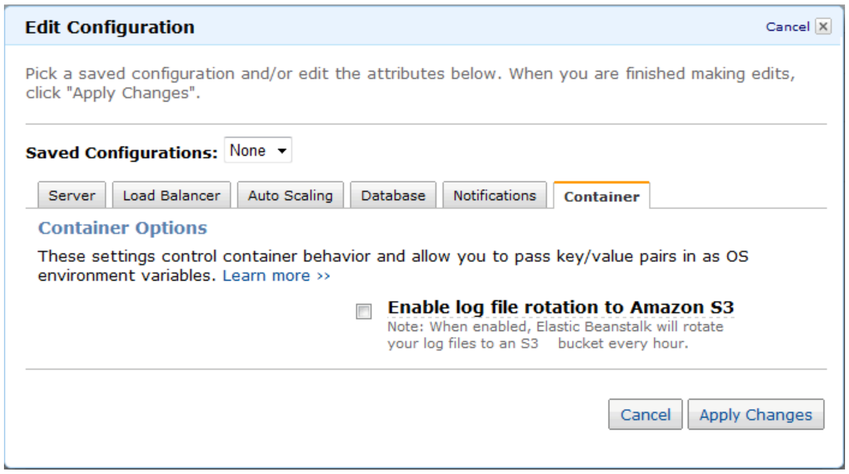
The **Container/Python Options** panel lets you enable or disable Amazon S3 log rotation.

To access the Container panel for your AWS Elastic Beanstalk application

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Select your application from the drop-down list.
3. Below the application selection drop-down list, click the **Actions** drop-down list for your application environment in the application's **Environments** list and select **Edit/Load Configuration**.


My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

4. In the **Edit Configuration** window, click the **Container** tab.



Pick a saved configuration and/or edit the attributes below. When you are finished making edits, click "Apply Changes".

Saved Configurations: None

Container Options

These settings control container behavior and allow you to pass key/value pairs in as OS environment variables. Learn more »

[ ] Enable log file rotation to Amazon S3
Note: When enabled, Elastic Beanstalk will rotate your log files to an S3 bucket every hour.

Container Options

On the **Container** tab, you can specify the following option:

- **Enable log file rotation to Amazon S3** — Specifies whether log files for your application's Amazon EC2 instances should be copied to the Amazon S3 bucket associated with your application on an hourly basis.

CLI

To edit an application's environment configuration settings

- Update an application's environment configuration settings.

  ```
  PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f
  "Options.txt"
  ```

Options.txt

```
[
   {"Namespace": "aws:elasticbeanstalk:hostmanager",
   "OptionName": "LogPublicationControl",
```

```
   "Value": "false"}
]
```

API

To edit an application's environment configuration settings

- Call `UpdateEnvironment` with the following parameters:
  - *`OptionSettings.member.1.Namespace`* = `aws:elasticbeanstalk:hostmanager`
  - *`OptionSettings.member.1.OptionName`* = `LogPublicationControl`
  - *`OptionSettings.member.1.Value`* = `false`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&OptionSettings.member.1.Namespace=aws%3Aelasticbeanstalk%3Ahostmanager
&OptionSettings.member.1.OptionName=LogPublicationControl
&OptionSettings.member.1.Value=false
&Operation=UpdateEnvironment
&AuthParams
```

Listing and Connecting to Server Instances

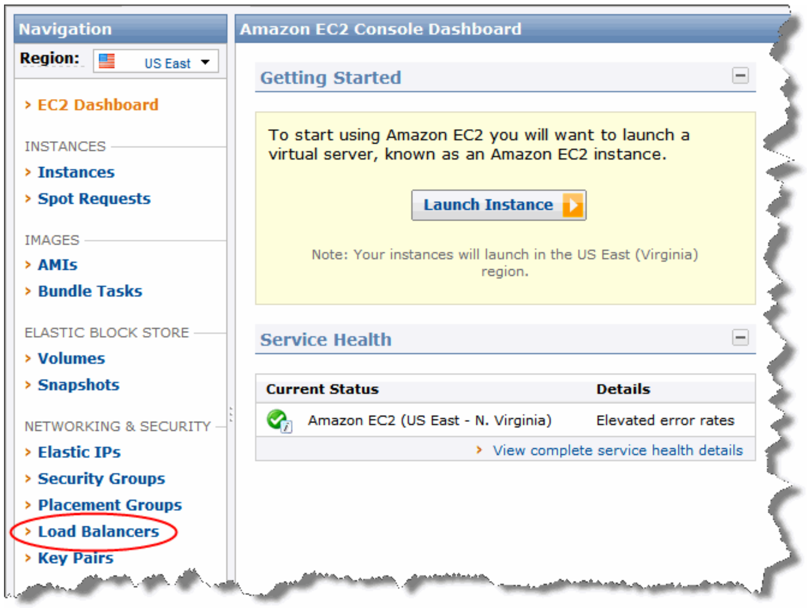
You can view a list of Amazon EC2 instances running your AWS Elastic Beanstalk application environment through the AWS Management Console. You can connect to the instances running Tomcat or PHP using any SSH client. For more information about listing and connecting to Server Instances using the AWS Toolkit for Eclipse, see Listing and Connecting to Server Instances (p. 48). You can connect to the instances running Windows using Remote Desktop. For more information about listing and connecting to Server Instances using the AWS Toolkit for Visual Studio, see Listing and Connecting to Server Instances (p. 86).

Important

You must create an Amazon EC2 key pair and configure your AWS Elastic Beanstalk–provisioned Amazon EC2 instances to use the Amazon EC2 key pair before you can access your AWS Elastic Beanstalk–provisioned Amazon EC2 instances. You can set up your Amazon EC2 key pairs using the AWS Management Console. For instructions on creating a key pair for Amazon EC2, go to the *Amazon EC2 Getting Started Guide*. For more information on how to configure your Amazon EC2 instances to use an Amazon EC2 key pair, see Amazon EC2 Key Pairs (p. 162).

To view and connect to Amazon EC2 instances for an environment

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the region list, select a region.
3. Select **Amazon EC2** from the drop-down list under the **Sign in to the AWS Management Console** button, and then click **Sign in to the AWS Management Console**.
4. Click the **Load Balancers** link in the **Navigation** list on the left side of the console window.



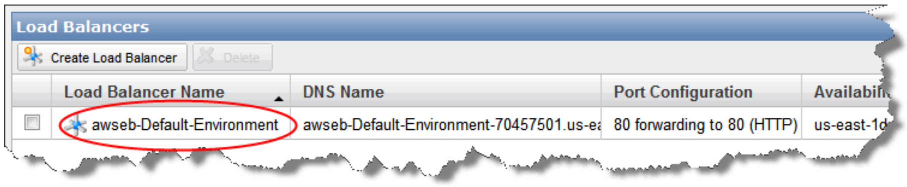
To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (Virginia) region.

Current Status	Details
Amazon EC2 (US East - N. Virginia)	Elevated error rates

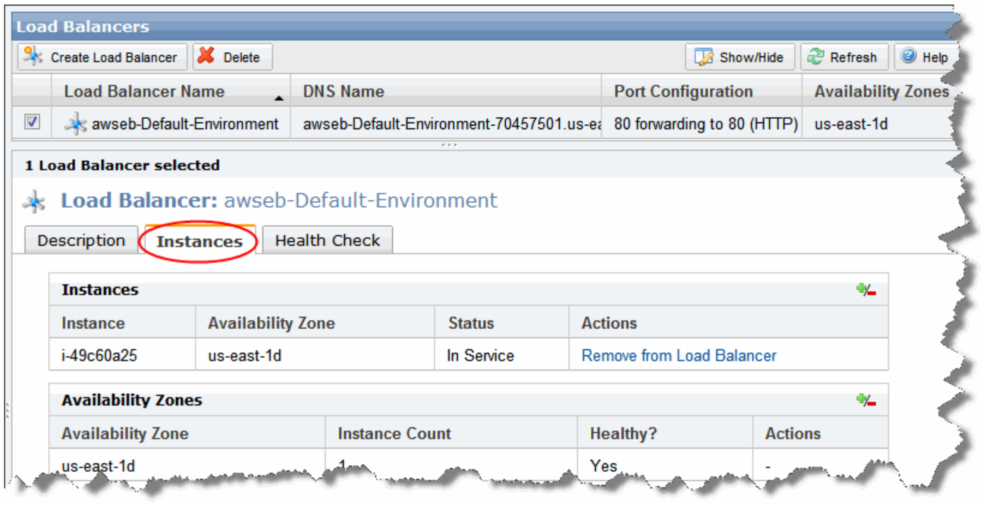
5. Load balancers created by AWS Elastic Beanstalk will have a *awseb-* prefix in the name, followed by the name of your environment. Find the load balancer for your environment and click it.

   In this example, we will click the *awseb-Default-Environment* load balancer.



Load Balancer Name	DNS Name	Port Configuration	Availabili
awseb-Default-Environment	awseb-Default-Environment-70457501.us-ea	80 forwarding to 80 (HTTP)	us-east-1d

6. Click the **Instances** tab in the bottom pane of the console window.



Load Balancer Name	DNS Name	Port Configuration	Availability Zones
awseb-Default-Environment	awseb-Default-Environment-70457501.us-ea	80 forwarding to 80 (HTTP)	us-east-1d

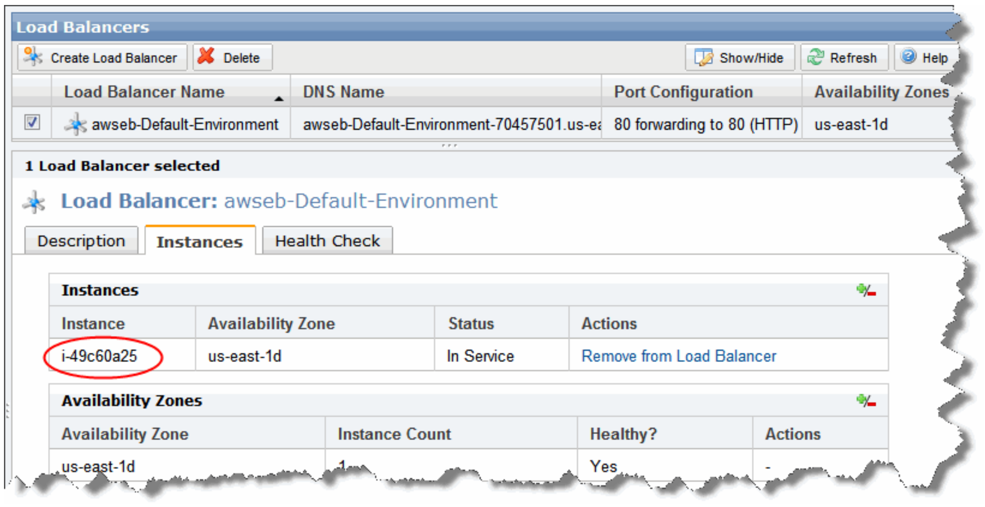
Instances

Instance	Availability Zone	Status	Actions
i-49c60a25	us-east-1d	In Service	Remove from Load Balancer

Availability Zones

Availability Zone	Instance Count	Healthy?	Actions
us-east-1d	1	Yes	-

A list of the instances that the load balancer for your AWS Elastic Beanstalk environment uses is displayed. Make a note of an instance ID that you want to connect to.



	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
✓	awseb-Default-Environment	awseb-Default-Environment-70457501.us-ea	80 forwarding to 80 (HTTP)	us-east-1d

1 Load Balancer selected

Load Balancer: awseb-Default-Environment

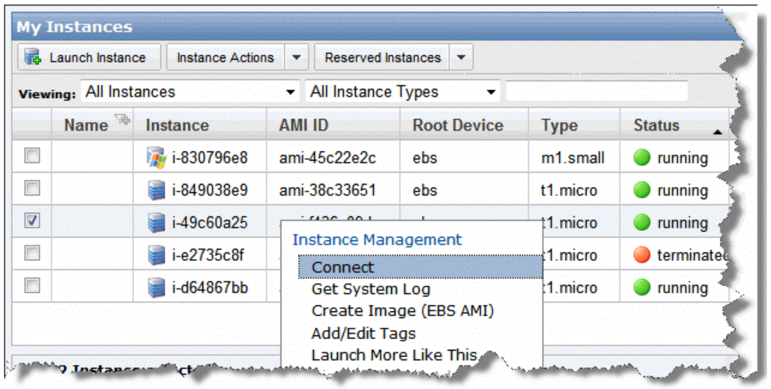
Instances

Instance	Availability Zone	Status	Actions
i-49c60a25	us-east-1d	In Service	Remove from Load Balancer

Availability Zones

Availability Zone	Instance Count	Healthy?	Actions
us-east-1d	1	Yes	-

7. Click the **Instances** link in the **Navigation** panel.
8. Right-click the instance ID for the Amazon EC2 instance running in your environment's load balancer in the **My Instances** list and select **Connect** from the context menu.



	Name	Instance	AMI ID	Root Device	Type	Status
		i-830796e8	ami-45c22e2c	ebs	m1.small	running
		i-849038e9	ami-38c33651	ebs	t1.micro	running
✓		i-49c60a25			t1.micro	running
		i-e2735c8f			t1.micro	terminated
		i-d64867bb			t1.micro	running

9. Make a note of the instance's public DNS address.
10. To connect to an instance running Linux, use the SSH client of your choice to connect to your instance and type **ssh -i .ec2/mykeypair.pem ec2-user@<public-DNS-of-the-instance>** . For instructions on how to connect to an instance running Windows, see Connect to your Windows Instance in the *Amazon Elastic Compute Cloud Getting Started Guide*.

For more information on connecting to an Amazon EC2 instance, see the Amazon Elastic Compute Cloud Getting Started Guide.

Working with Logs

You can configure your environment so that the logs from the Amazon EC2 instances running your applications are copied by AWS Elastic Beanstalk to the Amazon S3 bucket associated with your application. You can then view these logs from the AWS Elastic Beanstalk console.

To take a snapshot and view logs

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Navigate to the logs for your application.

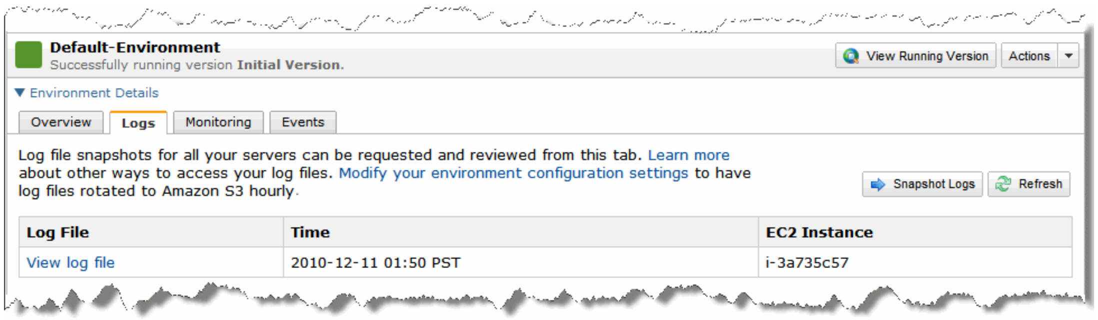
   a. Select your application from the drop-down list at the top.
   b. Below the application selection drop-down list, click the **Details** for your application in the application's **Environments** list.
   c. Click the **Logs** tab.

      The **Logs** panel appears.

3. Click the **Snapshot Logs** button.
   This action takes a snapshot of the logs for your AWS Elastic Beanstalk application. The log file appears in the **Logs** panel.

   Note

   It takes several seconds to retrieve the log files. You may need to click the **Refresh** button to see the contents of the log files.



Log file snapshots for all your servers can be requested and reviewed from this tab. Learn more about other ways to access your log files. Modify your environment configuration settings to have log files rotated to Amazon S3 hourly.

Log File	Time	EC2 Instance
View log file	2010-12-11 01:50 PST	i-3a735c57

4. Click the **View log file** link.

   A web page displaying the text output of the log file snapshot opens.

For Java applications, Tomcat uses java.util.logging package instead of the default JULI. Anything written to java.util.logging.Logger above the FINE level will be written to the appropriate log file, rotated, and viewable through the **Snapshot Logs** feature. Anything above the FINE level is currently written to the Amazon EC2 instance and be located at /var/log/tomcat6/tail_catalina.log for Tomcat 6 or /opt/tomcat7/logs for Tomcat 7. For information about accessing your server instance, go to Listing and Connecting to Server Instances (p. 197).

Terminating an Environment

You can terminate a running environment using the AWS Management Console to avoid incurring charges for unused AWS resources. For more information about terminating an environment using the AWS Toolkit for Eclipse, see Terminating an Environment (p. 49).

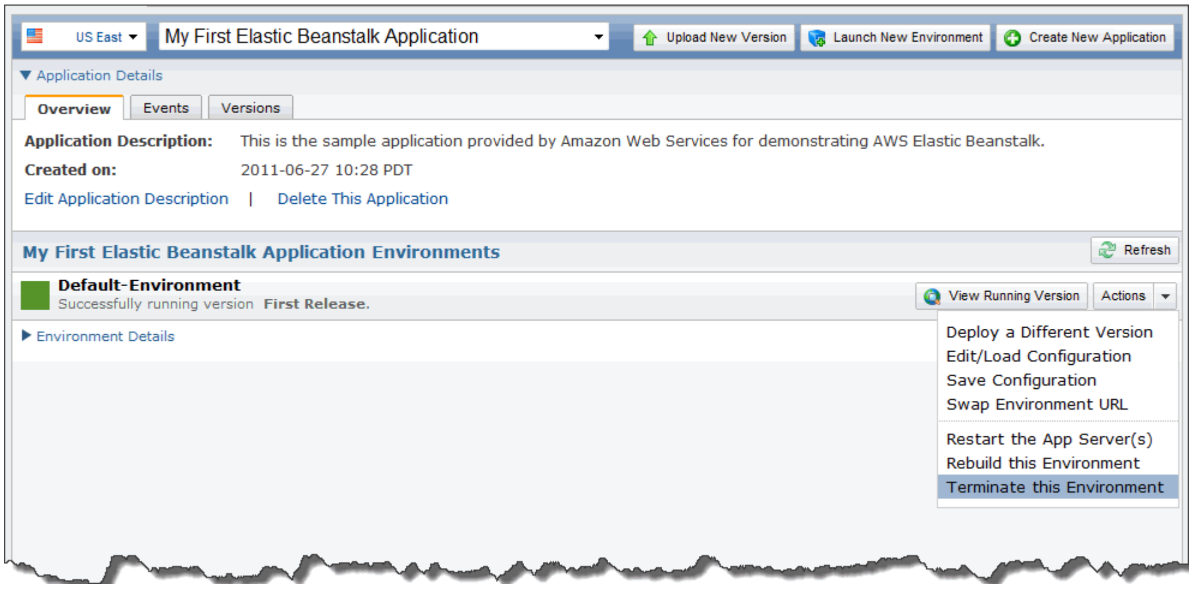
Note

You can always launch a new environment using the same version later.

AWS Management Console

To terminate an environment

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Select your application from the drop-down list.
4. Below the application selection drop-down list, click the **Details** for your application in the application's **Environments** list.
5. Click the **Actions** button at the right side of the window and select **Terminate this Environment** from the drop-down menu.



Application Description: This is the sample application provided by Amazon Web Services for demonstrating AWS Elastic Beanstalk.
Created on: 2011-06-27 10:28 PDT

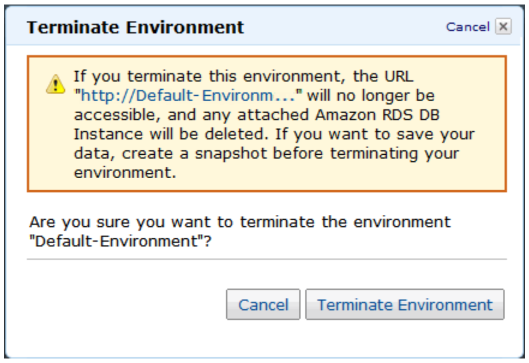
Edit Application Description | Delete This Application
My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

A dialog box appears, in which you can confirm that you want to terminate the environment.

Note

When you terminate your environment, the CNAME associated with the terminated environment becomes available for anyone to use.

6. Click **Terminate Environment** to confirm termination of the environment.



If you terminate this environment, the URL "http://Default-Environm..." will no longer be accessible, and any attached Amazon RDS DB Instance will be deleted. If you want to save your data, create a snapshot before terminating your environment.

Are you sure you want to terminate the environment "Default-Environment"?

It will take a few minutes for AWS Elastic Beanstalk to terminate the AWS resources running in the environment.

CLI

To terminate an environment

- `PROMPT> elastic-beanstalk-terminate-environment -e [Environment Name]`

API

To terminate an environment

- Call `TerminateEnvironment` with the following parameter:
  - *`EnvironmentName`* = `SampleAppEnv`

Example

```
https://elasticbeanstalk.us-east-1.amazon.com/?EnvironmentName=SampleAppEnv
&Operation=TerminateEnvironment
&AuthParams
```

Using Custom AMIs

You can customize the default Amazon Machine Image (AMI) that AWS Elastic Beanstalk uses for your applications. You can do this by customizing the existing AWS Elastic Beanstalk AMI. This section describes how to create and deploy a customized Amazon Machine Image (AMI) for use with AWS Elastic Beanstalk.

Note

If you are deploying your AWS Elastic Beanstalk application using a non-legacy container type, then you can use configuration files to customize and configure your Amazon EC2 instances. For more information about configuration files and supported container types, see Customizing and Configuring AWS Elastic Beanstalk Environments (p. 206).

Important

After you are running on your own custom AMI, you will no longer receive any automated updates to the operating system, software stack, or the AWS Elastic Beanstalk host manager.

To create and use a customized AMI

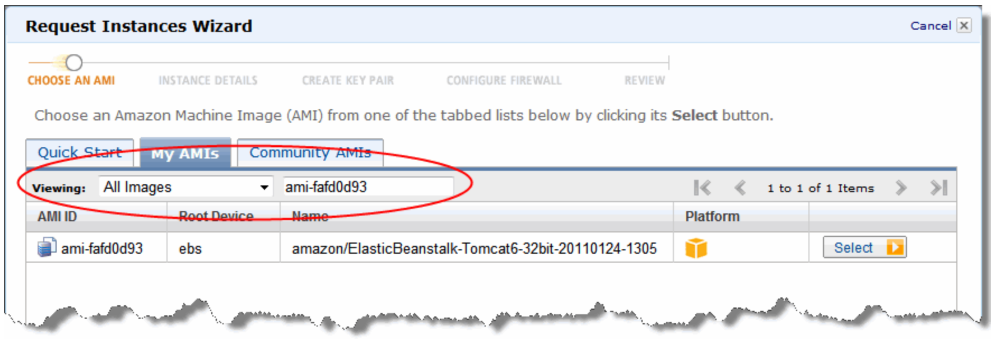
1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. From the region list, select a region.
3. Launch your AWS Elastic Beanstalk application. For more information on how to launch an AWS Elastic Beanstalk application, go to the Getting Started Using AWS Elastic Beanstalk (p. 5).
4. Find the AMI ID used by AWS Elastic Beanstalk to launch your application. This information can be viewed in the AWS Elastic Beanstalk **Edit Configuration** dialog box.

5. Use the Amazon EC2 console to launch an instance using that AMI ID:

   Note

   You must use the Amazon EC2 console to launch an instance with the AMI ID. You cannot custominze an instance that was launched by AWS Elastic Beanstalk.

   a. Open the Amazon EC2 console.
   b. Click the **Launch Instance** button. The **Request Instances Wizard** appears.



Request Instances Wizard

Choose an Amazon Machine Image (AMI) from one of the tabbed lists below by clicking its **Select** button.

Viewing: All Images ami-fafd0d93

AMI ID	Root Device	Name	Platform	
ami-fafd0d93	ebs	amazon/ElasticBeanstalk-Tomcat6-32bit-20110124-1305		Select

   c. Click the **My AMIs** tab.
   d. Select **All Images** from the **Viewing** drop-down list.
   e. Enter the AMI ID that AWS Elastic Beanstalk used to launch your application in the text box to the right of the **Viewing** drop-down list and wait for the AMI to appear in the list.
   f. Click the **Select** button next to the AMI in the list to continue.
   g. Continue using the Amazon EC2 **Request Instances Wizard** to launch the AWS Elastic Beanstalk AMI that you want to customize.

      Note

      When launching the instance using the AWS Management Console, make sure you create or specify a key pair, and that you select your EC2 security group for your AWS Elastic Beanstalk environment.

   For additional information on how to launch an Amazon EC2 instance, go to Running an Instance in the *Amazon Elastic Compute Cloud User Guide*.

6. Connect to the instance. For more information on connecting to an Amazon EC2 instance, go to Connecting to Instances in the *Amazon Elastic Compute Cloud User Guide*.

   Important

   You must leave the AWS Elastic Beanstalk host manager running on your instance. After you are running on your own custom AMI, you will no longer receive any automated updates to the operating system, software stack, or the AWS Elastic Beanstalk host manager.

7. If you are using an AMI with Apache and Tomcat, you will need to perform your customizations.

   Apache and Tomcat are not automatically started when you manually launch the AWS Elastic Beanstalk AMI using the Amazon EC2 tab on the AWS Management Console. Enter the following commands at your Amazon EC2 instance's command prompt to start Apache and Tomcat.

```
sudo -s
```

```
cd /etc/init.d
./httpd start
./tomcat7 start
```

8. From the Amazon EC2 console on the AWS Management Console, select the running instance that you've just modified and select **Create Image (EBS AMI)** from the **Instance Actions** menu. For more information on how to create an image from a running instance, go to Creating an Image from a Running Instance in the *Amazon Elastic Compute Cloud User Guide*. You can also view the Create Your Own Customized AMI video.
9. To avoid incurring additional AWS charges, terminate the Amazon EC2 instance you used to create the AMI. For instructions on how to terminate an instance, see Terminating an Environment (p. 200).
10. To use your custom AMI, specify your custom AMI ID in the **Custom AMI ID** text box in the AWS Elastic Beanstalk **Edit Configuration** dialog box. Existing instances will be replaced with new instances launched from the new custom AMI.

Migrating Your Application from a Legacy Container Type

If you have deployed an AWS Elastic Beanstalk Java application that uses a legacy container type, you should migrate your application to a new environment using a non-legacy container type so that you can get access to new features. If you are unsure if you are running your application using a legacy container type, you can check in the Elastic Beanstalk console. For instructions, see To check if you are using a legacy container type (p. 204).

Why are some container types marked legacy?

Some container types are marked **(legacy)** and do not support new Elastic Beanstalk functionality. Non-legacy containers are new container types that enable you to get access to new functionality such as attaching Amazon RDS DB Instances to your environment and using configuration files.

Currently, AWS Elastic Beanstalk supports the following non-legacy container types:

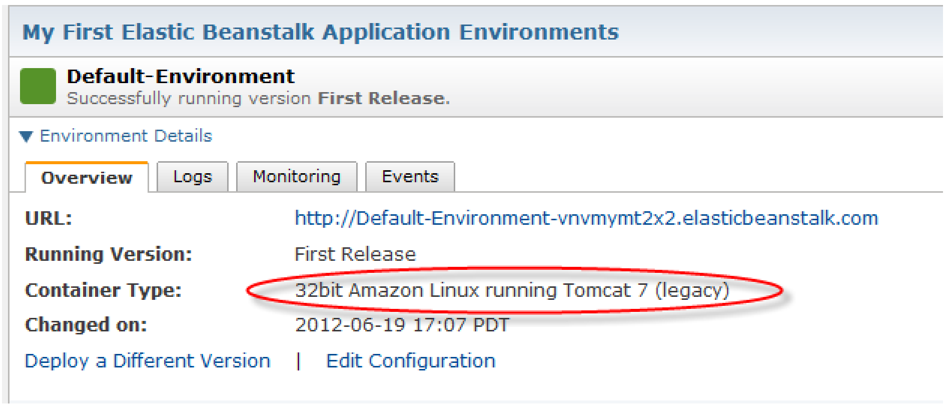
- Tomcat 6 and 7
- Python 2.6

AWS Elastic Beanstalk does not support non-legacy container types for PHP and .NET at this time. If you are using a Tomcat 6 or 7 container type, then check the Elastic Beanstalk console to see if it is a legacy container type.

To check if you are using a legacy container type

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Click the **Environment Details** link in the **Environments** pane for your application.

   The details appear for your application's environment.
3. Your application is using a legacy container type if you see **(legacy)** next to the container type as shown in the following figure.


My First Elastic Beanstalk Application Environments
Default-Environment
Successfully running version First Release.
Environment Details
Overview
Logs
Monitoring
Events

URL: http://Default-Environment-vnvmymt2x2.elasticbeanstalk.com
Running Version: First Release
Container Type: 32bit Amazon Linux running Tomcat 7 (legacy)
Changed on: 2012-06-19 17:07 PDT

Deploy a Different Version | Edit Configuration

To migrate your application

1. Deploy your application to a new environment. For instructions, go to Launching New Environments (p. 135).
2. If you have an Amazon RDS DB Instance, update your database security group to allow access to your EC2 security group for your new environment. For instructions on how to find the name of your EC2 security group using the AWS Management Console, see Amazon EC2 Security Groups (p. 161). For more information about configuring your EC2 security group, go to the "Authorizing Network Access to an Amazon EC2 Security Group" section of Working with DB Security Groups in the *Amazon Relational Database Service User Guide*.
3. Swap your environment URL. For instructions, go to Deploying Versions with Zero Downtime (p. 143).
4. Terminate your old environment. For instructions, go to Terminating an Environment (p. 200).

Customizing and Configuring AWS Elastic Beanstalk Environments

Topics

When deploying your applications, you may want to customize and configure the software that your application depends on. These files could be either dependencies required by the application—for example, additional packages from the yum repository—or they could be configuration files such as a replacement for httpd.conf to override specific settings that are defaulted by AWS Elastic Beanstalk. You can easily customize your environment at the same time that you deploy your application version by including a configuration file with your source bundle. Using a configuration file is more advantageous than customizing your own AMI because you do not need to maintain a set of AMIs. You can use a configuration file for the following container types:

- Tomcat 6 and 7 (non-legacy container types)
- Python 2.6

Currently, AWS Elastic Beanstalk does not support configuration files for PHP and .NET.

This section explains the file format of a configuration file. It also walks you through the process of creating a configuration file and bundling it with your source. Customizing and configuring your container involves two steps:

1. Create a configuration file with the extension **.config** and place it in an **.ebextensions** top-level directory of your source bundle. You can have multiple configuration files in your **.ebextensions** directory.
2. Deploy your application version.

Configuration File Format

When creating the configuration file, you can specify the following:

- **packages** — You can use the packages key to download and install pre-packaged applications and components. Supported package managers include apt, yum, rubygems, python, and rpm.
- **sources** — You can use the sources key to download an archive file and unpack it in a target directory on the EC2 instance. You can reference external locations such as Amazon Simple Storage Service (Amazon S3) or artifacts stored in the source bundle. Supported formats are tar, tar+gzip, tar+bz2 and zip.
- **files** — You can use the files key to create files on the EC2 instance. The content can be either inline in the template or the content can be pulled from a URL. The files are written to disk in lexicographic order. You can reference external locations such as Amazon S3 or artifacts stored in the source bundle.
- **users** — You can use the users key to create Linux/UNIX users on the EC2 instance.
- **groups** — You can use the groups key to create Linux/UNIX groups and to assign group IDs.
- **commands** — You can use the commands key to execute commands on the EC2 instance. The commands are processed in alphabetical order by name, and they run before the application and web server are set up and the application version file is extracted.
- **container_commands** — You can use the container_commands key to execute commands for your container. Container_commands are processed in alphabetical order by name. They run after the application and web server have been set up and the application version file has been extracted, but before the application version is deployed. They also have access to environment variables such as JVM options. Additionally, you can use **leader_only**. One instance is chosen to be the leader in an Auto Scaling group. If the **leader_only** value is set to true, then the command runs only on the instance that is marked as the leader.
- **services** — You can use the services key to define which services should be started or stopped when the instance is launched. The services key also allows you to specify dependencies on sources, packages, and files so that if a restart is needed due to files being installed, AWS Elastic Beanstalk will take care of the service restart.
- **option_settings** You can use the option_settings key to define container settings. The following table displays the namespaces that support option settings. Some namespaces allow you to extend the number of parameters, and specify the parameter names. These get passed in as environment variables on your Amazon EC2 instances.

Container	Namespace	Extend
Python	aws:elasticbeanstalk:container:python:environment	Yes
	aws:elasticbeanstalk:container:python	No
	aws:elasticbeanstalk:container:python:staticfiles	No
Java	aws:elasticbeanstalk:application:environment	Yes
	aws:elasticbeanstalk:container:tomcat:jvmoptions	Yes

The following is an example syntax for a configuration file.

```
packages:
  yum:
    package1: [3.2.1]
    package2: [1.0]
  rpm:
    package3
```

```
sources:
  /home/ec2-user/elasticache: https://s3.amazonaws.com/elasticache-down
loads/AmazonElastiCacheCli-2012-03-09-1.6.000.zip

files:
  “/home/ec2-user/myfile” :
    mode: 000777
    owner: ec2-user
    group: ec2-user
    source: http://foo.bar/myfile

 “/home/ec2-user/myfile2” :
    mode: 000777
    owner: ec2-user
    group: ec2-user
    content: |
      # this is my file
      # with content

users:
  - myuser :
      groups:
       - group1
       - group2
      uid: 50
      homedir: “/tmp”

groups:
  - group1 : 45
  - group2 : 99
  - group3

container_commands:
  replace-config-label:
    command: cp .ebextensions/server.xml /etc/tomcat7/server.xml

commands:
  myscript:
    command: myscript.py
    cwd: /home/ec2-user
    env:
      a: b

services:
  sysvinit:
    - myservice:
        enabled: true
        ensureRunning: true

option_settings:
  - namespace:  aws:elasticbeanstalk:container:tomcat:jvmoptions
    option_name:  Xmx
    value:  256m
```

Accessing Environment Variables

The parameters specified in the **option_settings** section of the configuration file are passed in as environment variables to the EC2 instances. For coding examples, see the following sections:

- Java
- Python

Example: Using Custom Amazon CloudWatch Metrics

Amazon CloudWatch is a web service that enables you to monitor, manage, and publish various metrics, as well as configure alarm actions based on data from metrics. You can define custom metrics for your own use, and AWS Elastic Beanstalk will push those metrics to Amazon CloudWatch. Once Amazon CloudWatch contains your custom metrics, you can view those in the Amazon CloudWatch console.

The Amazon CloudWatch Monitoring Scripts for Linux are available to demonstrate how to produce and consume Amazon CloudWatch custom metrics. The scripts comprise a fully functional example that reports memory, swap, and disk space utilization metrics for an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance. For more information about the Amazon CloudWatch Monitoring Scripts, go to Amazon CloudWatch Monitoring Scripts for Linux in the *Amazon CloudWatch Developer Guide*.

This section walks you through how to deploy a sample application to AWS Elastic Beanstalk and then add custom Amazon CloudWatch metrics using a configuration file.

Step 1: Initialize Your Git Repository

A Git client extension, AWS DevTools, is available as part of the CLI package. It enables you to deploy applications to AWS Elastic Beanstalk quickly. Before you can use the command line interface, you will need to install the following prerequisite software, and then intialize your Git repository.

To install the prerequisite software and initialize your Git repository

1. Install the following software onto your local computer:
   - Git. To download Git, go to http://git-scm.com/. Make sure you have at least version 1.6.6 or later.
   - For Linux, download Ruby version 1.8.7 or later. To view and download Ruby clients, go to http://www.ruby-lang.org/en/.
   - For Windows, download PowerShell 2.0. Windows 7 and Windows Server 2008 R2 come with PowerShell 2.0. For earlier versions of Windows, you can download PowerShell 2.0. Visit http://technet.microsoft.com/en-us/scriptcenter/dd742419.aspx for more details.

2. Initialize your Git repository. In this example, we use the following command from the directory where we installed the command line interface:

```
git init .
```

Step 2: Configure AWS Elastic Beanstalk

Use the `init` command, and AWS Elastic Beanstalk will prompt you to enter this information. If a default value is available, and you want to use it, press **Enter**.

To configure AWS Elastic Beanstalk

1. From your directory where you installed the command line interface, type the following command.

```
eb init
```

2. When you are prompted for the AWS access key, type your access key. To get your access key information, go to Access Credentials.

```
Enter your AWS Access Key ID (current value is "AKIAIOSFODNN7EXAMPLE"):
```

3. When you are prompted for the AWS secret key, type your secret key. To get your secret key information, go to Access Credentials.

```
Enter your AWS Secret Access Key (current value is "wJalrXUtnFEMI/K7MDENG/bPxR
fiCYEXAMPLEKEY"):
```

4. When you are prompted for the AWS Elastic Beanstalk region, type the number of the region. For information about this product's regions, go to Regions and Endpoints in the *Amazon Web Services General Reference*. For this example, we'll use **US East (Northern Virginia)**.
5. When you are prompted for the AWS Elastic Beanstalk application name, type the name of the application. AWS Elastic Beanstalk auto-generates an application name based on the current directory name if an application name has not been previously configured. In this example, we use HelloWorld.

```
Enter an AWS Elastic Beanstalk application name (auto-generated value is
"windows"): sampleapp
```

Note

If you have a space in your application name, make sure you do not use quotes.

6. When you are prompted for the AWS Elastic Beanstalk environment name, type the name of the environment. AWS Elastic Beanstalk automatically creates an environment name based on your application name. If you want to accept the default, press **Enter**.

```
Enter an AWS Elastic Beanstalk environment name (current value is "sampleapp-
env"):
```

Note

If you have a space in your application name, make sure you do not have a space in your environment name.

7. When you are prompted for the solution stack, type the number of the solution stack you want. For this example, we'll use **32bit Amazon Linux running Tomcat 7**.
8. When you are prompted to create an Amazon RDS database, type **Y** or **N**. For this example, we'll type **N**.

```
Create RDS instance? [y/n]: n
```

After configuring AWS Elastic Beanstalk, you are ready to deploy a sample application.

If you want to update your AWS Elastic Beanstalk configuration, you can use the **`init`** command again. When prompted, you can update your configuration options. If you want to keep any previous settings, press the **Enter** key.

Step 3: Create Application

Next, you will create and deploy a sample application. For this step, you use a sample application that is already prepared. AWS Elastic Beanstalk uses the configuration information you specified in the previous step to do the following:

- Creates an application using the application name you specified.
- Launches an environment using the environment name you specified that provisions the AWS resources to host the application.
- Deploys the application into the newly created environment.

Use the `start` command to create and deploy a sample application.

To create an application

- From your directory where you installed the command line interface, type the following command.

```
eb start
```

This process may take several minutes to complete. AWS Elastic Beanstalk will provide status updates during the process. If at any time you want to stop polling for status updates, press **Ctrl+C**. Once the environment status is Green, AWS Elastic Beanstalk will output a URL for the application.

Step 4: View Application

In the previous step, you created an application and deployed it to AWS Elastic Beanstalk. After the environment is ready and its status is Green, AWS Elastic Beanstalk provides a URL to view the application. In this step, you can check the status of the environment to make sure it is set to Green and then copy and paste the URL to view the application.

Use the `status` command to check the environment status, and then use the URL to view the application.

To view an application

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb status
   ```

   AWS Elastic Beanstalk displays the environment status. If the environment is set to Green, then AWS Elastic Beanstalk displays the URL for the application. If you attached an RDS DB Instance to your environment, your RDS DB information is displayed.
2. Copy and paste the URL into your web browser to view your application.

Step 5: Update Application

After you have deployed a sample application, you can update your AWS Elastic Beanstalk environment to push your custom metrics to Amazon CloudWatch. In this step, we create a configuration file, and then use eb to push the new file to AWS Elastic Beanstalk.

To update the AWS Elastic Beanstalk environment with Amazon CloudWatch metrics

1. Create an **.ebextensions** directory.

```
mkdir .ebextensions
```

2. Create a file with the **.config** extension and place it in the **.ebextensions** directory. For more information about the configuration file, see Configuration File Format (p. 206). The following command reports memory utilization, memory used, and memory available every 5 minutes to Amazon CloudWatch.

```
sources: /aws-scripts-mon: http://ec2-downloads.s3.amazonaws.com/cloudwatch-
samples/CloudWatchMonitoringScripts.zip

commands:
  01-setupcron:
    command: echo "*/5 * * * * root perl /aws-scripts-mon/mon-put-instance-
data.pl --mem-util --mem-used --mem-avail --aws-access-key-id AKIAIOSFOD
NN7EXAMPLE --aws-secret-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY >
/dev/null" > /etc/cron.d/cwpump
  02-changeperm:
    command: chmod 644 /etc/cron.d/cwpump
```

3. Add your file to your local Git repository, and then commit your change.

```
git add .
git commit -m "eb configuration"
```

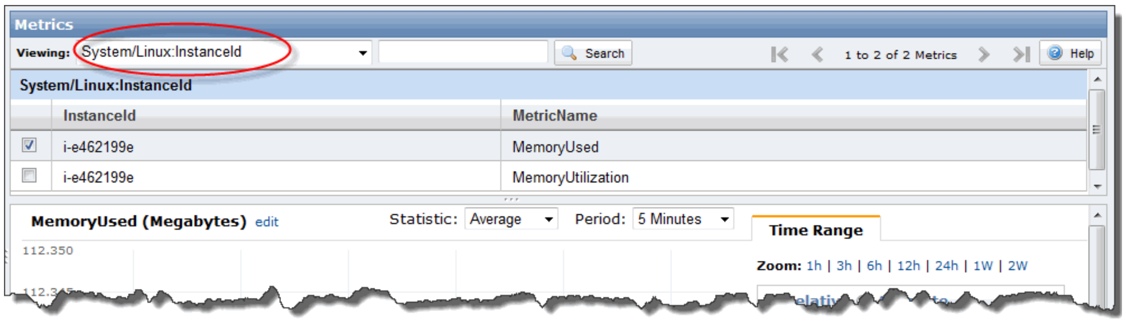
   Note

   For information about Git commands, go to Git - Fast Version Control System.

4. Deploy to AWS Elastic Beanstalk.

```
git aws.push
```

5. Use the `eb status --verbose` command to check your environment status. When your environment is green and ready, check the Amazon CloudWatch console to view your metrics. Custom metrics will have the prefix **System/Linux** in the **Viewing** list. You should something similar see the following.



InstanceId	MetricName
✓ i-e462199e	MemoryUsed
i-e462199e	MemoryUtilization

If you no longer want to run your application, you can clean up by terminating your environment and deleting your application.

Use the `stop` command to terminate your environment and the `delete` command to delete your application.

To delete the application

1. From your directory where you installed the command line interface, type the following command.

```
eb stop
```

   This process may take a few minutes. AWS Elastic Beanstalk will display a message once the environment has been successfully terminated.

   Note

   If you attached an RDS DB Instance to your environment, then your RDS DB will be deleted, and you will lose your data. To save your data, create a snapshot before you delete the application. For instructions on how to create a snapshot, go to Creating a DB Snapshot in the *Amazon Relational Database Service User Guide*.

2. From your directory where you installed the command line interface, type the following command.

```
eb delete
```

   AWS Elastic Beanstalk will display a message once it has successfully deleted the application.

Troubleshooting

Topics

This section provides a table of the most common AWS Elastic Beanstalk issues and how to resolve or work around them.

Issue	Workaround
Unable to connect to Amazon RDS from AWS Elastic Beanstalk.	To connect RDS to your AWS Elastic Beanstalk application, do the following: • Make sure RDS is in the same Region as your AWS Elastic Beanstalk application. • Make sure the RDS security group for your instance has an authorization for the Amazon EC2 security group you are using for your AWS Elastic Beanstalk environment. For instructions on how to find the name of your EC2 security group using the AWS Management Console, see Amazon EC2 Security Groups (p. 161). For more information about configuring your EC2 security group, go to the "Authorizing Network Access to an Amazon EC2 Security Group" section of Working with DB Security Groups in the *Amazon Relational Database Service User Guide*. • For Java, make sure the MySQL JAR file is in your WEB-INF/lib. See Using Amazon RDS and MySQL Connector/J (p. 33) for more details.
Experiencing a couple of seconds of downtime when updating the version of my application.	Because AWS Elastic Beanstalk uses a drop-in upgrade process, there might be a few seconds of downtime. There is no workaround at this time.
Unable to connect to another Amazon EC2 instance using the Amazon EC2 security group for your AWS Elastic Beanstalk environment.	Create a CNAME record and point this to the public DNS of the Amazon EC2 instance.

Issue	Workaround
How can I change my application URL from myapp.elasticbeanstalk.com to www.myapp.com?	Register in a DNS server a CNAME record such as: www.mydomain.com CNAME mydomain.elasticbeanstalk.com.
Unable to specify a specific Availability Zone for my AWS Elastic Beanstalk application.	This feature is not currently supported in the AWS Management Console. However you can pick specific AZs using the APIs, CLI, Eclipse plug-in, or Visual Studio plug-in.
Getting charged for my AWS Elastic Beanstalk application.	The default settings for AWS Elastic Beanstalk do not incur any additional charges. However, if you modified the default settings by changing the Amazon EC2 instance type or adding additional Amazon EC2 instance, charges may be accrued. For information about the free tier, see http://aws.amazon.com/free. If you have questions about your account, contact our customer service team directly.
How can I receive notifications by SMS?	If you specify an SMS email address, such as one constructed on http://www.makeuseof.com/tag/email-to-sms, you will receive the notifications by SMS. To subscribe to more than one email address, you can use the AWS Elastic Beanstalk command line to register an SNS topic with an environment.
How do I upgrade my instance type to an m1.large?	You can upgrade your instance type to an m1.large by launching a new environment using a 64-bit container. Once your environment is launched, you can select **Edit Configuration** to select m1.large. See Launching New Environments (p. 135) for instructions on launching a new environment. See Configuring EC2 Server Instances with AWS Elastic Beanstalk (p. 160) for instructions for editing your configuration.
Unable to connect to AWS Elastic Beanstalk when deploying using the AWS Toolkit for Eclipse.	Try one of the following: • Make sure you are running the latest distribution of Eclipse. • Make sure you've signed up your account for AWS Elastic Beanstalk (and have received an email confirmation). • Check the "Error Log" view in Eclipse to see if there's any additional information or stack traces.
How can I get Amazon EBS to work with AWS Elastic Beanstalk?	The default AMIs are EBS-backed; however, the root volume is deleted upon termination of the instance. You can alter the delete on termination behavior by using: `$ ec2-modify-instance-attribute -b '/dev/sdc=<vol-id>:false` as described in the EC2 Command Line Reference.
Servers that were created in the AWS Management Console do not appear in the Toolkit for Eclipse	You can manually import servers by following the instructions at Importing Existing Environments into Eclipse (p. 30).

Understanding Environment Launch Events

Topics

AWS Elastic Beanstalk monitors the environment launch process to ensure your application is healthy. To determine your application's health, AWS Elastic Beanstalkk sends periodic requests to the application's health check URL (by default root or '/'). If your application experiences problems and does not respond to the health check, a variety of launch events are published. This section describes the most common launch events, why they happen, and how to address environment launch-related problems. For instructions on how to view your events, see Viewing Events (p. 155). For more information about health check URL, see Health Checks (p. 167).

Failed to Perform HTTP HEAD Request to http://<yourapp>.elasticbeanstalk.com:80

AWS Elastic Beanstalk sends periodic HTTP HEAD requests to the health check URL. This event fires when the health check URL does not respond successfully (with HTTP code 200).

If you receive this event, try one or both of the following:

- Make sure that your application's health check URL exists. For example, if AWS Elastic Beanstalk makes a health check request to http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.jsp, ensure that /myapp/index.jsp exists and is accessible. Similarly, for PHP, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.php, make sure that /myapp/index.php exists and is accessbile. For ASP.NET, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/default.aspx, make sure that /myapp/default.aspx exists and is accessbile.
- Inspect previous events in the **Events** tab in the AWS Management Console to ensure that your environment is healthy. For example, if instances of your environments are running at close to 100 percent CPU utilization, they may become unresponsive. AWS Elastic Beanstalk will alert you via an event that reads **Instance <instance id> is experiencing CPU Utilization greater than 95.00%. Consider adjusting auto-scaling settings or upgrading to an instance type larger than a <instance type>.** See CPU Utilization Greater Than 95.00% (p. 216) for more information about this event.

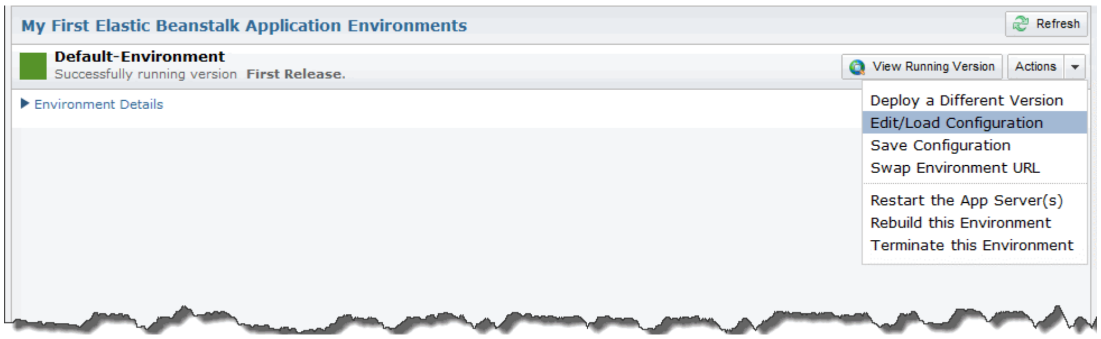
CPU Utilization Greater Than 95.00%

If you receive an event that reads **Instance <instance id> is experiencing CPU Utilization greater than 95.00%. Consider adjusting auto-scaling settings or upgrading to an instance type larger**

than a <instance type>., this means an instance is using the CPU for more than 95 percent of the time. If your application can be parallelized across multiple instances, you can adjust auto-scaling settings.

To adjust auto-scaling settings

1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Select your application from the list.
3. Below the application selection list, click the **Actions** list for your application environment in the application's **Environments** list and select **Edit Configuration**.


My First Elastic Beanstalk Application Environments
Refresh
Default-Environment
Successfully running version First Release.
View Running Version
Actions
Environment Details
Deploy a Different Version
Edit/Load Configuration
Save Configuration
Swap Environment URL
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment

   The **Edit Configuration** window is displayed.
4. Click the **Auto Scaling** tab. For instructions on how to adjust auto-scaling to increase your maximum number of instances and adjust your scaling trigger, see Configuring Auto Scaling with AWS Elastic Beanstalk (p. 170).

   If your application requires higher computing needs, you can upgrade the Amazon EC2 instance type for your environment. For instructions on how to upgrade your Amazon EC2 instance type, see Configuring EC2 Server Instances with AWS Elastic Beanstalk (p. 160).

Elastic Load Balancer awseb-<yourapp> Has Zero Healthy Instances

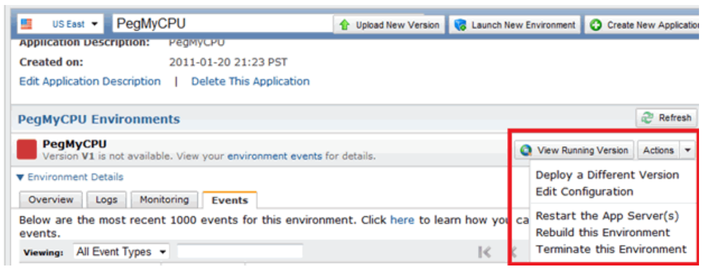
When the health status of all of your instances changes from green to red, AWS Elastic Beanstalk alerts you that your application has become unavailable. Make sure that your application’s health check URL exists. For example, if AWS Elastic Beanstalk makes a health check request to http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.jsp, ensure that /myapp/index.jsp exists and is accessible. Similarly, for PHP, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.php, make sure that /myapp/index.php exists and is accessbile. For ASP.NET, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/default.aspx, make sure that /myapp/default.aspx exists and is accessbile.

Elastic Load Balancer awseb-<yourapp> Cannot Be Found

If you receive the event **Elastic Load Balancer awseb-yourapp cannot be found. If this problem persists, try rebuilding your environment**., then it is because your Elastic Load Balancer for your environment has been removed. To resolve this issue, you need to rebuild your environment.

To rebuild your environment

1. In the AWS Management Console, go to the **Environments** panel and click the **Actions** drop-down.


US East
PegMyCPU
Upload New Version
Launch New Environment
Create New Application
Application Description: PegMyCPU
Created on: 2011-01-20 21:23 PST
Edit Application Description | Delete This Application
PegMyCPU Environments
Refresh
PegMyCPU
Version V1 is not available. View your environment events for details.
View Running Version
Actions
Deploy a Different Version
Edit Configuration
Restart the App Server(s)
Rebuild this Environment
Terminate this Environment
Environment Details
Overview
Logs
Monitoring
Events
Below are the most recent 1000 events for this environment. Click here to learn how you ca
events.
Viewing: All Event Types

2. Select **Rebuild this Environment**.

Failed to Retrieve Status of Instance <instance id> 4 Consecutive Time(s)

This event occurs when AWS Elastic Beanstalk is not getting a response from the health check URL. AWS Elastic Beanstalk will try to reach the health check URL 10 times over 10 minutes. If this event occurs, try one or both of the following:

- Make sure that your application’s health check URL exists. For example, if AWS Elastic Beanstalk makes a health check request to http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.jsp, ensure that /myapp/index.jsp exists and is accessible. Similarly, for PHP, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.php, make sure that /myapp/index.php exists and is accessbile. For ASP.NET, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/default.aspx, make sure that /myapp/default.aspx exists and is accessbile.
- Inspect previous events in the **Events** tab on the AWS Management Console to ensure that your environment is healthy. For example, if instances of your environments are running at close to 100 percent CPU utilization, they may become unresponsive. AWS Elastic Beanstalk will alert you via an event that reads **Instance <instance id> is experiencing CPU Utilization greater than 95.00%. Consider adjusting auto-scaling settings or upgrading to an instance type larger than a <instance type>**. For more information, see CPU Utilization Greater Than 95.00% (p. 216).

Failed to Launch Environment

This event occurs when AWS Elastic Beanstalk attempts to launch an environment and encounters failures along the way. Previous events in the **Events** tab will alert you to the root cause of this issue.

EC2 Instance Launch Failure. Waiting for a New EC2 Instance to Launch...

This event occurs when an Amazon EC2 instance fails to launch. If this event occurs, try one or both of the following:

- Check the service health dashboard to ensure that the Elastic Compute Cloud (Amazon EC2) service is green.
- Make sure that you are not over the Amazon EC2 instance limit for your account (the default is 10 instances). To request an increase to the Amazon EC2 instances, complete the form available at http://aws.amazon.com/contact-us/ec2-request/.

Exceeded Maximum Amount of Time to Wait for the Application to Become Available. Setting Environment Ready

During the launch of a new environment, AWS Elastic Beanstalk monitors the environment to make sure that the application is available. If the application is still unavailable after 6 minutes have passed, the environment enters the ready state; this allows you to take action and make configuration changes. If this event occurs, try one or both of the following:

- Make sure that your application's health check URL exists. For example, if AWS Elastic Beanstalk makes a health check request to http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.jsp, ensure that /myapp/index.jsp exists and is accessible. Similarly, for PHP, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.php, make sure that /myapp/index.php exists and is accessbile. For ASP.NET, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/default.aspx, make sure that /myapp/default.aspx exists and is accessbile.
- Make sure you have uploaded a valid .war file or a .zip file.

Launched Environment: <environment id>. However, There Were Issues During Launch. See Event Log for Details

During the launch of a new environment, AWS Elastic Beanstalk monitors the environment to make sure that the application is available. If the application is still unavailable after 6 minutes have passed, the environment enters the ready state; this allows you to take action and make configuration changes. If this event occurs, try one or both of the following:

- Make sure that your application's health check URL exists. For example, if AWS Elastic Beanstalk makes a health check request to http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.jsp, ensure that /myapp/index.jsp exists and is accessible. Similarly, for PHP, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/index.php, make sure that /myapp/index.php exists and is accessbile. For ASP.NET, if you have http://healthcheckrocks.elasticbeanstalk.com:80/myapp/default.aspx, make sure that /myapp/default.aspx exists and is accessbile.
- Make sure you have uploaded a valid .war file or .zip file.
- Check previous events in the **Events** tab.

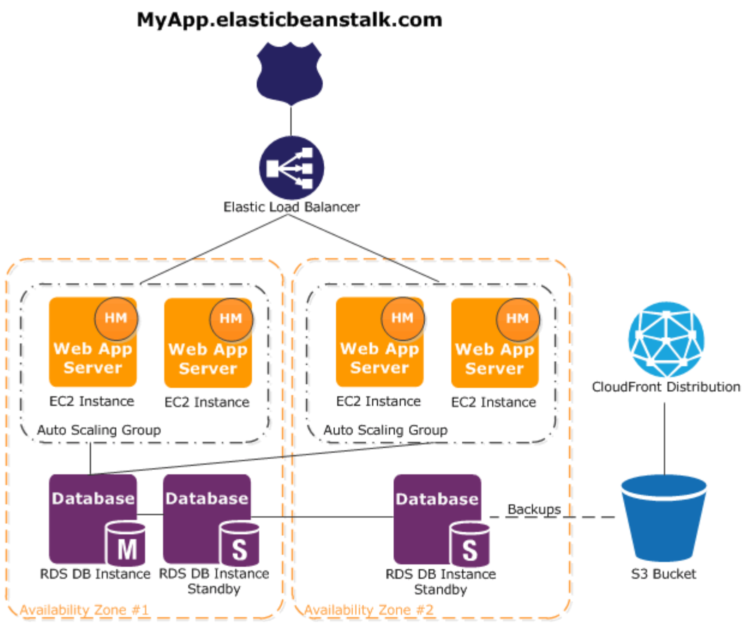
Using AWS Elastic Beanstalk with Other AWS Services

Topics

This topic discusses the integration of AWS Elastic Beanstalk with other AWS services.

Architectural Overview

The following diagram illustrates an example architecture of AWS Elastic Beanstalk across multiple Availability Zones working with other AWS products such as Amazon CloudFront, Amazon Simple Storage Service (Amazon S3), and Amazon Relational Database Service (Amazon RDS). For a more detailed discussion about Amazon Route 53, Elastic Load Balancing, Amazon Elastic Compute Cloud (Amazon EC2) and host manager (HM), see Architectural Overview (p. 18).



To plan for fault-tolerance, it is advisable to have N+1 Amazon EC2 instances and spread your instances across multiple Availability Zones. In the unlikely case that one Availability Zone goes down, you will still have your other Amazon EC2 instances running in another Availability Zone. You can adjust Auto Scaling to allow for a minimum number of instances as well as multiple Availability Zones. For instructions on how to do this, see Launch Configuration (p. 171). Fore more information about building fault-tolerant applications, go to Building Fault-Tolerant Applications on AWS.

The following sections discuss in more detail integration with Amazon Route 53, Amazon RDS, Amazon SimpleDB, Amazon ElastiCache, Amazon S3, Amazon CloudFront, and Amazon CloudWatch.

Using AWS Elastic Beanstalk with Amazon Route 53 to Map Your Root Domain to Your Load Balancer

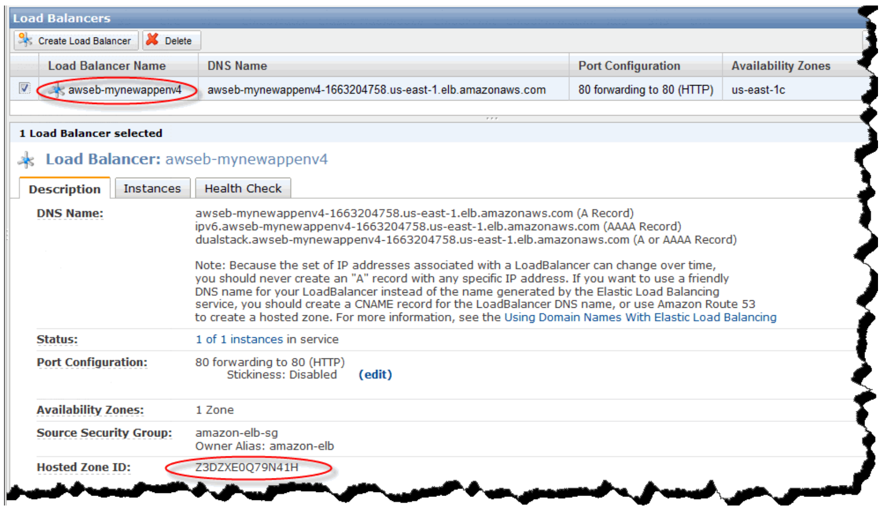
Due to the way the DNS protocol works, there is no way to refer your Elastic Load Balancer from the root (also known as the apex) of the domain. For instance, you can create a DNS entry that maps http://www.example.com to an Elastic Load Balancer, but you cannot do the same for http://example.com. Amazon Route 53 enables you to map the apex of a hosted zone to your Elastic Load Balancer using an Alias record. When Amazon Route 53 encounters an Alias record, it looks up the A records associated with the target DNS name in the Alias, and returns the IP addresses from that name. The following procedure steps you through how to use Amazon Route 53 to map your root domain to your Elastic Load Balancer.

To map your root domain to your Elastic Load Balancer

1. Follow the Amazon Route 53 Getting Started Guide instructions to sign up for Route 53, create a hosted zone, and then update your name server records with your registrar.
2. Get the value of the hosted zone ID for your Elastic Load Balancer.
    a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
    b. From the region list, select a region.

c. In the **Navigation** pane, click **Load Balancers** .

d. Select the Load Balancer associated with your AWS Elastic Beanstalk application. The load balancer will be in the format **awseb-<*`your app environment name`*>**. Your hosted ID will appear in the **Load Balancer** details pane on the **Description** tab. Make a note of your hosted ID.



Load Balancers

Create Load Balancer | Delete

	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
✓	awseb-mynewappenv4	awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com	80 forwarding to 80 (HTTP)	us-east-1c

1 Load Balancer selected

Load Balancer: awseb-mynewappenv4

Description | Instances | Health Check

DNS Name:	awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com (A Record) ipv6.awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com (AAAA Record) dualstack.awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com (A or AAAA Record) Note: Because the set of IP addresses associated with a LoadBalancer can change over time, you should never create an "A" record with any specific IP address. If you want to use a friendly DNS name for your LoadBalancer instead of the name generated by the Elastic Load Balancing service, you should create a CNAME record for the LoadBalancer DNS name, or use Amazon Route 53 to create a hosted zone. For more information, see the Using Domain Names With Elastic Load Balancing
Status:	1 of 1 instances in service
Port Configuration:	80 forwarding to 80 (HTTP) Stickiness: Disabled (edit)
Availability Zones:	1 Zone
Source Security Group:	amazon-elb-sg Owner Alias: amazon-elb
Hosted Zone ID:	Z3DZXE0Q79N41H

3. Create an alias resource record set in your hosted zone. For instructions, go to How to Create an Alias Resource Record Set in the *Amazon Route 53 Developer Guide*.

4. Check on the status of your changes. For instructions, see Creating, Changing, and Deleting Resource Record Sets in the *Amazon Route 53 Developer Guide*.

Your root domain is now mapped to your AWS Elastic Beanstalk Load Balancer.

If you launch a new environment, and you want your existing Amazon Route 53 Alias to point to the new Elastic Load Balancer, you will need to map your root domain to the Elastic Load Balancer in your new environment.

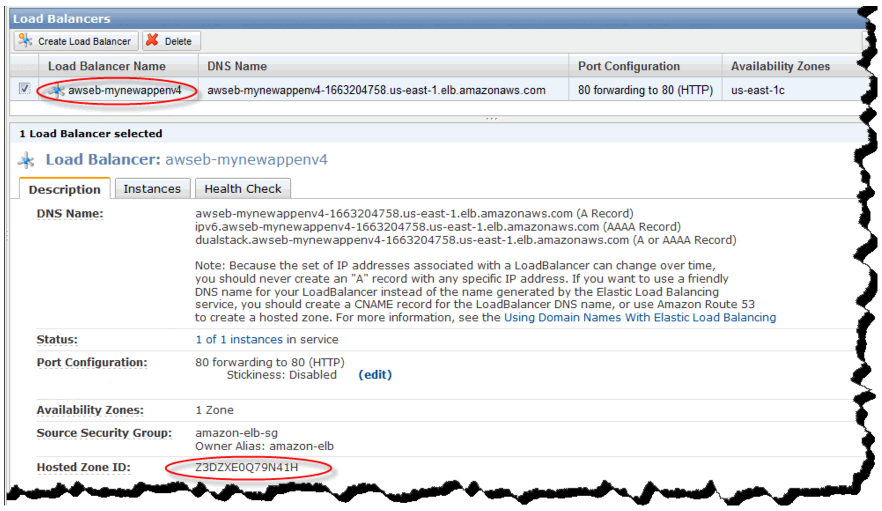
To map your root domain to your new Elastic Load Balancer

1. Get the value of the hosted zone ID for your old Elastic Load Balancer.

   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   b. In the **Navigation** pane, click **Load Balancers**.

   c. Select the load balancer associated with your AWS Elastic Beanstalk application. The load balancer will be in the format **awseb-<your app environment name>**. Your hosted ID will appear in the **Load Balancer** details pane on the **Description** tab. Make a note of your hosted ID.


Load Balancers

Create Load Balancer | Delete

	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
✓	awseb-mynewappenv4	awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com	80 forwarding to 80 (HTTP)	us-east-1c

1 Load Balancer selected

Load Balancer: awseb-mynewappenv4

Description | Instances | Health Check

DNS Name:	awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com (A Record) ipv6.awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com (AAAA Record) dualstack.awseb-mynewappenv4-1663204758.us-east-1.elb.amazonaws.com (A or AAAA Record) Note: Because the set of IP addresses associated with a LoadBalancer can change over time, you should never create an "A" record with any specific IP address. If you want to use a friendly DNS name for your LoadBalancer instead of the name generated by the Elastic Load Balancing service, you should create a CNAME record for the LoadBalancer DNS name, or use Amazon Route 53 to create a hosted zone. For more information, see the Using Domain Names With Elastic Load Balancing
Status:	1 of 1 instances in service
Port Configuration:	80 forwarding to 80 (HTTP) Stickiness: Disabled (edit)
Availability Zones:	1 Zone
Source Security Group:	amazon-elb-sg Owner Alias: amazon-elb
Hosted Zone ID:	Z3DZXE0Q79N41H

2. Change an alias resource record set in your hosted zone. For instructions, go to Creating, Changing, and Deleting Resource Record Sets in the *Amazon Route 53 Developer Guide*.
3. Check on the status of your changes. For instructions, go to Creating, Changing, and Deleting Resource Record Sets in the *Amazon Route 53 Developer Guide*.
4. After you have confirmed that the changes completed successfully, you can terminate your old environment. For instructions on how to terminate an environment, see Terminating an Environment (p. 200).

Using AWS Elastic Beanstalk with Amazon RDS

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you to focus on your applications and business.

If you plan to use Amazon RDS, it is advisable to configure Amazon RDS across multiple Availability Zones. This enables a synchronous standby replica of your database to be provisioned in a different Availability Zone. To keep both databases in sync, updates to your database instance are synchronously replicated across Availability Zones. In case of a failover scenario, the standby is promoted to be the primary and will handle the database operations. Running your database instance in multiple Availability Zones safeguards your data in the unlikely event of a database instance component failure or service health disruption in one Availability Zone.

The instructions for configuring your AWS Elastic Beanstalk application with Amazon RDS depend on the programming language you use.

- Java — Using Amazon RDS and MySQL Connector/J (p. 33)
- PHP — Using Amazon RDS with PHP (p. 100)
- Python — Deploying AWS Elastic Beanstalk Applications in Python Using Eb and Git (p. 103)
- .NET (SQL Server) — Get Started (p. 52)
- .NET (MySQL Server) — Amazon RDS for C# Developers

Using AWS Elastic Beanstalk with Amazon SimpleDB

Amazon SimpleDB is a fault-tolerant and durable structured data storage solution. This service works in close conjunction with Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2), collectively providing the ability to store, process, and query data sets in the cloud. These services are designed to make web-scale computing easier and more cost-effective for developers. In many scenarios, Amazon SimpleDB can be used to augment or even replace your use of traditional relational databases. Amazon SimpleDB is highly available for your use, just like Amazon S3 and the other services. By using Amazon SimpleDB, you can take advantage of a scalable service that has been designed for high availability and fault tolerance. Data stored in Amazon SimpleDB is stored redundantly without single points of failure. For more information about Amazon SimpleDB, go to the Amazon SimpleDB Getting Started Guide.

Using AWS Elastic Beanstalk with Amazon ElastiCache

Amazon ElastiCache is a web service that makes it easy to set up, manage, and scale distributed in-memory cache environments in the cloud. It provides a high performance, resizeable, and cost-effective in-memory cache, while removing the complexity associated with deploying and managing a distributed cache environment. Amazon ElastiCache is protocol-compliant with Memcached, so the code, applications, and most popular tools that you use today with your existing Memcached environments will work seamlessly with the service. For more information about Amazon ElastiCache, go to the Amazon ElastiCache product page.

To use AWS Elastic Beanstalk with Amazon ElastiCache

1. Create an ElastiCache cluster. For instructions on how to create an ElastiCache cluster, go to Create a Cache Cluster in the *Amazon ElastiCache Getting Started Guide*.
2. Configure your Amazon ElastiCache Security Group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application. For instructions on how to find the name of your EC2 security group using the AWS Management Console, see Amazon EC2 Security Groups (p. 161). For more information, go to Authorize Access in the *Amazon ElastiCache Getting Started Guide*.

Using AWS Elastic Beanstalk with Amazon S3

Amazon S3 is a simple web service that provides highly durable, fault-tolerant data storage. Behind the scenes, Amazon S3 stores objects redundantly on multiple devices across multiple facilities in an Amazon S3 Region. In the unlikely event of a failure in an Amazon Web Service data center, you will still have access to your data. AWS Elastic Beanstalk automatically signs you up for Amazon S3 when you sign up for AWS Elastic Beanstalk. When you create your application and deploy it to AWS Elastic Beanstalk your application will be automatically uploaded to an Amazon S3 bucket. To learn more about Amazon S3, go to the Amazon Simple Storage Service (Amazon S3) product page.

Using AWS Elastic Beanstalk with Amazon CloudFront

Amazon CloudFront distributes your web content (such as images, video, and so on) using a network of edge locations around the world. End users are routed to the nearest edge location, so content is delivered with the best possible performance. CloudFront works seamlessly with Amazon S3. After you create and deploy your AWS Elastic Beanstalk you can sign up for Amazon CloudFront and start using Amazon CloudFront to distribute your content. Create your distribution from a custom origin, and use an AWS Elastic Beanstalk domain name. To get started using Amazon CloudFront, go to the Amazon CloudFront Developer Guide.

Using AWS Elastic Beanstalk with Amazon CloudWatch

Amazon CloudWatch enables you to monitor, manage, and publish various metrics, as well as configure alarm actions based on data from metrics. Amazon CloudWatch monitoring enables you to collect, analyze, and view system and application metrics so that you can make operational and business decisions more quickly and with greater confidence. You can use Amazon CloudWatch to collect metrics about your Amazon Web Services (AWS) resources—such as the performance of your Amazon EC2 instances. You can also publish your own metrics directly to Amazon CloudWatch. Amazon CloudWatch alarms help you implement decisions more easily by enabling you to send notifications or automatically make changes to the resources you are monitoring, based on rules that you define. For example, you can create alarms that initiate Auto Scaling and Amazon Simple Notification Service (Amazon SNS) actions on your behalf. AWS Elastic Beanstalk automatically uses Amazon CloudWatch to help you monitor your application and environment status. You can navigate to the Amazon CloudWatch console to see your dashboard and get an overview of all of your resources as well as your alarms. You can also choose to view more metrics or add custom metrics. For more information about Amazon CloudWatch, go to the Amazon CloudWatch Developer Guide.

Using AWS Elastic Beanstalk with AWS Identity and Access Management (IAM)

You can use AWS Identity and Access Management (IAM) to control the ability of users to perform AWS Elastic Beanstalk API actions. IAM is automatically available with AWS Elastic Beanstalk. You do not need to sign up separately to use IAM.

Any time you need to grant permissions to a user or group, you can write a policy that contains the permissions and then attach the policy to the user or group. A policy can grant or deny the ability to perform actions on AWS Elastic Beanstalk resources.

There are two ways to control access to AWS Elastic Beanstalk resources:

- Use a policy template. (p. 226) The AWS Elastic Beanstalk policy templates enable you to easily give users and groups full or read-only access to all AWS Elastic Beanstalk resources.
- Create a policy that allows or denies permissions to perform specific actions on specific resources. (p. 227) A custom policy gives you the flexibility to specify exactly what actions can be performed on what resources.

Using Policy Templates to Control Access to All AWS Elastic Beanstalk Resources

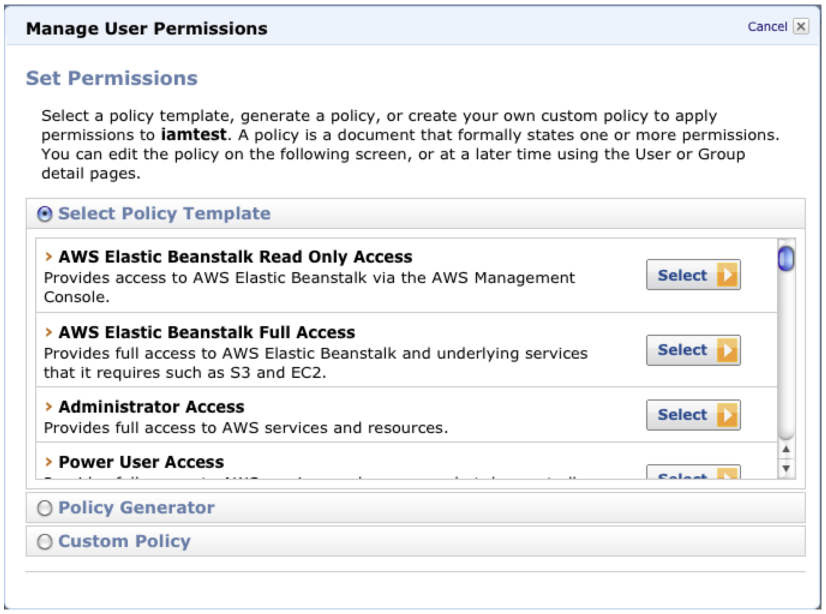
AWS Elastic Beanstalk provides two policy templates that enable you to assign full access or read-only access to all AWS Elastic Beanstalk resources. You can attach the policy templates to users or groups. You should use these templates if you want to grant broad permissions for all AWS Elastic Beanstalk in your AWS account. If you want to control permissions for specific resources, you need to create the policy.

The following table describes each policy template.

Template	Description
AWS Elastic Beanstalk Full Access	This template allows the user to create, modify, and delete applications, application versions, configuration settings, environments, and their underlying resources, including access required by AWS Elastic Beanstalk to provision and manage underlying resources (Elastic Load Balancing, Auto Scaling, Amazon EC2, Amazon SNS, Amazon CloudWatch, Amazon S3, and AWS CloudFormation (for non-legacy container types)) used by an environment. For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).
AWS Elastic Beanstalk Read Only Access	This template allows the user to view applications and environments but not to perform any operations on them. It provides read-only access to all applications, application versions, events, and environments.

To apply a policy template to a user or group

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the **Navigation** pane, click **Users** or **Groups**, as appropriate.
3. In the **Users** or **Groups** pane, click the user or group that you want to apply the policy template to.
4. Under the user or group name, click the **Permissions** tab.
5. Click **Attach User Policy** for a user or **Attach Policy** for a group, as appropriate.
6. On the **Manage Policy** page, click **Select Policy Template**.
7. Locate the policy that you want to assign, and then click the corresponding **Select** button.



Manage User Permissions

Set Permissions

Select a policy template, generate a policy, or create your own custom policy to apply permissions to **iamtest**. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the User or Group detail pages.

◉ Select Policy Template

AWS Elastic Beanstalk Read Only Access
Provides access to AWS Elastic Beanstalk via the AWS Management Console.

AWS Elastic Beanstalk Full Access
Provides full access to AWS Elastic Beanstalk and underlying services that it requires such as S3 and EC2.

Administrator Access
Provides full access to AWS services and resources.

Power User Access

8. Review the policy document, and then click **Apply Policy**.

Creating Policies to Control Access to Specific AWS Elastic Beanstalk Resources

You can create your own IAM policy to allow or deny specific AWS Elastic Beanstalk API actions on specific AWS Elastic Beanstalk resources. To put the policy into effect, you attach it to a user or group using the IAM console, command line tools, or API. For more information about attaching a policy to a user or group, see Managing IAM Policies in *Using AWS Identity and Access Management*.

An IAM policy contains policy statements that describe the specific permissions you want to grant. When you create a policy statement for AWS Elastic Beanstalk, there are four parts of a statement that you need to know how to use:

- **Effect** specifies whether to allow or deny the actions in the statement.
- **Action** specifies the actions you want to control. To specify AWS Elastic Beanstalk actions, the action name must be prefixed with the lowercase string `elasticbeanstalk`. You use wildcards to specify all actions related to AWS Elastic Beanstalk. The wildcard `"*"` matches zero or multiple characters. For example, to grant all create action permissions, you can specify `elasticbeanstalk:create*` in your IAM policy.

  Note

  If your policy uses a wildcard to specify all actions instead of explicitly listing each action, be aware that if an update to AWS Elastic Beanstalk were to add any new actions, this policy would automatically give the grantee access to those new actions.

  For a complete list of AWS Elastic Beanstalk actions, see the API action names in the AWS Elastic Beanstalk API Reference. For more information about permissions and policies, go to Permissions and Policies in *Using AWS Identity and Access Management*.

  Users with permission to use specific AWS Elastic Beanstalk API actions can perform those actions. Certain operations, such as creating an environment, may require additional permissions to perform those actions. To check if an API action depends on permissions to other actions and to ensure all required permissions are assigned, use the information in section Resources and Conditions for AWS Elastic Beanstalk Actions (p. 231).

- **Resource** specifies the resources that you want to control access to. To specify AWS Elastic Beanstalk resources, you list the Amazon Resource Name (ARN) of each resource. For more information, see Amazon Resource Name (ARN) Format for AWS Elastic Beanstalk (p. 230). Each AWS Elastic Beanstalk action operates on a specific resource. For example, the `UpdateApplicationVersion` action operates on application versions, which you would specify as one or more version resources. For more information, see Amazon Resource Name (ARN) Format for AWS Elastic Beanstalk (p. 230). To specify multiple ARNs, you can list each resource's ARN or use the `"*"` wildcard, which matches zero or multiple characters.
- **Condition** specifies restrictions on the permission granted in the statement. As discussed earlier, an action operates on a specific resource. However, that action may have dependencies on other AWS Elastic Beanstalk resources such as where the action occurs (for example, creating an environment within an application) or which other resources the action needs access to in order to complete its operation (for example, updating an environment from a configuration template or application version). For more information, see Resources and Conditions for AWS Elastic Beanstalk Actions (p. 231).

IAM policies are expressed in JSON format. For information about the structure of IAM policies and statements, see Basic Policy Structure in *Using AWS Identity and Access Management*. The following example policy contains three sets of statements that enable a user who has this policy to call the `CreateEnvironment` action to create an environment whose name begins with **`Test`** in the application **`My First Elastic Beanstalk Application`** using the application version **`First Release`**. The policy also allows the user to perform actions on the resources required to create the environment. The `CreateEnvironmentPerm` statement allows the `elasticbeanstalk:CreateEnvironment` action to create an environment with the constraints specified above. The `AllNonResourceCalls` statement allows `elasticbeanstalk:CreateEnvironment` to perform the AWS Elastic Beanstalk actions required to create the environment. The `OtherServicePerms` statement allows `elasticbeanstalk:CreateEnvironment` to call the appropriate actions to create resources in other AWS services to complete the creation of the environment.

Note

The following policy is an example. It gives a broad set of permissions to the AWS products that AWS Elastic Beanstalk uses to manage applications and environments. For example, `ec2:*` allows an IAM user to perform any action on any Amazon EC2 resource in the AWS account. These permissions are not limited to the resources that you use with AWS Elastic Beanstalk. As a best practice, you should grant individuals only the permissions they need to perform their duties.

```
{
  "Statement": [
    {
      "Sid":"CreateEnvironmentPerm",
      "Action": [
        "elasticbeanstalk:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My First
Elastic Beanstalk Application/Test*"
      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My First Elastic Beanstalk Application"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbean
stalk:us-east-1:123456789012:applicationversion/My First Elastic Beanstalk Ap
plication/First Release"]
        }
       }
      },
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability",
            "elasticbeanstalk:CreateStorageLocation"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      },
      {
      "Sid":"OtherServicePerms",
         "Effect":"Allow",
         "Action":[
            "ec2:*",
            "elasticloadbalancing:*",
            "autoscaling:*",
            "cloudwatch:*",
            "s3:*",
            "sns:*",
            "cloudformation:*"
         ],
         "Resource":"*"
      }
   ]
}
```

Note that the policy above enables the user to create an environment using the AWS Elastic Beanstalk CreateEnvironment API and the elastic-beanstalk-create-environment (p. 294) command. However, if you want that user to be able to use the AWS Elastic Beanstalk console to create an environment, you must also add the following policy to the user. When the user creates an environment in the AWS Elastic Beanstalk console, the user must be able to navigate to the application **`My First Elastic Beanstalk Application`** (`elasticbeanstalk:DescribeApplications`). When the user clicks **Launch New Environment**, the AWS Elastic Beanstalk console needs to get information about existing environments (`elasticbeanstalk:DescribeEnvironments`), the application version to use (`elasticbeanstalk:DescribeApplicationVersions`), and solution stacks (`elasticbeanstalk:ListAvailableSolutionStacks` and `elasticbeanstalk:DescribeConfigurationOptions`). If you want to enable specific actions within the AWS Elastic Beanstalk console, you need to consider the types of dependencies described in this example.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:DescribeApplications"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:155363561088:application/My First
Elastic Beanstalk Application"
      ]
    },
    {
      "Action": "elasticbeanstalk:DescribeEnvironments",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My First
Elastic Beanstalk Application/Test*"
      ]
    },
    {
      "Action": "elasticbeanstalk:DescribeApplicationVersions",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:applicationversion/My
 First Elastic Beanstalk Application/First Release"
      ]
    },
    {
      "Action": [
        "elasticbeanstalk:ListAvailableSolutionStacks"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:elasticbeanstalk:us-east-1::solutionstack/*"
    },
    {
      "Action": "elasticbeanstalk:DescribeConfigurationOptions",
      "Effect": "Allow",
      "Resource": [
         "arn:aws:elasticbeanstalk:us-east-1::solutionstack/*"
      ]
    }
  ]
}
```

Amazon Resource Name (ARN) Format for AWS Elastic Beanstalk

You specify a resource for an IAM policy using that resource's Amazon Resource Name (ARN). For AWS Elastic Beanstalk, the ARN has the following format.

```
arn:aws:elasticbeanstalk:region:accountid:resourcetype/resourcepath
```

Where:

- *region* is the Region the resource resides in (for example, **us-east-1**).
- *accountid* is the AWS account ID with no hyphens (for example, **123456789012**)
- *resourcetype* identifies the type of the AWS Elastic Beanstalk resource (for example, environment). See the table below for a list of all AWS Elastic Beanstalk resource types.
- *resourcepath* is the portion that identifies the specific resource. An AWS Elastic Beanstalk resource has a path that uniquely identifies that resource. See the table below for the format of the resource path for each resource type. For example, an environment is always associated with an application. The resource path for the environment **myEnvironment** in the application **myApp** would look like this:

```
myApp/myEnvironment
```

AWS Elastic Beanstalk has several types of resources you can specify in a policy. The following table shows the ARN format for each resource type and an example.

Resource Type	Format for ARN
`application`	`arn:aws:elasticbeanstalk:`*`region`*`:`*`accountid`*`:application/`*`applicationname`* Example: **`arn:aws:elasticbeanstalk:us-east-1:123456789012:application/My App`**
`applicationversion`	`arn:aws:elasticbeanstalk:`*`region`*`:`*`accountid`*`:applicationversion/`*`applicationname`*`/`*`versionlabel`* Example: **`arn:aws:elasticbeanstalk:us-east-1:123456789012:applicationversion/My App/My Version`**
`environment`	`arn:aws:elasticbeanstalk:`*`region`*`:`*`accountid`*`:environment/`*`applicationname`*`/`*`environmentname`* Example: **`arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My App/MyEnvironment`**
`solutionstack`	`arn:aws:elasticbeanstalk:`*`region`*`::solutionstack/`*`solutionstackname`* Example: **`arn:aws:elasticbeanstalk:us-east-1::solutionstack/32bit Amazon Linux running Tomcat 7`**
`template`	`arn:aws:elasticbeanstalk:`*`region`*`:`*`accountid`*`:template/`*`applicationname`*`/`*`templatename`* Example: **`arn:aws:elasticbeanstalk:us-east-1:123456789012:template/My App/My Template`**

An environment, application version, and configuration template are always contained within a specific application. You'll notice that these resources all have an application name in their resource path so that they are uniquely identified by their resource name and the containing application. Although solution stacks are used by configuration templates and environments, solution stacks are not specific to an application or AWS account and do not have the application or AWS account in their ARNs.

Resources and Conditions for AWS Elastic Beanstalk Actions

Topics

This section describes the resources and conditions that you can use in policy statements to grant permissions that allow specific AWS Elastic Beanstalk actions to be performed on specific AWS Elastic Beanstalk resources.

Note

Some AWS Elastic Beanstalk actions may require permissions to other AWS services. For example, the following policy gives permissions for all Auto Scaling, Amazon S3, Amazon EC2, Amazon CloudWatch, Amazon SNS, Elastic Load Balancing, and AWS CloudFormation (for non-legacy container types) actions required to complete any AWS Elastic Beanstalk action. AWS Elastic Beanstalk relies on these additional services to provision underlying resources when creating an environment. For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).

The following policy is an example. It gives a broad set of permissions to the AWS products that AWS Elastic Beanstalk uses to manage applications and environments. For example, `ec2:*` allows an IAM user to perform any action on any Amazon EC2 resource in the AWS account. These permissions are not limited to the resources that you use with AWS Elastic Beanstalk. As a best practice, you should grant individuals only the permissions they need to perform their duties.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
       "ec2:*",
       "elasticloadbalancing:*",
       "autoscaling:*",
       "cloudwatch:*",
       "s3:*",
       "sns:*",
       "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

The following policy is an example. It gives a broad set of permissions to the AWS products that AWS Elastic Beanstalk uses to manage applications and environments. For example, `ec2:*` allows an IAM user to perform any action on any Amazon EC2 resource in the AWS account. These permissions are not limited to the resources that you use with AWS Elastic Beanstalk. As a best practice, you should grant individuals only the permissions they need to perform their duties.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
       "ec2:*",
       "elasticloadbalancing:*",
       "autoscaling:*",
       "cloudwatch:*",
       "s3:*",
       "sns:*",
       "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Policy Information for AWS Elastic Beanstalk Actions

The following table lists all AWS Elastic Beanstalk actions, the resource that each action acts upon, and the additional contextual information that can be provided using conditions.

Conditions enable you to specify permissions to resources that the action needs to complete. For example, when you can call the `CreateEnvironment` action, you must also specify the application version to deploy as well as the application that contains that application name. When you set permissions for the `CreateEnvironment` action, you specify the application and application version that you want the action to act upon by using the `InApplication` and `FromApplicationVersion` conditions. In addition, you can specify the environment configuration with a solution stack (`FromSolutionStack`) or a configuration template (`FromConfigurationTemplate`). The following policy statement allows the `CreateEnvironment` action to create an environment with the name **`myenv`** (specified by `Resource`) in the application **`My App`** (specified by the `InApplication` condition) using the application version **`My Version`** (`FromApplicationVersion`) with a **`32bit Amazon Linux running Tomcat 7`** configuration (`FromSolutionStack`):

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My
App/myenv"
      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbean
stalk:us-east-1:123456789012:applicationversion/My App/My Version"],
          "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-
east-1::solutionstack/32bit Amazon Linux running Tomcat 7"]
        }
      }
    }
  ]
}
```

As you can see in the example above, resources are specified using their Amazon Resource Name (ARN). For more information about the ARN format for AWS Elastic Beanstalk resources, see Amazon Resource Name (ARN) Format for AWS Elastic Beanstalk (p. 230).

The Comments column contains a simple example statement that grants permission to use the action on a specific resource with the appropriate contextual information provided through one or more conditions. The Comments column also lists dependencies that the action may have on permissions to perform other actions or to access other resources.

Note

If you set a policy on `elasticbeanstalk:Describe*` actions, those actions return only values that are permitted through the policy. For example, the following policy allows the `elasticbeanstalk:DescribeEvents` action to return a list of event descriptions for the environment **`myenv`** in the application **`My App`**. If you applied this policy to a user, that user could successfully perform the `elasticbeanstalk:DescribeEvents` action using **`myenv`** for the `EnvironmentName` parameter to get the list of events for **`myenv`**. However, if the user used another environment name for `EnvironmentName` or specified different parameters such as one for a specific application version, the action would return no event descriptions because the user has permission to view only**`myenv`** events. If the user specified no parameters for `elasticbeanstalk:DescribeEvents`, the action would return only the events for **`myenv`** because that is the only resource the user has permissions for.

```
{
  "Statement": [
    {
      "Action": "elasticbeanstalk:DescribeEvents",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My
 App/myenv"
      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

Policy information for AWS Elastic Beanstalk actions, including resources, conditions, examples, and dependencies

Resource	Conditions	Comments
Action: `CheckDNSAvailability`		

Resource	Conditions	Comments
`"*"`	N/A	This example allows the `CheckDNSAvailability` action to check if a CNAME is available. Note that permission to a resource is not required for this action and the resource should be specified by `"*"`.

```
{
  "Statement": [
    {
      "Action": [
       "elasticbeanstalk:CheckDNSAvailabil
ity"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Action: `CreateApplication`		
`application`	N/A	This example allows the `CreateApplication` action to create applications whose names begin with **`DivA`**:

```
{
  "Statement": [
    {
      "Action": [
       "elasticbeanstalk:CreateApplication"

      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/DivA*"
      ]
    }
  ]
}
```

Action: `CreateApplicationVersion`		

Resource	Conditions	Comments
applicationversion	`InApplication`	This example allows the `CreateApplicationVersion` action to create application versions with any name (**`*`**) in the application **`My App`**: `{ "Statement": [{ "Action": ["elasticbeanstalk:CreateApplicationVersion"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-1:123456789012:applicationversion/My App/*"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-1:123456789012:application/My App"] } } }] }`
Action: `CreateConfigurationTemplate`		

```
{
  "Statement": [
    {
      "Action": [
       "elasticbeanstalk:CreateApplication
Version"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:applicationversion/My App/*"

      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication":
 ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

Resource	Conditions	Comments

configurationtemplate	InApplication
	FromApplication
	FromApplicationVersion
	FromConfigurationTemplate
	FromEnvironment
	FromSolutionStack
		This example allows the `CreateConfigurationTemplate` action to create configuration templates whose name begins with **`My Template`** (`My Template*`) in the application **`My App`**:
		```
		{
		  "Statement": [
		    {
		      "Action": [
		        "elasticbeanstalk:CreateConfigura
		tionTemplate"
		      ],
		      "Effect": "Allow",
		      "Resource": [
		        "arn:aws:elasticbeanstalk:us-east-
		1:123456789012:configurationtemplate/My
		App/My Template*"
		      ],
		      "Condition": {
		        "StringEquals": {
		         "elasticbeanstalk:InApplication":
		 ["arn:aws:elasticbeanstalk:us-east-
		1:123456789012:application/My App"],
		          "elasticbeanstalk:FromSolution
		Stack": ["arn:aws:elasticbeanstalk:us-east-
		1::solutionstack/32bit Amazon Linux running
		 Tomcat 7"]
		        }
		      }
		    }
		  ]
		}
		```

Action: `CreateEnvironment`		

Resource	Conditions	Comments

`environment`	`InApplication`
	
	`FromApplicationVersion`
	
	`FromConfigurationTemplate`
	
	`FromSolutionStack`
		This example allows the `CreateEnvironment` action to create an environment whose name is **`myenv`** in the application **`My App`** and using the solution stack **`32bit Amazon Linux running Tomcat 7`**:
		
		`{`
		`  "Statement": [`
		`    {`
		`      "Action": [`
		`       "elasticbeanstalk:CreateEnvironment"`
		
		`      ],`
		`      "Effect": "Allow",`
		`      "Resource": [`
		`        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My App/myenv"`
		`      ],`
		`      "Condition": {`
		`        "StringEquals": {`
		`         "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-1:123456789012:application/My App"],`
		`         "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-east-1:123456789012:applicationversion/My App/My Version"],`
		`          "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-east-1::solutionstack/32bit Amazon Linux running Tomcat 7"]`
		`        }`
		`      }`
		`    }`
		`  ]`
		`}`

Action: `CreateStorageLocation`		

Resource	Conditions	Comments

`"*"`	N/A	This example allows the `CreateStorageLocation` action to create an Amazon S3 storage location. Note that permission to an AWS Elastic Beanstalk resource is not required for this action, and the resource should be specified by `"*"`.
		```
		{
		  "Statement": [
		    {
		      "Action": [
		       "elasticbeanstalk:CreateStorageLoca
		tion"
		      ],
		      "Effect": "Allow",
		      "Resource": "*"
		    }
		  ]
		}
		```

Action: `DeleteApplication`		

`application`	N/A	This example allows the `DeleteApplication` action to delete the application **`My App`**:
		```
		{
		  "Statement": [
		    {
		      "Action": [
		       "elasticbeanstalk:DeleteApplication"

		      ],
		      "Effect": "Allow",
		      "Resource": [
		        "arn:aws:elasticbeanstalk:us-east-
		1:123456789012:application/My App"
		      ]
		    }
		  ]
		}
		```

Action: `DeleteApplicationVersion`		

Resource	Conditions	Comments
applicationversion	`InApplication`	This example allows the `DeleteApplicationVersion` action to delete an application version whose name is **`My Version`** in the application **`My App`**:

```
{
  "Statement": [
    {
      "Action": [
       "elasticbeanstalk:DeleteApplication
Version"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:applicationversion/My App/My
 Version"
      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication":
 ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

Action: `DeleteConfigurationTemplate`

Resource	Conditions	Comments
configurationtemplate	`InApplication` (Optional)	This example allows the `DeleteConfigurationTemplate` action to delete a configuration template whose name is **`My Template`** in the application **`My App`**. Specifying the application name as a condition is optional.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:DeleteConfigura
tionTemplate"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:configurationtemplate/My
App/My Template"
      ]
    }
  ]
}
```

Resource	Conditions	Comments
Action: `DeleteEnvironmentConfiguration`		
`environment`	`InApplication` (Optional)	This example allows the `DeleteEnvironmentConfiguration` action to delete a draft configuration for the environment **`myenv`** in the application **`My App`**. Specifying the application name as a condition is optional.

```
{
  "Statement": [
    {
      "Action": [
       "elasticbeanstalk:DeleteEnvironment
Configuration"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv"
      ]
    }
  ]
}
```

Resource	Conditions	Comments
Action: `DescribeApplicationVersions`		
applicationversion	`InApplication` (Optional)	This example allows the `DescribeApplicationVersions` action to describe the application version **`My Version`** in the application **`My App`**. Specifying the application name as a condition is optional.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:DescribeApplica
tionVersions"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:applicationversion/My App/My
 Version"
      ]
    }
  ]
}
```

Resource	Conditions	Comments
Action: `DescribeApplications`		

Resource	Conditions	Comments

`application`	N/A	This example allows the `DescribeApplications` action to describe the application My App.
		```
		{
		  "Statement": [
		    {
		      "Action": [
		        "elasticbeanstalk:DescribeApplica
		tions"
		      ],
		      "Effect": "Allow",
		      "Resource": [
		        "arn:aws:elasticbeanstalk:us-east-
		1:123456789012:application/My App"
		      ]
		    }
		  ]
		}
		```

Action: `DescribeConfigurationOptions`		

`environment`, `configurationtemplate`, `solutionstack`	`InApplication` (Optional)	This example allows the `DescribeConfigurationOptions` action to describe the configuration options for the environment **`myenv`** in the application **`My App`**. Specifying the application name as a condition is optional.
		```
		{
		  "Statement": [
		    {
		     "Action": "elasticbeanstalk:Describe
		ConfigurationOptions",
		      "Effect": "Allow",
		      "Resource": [
		        "arn:aws:elasticbeanstalk:us-east-
		1:123456789012:environment/My App/myenv"
		      ]
		    }
		  ]
		}
		```

Action: `DescribeConfigurationSettings`		

Resource	Conditions	Comments
`environment`, `configurationtemplate`	`InApplication` (Optional)	This example allows the `DescribeConfigurationSettings` action to describe the configuration settings for the environment **`myenv`** in the application **`My App`**. Specifying the application name as a condition is optional.

```
{
  "Statement": [
    {
     "Action": "elasticbeanstalk:Describe
ConfigurationSettings",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv"
      ]
    }
  ]
}
```

Resource	Conditions	Comments
Action: `DescribeEnvironmentResources`		
`environment`	`InApplication` (Optional)	This example allows the `DescribeEnvironmentResources` action to return list of AWS resources for the environment **`myenv`** in the application **`My App`**. Specifying the application name as a condition is optional.

```
{
  "Statement": [
    {
     "Action": "elasticbeanstalk:DescribeEn
vironmentResources",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv"
      ]
    }
  ]
}
```

Resource	Conditions	Comments
Action: `DescribeEnvironments`		

Resource	Conditions	Comments
`environment`	`InApplication` (Optional)	This example allows the `DescribeEnvironments` action to describe the environments **`myenv`** and **`myotherenv`** in the application **`My App`**. Specifying the application name as a condition is optional.

```
{
  "Statement": [
    {
     "Action": "elasticbeanstalk:DescribeEn
vironments",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv",
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App2/myother
env"
      ]
    }
  ]
}
```

Action: DescribeEvents

Resource	Conditions	Comments
`application`, `applicationversion`, `configurationtemplate`, `environment`	`InApplication`	This example allows the `DescribeEvents` action to list event descriptions for the environment **`myenv`** and the application version **`My Version`** in the application **`My App`**.

```
{
  "Statement": [
    {
      "Action": "elasticbeanstalk:Descri
beEvents",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv",
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:applicationversion/My App/My
 Version"
      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication":
 ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

Resource	Conditions	Comments
Action: `ListAvailableSolutionStacks`		
`solutionstack`	N/A	This example allows the `ListAvailableSolutionStacks` action to return only the solution stack **`32bit Amazon Linux running Tomcat 7`**.

```
{
  "Statement": [
    {
      "Action": [
       "elasticbeanstalk:ListAvailableSol
utionStacks"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:elasticbean
stalk:us-east-1::solutionstack/32bit Amazon
 Linux running Tomcat 7"
    }
  ]
}
```

Resource	Conditions	Comments
Action: `RebuildEnvironment`		
`environment`	`InApplication`	This example allows the `RebuildEnvironment` action to rebuild the environment **`myenv`** in the application **`My App`**.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:RebuildEnviron
ment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication":
 ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

Resource	Conditions	Comments
Action: RequestEnvironmentInfo		

environment	InApplication	This example allows the RequestEnvironmentInfo action to compile information about the environment myenv in the application My App.
		
		`{`
		`  "Statement": [`
		`    {`
		`      "Action": [`
		`        "elasticbeanstalk:RequestEnviron`
		`mentInfo"`
		`      ],`
		`      "Effect": "Allow",`
		`      "Resource": [`
		`        "arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:environment/My App/myenv"`
		`      ],`
		`      "Condition": {`
		`        "StringEquals": {`
		`          "elasticbeanstalk:InApplication":`
		` ["arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:application/My App"]`
		`        }`
		`      }`
		`    }`
		`  ]`
		`}`

Action: RestartAppServer		

Resource	Conditions	Comments
`environment`	`InApplication`	This example allows the `RestartAppServer` action to restart the application container server for the environment **`myenv`** in the application **`My App`**.

```
{
  "Statement": [
    {
      "Action": [
       "elasticbeanstalk:RestartAppServer"

      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
         "elasticbeanstalk:InApplication":
 ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

Action: `RetrieveEnvironmentInfo`

Resource	Conditions	Comments

environment	InApplication	This example allows the RetrieveEnvironmentInfo action to retrieve the compiled information for the environment **myenv** in the application **My App**.
		
		`{`
		`  "Statement": [`
		`    {`
		`      "Action": [`
		`        "elasticbeanstalk:RetrieveEnviron`
		`mentInfo"`
		`      ],`
		`      "Effect": "Allow",`
		`      "Resource": [`
		`        "arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:environment/My App/myenv"`
		`      ],`
		`      "Condition": {`
		`        "StringEquals": {`
		`         "elasticbeanstalk:InApplication":`
		` ["arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:application/My App"]`
		`        }`
		`      }`
		`    }`
		`  ]`
		`}`

Action: SwapEnvironmentCNAMEs		

environment	InApplication (Optional)
	
	FromEnvironment (Optional)
		This example allows the SwapEnvironmentCNAMEs action to swap the CNAMEs for the environments **mysrcenv** and **mydestenv**.
		
		`{`
		`  "Statement": [`
		`    {`
		`      "Action": [`
		`       "elasticbeanstalk:SwapEnvironmentC`
		`NAMEs"`
		`      ],`
		`      "Effect": "Allow",`
		`      "Resource": [`
		`        "arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:environment/My App/mysrcenv",`
		
		`        "arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:environment/My App/mydestenv"`
		
		`      ]`
		`    }`
		`  ]`
		`}`

Resource	Conditions	Comments
Action: `TerminateEnvironment`		
`environment`	`InApplication`	This example allows the `TerminateEnvironment` action to terminate the environment **`myenv`** in the application **`My App`**.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:TerminateEnviron
ment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication":
 ["arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

Resource	Conditions	Comments
Action: UpdateApplication		
`application`	N/A	This example allows the `UpdateApplication` action to update properties of the application **`My App`**.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateApplication"

      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-
1:123456789012:application/My App"
      ]
    }
  ]
}
```

Resource	Conditions	Comments
Action: `UpdateApplicationVersion`		

Resource	Conditions	Comments

applicationversion	`InApplication`	This example allows the `UpdateApplicationVersion` action to update the properties of the application version **`My Version`** in the application **`My App`**.
		
		`{`
		`  "Statement": [`
		`    {`
		`      "Action": [`
		`       "elasticbeanstalk:UpdateApplication Version"`
		`      ],`
		`      "Effect": "Allow",`
		`      "Resource": [`
		`        "arn:aws:elasticbeanstalk:us-east-1:123456789012:applicationversion/My App/My Version"`
		`      ],`
		`      "Condition": {`
		`        "StringEquals": {`
		`         "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-1:123456789012:application/My App"]`
		`        }`
		`      }`
		`    }`
		`  ]`
		`}`

Action: `UpdateConfigurationTemplate`		

Resource	Conditions	Comments

configurationtemplate	InApplication	This example allows the `UpdateConfigurationTemplate` action to update the properties or options of the configuration template **`My Template`** in the application **`My App`**.
		
		`{`
		`  "Statement": [`
		`    {`
		`      "Action": [`
		`        "elasticbeanstalk:UpdateConfigura`
		`tionTemplate"`
		`      ],`
		`      "Effect": "Allow",`
		`      "Resource": [`
		`        "arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:configurationtemplate/My`
		`App/My Template"`
		`      ],`
		`      "Condition": {`
		`        "StringEquals": {`
		`         "elasticbeanstalk:InApplication":`
		` ["arn:aws:elasticbeanstalk:us-east-`
		`1:123456789012:application/My App"]`
		`        }`
		`      }`
		`    }`
		`  ]`
		`}`

Action: `UpdateEnvironment`		

Resource	Conditions	Comments

`environment`	`InApplication`
	
	`FromApplicationVersion`
	
	`FromConfigurationTemplate`
		This example allows the `UpdateEnvironment` action to update the environment **`myenv`** in the application **`My App`** by deploying the application version **`My Version`**.
		
		`{`
		`  "Statement": [`
		`    {`
		`      "Action": [`
		`       "elasticbeanstalk:UpdateEnvironment"`
		
		`      ],`
		`      "Effect": "Allow",`
		`      "Resource": [`
		`        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My App/myenv"`
		`      ],`
		`      "Condition": {`
		`        "StringEquals": {`
		`          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-1:123456789012:application/My App"],`
		`         "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-east-1:123456789012:applicationversion/My App/My Version"]`
		`        }`
		`      }`
		`    }`
		`  ]`
		`}`

Action: `ValidateConfigurationSettings`		

Resource	Conditions	Comments

`template`, `environment`	`InApplication`	This example allows the `ValidateConfigurationSettings` action to validates configuration settings against the environment **`myenv`** in the application **`My App`**.
		```
		{
		  "Statement": [
		    {
		      "Action": [
		        "elasticbeanstalk:ValidateConfigur
		ationSettings"
		      ],
		      "Effect": "Allow",
		      "Resource": [
		        "arn:aws:elasticbeanstalk:us-east-
		1:123456789012:environment/My App/myenv"
		      ],
		      "Condition": {
		        "StringEquals": {
		          "elasticbeanstalk:InApplication":
		 ["arn:aws:elasticbeanstalk:us-east-
		1:123456789012:application/My App"]
		        }
		      }
		    }
		  ]
		}
		```

Condition Keys for AWS Elastic Beanstalk Actions

Keys enable you to specify conditions that express dependencies, restrict permissions, or specify constraints on the input parameters for an action. AWS Elastic Beanstalk supports the following keys.

`InApplication`

Specifies the application that contains the resource that the action operates on.

The following example allows the `UpdateApplicationVersion` action to update the properties of the application version **`My Version`**. The `InApplication` condition specifies **`My App`** as the container for **`My Version`**.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateApplicationVersion"
      ],
      "Effect": "Allow",
      "Resource": [
       "arn:aws:elasticbeanstalk:us-east-1:123456789012:applicationversion/My
 App/My Version"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-1:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

`FromApplicationVersion`

Specifies an application version as a dependency or a constraint on an input parameter.

The following example allows the `UpdateEnvironment` action to update the environment **`myenv`** in the application **`My App`**. The `FromApplicationVersion` condition constrains the `VersionLabel` parameter to allow only the application version **`My Version`** to update the environment.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My
App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-1:123456789012:application/My App"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbean
stalk:us-east-1:123456789012:applicationversion/My App/My Version"]
        }
      }
    }
  ]
}
```

`FromConfigurationTemplate`

Specifies a configuration template as a dependency or a constraint on an input parameter.

The following example allows the `UpdateEnvironment` action to update the environment **`myenv`** in the application **`My App`**. The `FromConfigurationTemplate` condition constrains the `TemplateName` parameter to allow only the configuration template **`My Template`** to update the environment.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My
App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-1:123456789012:application/My App"],
          "elasticbeanstalk:FromConfigurationTemplate": ["arn:aws:elastic
beanstalk:us-east-1:123456789012:configurationtemplate/My App/My Template"]

        }
      }
    }
  ]
}
```

`FromEnvironment`

Specifies an environment as a dependency or a constraint on an input parameter.

The following example allows the `SwapEnvironmentCNAMEs` action to swap the CNAMEs in **`My App`** for all environments whose names begin with **`mysrcenv`** and **`mydestenv`** but not those environments whose names begin with **`mysrcenvPROD*`** and **`mydestenvPROD*`**.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:SwapEnvironmentCNAMEs"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My
App/mysrcenv*",
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:environment/My
App/mydestenv*"
      ],
      "Condition": {
        "StringNotLike": {
         "elasticbeanstalk:FromEnvironment": ["arn:aws:elasticbeanstalk:us-
east-1:123456789012:environment/My App/mysrcenvPROD*",
         "elasticbeanstalk:FromEnvironment": ["arn:aws:elasticbeanstalk:us-
east-1:123456789012:environment/My App/mydestenvPROD*"
          ]
        }
      }
    }
  ]
}
```

`FromSolutionStack`

Specifies a solution stack as a dependency or a constraint on an input parameter.

This example allows the `CreateConfigurationTemplate` action to create configuration templates whose name begins with **`My Template`** (`My Template*`) in the application **`My App`**. The

`FromSolutionStack` condition constrains the `solutionstack` parameter to allow only the solution stack **`32bit Amazon Linux running Tomcat 7`** as the input value for that parameter.

```
{
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:CreateConfigurationTemplate"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-1:123456789012:configurationtem
plate/My App/My Template*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-1:123456789012:application/My App"],
         "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-
east-1::solutionstack/32bit Amazon Linux running Tomcat 7"]
        }
      }
    }
  ]
}
```

Example Policies Based on Policy Templates

This section walks through a use case for controlling user access to AWS Elastic Beanstalk and sample policies that support the use case. These policies use the AWS Elastic Beanstalk policy templates as a starting point. For information about attaching policies to users and groups, go to Managing IAM Policies in the *Using AWS Identity and Access Management Guide*.

In our use case, Example Corp. is a software company with three teams responsible for their website: administrators who manage the infrastructure, developers who build the software for the website, and a QA team that tests the website. To help manage permissions to their AWS Elastic Beanstalk assets, Example Corp. creates groups that contain the members of each team: Admins, Developers, and Testers. Example Corp. wants to enable the Admins group to have full access to all applications, environments, and their underlying resources so that they can create, troubleshoot, and delete all of their AWS Elastic Beanstalk assets. Developers require permissions to view all AWS Elastic Beanstalk assets and to create and deploy application versions. Developers should not be able to create new applications or environments and cannot terminate running environments since they are not part of the Admins group. Testers need to view all AWS Elastic Beanstalk resources in order to monitor and test applications so that they can run automated tests and access the web application. However, the Testers group should not be able to make changes to any AWS Elastic Beanstalk resources.

Example 1: Allow the Admins group to use all AWS Elastic Beanstalk and related service APIs

The following policy gives permissions for all actions required to use AWS Elastic Beanstalk. This policy includes actions for Auto Scaling, Amazon S3, Amazon EC2, Amazon CloudWatch, Amazon SNS, Elastic Load Balancing, and AWS CloudFormation (for non-legacy container types), as well as for all AWS Elastic Beanstalk actions. AWS Elastic Beanstalk relies on these additional services to provision underlying resources when creating an environment. For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).

Note

The following policy is an example. It gives a broad set of permissions to the AWS products that AWS Elastic Beanstalk uses to manage applications and environments. For example, `ec2:*` allows an IAM user to perform any action on any Amazon EC2 resource in the AWS account. These permissions are not limited to the resources that you use with AWS Elastic Beanstalk. As a best practice, you should grant individuals only the permissions they need to perform their duties.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
       "elasticbeanstalk:*",
       "ec2:*",
       "elasticloadbalancing:*",
       "autoscaling:*",
       "cloudwatch:*",
       "s3:*",
       "sns:*",
       "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Example 2: Allow the Developers group to do all actions except highly privileged operations such as creating applications and environments

The following policy denies permission to create applications and environments but allows all other AWS Elastic Beanstalk actions.

> **Note**
>
> The following policy is an example. It gives a broad set of permissions to the AWS products that AWS Elastic Beanstalk uses to manage applications and environments. For example, `ec2:*` allows an IAM user to perform any action on any Amazon EC2 resource in the AWS account. These permissions are not limited to the resources that you use with AWS Elastic Beanstalk. As a best practice, you should grant individuals only the permissions they need to perform their duties.

```
{
   "Statement":[
     {
      "Action":["elasticbeanstalk:CreateApplication",
                                 "elasticbeanstalk:CreateEnvironment",
                                 "elasticbeanstalk:DeleteApplication",
                                 "elasticbeanstalk:RebuildEnvironment",
                                 "elasticbeanstalk:SwapEnvironmentCNAMEs",
                                 "elasticbeanstalk:TerminateEnvironment"],
      "Effect":"Deny",
      "Resource":"*"
      },
      {
      "Action":["elasticbeanstalk:*",
                                 "ec2:*",
                                 "elasticloadbalancing:*",
                                 "autoscaling:*",
                                 "cloudwatch:*",
                                 "s3:*",
                                 "sns:*",
                                 "cloudformation:*"],
      "Effect":"Allow",
      "Resource":"*"
      }
   ]
}
```

Example 3: Allow the Testers group to view all AWS Elastic Beanstalk assets but not to perform any actions.

The following policy allows read-only access to all applications, application versions, events, and environments.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
       "elasticbeanstalk:Check*",
        "elasticbeanstalk:Describe*",
        "elasticbeanstalk:List*",
        "elasticbeanstalk:RequestEnvironmentInfo",
        "elasticbeanstalk:RetrieveEnvironmentInfo",
        "ec2:Describe*",
        "elasticloadbalancing:Describe*",
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
       "cloudwatch:List*",
        "cloudwatch:Get*",
        "s3:Get*",
        "s3:List*",
        "sns:Get*",
        "sns:List*",
        "cloudformation:Describe*",
        "cloudformation:Get*",
        "cloudformation:List*",
        "cloudformation:Validate*",
        "cloudformation:Estimate*"
      ],
      "Resource": "*"
    }
  ]
}
```

Example Policies Based on Resource Permissions

This section walks through a use case for controlling user permissions for AWS Elastic Beanstalk actions that access specific AWS Elastic Beanstalk resources. We'll walk through the sample policies that support the use case. For more information policies on AWS Elastic Beanstalk resources, see Creating Policies to Control Access to Specific AWS Elastic Beanstalk Resources (p. 227). For information about attaching policies to users and groups, go to Managing IAM Policies in the *Using AWS Identity and Access Management Guide*.

In our use case, Example Corp. is a small consulting firm developing applications for two different customers. John is the development manager overseeing the development of the two AWS Elastic Beanstalk applications, app1 and app2. John does development and some testing on the two applications, and only he can update the production environment for the two applications. These are the permissions that he needs for app1 and app2:

- View application, application versions, environments, and configuration templates
- Create application versions and deploy them to the staging environment
- Update the production environment
- Create and terminate environments

Jill is a tester who needs access to view the following resources in order to monitor and test the two applications: applications, application versions, environments, and configuration templates. However, she should not be able to make changes to any AWS Elastic Beanstalk resources.

Jack is the developer for app1 who needs access to view all resources for app1 and also needs to create application versions for app1 and deploy them to the staging environment.

Joe is the administrator of the AWS account for Example Corp. He has created IAM users for John, Jill, and Jack and attaches the following policies to those users to grant the appropriate permissions to the app1 and app2 applications.

Example 1: Policies that allow John to perform his development, test, and deployment actions on app1 and app2

We have broken down John's policy into three separate policies so that they are easier to read and manage. Together, they give John the permissions he needs to perform the AWS Elastic Beanstalk actions on the two applications.

The first policy specifies actions for Auto Scaling, Amazon S3, Amazon EC2, Amazon CloudWatch, Amazon SNS, Elastic Load Balancing, and AWS CloudFormation (for non-legacy container types). AWS Elastic Beanstalk relies on these additional services to provision underlying resources when creating an environment. For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).

> **Note**
>
> The following policy is an example. It gives a broad set of permissions to the AWS products that AWS Elastic Beanstalk uses to manage applications and environments. For example, `ec2:*` allows an IAM user to perform any action on any Amazon EC2 resource in the AWS account. These permissions are not limited to the resources that you use with AWS Elastic Beanstalk. As a best practice, you should grant individuals only the permissions they need to perform their duties.

```
{
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:*",
            "elasticloadbalancing:*",
            "autoscaling:*",
            "cloudwatch:*",
            "s3:*",
            "sns:*",
            "cloudformation:*"
         ],
         "Resource":"*"
      }
   ]
}
```

The second policy specifies the AWS Elastic Beanstalk actions that John is allowed to perform on the app1 and app2 resources. The `AllCallsInApplications` statement allows all AWS Elastic Beanstalk actions (`"elasticbeanstalk:*"`) performed on all resources within app1 and app2 (for example, `elasticbeanstalk:CreateEnvironment`). The `AllCallsOnApplications` statement allows all AWS Elastic Beanstalk actions (`"elasticbeanstalk:*"`) on the app1 and app2 application resources (for example, `elasticbeanstalk:DescribeApplications`, `elasticbeanstalk:UpdateApplication`, etc.). The `AllCallsOnSolutionStacks` statement allows all AWS Elastic Beanstalk actions (`"elasticbeanstalk:*"`) for solution stack resources (for example, `elasticbeanstalk:ListAvailableSolutionStacks`).

```
{
   "Statement":[
      {
         "Sid":"AllCallsInApplications",
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-1:123456789012:applica
tion/app1",
                  "arn:aws:elasticbeanstalk:us-east-1:123456789012:applica
tion/app2"
               ]
            }
         }
      },
      {
         "Sid":"AllCallsOnApplications",
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
           "arn:aws:elasticbeanstalk:us-east-1:123456789012:application/app1",

            "arn:aws:elasticbeanstalk:us-east-1:123456789012:application/app2"

         ]
      },
      {
         "Sid":"AllCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-1::solutionstack/*"
         ]
      }
   ]
}
```

The third policy specifies the AWS Elastic Beanstalk actions that the second policy needs permissions to in order to complete those AWS Elastic Beanstalk actions. The `AllNonResourceCalls` statement allows the `elasticbeanstalk:CheckDNSAvailability` action, which is required to call `elasticbeanstalk:CreateEnvironment` and other actions. It also allows the `elasticbeanstalk:CreateStorageLocation` action, which is required for `elasticbeanstalk:CreateApplication`, `elasticbeanstalk:CreateEnvironment`, and other actions.

```
{
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability",
            "elasticbeanstalk:CreateStorageLocation"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example 2: Policies that allow Jill to test and monitor app1 and app2

We have broken down Jill's policy into three separate policies so that they are easier to read and manage. Together, they give Jill the permissions she needs to perform the AWS Elastic Beanstalk actions on the two applications.

The first policy specifies `Describe*`, `List*`, and `Get*` actions on Auto Scaling, Amazon S3, Amazon EC2, Amazon CloudWatch, Amazon SNS, Elastic Load Balancing, and AWS CloudFormation (for non-legacy container types) so that the AWS Elastic Beanstalk actions are able to retrieve the relevant information about the underlying resources of the app1 and app2 applications.

```
{
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:Describe*",
            "elasticloadbalancing:Describe*",
            "autoscaling:Describe*",
            "cloudwatch:Describe*",
            "cloudwatch:List*",
            "cloudwatch:Get*",
            "s3:Get*",
            "s3:List*",
            "sns:Get*",
            "sns:List*",
            "cloudformation:Describe*",
         "cloudformation:Get*",
         "cloudformation:List*",
         "cloudformation:Validate*",
         "cloudformation:Estimate*"
         ],
         "Resource":"*"
      }
   ]
}
```

The second policy specifies the AWS Elastic Beanstalk actions that Jill is allowed to perform on the app1 and app2 resources. The `AllReadCallsInApplications` statement allows her to call the `Describe*` actions and the environment info actions. The `AllReadCallsOnApplications` statement allows her to call the `DescribeApplications` and `DescribeEvents` actions on the app1 and app2 application resources. The `AllReadCallsOnSolutionStacks` statement allows viewing actions that involve solution stack resources (`ListAvailableSolutionStacks`, `DescribeConfigurationOptions`, and `ValidateConfigurationSettings`).

```
{
   "Statement":[
      {
         "Sid":"AllReadCallsInApplications",
         "Action":[
            "elasticbeanstalk:Describe*",
            "elasticbeanstalk:RequestEnvironmentInfo",
            "elasticbeanstalk:RetrieveEnvironmentInfo"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-1:123456789012:applica
tion/app1",
                  "arn:aws:elasticbeanstalk:us-east-1:123456789012:applica
tion/app2"
               ]
            }
         }
      },
      {
         "Sid":"AllReadCallsOnApplications",
         "Action":[
            "elasticbeanstalk:DescribeApplications",
            "elasticbeanstalk:DescribeEvents"
         ],
         "Effect":"Allow",
         "Resource":[
           "arn:aws:elasticbeanstalk:us-east-1:123456789012:application/app1",

            "arn:aws:elasticbeanstalk:us-east-1:123456789012:application/app2"

         ]
      },
      {
         "Sid":"AllReadCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:ListAvailableSolutionStacks",
            "elasticbeanstalk:DescribeConfigurationOptions",
            "elasticbeanstalk:ValidateConfigurationSettings"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-1::solutionstack/*"
         ]
      }
   ]
}
```

The third policy specifies the AWS Elastic Beanstalk actions that the second policy needs permissions to in order to complete those AWS Elastic Beanstalk actions. The `AllNonResourceCalls` statement allows the `elasticbeanstalk:CheckDNSAvailability` action, which is required for some viewing actions.

```
{
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example 3: Policies that allow Jack to access app1 to test, monitor, create application versions, and deploy to the staging environment

We have broken down Jack's policy into three separate policies so that they are easier to read and manage. Together, they give Jack the permissions he needs to perform the AWS Elastic Beanstalk actions on the app1 resource.

The first policy specifies the actions on Auto Scaling, Amazon S3, Amazon EC2, Amazon CloudWatch, Amazon SNS, Elastic Load Balancing, and AWS CloudFormation (for non-legacy container types) so that the AWS Elastic Beanstalk actions are able to view and work with the underlying resources of app1. For a list of supported non-legacy container types, see Why are some container types marked legacy? (p. 204).

Note

The following policy is an example. It gives a broad set of permissions to the AWS products that AWS Elastic Beanstalk uses to manage applications and environments. For example, `ec2:*` allows an IAM user to perform any action on any Amazon EC2 resource in the AWS account. These permissions are not limited to the resources that you use with AWS Elastic Beanstalk. As a best practice, you should grant individuals only the permissions they need to perform their duties.

```
{
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:*",
            "elasticloadbalancing:*",
            "autoscaling:*",
            "cloudwatch:*",
            "s3:*",
            "sns:*",
            "cloudformation:*"
         ],
         "Resource":"*"
      }
   ]
}
```

The second policy specifies the AWS Elastic Beanstalk actions that Jack is allowed to perform on the app1 resource.

```
{
   "Statement":[
      {
         "Sid":"AllReadCallsAndAllVersionCallsInApplications",
         "Action":[
            "elasticbeanstalk:Describe*",
            "elasticbeanstalk:RequestEnvironmentInfo",
            "elasticbeanstalk:RetrieveEnvironmentInfo",
            "elasticbeanstalk:CreateApplicationVersion",
            "elasticbeanstalk:DeleteApplicationVersion",
            "elasticbeanstalk:UpdateApplicationVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-1:123456789012:applica
tion/app1"
               ]
            }
         }
      },
      {
         "Sid":"AllReadCallsOnApplications",
         "Action":[
            "elasticbeanstalk:DescribeApplications",
            "elasticbeanstalk:DescribeEvents"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-1:123456789012:application/app1"

         ]
      },
      {
         "Sid":"UpdateEnvironmentInApplications",
         "Action":[
            "elasticbeanstalk:UpdateEnvironment"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-1:123456789012:environ
ment/app1/app1-staging*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-1:123456789012:applica
tion/app1"
               ]
            },
            "StringLike":{
               "elasticbeanstalk:FromApplicationVersion":[
                 "arn:aws:elasticbeanstalk:us-east-1:123456789012:application
version/app1/*"
               ]
            }
         }
      },
      {
         "Sid":"AllReadCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:ListAvailableSolutionStacks",
            "elasticbeanstalk:DescribeConfigurationOptions",
            "elasticbeanstalk:ValidateConfigurationSettings"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-1::solutionstack/*"
         ]
      }
   ]
}
```

The third policy specifies the AWS Elastic Beanstalk actions that the second policy needs permissions to in order to complete those AWS Elastic Beanstalk actions. The `AllNonResourceCalls` statement allows the `elasticbeanstalk:CheckDNSAvailability` action, which is required to call `elasticbeanstalk:CreateEnvironment` and other actions. It also allows the `elasticbeanstalk:CreateStorageLocation` action, which is required for `elasticbeanstalk:CreateEnvironment`, and other actions.

```
{
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability",
            "elasticbeanstalk:CreateStorageLocation"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Tools

Topics

Eb is an updated command line interface (CLI) for AWS Elastic Beanstalk that enables you to deploy applications quickly and more easily. For a walkthrough of how to deploy a sample application using eb, see Getting Started with Eb (p. 270). For a complete CLI reference for more advanced scenarios, see AWS Elastic Beanstalk API Command Line Interface (p. 274).

Getting Started with Eb

Eb is an updated command line interface (CLI) for AWS Elastic Beanstalk that enables you to deploy applications quickly and more easily. This section provides an end-to-end walkthrough using eb to launch a sample application, view it, update it, and then delete it. To complete this walkthrough, you will need to download the command line tools at the AWS Sample Code & Libraries website. For a complete CLI reference for more advanced scenarios, see Operations (p. 286), and see Getting Set Up (p. 275) for instructions on how to get set up.

Step 1: Initialize Your Git Repository

A Git client extension, AWS DevTools, is available as part of the CLI package. It enables you to deploy applications to AWS Elastic Beanstalk quickly. Before you can use the command line interface, you will need to install the following prerequisite software, and then intialize your Git repository.

To install the prerequisite software and initialize your Git repository

1. Install the following software onto your local computer:

   - Git. To download Git, go to http://git-scm.com/. Make sure you have at least version 1.6.6 or later.
   - For Linux, download Ruby version 1.8.7 or later. To view and download Ruby clients, go to http://www.ruby-lang.org/en/.

- For Windows, download PowerShell 2.0. Windows 7 and Windows Server 2008 R2 come with PowerShell 2.0. For earlier versions of Windows, you can download PowerShell 2.0. Visit http://technet.microsoft.com/en-us/scriptcenter/dd742419.aspx for more details.

2. Initialize your Git repository. In this example, we use the following command from the directory where we installed the command line interface:

```
git init .
```

Step 2: Configure AWS Elastic Beanstalk

AWS Elastic Beanstalk needs the following information to deploy an application:

- AWS access key ID
- AWS secret key
- Service region
- Application name
- Environment name
- Solution stack

Use the `init` command, and AWS Elastic Beanstalk will prompt you to enter this information. If a default value is available, and you want to use it, press `Enter`.

To configure AWS Elastic Beanstalk

1. From your directory where you installed the command line interface, type the following command.

```
eb init
```

2. When you are prompted for the AWS access key, type your access key. To get your access key information, go to Access Credentials.

```
Enter your AWS Access Key ID (current value is "AKIAIOSFODNN7EXAMPLE"):
```

3. When you are prompted for the AWS secret key, type your secret key. To get your secret key information, go to Access Credentials.

```
Enter your AWS Secret Access Key (current value is "wJalrXUtnFEMI/K7MDENG/bPxR
fiCYEXAMPLEKEY"):
```

4. When you are prompted for the AWS Elastic Beanstalk region, type the number of the region. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference. For this example, we'll use **US East (Virginia)**.
5. When you are prompted for the AWS Elastic Beanstalk application name, type the name of the application. AWS Elastic Beanstalk auto-generates an application name based on the current directory name if an application name has not been previously configured. In this example, we use HelloWorld.

```
Enter an AWS Elastic Beanstalk application name (auto-generated value is
"windows"): HelloWorld
```

Note

If you have a space in your application name, make sure you do not use quotes.

6. When you are prompted for the AWS Elastic Beanstalk environment name, type the name of the environment. AWS Elastic Beanstalk automatically creates an environment name based on your application name. If you want to accept the default, press `Enter`.

```
Enter an AWS Elastic Beanstalk environment name (current value is "HelloWorld-
env"):
```

Note

If you have a space in your application name, make sure you do not have a space in your environment name.

7. When you are prompted for the solution stack, type the number of the solution stack you want. For this example, we'll use **64bit Amazon Linux running PHP 5.3**.

After configuring AWS Elastic Beanstalk, you are ready to deploy a sample application.

If you want to update your AWS Elastic Beanstalk configuration, you can use the **`init`** command again. When prompted, you can update your configuration options. If you want to keep any previous settings, press the **Enter** key.

Step 3: Create Application

Next, you will create and deploy a sample application. For this step, you use a sample application that is already prepared. AWS Elastic Beanstalk uses the configuration information you specified in the previous step to do the following:

- Creates an application using the application name you specified.
- Launches an environment using the environment name you specified that provisions the AWS resources to host the application.
- Deploys the application into the newly created environment.

Use the `start` command to create and deploy a sample application.

To create an application

- From your directory where you installed the command line interface, type the following command.

```
eb start
```

This process may take several minutes to complete. AWS Elastic Beanstalk will provide status updates during the process. If at any time you want to stop polling for status updates, press **Ctrl+C**. Once the environment status is Green, AWS Elastic Beanstalk will output a URL for the application.

Step 4: View Application

In the previous step, you created an application and deployed it to AWS Elastic Beanstalk. After the environment is ready and its status is Green, AWS Elastic Beanstalk provides a URL to view the application. In this step, you can check the status of the environment to make sure it is set to Green and then copy and paste the URL to view the application.

Use the `status` command to check the environment status, and then use the URL to view the application.

To view an application

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb status
   ```

   AWS Elastic Beanstalk displays the environment status. If the environment is set to Green, then AWS Elastic Beanstalk displays the URL for the application. If you attached an RDS DB Instance to your environment, your RDS DB information is displayed.
2. Copy and paste the URL into your web browser to view your application.

Step 5: Update Your Application

After you have deployed a sample application, you can update the sample application with your own application. In this step, we'll update the sample PHP application with a simple HelloWorld application.

To update the sample application

1. Create a simple PHP file that displays "Hello World" and add it to your Git repository.

   ```
   <html>
    <head>
     <title>PHP Test</title>
    </head>
    <body>
    <?php echo '<p>Hello World</p>'; ?>
    </body>
   </html>
   ```

   Next, add your new program to your local Git repository, and then commit your change.

   ```
   git add index.php
   git commit -m "initial check-in"
   ```

   Note

   For information about Git commands, go to Git - Fast Version Control System.
2. Deploy to AWS Elastic Beanstalk.

   ```
   git aws.push
   ```

3. View your updated application. Copy and paste the same URL in your web browser as you did in Step 4: View Application (p. 273).

Step 6: Update Your Environment

At any time, you can quickly update your environment settings. In this step, we update the the maximum size of your Auto Scaling group to 2.

To update your AWS Elastic Beanstalk environment

1. Open the **OPTIONSETTINGS** file in the `.elasticbeanstalk` directory.
2. Type **2** for the **maxsize** option name of the **aws:autoscaling:asg** namespace.
3. Type the following command:

   ```
   eb update
   ```

   AWS Elastic Beanstalk displays a message once it has successfully updated the environment.

Step 7: Clean Up

If you no longer want to run your application, you can clean up by terminating your environment and deleting your application.

Use the `stop` command to terminate your environment and the `delete` command to delete your application.

To delete the application

1. From your directory where you installed the command line interface, type the following command.

   ```
   eb stop
   ```

   This process may take a few minutes. AWS Elastic Beanstalk will display a message once the environment has been successfully terminated.

   Note

   If you attached an RDS DB Instance to your environment, then your RDS DB will be deleted, and you will lose your data. To save your data, create a snapshot before you delete the application. For instructions on how to create a snapshot, go to Creating a DB Snapshot in the *Amazon Relational Database Service User Guide*.
2. From your directory where you installed the command line interface, type the following command.

   ```
   eb delete
   ```

   AWS Elastic Beanstalk will display a message once it has successfully deleted the application.

AWS Elastic Beanstalk API Command Line Interface

Topics

You can use the command line to create and deploy applications to AWS Elastic Beanstalk. This section contains a complete reference for the API command line interface. If you want a quick and easy way to deploy your applications without needing to know which command line operations to use, you can use eb. Eb is an updated command line interface for AWS Elastic Beanstalk that is interactive and asks you the necessary questions to deploy your application. For more information, see Getting Started with Eb (p. 270). This section discusses how to set up the API command line interface and the common options. For a complete reference list, see Operations (p. 286).

Getting Set Up

AWS Elastic Beanstalk provides a command line interface (CLI) to access AWS Elastic Beanstalk functionality without using the AWS Management Console or the APIs. This section describes the prerequisites for running the CLI tools (or command line tools), where to get the tools, how to set them up and their environment, and includes a series of common examples of tool usage.

Prerequisites

This document assumes you can work in a Linux/UNIX or Windows environment. The AWS Elastic Beanstalk command line interface also works correctly on Mac OS X (which resembles the Linux and UNIX command environment), but no specific Mac OS X instructions are included in this guide.

As a convention, all command line text is prefixed with a generic **`PROMPT>`** command line prompt. The actual command line prompt on your machine is likely to be different. We also use **`$`** to indicate a Linux/UNIX-specific command and **`C:\>`** for a Windows-specific command. The example output resulting from the command is shown immediately thereafter without any prefix.

Important

The command line tools used in this guide require Ruby version 1.8.7+ or Ruby version 1.9.2+ to run. To view and download Ruby clients for a range of platforms, including Linux/UNIX and Windows, go to http://www.ruby-lang.org/en/.

Getting the Command Line Tools

The command line tools are available as a .zip file on the AWS Sample Code & Libraries website. These tools are written in Ruby, and include shell scripts for Windows 2000, Windows XP, Windows Vista, Windows 7, Linux/UNIX, and Mac OS X. The .zip file is self-contained and no installation is required; simply download the .zip file and unzip it to a directory on your local machine. You can find the tools in the **api** directory.

Providing Credentials for the Command Line Interface

The command line interface requires the Access Key ID and Secret Access Key provided with your AWS account.

You need to create a file containing your Access Key ID and Secret Access Key. The contents of the file should look like this:

```
AWSAccessKeyId=Write your AWS access ID
```

```
AWSSecretKey=Write your AWS secret key
```

Important

On UNIX, limit permissions to the owner of the credential file:

```
$ chmod 600 <the file created above>
```

With the credentials file set up, you'll need to set the AWS_CREDENTIAL_FILE environment variable so that the AWS Elastic Beanstalk CLI tools can find your information.

To set the AWS_CREDENTIAL_FILE environment variable

- Set the environment variable using the following command:

On Linux and UNIX	On Windows
`$ export AWS_CREDENTIAL_FILE=<the file created above>`	`C:\> set AWS_CREDENTIAL_FILE=<the file created above>`

Set the Service Endpoint URL

By default, the AWS Elastic Beanstalk uses the US-East (Northern Virginia) Region (us-east-1) with the elasticbeanstalk.us-east-1.amazonaws.com service endpoint URL. This section describes how to specify a different region by setting the service endpoint URL. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference.

To set the service endpoint URL

- Set the environment variable using the following command:

On Linux and UNIX	On Windows
`$ export ELASTICBEANSTALK_URL=<service_endpoint>`	`C:\> set ELASTICBEANSTALK_URL=<service_endpoint>`

Common Options

The command line operations accept the set of optional parameters described in the following table.

Option	Description
`--help` `-h`	Displays help text for the command. You can also use `help` *`commandname`*. This option applies to eb and the original command line interface. Default: off

Option	Description
`--show-json` `-j`	Displays the raw json response. This option applies only to the original command line interface. Default: off

Option Values

This section covers the possible option values that can be specified in the options file that is passed in as the `options-file` parameter. This options file can be passed in with the following command line operations:

- elastic-beanstalk-create-configuration-template (p. 291)
- elastic-beanstalk-create-environment (p. 294)
- elastic-beanstalk-describe-configuration-options (p. 307)
- elastic-beanstalk-update-configuration-template (p. 330)
- elastic-beanstalk-update-environment (p. 332)
- elastic-beanstalk-validate-configuration-settings (p. 335)

General Option Values

Name	Description	Default	Options
Namespace: `aws:autoscaling:asg`			
Availability Zones	Availability Zones are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low-latency network connectivity to other Availability Zones in the same Region. Choose the number of Availability Zones you want to launch your instances in.	Any 1	Any 1 Any 2
Cooldown	Cooldown periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible.	360	0 to 10000

Name	Description	Default	Options
Custom Availability Zones	Define which Availability Zones you want to launch your instances in.	n/a	us-east-1a us-east-1b us-east-1c us-east-1d us-east-1e
MinSize	Minimum number of instances you want in your Auto Scaling group.	1	1 to 10000
MaxSize	Maximum number of instances you want in your Auto Scaling group.	4	1 to 10000
Namespace: `aws:autoscaling:launchconfiguration`			
EC2KeyName	A key pair enables you to securely log into your Amazon EC2 instance.	n/a	n/a
ImageId	You can override the default Amazon Machine Image (AMI) by specifying your own custom AMI ID.	Example: ami-cbab67a2	n/a
InstanceType	Choose from a number of different instance types to meet your computing needs. Each instance provides a predictable amount of dedicated compute capacity.	32-bit: t1.micro 64-bit: t1.micro	32-bit: t1.micro m1.small c1.medium 64-bit: t1.micro m1.small c1.medium m1.medium m1.large m1.xlarge c1.xlarge m2.xlarge m2.2xlarge m2.4xlarge

Name	Description	Default	Options
MonitoringInterval	Interval at which you want Amazon CloudWatch metrics returned.	5 minutes	1 minute 5 minute
SecurityGroups	Defines firewall rules for your instances.	elasticbeanstalk-default	n/a
Namespace: `aws:autoscaling:trigger`			
BreachDuration	Amount of time a metric can be beyond its defined limit (as specified in the Upper Threshold and Lower Threshold) before the trigger fires.	5	1 to 600
LowerBreachScaleIncrement	How many Amazon EC2 instances to remove when performing a scaling activity.	-1	n/a
LowerThreshold	If the measurement falls below this number for the breach duration, a trigger is fired.	2000000	0 to 20000000
MeasureName	Metric used for your auto scaling trigger.	NetworkOut	CPUUtilization NetworkIn NetworkOut DiskWriteOps DiskReadBytes DiskReadOps DiskWriteBytes Latency RequestCount HealthyHostCount UnhealthyHostCount
Period	Specifies how frequently Amazon CloudWatch measures the metrics for your trigger	5	n/a

Name	Description	Default	Options
Statistic	Statistic the trigger should use, such as Average.	Average	Minimum Maximum Sum Average
Unit	Unit for the trigger measurement, such as Bytes.	Bytes	Seconds Percent Bytes Bits Count Bytes/Second Bits/Second Count/Second None
UpperBreachScaleIncrement	How many Amazon EC2 instances to add when performing a scaling activity.	1	n/a
UpperThreshold	If the measurement is higher than this number for the breach duration, a trigger is fired.	6000000	0 to 20000000
Namespace: `aws:elasticbeanstalk:application`			
Application Healthcheck URL	The URL the Elastic Load Balancer uses to query for instance health.	/	A blank string is treated as "/", or specify a string starting with "/".
Namespace: `aws:elasticbeanstalk:monitoring`			
Automatically Terminate Unhealthy Instances	Specify if you want to terminate unhealthy instances automatically.	true	true false
Namespace: `aws:elasticbeanstalk:sns:topics`			
Notification Endpoint	Endpoint where you want to be notified of important events affecting your application.	n/a	n/a

Name	Description	Default	Options
Notification Protocol	Protocol used to send notifications to your endpoint.	email	http https email email-json sqs
Notification Topic ARN	Amazon Resource Name for the topic you subscribed to.	n/a	n/a
Notification Topic Name	Name of the topic you subscribed to.	n/a	n/a
Namespace: `aws:elb:loadbalancer:healthcheck`			
HealthyThreshold	Consecutive successful URL probes before Elastic Load Balancing changes the instance health status.	3	2 to 10
Timeout	Number of seconds Elastic Load Balancing will wait for a response before it considers the instance non-responsive.	5	2 to 60
UnhealthyThreshold	Consecutive unsuccessful URL probes before Elastic Load Balancing changes the instance health status.	5	2 to 10
Namespace: `aws:elb:loadbalancer`			
Interval	Define the interval at which Elastic Load Balancing will check the health of your application's Amazon EC2 instances.	30	5 to 300
LoadBalancerHTTPPort	Port that load balancer listens for requests over HTTP.	80	OFF 80
LoadBalancerHTTPSPort	Port that load balancer listens for requests over HTTPS.	OFF	OFF 443 8443

Name	Description	Default	Options
SSLCertificateId	Amazon Resource Name (ARN) for the SSL certificate you've uploaded for AWS Access and Identity Management.	n/a	n/a
Namespace: `aws:elb:policies`			
Stickiness Cookie Expiration	Duration of validity for each cookie.	0	0 to 1000000
Stickiness Policy	Binds a user's session to a specific server instance so that all requests coming from the user during the session will be sent to the same server instance.	false	true false

Java Container Options

Name	Description	Default	Options
Namespace: `aws:elasticbeanstalk:application:environment`			
AWS_SECRET_KEY	You Amazon Access Secret Key.	n/a	n/a
AWS_ACCESS_KEY_ID	Your Amazon Access Key ID.	n/a	n/a
JDBC_CONNECTION_STRING	Connection string to an external database.	n/a	n/a
PARAM1 – PARAM5	System properties passed in to the JVM at startup. You can use any number of parameters you want and you can specify any name you want.	n/a	n/a
Namespace: `aws:elasticbeanstalk:container:tomcat:jvmoptions`			
JVM Options	Pass command line options to the JVM at startup.	n/a	n/a
Xmx	Maximum JVM heap sizes.	256m	n/a
XX:MaxPermSize	Section of the JVM heap that is used to store class definitions and associated metadata.	64m	n/a
Xms	Initial JVM heap sizes.	256m	n/a
Namespace: `aws:elasticbeanstalk:hostmanager`			

Name	Description	Default	Options
LogPublicationControl	Copy the log files for your application's Amazon EC2 instances to the Amazon S3 bucket associated with your application on an hourly basis.	false	true false

Note

You can extend the number of parameters and specify the parameter names in the `aws:elasticbeanstalk:container:tomcat:jvmoptions` and `aws:elasticbeanstalk:application:environment` namespaces.

.NET Container Options

Name	Description	Default	Options
Namespace: `aws:elasticbeanstalk:application:environment`			
AWS_SECRET_KEY	You Amazon Access Secret Key.	n/a	n/a
AWS_ACCESS_KEY_ID	Your Amazon Access Key ID.	n/a	n/a
PARAM1 – PARAM5	Pass in key-value pairs as environment variables.	n/a	n/a
Namespace: `aws:elasticbeanstalk:container:net:apppool`			
Target Runtime	You can choose the version of .NET Framework for your application.	4.0	2.0 4.0
Enable 32-bit Applications	Enable 32-bit applications.	false	true false
Namespace: `aws:elasticbeanstalk:hostmanager`			
LogPublicationControl	Copy the log files for your application's Amazon EC2 instances to the Amazon S3 bucket associated with your application on an hourly basis.	false	true false

Note

You can extend the number of parameters and specify the parameter names in the `aws:elasticbeanstalk:container:net:apppool` and `aws:elasticbeanstalk:application:environment` namespaces.

PHP Container Options

Name	Description	Default	Options
Namespace: `aws:elasticbeanstalk:application:environment`			
AWS_SECRET_KEY	Your Amazon Access Secret Key.	n/a	n/a
AWS_ACCESS_KEY_ID	Your Amazon Access Key ID.	n/a	n/a
PARAM1 – PARAM5	Pass in key-value pairs as environment variables.	n/a	n/a
Namespace: `aws:elasticbeanstalk:container:php:phpini`			
document_root	Specify the child directory of your project that is treated as the public-facing web root.	/	A blank string is treated as "/", or specify a string starting with "/".
memory_limit	Amount of memory allocated to the PHP environment.	128M	n/a
zlib.output_compression	Specifies whether or not PHP should use compression for output.	false	true false
allow_url_fopen	Specifies if PHP's file functions are allowed to retrieve data from remote locations, such as websites or FTP servers.	true	true false
display_errors	Specifies if error messages should be part of the output.	Off	On Off stderr
max_execution_time	Sets the maximum time, in seconds, a script is allowed to run before it is terminated by the environment.	Off	On Off stderr
Namespace: `aws:elasticbeanstalk:hostmanager`			
LogPublicationControl	Copy the log files for your application's Amazon EC2 instances to the Amazon S3 bucket associated with your application on an hourly basis.	false	true false

Note

You can extend the number of parameters and specify the parameter names in the `aws:elasticbeanstalk:container:php:phpini` and `aws:elasticbeanstalk:application:environment` namespaces.

Python Container Options

Name	Description	Default	Options
Namespace: `aws:elasticbeanstalk:container:python:environment`			
AWS_SECRET_KEY	Your Amazon Access Secret Key.	n/a	n/a
AWS_ACCESS_KEY_ID	Your Amazon Access Key ID.	n/a	n/a
DJANGO_SETTINGS_MODULE	Specifies which settings file to use.	n/a	n/a
Any arbitrary parameter name, such as "application_stage"	Pass in key-value pairs as environment variables.	n/a	n/a
Namespace: `aws:elasticbeanstalk:container:python`			
WSGIPath	The file that contains the wsgi app. This file must have an "application" callable.	application.py	n/a
NumProcesses	The number of daemon processes that should be started for the process group when running WSGI applications.	1	n/a
NumThreads	The number of threads to be created to handle requests in each daemon process within the process group when running WSGI applications.	15	n/a
Namespace: `aws:elasticbeanstalk:container:python:staticfiles`			
/static/	A mapping of the url to a local directory.	n/a	n/a
Namespace: `aws:elasticbeanstalk:hostmanager`			
LogPublicationControl	Copy the log files for your application's Amazon EC2 instances to the Amazon S3 bucket associated with your application on an hourly basis.	false	true false

Note

You can extend the number of parameters and specify the parameter names in the `aws:elasticbeanstalk:container:python:environment` namespace using a configuration file. For instructions, see Customizing and Configuring a Python Container (p. 103). The parameters will be passed in as environment variables on your Amazon EC2 instances.

Operations

Topics

elastic-beanstalk-check-dns-availability

Description

Checks if the specified CNAME is available.

Syntax

`elastic-beanstalk-check-dns-availability -c [`*`CNAMEPrefix`*`]`

Options

Name	Description	Required

`-c`
`--cname-prefix` *`CNAMEPrefix`*
	The name of the CNAME to check.
	Type: String
	Default: None
		Yes

Output

The command returns a table with the following information:

- **Available—**Shows `true` if the CNAME is available; otherwise shows `false`.
- **FullyQualifiedCNAME—**Shows the fully qualified CNAME if it is available; otherwise shows N/A.

Examples

Checking to Availability of a CNAME

This example shows how to check to see if the CNAME prefix "myapp23" is available.

```
PROMPT> elastic-beanstalk-check-dns-availability -c myapp23
```

elastic-beanstalk-create-application

Description

Creates an application that has one configuration template named `default` and no application versions.

> **Note**
>
> The `default` configuration template is for a 32-bit version of the Amazon Linux operating system running the Tomcat 6 application container.

Syntax

`elastic-beanstalk-create-application -a [`*`name`*`] -d [`*`desc`*`]`

Options

Name	Description	Required

`-a`
`--application-name` *`name`*
	The name of the application.
	Constraint: This name must be unique within your account. If the specified name already exists, the action returns an `InvalidParameterValue` error.
	Type: String
	Default: None
		Yes

`-d`
`--description` *`desc`*
	The description of the application.
	Type: String
	Default: None
		No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application. If no application is found with this name, and `AutoCreateApplication` is `false`, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
- **ConfigurationTemplates—**A list of the configuration templates used to create the application.
- **DateCreated—**The date the application was created.
- **DateUpdated—**The date the application was last updated.
- **Description—**The description of the application.
- **Versions—**The versions of the application.

Examples

Creating an Application

This example shows how to create an application.

```
PROMPT> elastic-beanstalk-create-application -a MySampleApp -d "My description"
```

elastic-beanstalk-create-application-version

Description

Creates an application version for the specified application.

Note

Once you create an application version with a specified Amazon S3 bucket and key location, you cannot change that Amazon S3 location. If you change the Amazon S3 location, you receive an exception when you attempt to launch an environment from the application version.

Syntax

```
elastic-beanstalk-create-application-version -a [name] -l [label] -c -d [desc]
-s [location]
```

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application. If no application is found with this name, and `AutoCreateApplication` is `false`, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-c` `--auto-create`	Determines how the system behaves if the specified application for this version does not already exist: • `true`: Automatically creates the specified application for this release if it does not already exist. • `false`: Throws an `InvalidParameterValue` if the specified application for this release does not already exist. Type: Boolean Valid Values: `true` | `false` Default: `false`	No
`-d` `--description` *desc*	The description of the version. Type: String Default: None Length Constraints: Minimum value of 0. Maximum value of 200.	No

Name	Description	Required

`-l`
`--version-label` *`label`*
	A label identifying this version.
	Type: String
	Default: None
	Constraint: Must be unique per application. If an application version already exists with this label for the specified application, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
	Length Constraints: Minimum value of 1. Maximum value of 100.
		Yes

`-s`
`--source-location` *`location`*
	The name of the Amazon S3 bucket and key that identify the location of the source bundle for this version, in the format `bucketname/key`.
	If data found at the Amazon S3 location exceeds the maximum allowed source bundle size, AWS Elastic Beanstalk returns an `InvalidParameterCombination` error.
	Type: String
	Default: If not specified, AWS Elastic Beanstalk uses a sample application. If only partially specified (for example, a bucket is provided but not the key) or if no data is found at the Amazon S3 location, AWS Elastic Beanstalk returns an `InvalidParameterCombination` error.
		No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application.
- **DateCreated—**The date the application was created.
- **DateUpdated—**The date the application was last updated.
- **Description—**The description of the application.
- **SourceBundle—**The location where the source bundle is located for this version.
- **VersionLabel—**A label uniquely identifying the version for the associated application.

Examples

Creating a Version from a Source Location

This example shows create a version from a source location.

```
PROMPT> elastic-beanstalk-create-application-version -a MySampleApp -d "My
version" -l "TestVersion 1" -s amazonaws.com/sample.war
```

elastic-beanstalk-create-configuration-template

Description

Creates a configuration template. Templates are associated with a specific application and are used to deploy different versions of the application with the same configuration settings.

Syntax

elastic-beanstalk-create-configuration-template -a [*name*] -t [*name*] -E [*id*] -d [*desc*] -s [*stack*] -f [*filename*] -A [*name*] -T [*name*]

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application to associate with this configuration template. If no application is found with this name, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None	Yes
`-t` `--template-name` *name*	The name of the configuration template. If a configuration template already exists with this name, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None Constraint: Must be unique for this application. Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-E` `--environment-id` *id*	The environment ID of the configuration template. Type: String Default: None	No
`-d` `--description` *desc*	The description of the configuration. Type: String Default: None	No

Name	Description	Required

`-s`
`--solution-stack` *`stack`*
	The name of the solution stack used by this configuration. The solution stack specifies the operating system, architecture, and application server for a configuration template. It determines the set of configuration options as well as the possible and default values.
	Use `elastic-beanstalk-list-available-solution-stacks` to obtain a list of available solution stacks.
	A solution stack name or a source configuration parameter must be specified; otherwise, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
	If a solution stack name is not specified and the source configuration parameter is specified, AWS Elastic Beanstalk uses the same solution stack as the source configuration template.
	Type: String
	Length Constraints: Minimum value of 0. Maximum value of 100.
		No

`-f`
`--options-file` *`filename`*
	The name of a JSON file that contains a set of key-value pairs defining configuration options for the configuration template. The new values override the values obtained from the solution stack or the source configuration template.
	Type: String
		No

`-A`
`--source-application-name` *`name`*
	The name of the application to use as the source for this configuration template.
	Type: String
	Default: None
		No

`-T`
`--source-template-name` *`name`*
	The name of the template to use as the source for this configuration template.
	Type: String
	Default: None
		No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with the configuration set.
- **DateCreated—**The date (in UTC time) when this configuration set was created.
- **DateUpdated—**The date (in UTC time) when this configuration set was last modified.
- **DeploymentStatus—**If this configuration set is associated with an environment, the deployment status parameter indicates the deployment status of this configuration set:
  - `null`: This configuration is not associated with a running environment.
  - `pending`: This is a draft configuration that is not deployed to the associated environment but is in the process of deploying.

  - `deployed`: This is the configuration that is currently deployed to the associated running environment.
  - `failed`: This is a draft configuration that failed to successfully deploy.
- **Description—**The description of the configuration set.
- **EnvironmentName—**If not `null`, the name of the environment for this configuration set.
- **OptionSettings—**A list of configuration options and their values in this configuration set.
- **SolutionStackName—**The name of the solution stack this configuration set uses.
- **TemplateNamel—**If not `null`, the name of the configuration template for this configuration set.

Examples

Creating a Basic Configuration Template

This example shows how to create a basic configuration template.

```
PROMPT> elastic-beanstalk-create-configuration-template -a MySampleApp -t mycon
figtemplate
```

Related Operations

- elastic-beanstalk-describe-configuration-options (p. 307)
- elastic-beanstalk-describe-configuration-settings (p. 309)
- elastic-beanstalk-list-available-solution-stacks (p. 317)

elastic-beanstalk-create-environment

Description

Launches an environment for the specified application using the specified configuration.

Syntax

elastic-beanstalk-create-environment -a [*name*] -l [*label*] -e [*name*] [-t [*name*] | -s [*stack*]] -c [*prefix*] -d [*desc*] -f[*filename*] -F [*filename*]

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application that contains the version to be deployed. If no application is found with this name, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-l` `--version-label` *label*	The name of the application version to deploy. If the specified application has no associated application versions, AWS Elastic Beanstalk `UpdateEnvironment` returns an `InvalidParameterValue` error. Default: If not specified, AWS Elastic Beanstalk attempts to launch the most recently created application version. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	No
`-e` `--environment-name` *name*	A unique name for the deployment environment. Used in the application URL. Constraint: Must be from 4 to 23 characters in length. The name can contain only letters, numbers, and hyphens. It cannot start or end with a hyphen. This name must be unique in your account. If the specified name already exists, AWS Elastic Beanstalk returns an `InvalidParameterValue`. Type: String Default: If the CNAME parameter is not specified, the environment name becomes part of the CNAME, and therefore part of the visible URL for your application.	Yes

Name	Description	Required

`-t`
`--template-name` *`name`*
	The name of the configuration template to use in the deployment. If no configuration template is found with this name, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
	Conditional: You must specify either this parameter or a solution stack name, but not both. If you specify both, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. If you do not specify either, AWS Elastic Beanstalk returns a `MissingRequiredParameter`.
	Type: String
	Default: None
	Constraint: Must be unique for this application.
		Conditional

`-s`
`--solution-stack` *`stack`*
	This is the alternative to specifying a configuration name. If specified, AWS Elastic Beanstalk sets the configuration values to the default values associated with the specified solution stack.
	Condition: You must specify either this or a `TemplateName`, but not both. If you specify both, AWS Elastic Beanstalk returns an `InvalidParameterCombination` error. If you do not specify either, AWS Elastic Beanstalk returns `MissingRequiredParameter` error.
	Type: String
	Default: None
		Conditional

`-c`
`--cname-prefix` *`prefix`*
	If specified, the environment attempts to use this value as the prefix for the CNAME. If not specified, the environment uses the environment name.
	Type: String
	Default: None
	Length Constraints: Minimum value of 4. Maximum value of 23.
		No

`-d`
`--description` *`desc`*
	The description of the environment.
	Type: String
	Default: None
		No

`-f`
`--options-file` *`filename`*
	The name of a JSON file that contains a set of key-value pairs defining configuration options for this new environment. These override the values obtained from the solution stack or the configuration template.
	Type: String
		No

`-F`
`--options-to-remove-file` *`value`*
	The name of a JSON file that contains configuration options to remove from the configuration set for this new environment.
	Type: String
	Default: None
		No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with this environment.
- **CNAME—**The URL to the CNAME for this environment.
- **DateCreated—**The date the environment was created.
- **DateUpdated—**The date the environment was last updated.
- **Description—**The description of the environment.
- **EndpointURL—**The URL to the LoadBalancer for this environment.
- **EnvironmentID—**The ID of this environment.
- **EnvironmentName—**The name of this environment.
- **Health—**Describes the health status of the environment. AWS Elastic Beanstalk indicates the failure levels for a running environment:
  - `Red`: Indicates the environment is not responsive. Occurs when three or more consecutive failures occur for an environment.
  - `Yellow`: Indicates that something is wrong. Occurs when two consecutive failures occur for an environment.
  - `Green`: Indicates the environment is healthy and fully functional.
  - `Grey`: Default health for a new environment. The environment is not fully launched and health checks have not started or health checks are suspended during an `UpdateEnvironment` or `RestartEnvironment`request.
- **Resources—**A list of AWS resources used in this environment.
- **SolutionStackName—**The name of the solution stack deployed with this environment.
- **Status—**The current operational status of the environment:
  - `Launching`: Environment is in the process of initial deployment.
  - `Updating`: Environment is in the process of updating its configuration settings or application version.
  - `Ready`: Environment is available to have an action performed on it, such as update or terminate.
  - `Terminating`: Environment is in the shut-down process.
  - `Terminated`: Environment is not running.
- **TemplateName—**The name of the configuration template used to originally launch this environment.
- **VersionLabel—**The application version deployed in this environment.

Examples

Creating an Environment Using a Basic Configuration Template

This example shows how to create an environment using a basic configuration template as well as pass in a file to edit configuration settings and a file to remove configuration settings.
Options.txt

```
[
  {"Namespace": "aws:autoscaling:asg",
   "OptionName": "MinSize",
   "Value": "2"},
  {"Namespace": "aws:autoscaling:asg",
   "OptionName": "MaxSize",
   "Value": "3"}
]
```

Options_remove.txt

```
[
  {"Namespace": "aws:elasticbeanstalk:sns:topics",
   "OptionName": "PARAM4"}
]
```

```
PROMPT> elastic-beanstalk-create-environment -a MySampleApp -t myconfigtemplate
 -e MySampleAppEnv -f options.text -F options_remove.txt
```

elastic-beanstalk-create-storage-location

Description

Creates the Amazon S3 storage location for the account. This location is used to store user log files and is used by the AWS Management Console to upload application versions. You do not need to create this bucket in order to work with AWS Elastic Beanstalk.

Syntax

`elastic-beanstalk-create-storage-location`

Examples

Creating the Storage Location

This example shows how to create a storage location.

```
PROMPT> elastic-beanstalk-create-storage-location
```

This command will output the name of the Amazon S3 bucket created.

elastic-beanstalk-delete-application

Description

Deletes the specified application along with all associated versions and configurations.

Note

You cannot delete an application that has a running environment.

Syntax

`elastic-beanstalk-delete-application -a [`*`name`*`] -f`

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application to delete. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-f` `--force-terminate-env`	Determines if all running environments should be deleted before deleting the application. Type: Boolean Valid Values: `true | false` Default: `false`	No

Output

The command returns the string `Application deleted.`

Examples

Deleting an Application

This example shows how to delete an application.

```
PROMPT> elastic-beanstalk-delete-application -a MySampleApp
```

elastic-beanstalk-delete-application-version

Description

Deletes the specified version from the specified application.

> **Note**
>
> You cannot delete an application version that is associated with a running environment.

Syntax

`elastic-beanstalk-delete-application-version -a [`*`name`*`] -l [`*`label`*`] -d`

Options

Name	Description	Required

`-a`
`--application-name` *`name`*
	The name of the application to delete releases from.
	Type: String
	Default: None
		Yes

`-l` `--version-label`	The label of the version to delete. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-d` `--delete-source-bundle`	Indicates whether to delete the associated source bundle from Amazon S3. `true`: An attempt is made to delete the associated Amazon S3 source bundle specified at time of creation. `false`: No action is taken on the Amazon S3 source bundle specified at time of creation. Type: Boolean Valid Values: `true` | `false` Default: `false`	No

Output

The command returns the string `Application version deleted.`

Examples

Deleting an Application Version

This example shows how to delete an application version.

```
PROMPT> elastic-beanstalk-delete-application-version -a MySampleApp -l MyAppVer
sion
```

Deleting an Application Version and Amazon S3 Source Bundle

This example shows how to delete an application version.

```
PROMPT> elastic-beanstalk-delete-application-version -a MySampleApp -l MyAppVer
sion -d
```

elastic-beanstalk-delete-configuration-template

Description

Deletes the specified configuration template.

Note

When you launch an environment using a configuration template, the environment gets a copy of the template. You can delete or modify the environment's copy of the template without affecting the running environment.

Syntax

`elastic-beanstalk-delete-configuration-template -a [name] -t [name]`

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application to delete the configuration template from. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-t` `--template-name`	The name of the configuration template to delete. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes

Output

The command returns the string `Configuration template deleted.`

Examples

Deleting a Configuration Template

This example shows how to delete a configuration template.

```
PROMPT> elastic-beanstalk-delete-configuration-template -a MySampleApp -t MyCon
figTemplate
```

elastic-beanstalk-delete-environment-configuration

Description

Deletes the draft configuration associated with the running environment.

Note

Updating a running environment with any configuration changes creates a draft configuration set. You can get the draft configuration using `elastic-beanstalk-describe-configuration-settings` while the update is in progress or if the update fails. The deployment status for the draft configuration indicates whether the deployment is in process or has failed. The draft configuration remains in existence until it is deleted with this action.

Syntax

`elastic-beanstalk-delete-environment-configuration -a [`*`name`*`] -e [`*`name`*`]`

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application the environment is associated with. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-e` `--environment-name` *name*	The name of the environment to delete the draft configuration from. Type: String Default: None Length Constraints: Minimum value of 4. Maximum value of 23.	Yes

Output

The command returns the string `Environment configuration deleted.`

Examples

Deleting a Configuration Template

This example shows how to delete a configuration template.

```
PROMPT> elastic-beanstalk-delete-environment-configuration -a MySampleApp -e
MyEnvConfig
```

elastic-beanstalk-describe-application-versions

Description

Returns information about existing application versions.

Syntax

elastic-beanstalk-describe-application-versions -a [*name*] -l [*labels [,label..]*]

Options

Name	Description	Required
-a --application-name *value*	The name of the application. If specified, AWS Elastic Beanstalk restricts the returned descriptions to only include ones that are associated with the specified application. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	No
-l --version-label *labels*	Comma-delimited list of version labels. If specified, restricts the returned descriptions to only include ones that have the specified version labels. Type: String[] Default: None	No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with this release.
- **DateCreated—**The date the application was created.
- **DateUpdated—**The date the application version was last updated.
- **Description—**The description of the application version.
- **SourceBundle—**The location where the source bundle is located for this version.
- **VersionLabel—**A label uniquely identifying the version for the associated application.

Examples

Describing Application Versions

This example shows how to describe all application versions for this account.

```
PROMPT> elastic-beanstalk-describe-application-versions
```

Describing Application Versions for a Specified Application

This example shows how to describe application versions for a specific application.

```
PROMPT> elastic-beanstalk-describe-application-versions -a MyApplication
```

Describing Multiple Application Versions

This example shows how to describe multiple specified application versions.

```
PROMPT> elastic-beanstalk-describe-application-versions -l MyAppVersion1,
MyAppVersion2
```

elastic-beanstalk-describe-applications

Description

Returns descriptions about existing applications.

Syntax

elastic-beanstalk-describe-applications -a [*names [,name..]*]

Options

Name	Description	Required
`-a` `--application-names` *name*	The name of of one or more applications, separated by commas. If specified, AWS Elastic Beanstalk restricts the returned descriptions to only include those with the specified names. Type: String[] Default: None	No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application.
- **ConfigurationTemplates—**A list of the configuration templates used to create the application.
- **DateCreated—**The date the application was created.
- **DateUpdated—**The date the application was last updated.
- **Description—**The description of the application.
- **Versions—**The names of the versions for this application.

Examples

Describing the Applications

This example shows how to describe all applications for this account.

```
PROMPT> elastic-beanstalk-describe-applications
```

Describing a Specific Application

This example shows how to describe a specific application.

```
PROMPT> elastic-beanstalk-describe-applications -a MyApplication
```

elastic-beanstalk-describe-configuration-options

Description

Describes the configuration options that are used in a particular configuration template or environment, or that a specified solution stack defines. The description includes the values, the options, their default values, and an indication of the required action on a running environment if an option value is changed.

Syntax

```
elastic-beanstalk-describe-configuration-options -a [name] -t [name] -e [name]
-s [stack] -f [filename]
```

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application associated with the configuration template or environment. Only needed if you want to describe the configuration options associated with either the configuration template or environment. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	No
`-t` `--template-name` *name*	The name of the configuration template whose configuration options you want to describe. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	No
`-e` `--environment-name` *name*	The name of the environment whose configuration options you want to describe. Type: String Length Constraints: Minimum value of 4. Maximum value of 23.	No
`-s` `--solution-stack` *stack*	The name of the solution stack whose configuration options you want to describe. Type: String Default: None Length Constraints: Minimum value of 0. Maximum value of 100.	No
`-f` `--options-file` *filename*	The name of a JSON file that contains the options you want described. Type: String	No

Output

The command returns a table with the following information:

- **Options—**A list of the configuration options.
- **SolutionStackName—**The name of the `SolutionStack` these configuration options belong to.

Examples

Describing Configuration Options for an Environment

This example shows how to describe configuration options for an environment.

```
PROMPT> elastic-beanstalk-describe-configuration-options -a MySampleApp -t my
configtemplate -e MySampleAppEnv
```

elastic-beanstalk-describe-configuration-settings

Description

Returns a description of the settings for the specified configuration set, that is, either a configuration template or the configuration set associated with a running environment.

When describing the settings for the configuration set associated with a running environment, it is possible to receive two sets of setting descriptions. One is the deployed configuration set, and the other is a draft configuration of an environment that is either in the process of deployment or that failed to deploy.

Syntax

```
elastic-beanstalk-describe-configuration-settings -a [name] [-t [name] | -e
[name]]
```

Options

Name	Description	Required
`-a` `--application-name` *name*	The application name for the environment or configuration template. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-t` `--template-name` *name*	The name of the configuration template to describe. If no configuration template is found with this name, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Conditional: You must specify either this parameter or an environment name, but not both. If you specify both, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. If you do not specify either, AWS Elastic Beanstalk returns a `MissingRequiredParameter` error. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Conditional
`-e` `--environment-name` *name*	The name of the environment to describe. Type: String Default: None Length Constraints: Minimum value of 4. Maximum value of 23.	Conditional

Output

The command returns a table with the following information:

- **ConfigurationSettings—**A list of the configuration settings.

Examples

Describing Configuration Settings for an Environment

This example shows how to describe the configuration options for an environment.

```
PROMPT> elastic-beanstalk-describe-configuration-settings -a MySampleApp -e
MySampleAppEnv
```

Related Operations

- elastic-beanstalk-delete-environment-configuration (p. 303)

elastic-beanstalk-describe-environment-resources

Description

Returns AWS resources for this environment.

Syntax

elastic-beanstalk-describe-environment-resources [-e [*name*] | -E [*id*]]

Options

Name	Description	Required
`-e` `--environment-name` *name*	The name of the environment to retrieve AWS resource usage data. Type: String Default: None Length Constraints: Minimum value of 4. Maximum value of 23.	Conditional
`-E` `--environment-id` *id*	The ID of the environment to retrieve AWS resource usage data. Type: String Default: None	Conditional

Output

The command returns a table with the following information:

- **AutoScalingGroups—**A list of AutoScalingGroups used by this environment.
- **EnvironmentName—**The name of the environment.
- **Instances—**The Amazon EC2 instances used by this environment.
- **LaunchConfigurations—**The Auto Scaling launch configurations in use by this environment.
- **LoadBalancers—**The LoadBalancers in use by this environment.
- **Triggers—**The AutoScaling triggers in use by this environment.

Examples

Describing Environment Resources for an Environment

This example shows how to describe environment resources for an environment.

```
PROMPT> elastic-beanstalk-describe-environment-resources -e MySampleAppEnv
```

elastic-beanstalk-describe-environments

Description

Returns descriptions for existing environments.

Syntax

elastic-beanstalk-describe-environments -e [*names [,name...]*] -E [*ids [,id...]*] -a [*name*] -l [*label*] -d -D [*timestamp*]

Options

Name	Description	Required

`-e`
`--environment-names` *`names`*
	A list of environment names.
	Type: String[]
	Default: None
		No

`-E`
`--environment-ids` *`ids`*
	A list of environment IDs.
	Type: String[]
	Default: None
		No

`-a`
`--application-name` *`name`*
	A list of descriptions associated with the application.
	Type: String
	Default: None
	Length Constraints: Minimum value of 1. Maximum value of 100.
		No

`-l`
`--version-label` *`label`*
	A list of descriptions associated with the application version.
	Type: String
	Default: None
	Length Constraints: Minimum value of 1. Maximum value of 100.
		No

`-d` `--include-deleted`	Indicates whether to include deleted environments. `true`: Environments that have been deleted after `--include-deleted-back-to` are displayed. `false`: Do not include deleted environments. Type: Boolean Default: `true`	No

`-D`
`--include-deleted-back-to` *`timestamp`*
	If --include-deleted is set to `true`, then a list of environments that were deleted after this date are displayed.
	Type: Date Time
	Default: None
		No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with this environment.
- **CNAME—**The URL to the CNAME for this environment.
- **DateCreated—**The date the environment was created.
- **DateUpdated—**The date the environment was last updated.
- **Description—**The description of the environment.
- **EndpointURL—**The URL to the LoadBalancer for this environment.
- **EnvironmentID—**The ID of this environment.
- **EnvironmentName—**The name of this environment.
- **Health—**Describes the health status of the environment. AWS Elastic Beanstalk indicates the failure levels for a running environment:
  - `Red`: Indicates the environment is not responsive. Occurs when three or more consecutive failures occur for an environment.
  - `Yellow`: Indicates that something is wrong. Occurs when two consecutive failures occur for an environment.
  - `Green`: Indicates the environment is healthy and fully functional.
  - `Grey`: Default health for a new environment. The environment is not fully launched and health checks have not started or health checks are suspended during an `UpdateEnvironment` or `RestartEnvironment`request.
- **Resources—**A list of AWS resources used in this environment.
- **SolutionStackName—**The name of the `SolutionStack` deployed with this environment.
- **Status—**The current operational status of the environment:
  - `Launching`: Environment is in the process of initial deployment.
  - `Updating`: Environment is in the process of updating its configuration settings or application version.
  - `Ready`: Environment is available to have an action performed on it, such as update or terminate.
  - `Terminating`: Environment is in the shut-down process.
  - `Terminated`: Environment is not running.
- **TemplateName—**The name of the configuration template used to originally launch this environment.
- **VersionLabel—**The application version deployed in this environment.

Examples

Describing Environments

This example shows how to describe existing environments.

```
PROMPT> elastic-beanstalk-describe-environments
```

elastic-beanstalk-describe-events

Description

Returns a list of event descriptions matching criteria up to the last 6 weeks.

Note

This action returns the most recent 1,000 events from the specified `NextToken`.

Syntax

```
elastic-beanstalk-describe-events -a [name] -e [name] -E [id] -l [label] -L
[timestamp] -m [count] -n [token] -r [id] -s [level] -S [timestamp] -t [name]
```

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	No
`-e` `--environment-name` *name*	The name of the environment. Type: String Default: None Length Constraints: Minimum value of 4. Maximum value of 23.	No
`-E` `--environment-id` *id*	The ID of the environment. Type: String Default: None	No
`-l` `--version-label` *label*	The application version. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	No
`-L` `--end-time` *timestamp*	If specified, a list of events that occurred up to but not including the specified time is returned. Type: Date Time Default: None	No
`-m` `--max-records` *count*	Specifies the maximum number of events that can be returned, beginning with the most recent event. Type: Integer Default: None	No

Name	Description	Required

`-n`
`--next-token` *`token`*
	Pagination token. Used to return the next batch of results.
	Type: String
	Default: None
		No

`-r`
`--request-id` *`id`*
	The request ID.
	Type: String
	Default: None
		No

`-s`
`--severity` *`level`*
	If specified, a list of events with the specified severity level or higher is returned.
	Type: String
	Valid Values: `TRACE` | `DEBUG` | `INFO` | `WARN` | `ERROR` | `FATAL`
	Default: None
		No

`-S`
`--start-time` *`timestamp`*
	If specified, a list of events that occurred after the specified time is returned.
	Type: Date Time
	Default: None
		No

`-t`
`--template-name` *`name`*
	The name of the configuration template.
	Type: String
	Default: None
	Length Constraints: Minimum value of 1. Maximum value of 100.
		No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with the event.
- **EnvironmentName—**The name of the environment associated with the event.
- **EventDate—**The date of the event.
- **Message—**The event's message.
- **RequestID—**The web service request ID for the activity of this event.
- **Severity—**The severity level of the event.
- **TemplateName—**The name of the configuration associated with this event.
- **VersionLabel—**The release label for the application version associated with this event.

Examples

Describing Events for an Environment with a Security Level

This example shows how to describe events that have a severity level of WARN or higher for an environment.

```
PROMPT> elastic-beanstalk-describe-events -e MySampleAppEnv -s WARN
```

elastic-beanstalk-list-available-solution-stacks

Description

Returns a list of available solution stack names.

Syntax

`elastic-beanstalk-list-available-solution-stacks`

Output

The command returns of available solution stack names.

Examples

Listing the Available Solution Stacks

This example shows how to get the list of available solution stacks.

```
PROMPT> elastic-beanstalk-list-available-solution-stacks
```

elastic-beanstalk-rebuild-environment

Description

Deletes and recreates all of the AWS resources (for example: the Auto Scaling group, LoadBalancer, etc.) for a specified environment and forces a restart.

Syntax

`elastic-beanstalk-rebuild-environment [-e [`*`name`*`] | -E [`*`id`*`]]`

Options

Name	Description	Required

`-e`
`--environment-name` *`name`*
	A name of the environment to rebuild.
	Type: String
	Default: None
	Length Constraints: Minimum value of 4. Maximum value of 23.
		Conditional

`-E`
`--environment-id` *`id`*
	The ID of the environment to rebuild.
	Type: String
	Default: None
		Conditional

Output

The command outputs `Rebuilding environment`.

Examples

Rebuilding an Environment

This example shows how to rebuild an environment.

```
PROMPT> elastic-beanstalk-rebuild-environment -e MySampleAppEnv
```

elastic-beanstalk-request-environment-info

Description

Initiates a request to compile the specified type of information of the deployed environment.

Setting the InfoType to `tail` compiles the last lines from the application server log files of every Amazon EC2 instance in your environment. Use RetrieveEnvironmentInfo to access the compiled information.

Syntax

`elastic-beanstalk-request-environment-info [-e [`*`name`*`] | -E [`*`id`*`]] -i [`*`type`*`]`

Options

Name	Description	Required

`-e`
`--environment-name` *`name`*
	The name of the environment of the requested data.
	Type: String
	Default: None
	Length Constraints: Minimum value of 4. Maximum value of 23.
		Conditional

`-E`
`--environment-id` *`id`*
	The ID of the environment of the requested data.
	Type: String
	Default: None
		Conditional

`-i`
`--info-type` *`type`*
	The type of information to request.
	Type: String
	Valid Values: `tail`
	Default: None
		Yes

Examples

Requesting Environment Information

This example shows how to request environment information.

```
PROMPT> elastic-beanstalk-request-environment-info -e MySampleAppEnv -i tail
```

Related Operations

- elastic-beanstalk-retrieve-environment-info (p. 321)

elastic-beanstalk-restart-app-server

Description

Causes the environment to restart the application container server running on each Amazon EC2 instance.

Syntax

`elastic-beanstalk-restart-app-server [-e [`*`name`*`] | -E [`*`id`*`]]`

Options

Name	Description	Required

`-e`
`--environment-name` *`name`*
	The name of the environment to restart the server for.
	Type: String
	Default: None
	Length Constraints: Minimum value of 4. Maximum value of 23.
		Conditional

`-E`
`--environment-id` *`id`*
	The ID of the environment to restart the server for.
	Type: String
	Default: None
		Conditional

Examples

Restarting the Application Server

This example shows how to restart the application server.

```
PROMPT> elastic-beanstalk-restart-app-server -e MySampleAppEnv
```

elastic-beanstalk-retrieve-environment-info

Description

Retrieves the compiled information from a RequestEnvironmentInfo request.

Syntax

elastic-beanstalk-retrieve-environment-info [-e [*name*] | -E [*id*]] -i [*type*]

Options

Name	Description	Required
`-e` `--environment-name` *name*	The name of the data's environment. If no environments are found, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None Length Constraints: Minimum value of 4. Maximum value of 23.	Conditional
`-E` `--environment-id` *id*	The ID of the data's environment. The name of the data's environment. If no environments are found, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None	Conditional
`-i` `--info-type` *type*	The type of information to retrieve. Type: String Valid Values: tail Default: None	Yes

Output

The command returns a table with the following information:

- **EC2InstanceId—**The Amazon EC2 instance ID for this information.
- **InfoType—**The type of information retrieved.
- **Message—**The retrieved information.
- **SampleTimestamp—**The time stamp when this information was retrieved.

Examples

Retrieving Environment Information

This example shows how to retrieve environment information.

```
PROMPT> elastic-beanstalk-retrieve-environment-info -e MySampleAppEnv -i tail
```

Related Operations

- elastic-beanstalk-request-environment-info (p. 319)

elastic-beanstalk-swap-environment-cnames

Description

Swaps the CNAMEs of two environments.

Syntax

elastic-beanstalk-swap-environment-cnames [-s [*name*] | -S [*desc*]] [-d [*desc*] | -D [*desc*]]

Options

Name	Description	Required

`-s`
`--source-environment-name` *`name`*
	The name of the source environment.
	Type: String
	Default: None
		Conditional

`-S`
`--source-environment-id` *`id`*
	The ID of the source environment.
	Type: String
	Default: None
		Conditional

`-d`
`--destination-environment-name` *`name`*
	The name of the destination environment.
	Type: String
	Default: None
		Conditional

`-D`
`--destination-environment-id` *`id`*
	The ID of the destination environment.
	Type: String
	Default: None
		Conditional

Examples

Swapping Environment CNAMEs

This example shows how to swap the CNAME for two environments.

```
PROMPT> elastic-beanstalk-swap-environment-cnames -s MySampleAppEnv -d
MySampleAppEnv2
```

elastic-beanstalk-terminate-environment

Description

Terminates the specified environment.

Syntax

`elastic-beanstalk-terminate-environment [-e [name] | -E [id]] -t`

Options

Name	Description	Required

`-e`
`--environment-name` *`name`*
	The name of the environment to terminate.
	Type: String
	Default: None
	Length Constraints: Minimum value of 4. Maximum value of 23.
		Conditional

`-E`
`--environment-id` *`id`*
	The ID of the environment to terminate.
	Type: String
	Default: None
		Conditional

`-t` `--terminate-resources`	Indicates whether the associated AWS resources should shut down when the environment is terminated: • `true`: The specified environment as well as the associated AWS resources, such as Auto Scaling group and LoadBalancer, are terminated. • `false`: AWS Elastic Beanstalk resource management is removed from the environment, but the AWS resources continue to operate. Type: Boolean Valid Values: `true` | `false` Default: true	No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with this environment.
- **CNAME—**The URL to the CNAME for this environment.
- **DateCreated—**The date the environment was created.
- **DateUpdated—**The date the environment was last updated.
- **Description—**The description of the environment.
- **EndpointURL—**The URL to the LoadBalancer for this environment.
- **EnvironmentID—**The ID of this environment.

- **EnvironmentName—**The name of this environment.
- **Health—**Describes the health status of the environment. AWS Elastic Beanstalk indicates the failure levels for a running environment:
  - `Red`: Indicates the environment is not responsive. Occurs when three or more consecutive failures occur for an environment.
  - `Yellow`: Indicates that something is wrong. Occurs when two consecutive failures occur for an environment.
  - `Green`: Indicates the environment is healthy and fully functional.
  - `Grey`: Default health for a new environment. The environment is not fully launched and health checks have not started or health checks are suspended during an `UpdateEnvironment` or `RestartEnvironment`request.
- **Resources—**A list of AWS resources used in this environment.
- **SolutionStackName—**The name of the `SolutionStack` deployed with this environment.
- **Status—**The current operational status of the environment:
  - `Launching`: Environment is in the process of initial deployment.
  - `Updating`: Environment is in the process of updating its configuration settings or application version.
  - `Ready`: Environment is available to have an action performed on it, such as update or terminate.
  - `Terminating`: Environment is in the shut-down process.
  - `Terminated`: Environment is not running.
- **TemplateName—**The name of the configuration template used to originally launch this environment.
- **VersionLabel—**The application version deployed in this environment.

Examples

Terminating an Environment

This example shows how to terminate an environment.

```
PROMPT> elastic-beanstalk-terminate-environment -e MySampleAppEnv
```

elastic-beanstalk-update-application

Description

Updates the specified application to have the specified properties.

Note

If a property (for example, `description`) is not provided, the value remains unchanged. To clear these properties, specify an empty string.

Syntax

`elastic-beanstalk-update-application -a [`*`name`*`] -d [`*`desc`*`]`

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application to update. If no such application is found, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-d` `--description` *desc*	A new description for the application. Type: String Default: If not specified, AWS Elastic Beanstalk does not update the description. Length Constraints: Minimum value of 0. Maximum value of 200.	No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application.
- **ConfigurationTemplate—**The names of the configuration templates associated with this application.
- **DateCreated—**The date the environment was created.
- **DateUpdated—**The date the environment was last updated.
- **Description—**The description of the environment.
- **Versions—**The names of the versions for this application.

Examples

Updating an Application

This example shows how to update an application.

```
PROMPT> elastic-beanstalk-update-application -a MySampleApp -d "My new descrip
tion"
```

elastic-beanstalk-update-application-version

Description

Updates the specified application version to have the specified properties.

Note

If a property (for example, `description`) is not provided, the value remains unchanged. To clear these properties, specify an empty string.

Syntax

`elastic-beanstalk-update-application-version -a [`*`name`*`] -l [`*`label`*`] -d [`*`desc`*`]`

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application associated with this version. If no such application is found, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-l` `--version-label`	The name of the version to update. If no application version is found with this label, AWS Elastic Beanstalk returns an `InvalidParaemterValue` error. Type: String Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-d` `--description`	A new description for the release. Type: String Default: If not specified, AWS Elastic Beanstalk does not update the description. Length Constraints: Minimum value of 0. Maximum value of 200.	No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with this release.
- **DateCreated—**The creation date of the application version.
- **DateUpdated—**The last modified date of the application version.
- **Description—**The description of this application version.
- **SourceBundle—**The location where the source bundle is located for this version.
- **VersionLabel—**A label identifying the version for the associated application.

Examples

Updating an Application Version

This example shows how to update an application version.

```
PROMPT> elastic-beanstalk-update-application-version -a MySampleApp -d "My new
 version" -l "TestVersion 1"
```

elastic-beanstalk-update-configuration-template

Description

Updates the specified configuration template to have the specified properties or configuration option values.

Note

If a property (for example, `ApplicationName`) is not provided, its value remains unchanged. To clear such properties, specify an empty string.

Syntax

```
elastic-beanstalk-update-configuration-template -a [name] -t [name] -d [desc]
-f [filename] -F [filename]
```

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application associated with the configuration template to update. If no application is found with this name, AWS Elastic Beanstalk returns an `InvalidParameterValue` error. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-t` `--template-name` *name*	The name of the configuration template to update. If no configuration template is found with this name, UpdateConfigurationTemplate returns an `InvalidParameterValue` error. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-d` `--description` *desc*	A new description for the configuration. Type: String Default: None Length Constraints: Minimum value of 0. Maximum value of 200.	No
`-f` `--options-file` *filename*	The name of a JSON file that contains option settings to update with the new specified option value. Type: String	No
`-F` `--options-to-remove-file` *value*	The name of a JSON file that contains configuration options to remove. Type: String Default: None	No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with this configuration set.
- **DateCreated—**The date (in UTC time) when this configuration set was created.
- **DateUpdated—**The date (in UTC time) when this configuration set was last modified.
- **DeploymentStatus—**If this configuration set is associated with an environment, the *DeploymentStatus* parameter indicates the deployment status of this configuration set:
  - `null`: This configuration is not associated with a running environment.
  - `pending`: This is a draft configuration that is not deployed to the associated environment but is in the process of deploying.
  - `deployed`: This is the configuration that is currently deployed to the associated running environment.
  - `failed`: This is a draft configuration that failed to successfully deploy.
- **Description—**The description of the configuration set.
- **EnvironmentName—**If not null, the name of the environment for this configuration set.
- **OptionSettings—**A list of configuration options and their values in this configuration set.
- **SolutionStackName—**The name of the solution stack this configuration set uses.
- **TemplateName—**If not null, the name of the configuration template for this configuration set.

Examples

Updating a Configuration Template

This example shows how to update a configuration template.

```
PROMPT> elastic-beanstalk-update-configuration-template -a MySampleApp -t mycon
figtemplate -d "My updated configuration template" -f "Options.txt"
```

Options.txt

```
[
  {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "my_custom_param_1",
   "Value": "firstvalue"},
  {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "my_custom_param_2",
   "Value": "secondvalue"}
]
```

Related Operations

- elastic-beanstalk-describe-configuration-options (p. 307)

elastic-beanstalk-update-environment

Description

Updates the environment description, deploys a new application version, updates the configuration settings to an entirely new configuration template, or updates select configuration option values in the running environment.

Attempting to update both the release and configuration is not allowed and AWS Elastic Beanstalk returns an `InvalidParameterCombination` error.

When updating the configuration settings to a new template or individual settings, a draft configuration is created and `DescribeConfigurationSettings` for this environment returns two setting descriptions with different `DeploymentStatus` values.

Syntax

`elastic-beanstalk-update-environment [-e [`*`name`*`] | -E [`*`id`*`]] -l [`*`label`*`] -t [`*`name`*`] -d [`*`desc`*`] -f [`*`filename`*`] -F [`*`filename`*`]`

Options

Name	Description	Required

`-e`
`--environment-name` *`name`*
	The name of the environment to update. If no environment with this name exists, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
	Type: String
	Default: None
	Length Constraints: Minimum value of 4. Maximum value of 23.
		Conditional

`-E`
`--environment-id` *`id`*
	The ID of the environment to update. If no environment with this ID exists, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
	Type: String
	Default: None
		Conditional

`-l`
`--version-label` *`label`*
	If this parameter is specified, AWS Elastic Beanstalk deploys the named application version to the environment. If no such application version is found, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
	Type: String
	Default: None
	Length Constraints: Minimum value of 1. Maximum value of 100.
		No

Name	Description	Required

`-t`
`--template-name` *`name`*
	If this parameter is specified, AWS Elastic Beanstalk deploys this configuration template to the environment. If no such configuration template is found, AWS Elastic Beanstalk returns an `InvalidParameterValue` error.
	Type: String
	Default: None
	Length Constraints: Minimum value of 1. Maximum value of 100.
		No

`-d`
`--description` *`desc`*
	If this parameter is specified, AWS Elastic Beanstalk updates the description of this environment.
	Type: String
	Default: None
	Length Constraints: Minimum value of 0. Maximum value of 200.
		No

`-f`
`--options-file` *`filename`*
	A file containing option settings to update. If specified, AWS Elastic Beanstalk updates the configuration set associated with the running environment and sets the specified configuration options to the requested values.
	Type: String
	Default: None
		No

`-F`
`--options-to-remove-file` *`filename`*
	A file containing options settings to remove. If specified, AWS Elastic Beanstalk removes the option settings from the configuration set associated with the running environment.
	Type: String
	Default: None
		No

Output

The command returns a table with the following information:

- **ApplicationName—**The name of the application associated with this environment.
- **CNAME—**The URL to the CNAME for this environment.
- **DateCreated—**The date the environment was created.
- **DateUpdated—**The date the environment was last updated.
- **Description—**The description of the environment.
- **EndpointURL—**The URL to the LoadBalancer for this environment.
- **EnvironmentID—**The ID of this environment.
- **EnvironmentName—**The name of this environment.
- **Health—**Describes the health status of the environment. AWS Elastic Beanstalk indicates the failure levels for a running environment:
  - `Red`: Indicates the environment is not responsive. Occurs when three or more consecutive failures occur for an environment.

  - `Yellow`: Indicates that something is wrong. Occurs when two consecutive failures occur for an environment.
  - `Green`: Indicates the environment is healthy and fully functional.
  - `Grey`: Default health for a new environment. The environment is not fully launched and health checks have not started or health checks are suspended during an `UpdateEnvironment` or `RestartEnvironment`request.
- **Resources—**A list of AWS resources used in this environment.
- **SolutionStackName—**The name of the `SolutionStack` deployed with this environment.
- **Status—**The current operational status of the environment:
  - `Launching`: Environment is in the process of initial deployment.
  - `Updating`: Environment is in the process of updating its configuration settings or application version.
  - `Ready`: Environment is available to have an action performed on it, such as update or terminate.
  - `Terminating`: Environment is in the shut-down process.
  - `Terminated`: Environment is not running.
- **TemplateName—**The name of the configuration template used to originally launch this environment.
- **VersionLabel—**The application version deployed in this environment.

Examples

Updating an Existing Environment

This example shows how to update an existing environment.

```
PROMPT> elastic-beanstalk-update-environment -e MySampleAppEnv -f "Options.txt"
```

Options.txt

```
[
  {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "my_custom_param_1",
   "Value": "firstvalue"},
  {"Namespace": "aws:elasticbeanstalk:application:environment",
   "OptionName": "my_custom_param_2",
   "Value": "secondvalue"}
]
```

elastic-beanstalk-validate-configuration-settings

Description

Takes a set of configuration settings and either a configuration template or environment, and determines whether those values are valid.

This action returns a list of messages indicating any errors or warnings associated with the selection of option values.

Syntax

```
elastic-beanstalk-validate-configuration-settings -a [name] -t [name] -e [name]
-f [filename]
```

Options

Name	Description	Required
`-a` `--application-name` *name*	The name of the application that the configuration template or environment belongs to. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	Yes
`-t` `--template-name` *name*	The name of the configuration template to validate the settings against. Condition: You cannot specify both this and the environment name. Type: String Default: None Length Constraints: Minimum value of 1. Maximum value of 100.	No
`-e` `--environment-name` *name*	The name of the environment to validate the settings against. Type: String Default: None Length Constraints: Minimum value of 4. Maximum value of 23.	No
`-f` `--options-file` *filename*	The name of a JSON file that contains a list of options and desired values to evaluate. Type: String	Yes

Output

The command returns a table with the following information:

- **Message—**A message describing the error or warning.
- **Namespace**
- **OptionName**

- **Severity—**An indication of the severity of this message:
  - `error`: This message indicates that this is not a valid settings for an option.
  - `warning`: This message provides information you should take into account.

Examples

Validating Configuration Settings for an Environment

This example shows how to validate the configuration settings for an environment.

```
PROMPT> elastic-beanstalk-validate-configuration-settings -a MySampleApp -e
MySampleAppEnv -f MyOptionSettingsFile.json
```

AWS DevTools

Topics

This section provides step by step instructions for deploying your PHP web application to AWS Elastic Beanstalk using AWS DevTools, a Git client extension. For more information on prerequisites and installation instructions for AWS DevTools, see Get Set Up (p. 336). You can also use the AWS Management Console, CLIs, or APIs to upload your PHP files using a .zip file. For more information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

After you deploy your AWS Elastic Beanstalk application, you can use the AWS Management Console, CLIs, or the APIs to manage your AWS Elastic Beanstalk environment. For more information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Get Set Up

AWS DevTools is a Git client extension that enables you to deploy applications to AWS Elastic Beanstalk quickly. This section describes the prerequisites for running AWS DevTools, where to get it, and how to set it up. For an example of how to configure your Git environment and deploy your AWS Elastic Beanstalk application using AWS DevTools, see AWS DevTools (p. 336).

Linux/Unix and Mac

The AWS DevTools works on Linux, Unix, and Mac OS X operating systems. This document assumes you can work in one of these environments. You can set up AWS DevTools in a few basic steps:

- Install the prerequisite software
- Download AWS DevTools
- Create a Git repository directory
- Run the AWS Devtools setup script

To set up AWS DevTools on a Linux/Unix or Mac computer

1. Install the following software on to your local computer:

- Git. To download Git, go to http://git-scm.com/. Make sure you have at least version 1.6.6 or later.

  To verify if Git is already installed, type the following command at the command prompt:

  ```
  git
  ```

  If Git is installed, you should get a list of the most commonly used commands.
- Ruby version 1.8.7 or later. To view and download Ruby clients, go to http://www.ruby-lang.org/en/.

  To verify if Ruby is already installed, type the following command at the command prompt:

  ```
  ruby -v
  ```

  If Ruby responds, and if it shows a version number at or above 1.8.7, then you have the correct version installed.

2. Download AWS DevTools, which is part of the command line interface package, at the AWS Sample Code & Libraries website. The .zip file is self-contained, and no installation is required; simply download the .zip file, and unzip it to a directory on your local machine.
3. If you haven't already set up a Git respository, you'll need to first create one. If you have an application in a directory, you can change to that directory and then type the following command to initialize your Git repository.

   ```
   git init .
   ```

4. From your Git repository directory, run **AWSDevTools-RepositorySetup.sh**. You can find **AWSDevTools-RepositorySetup.sh** in the AWS DevTools/Linux directory. You need to run this script for each Git repository.

   To learn how to create and deploy an application using AWS DevTools, see AWS DevTools (p. 336).

Windows

The AWS DevTools works on Windows operating systems. This document assumes you can work in a Windows environment. You can set up AWS DevTools in a few basic steps:

- Install the prerequisite software
- Download AWS DevTools
- Run the setup script
- Create a Git repository directory
- Run the repository setup script

To set up AWS DevTools on a Windows computer

1. Install the following prerequisites:

   - Git. To download Git, go to http://git-scm.com/. Make sure you have version 1.6.6 or later.
   - PowerShell 2.0. Windows 7 and Windows Server 2008 R2 comes with PowerShell 2.0. For earlier versions of Windows, you can download PowerShell 2.0. Visit http://technet.microsoft.com/en-us/scriptcenter/dd742419.aspx for more details.

Note

You can check **Control Panel | Programs | Programs and Features** to verify if you have previously installed these applications.

2. Download AWS DevTools, which is part of the command line interface package, at the AWS Sample Code & Libraries website. The .zip file is self-contained, and no installation is required; simply download the .zip file, and unzip it to a directory on your local machine.
3. Double-click **AWSDevTools-OneTimeSetup.bat**. You can find **AWSDevTools-OneTimeSetup.bat** in the AWS DevTools/Windows directory. The setup script installs all of the necessary pieces for pushing AWS Elastic Beanstalk applications. You need to run this setup script only once.
4. If you haven't already set up a Git repository, you'll need to first create one. If you have an application in a directory, you can change to that directory and then type the following command to initialize your Git repository.

```
git init .
```

Note

You need to run the PowerShell commands from an environment that has access to Git. If you installed Git as a native Windows program, that would be cmd.exe. If you installed Git wrapped inside Git Bash, that would be Git Bash.

5. Copy the **AWSDevTools-RepositorySetup.bat** from the AWS DevTools/Windows directory to your Git repository directory, and then double-click **AWSDevTools-RepositorySetup.bat**. You need to run this script for each Git repository.

Note

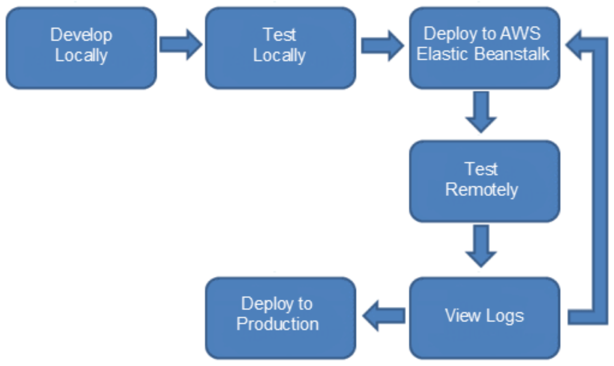
If you receive an error, try running the **AWSDevTools-OneTimeSetup.bat** file again.

To learn how to create and deploy an application using AWS DevTools, see AWS DevTools (p. 336).

Develop, Test, and Deploy

Topics

The following diagram illustrates a typical software development life cycle including deploying your application to AWS Elastic Beanstalk.



Typically, after developing and testing your application locally, you will deploy your application to AWS Elastic Beanstalk. At this point, your application will be live at a URL such as http://myexampleapp-wpams3yrvj.elasticbeanstalk.com. Because your application will be live, you should consider setting up multiple environments, such as a testing environment and a production environment. You can point your domain name to the Amazon Route 53 (a highly available and scalable Domain Name System (DNS) web service) CNAME <*yourappname*>.elasticbeanstalk.com. Contact your DNS provider to set this up. For information about how to map your root domain to your Elastic Load Balancer, see Using AWS Elastic Beanstalk with Amazon Route 53 to Map Your Root Domain to Your Load Balancer (p. 221). After you remotely test and debug your AWS Elastic Beanstalk application, you can then make any updates and redeploy to AWS Elastic Beanstalk. After you are satisfied with all of your changes, you can upload your latest version to your production environment. The following sections provide more details explaining each stage of the software development life cycle.

Develop Locally

After installing AWS DevTools on your local computer, you use the Git command line as you normally would to create your local repository and add and commit changes. You create your PHP application as you normally would with your favorite editor. If you don't already have a PHP application ready, you can use a simple "Hello World" application. Type the following program into your favorite editor, and save it as a PHP file.

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

Next, create a new local repository, add your new program, and commit your change.

```
git add .
git commit -m "initial check-in"
```

Note

For information about Git commands, go to Git - Fast Version Control System.

Test Locally

Normally, at this point you would test your application locally before deploying to AWS Elastic Beanstalk. Suppose you find a few issues you would like to fix. Using the above "Hello World" application, add a "!" after "Hello World" and check in your changes. Update your index.php file, and then type the following commands to check in your updated file.

```
git add .
git commit -m "my second check-in"
```

After you commit your changes, you should see a response similar to the following:

```
[master 0535814] my second check-in
1 files changed, 1 insertions(+), 1 deletions(-)
```

Note the commit ID that is generated. AWS DevTools will use this ID to generate a version label for your application.

Deploy to AWS Elastic Beanstalk

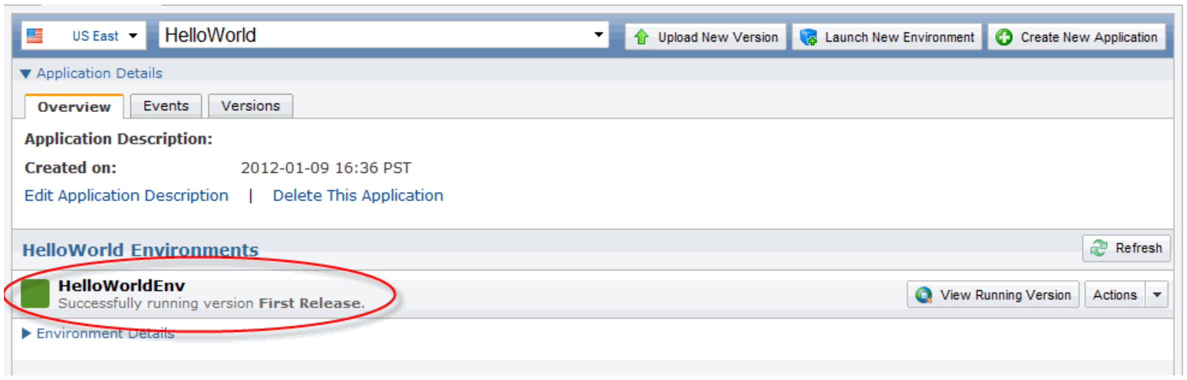
After testing your application, you are ready to deploy it to AWS Elastic Beanstalk. Deploying requires the following steps:

- Use the AWS Management Console, CLI, or APIs to create a sample application.
- Configure your Git environment. AWS DevTools is configured through `git config` commands.
- Update the sample application with your application.

When you update the sample application with your application, AWS Elastic Beanstalk replaces the existing sample application version with your new application version in the existing environment.

To create a sample application

1. Follow the instructions at Creating New Applications (p. 124) to create a sample AWS Elastic Beanstalk application using the AWS Management Console, CLI, or APIs. When selecting a container, select either 32-bit or 64-bit Amazon Linux running PHP.
2. Verify that your AWS Elastic Beanstalk environment is healthy. You should see something similar to the following illustration. Note that it also says running version First Release.



Application Description:
Created on: 2012-01-09 16:36 PST

HelloWorldEnv
Successfully running version First Release.

After you have created a sample application to AWS Elastic Beanstalk, you need to configure your Git environment.

To configure your Git environment

1. From your Git repository directory, type the following command.

   ```
   git aws.config
   ```

2. When you are prompted for the AWS access key, type your access key. To get your access key information, go to Access Credentials.

   ```
   AWS Access Key: AKIAIOSFODNN7EXAMPLE
   ```

   Note

   If you are not prompted to enter your AWS access key, there may be an issue with your setup. Try installing AWS DevTools again. For instructions, see Get Set Up (p. 336).
3. When you are prompted for the AWS secret key, type your secret key. To get your secret key information, go to Access Credentials.

   ```
   AWS Secret Key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   ```

4. When you are prompted for the AWS Elastic Beanstalk region, type the region or press **`Enter`** to accept the default region. For information about this product's regions, go to Regions and Endpoints in the Amazon Web Services General Reference.

   ```
   AWS Region [default to us-east-1]:
   ```

5. When you are prompted for the AWS Elastic Beanstalk application name, type the name of the application. The application name should match the application name that you used when you created your sample application in Creating New Applications (p. 124). In this example, we use HelloWorld.

   ```
   AWS Elastic Beanstalk Application: HelloWorld
   ```

   Note

   If you have a space in your application name, make sure you do not use quotes.
6. When you are prompted for the AWS Elastic Beanstalk environment name, type the name of the environment. The environment name should match the environment name you used when you created your sample application in Creating New Applications (p. 124). In this example, we use HelloWorldEnv.

   ```
   AWS Elastic Beanstalk Environment: HelloWorldEnv
   ```

After configuring your Git environment, you are ready to update the sample application with your application.

If you want to update your Git environment, you can use the **`git aws.config`** command. When prompted, you can update your configuration options. If you want to keep any previous settings, press the **Enter** key. If you want to update a particular configuration option, refer to the table below for the commands.

Command	Description	Example
`aws.push`	Deploy your application to AWS Elastic Beanstalk	`git aws.push`
`aws.config`	Run through all configuration options	`git aws.config`
`aws.accesskey`	Set your AWS Access Key ID	`git config aws.accesskey AKIAIOSFODNN7EXAMPLE`
`aws.secretkey`	Set your AWS Secret Access Key	`git config aws.secretkey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
`aws.region`	Set the region where your environment is located	`git config aws.region us-east-1`
`aws.elasticbeanstalk.host`	Set the endpoint where your environment is located	`git config aws.elasticbeanstalk.host git.elasticbeanstalk.us-east-1.amazonaws.com`
`aws.elasticbeanstalk.application`	Set your application name	`git config aws.elasticbeanstalk.application myappname`
`aws.elasticbeanstalk.environment`	Set your environment name	`git config aws.elasticbeanstalk.environment myappnameenv`
`aws.elasticbeanstalk.remote`	Generate a fresh password and print out a signed URL for the remote	`git aws.elasticbeanstalk.remote`

To remove the settings created by the repository setup, issue the following commands:

```
git config --remove-section aws
git config --remove-section aws.elasticbeanstalk
git config --remove-section alias.aws
git config --remove-section alias.aws.elasticbeanstalk
```

Note

AWS DevTools is write-only. You cannot clone a Git repository using AWS DevTools.

To update the sample application with your local application

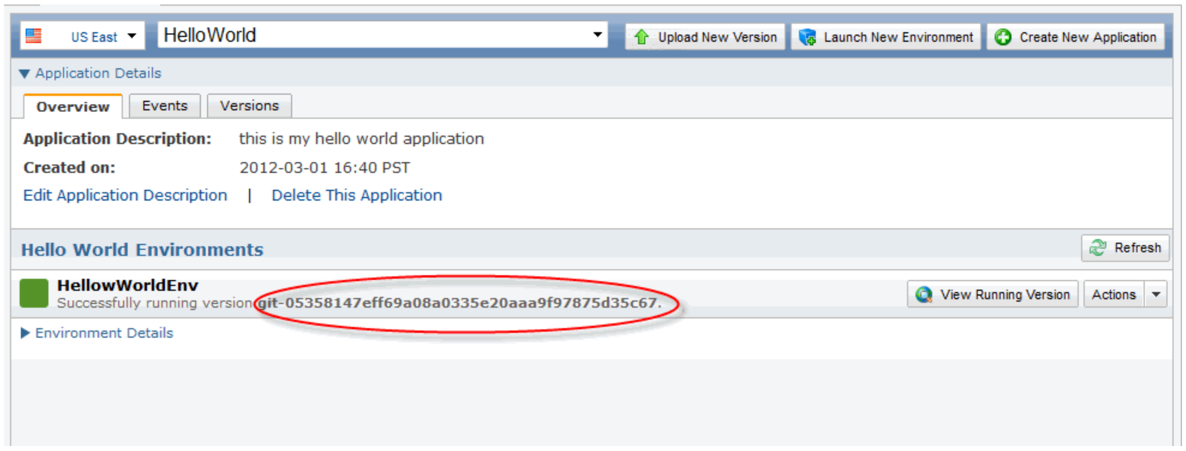
1. Type the following command.

```
git aws.push
```

2. If everything worked as expected, you should see something similar to the following:

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects:100% (2/2), done.
Writing objects: 100% (3/3), 298 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To https://<some long string>@git.elasticbeanstalk.us-east-
1.amazon.com/helloworld/helloworldenv
 44c7066..b1f11a1 master -> master
```

3. Verify that your application has been updated. In the AWS Elastic Beanstalk Console, you should now see something similar to the following illustration. Note that the running version has been updated and begins with the commit ID from your last commit.



Application Description: this is my hello world application
Created on: 2012-03-01 16:40 PST

Hello World Environments

HellowWorldEnv
Successfully running version git-05358147eff69a08a0335e20aaa9f97875d35c67.

Test Remotely

To investigate any issues, you can connect to your Amazon EC2 instance. For instructions on how to connect to your instance, see Listing and Connecting to Server Instances (p. 197). You can also view logs to help you with debugging. For more information about viewing logs, see Debug/View Logs (p. 343).

Debug/View Logs

You can configure your environment so that the logs from the Amazon EC2 instances running your applications are copied by AWS Elastic Beanstalk to the Amazon S3 bucket associated with your application. For instructions on how to view these logs from the AWS Management Console, see Working with Logs (p. 199).

Edit the Application and Redeploy

Now that you have tested your application, it is easy to edit your application, redeploy, and see the results in moments. First, make changes to your application and commit your changes. Then deploy a new application version to your existing AWS Elastic Beanstalk environment.

```
git add .
git commit -m "my third check-in"
git aws.push
```

A new application version will be uploaded to your AWS Elastic Beanstalk environment.

You can use the AWS Management Console, CLIs, or APIs to manage your AWS Elastic Beanstalk environment. For more information, see Managing and Configuring Applications and Environments Using the Console, CLI, and APIs (p. 124).

Deploy to Production

When you are satisfied with all of the changes you want to make to your application, you can deploy it to your production environment. First, you'll need to create a new production environment using the AWS Management Console, CLIs, or APIs. Then you can update your application in your production environment using AWS DevTools. When you update your application using AWS DevTools, AWS Elastic Beanstalk will create a new application version. For information on how to deploy an already existing application version to a new environment, see Launching New Environments (p. 135). The following steps walk you through creating a new environment, and then updating your application in that environment with a new application version using AWS DevTools.

To deploy to production using AWS DevTools

1. To launch a new environment, follow the steps at Launching New Environments (p. 135).
2. Update the Git environment to point to the production environment.

```
git config aws.elasticbeanstalk.environment helloworldprod
```

3. Check in final changes.

```
git add .
git commit -m "final check-in"
```

4. Deploy your application to AWS Elastic Beanstalk.

```
git aws.push
```

Note

Make sure when you call `git aws.push` that your directory is not empty. You cannot push empty directories.

Deploy an Existing Application Version to an Existing Environment

If you need to deploy an existing application to an existing environment, you can do so using the AWS Management Console, CLI, or APIs. You may want to do this if, for instance, you need to roll back to a previous application version. For instructions on how to deploy an existing application version to an existing environment, see Deploying Versions to Existing Environments (p. 141).

Related AWS Elastic Beanstalk Resources

The following table lists related resources that you'll find useful as you work with this service.

Resource	Description
AWS Elastic Beanstalk API Reference	Contains a comprehensive description of all SOAP and Query APIs. Additionally, it contains a list of all SOAP data types.
AWS Elastic Beanstalk Technical FAQ	Covers the top questions developers have asked about this product.
AWS Elastic Beanstalk Release Notes	Provides a high-level overview of the current release. This document specifically notes any new features, corrections, and known issues.
AWS Developer Resource Centers	From here, you will find links to developer centers that provide documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
AWS Management Console	The console enables you to use most of the functions of AWS Elastic Beanstalk and other AWS products without programming.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and AWS Premium Support (if you are subscribed to this program).
AWS Premium Support Information	The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services.
AWS Elastic Beanstalk Product Information	The primary web page for information about AWS Elastic Beanstalk.

Resource	Description
Form for questions related to your AWS account: Contact Us	This form is *only* for account questions. For technical questions, use the Discussion Forums.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

Sample Applications

The following are links to the sample applications that are deployed as part of Getting Started Using AWS Elastic Beanstalk (p. 5).

- Java
- .NET
- PHP
- Python

Document History

The following table describes the important changes to the documentation since the last release of AWS Elastic Beanstalk.

API version: 2010-12-01

Latest documentation update: October 2, 2012

Change	Description	Date Changed
New content	Added new topic Using Amazon RDS and MySQL Connector/J (p. 33).	04 March 2011
New content	Updated some topics for new console user interface. Added new topics Using Custom AMIs (p. 202) and Using Custom Environment Properties with AWS Elastic Beanstalk (p. 30).	17 February 2011
New service	This is the initial release of AWS Elastic Beanstalk.	18 January 2011
New container type	This is the added support of Java Tomcat 7 for AWS Elastic Beanstalk.	21 April 2011
New content	Combined AWS Elastic Beanstalk Getting Started Guide and AWS Elastic Beanstalk User Guide into one guide: AWS Elastic Beanstalk Developer Guide. Added new content for saving and editing environment configuration settings as well as swapping environment URLs. Added new section for CLI reference.	29 June 2011
New content	Added new section for managing and configuration application and environments using the AWS Toolkit for Eclipse.	10 August 2011
New content	You can create a policy that allows or denies permissions to perform specific AWS Elastic Beanstalk actions on specific AWS Elastic Beanstalk resources.	6 March 2012
New content	Added new section for deploying PHP applications using Git.	20 March 2012
New content	Added content to support the Asia Pacific (Tokyo) Region.	23 April 2012

Change	Description	Date Changed
New content	Added new section for deploying .NET applications.	08 May 2012
New content	Added content to support the EU West (Ireland) Region.	16 May 2012
New content	Added content for deploying .NET applications using the standalone deployment tool.	29 May 2012
New content	Added Get Started section for command line interface.	27 June 2012
New content	Added content to support US West (Oregon) Region and US West (Northern California) Region.	10 July 2012
New content	Added content for deploying Python applications and Amazon RDS integration.	19 August 2012
New content	Added content to support Asia Pacific (Singapore) Region.	4 September 2012
New content	Added content for customizing container types. Added content for migrating applications using legacy container types to non-legacy container types.	02 October 2012

Appendix

Topics

Using Amazon RDS and MySQL Connector/J (Legacy Container Types)

Amazon Relational Database Service (Amazon RDS) lets you quickly and easily provision and maintain a MySQL Server instance in the cloud. This topic discusses how you can use Amazon RDS and the MySQL Connector/J with your AWS Elastic Beanstalk application.

To use Amazon RDS from your AWS Elastic Beanstalk application, you need to do the following:

- Create an Amazon RDS DB Instance.
- Configure your Amazon RDS DB security group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application.
- Create a JDBC connection string using your Amazon RDS DB Instance's public DNS name and configure your AWS Elastic Beanstalk environment to pass the string to your AWS Elastic Beanstalk application as an environment variable.
- Download and install MySQL Connector/J.
- Retrieve the JDBC connection string from the environment property passed to your server instance from AWS Elastic Beanstalk and use MySQL Connector/J to access your Amazon RDS database.

To use Amazon RDS with MySQL Connector/J from your AWS Elastic Beanstalk application

1. Create an Amazon RDS DB Instance. For instructions on how to do this, go to the Amazon Relational Database Service Getting Started Guide.
2. Configure your Amazon RDS DB security group to allow access from the Amazon EC2 security group used by your AWS Elastic Beanstalk application. For instructions on how to find the name of your EC2 security group using AWS Toolkit for Eclipse, see Amazon EC2 Security Groups (p. 39). For instructions on how to find the name of your EC2 security group using the AWS Management Console, see Amazon EC2 Security Groups (p. 161). For more information, go to the "Authorizing Network

Access to an Amazon EC2 Security Group" section of Working with DB Security Groups in the *Amazon Relational Database Service User Guide*.

3. Download and install MySQL Connector/J for your development environment. For download and installation instructions, go to http://dev.mysql.com/downloads/connector/j.
4. Create a JDBC connection string using your Amazon RDS DB Instance's public DNS name, port number, and (optionally) database name and login credentials. The following example shows a JDBC connection string that would connect to the employees database on an RDS instance at mydbinstance.abcdefghijkl.us-east-1.rds.amazonaws.com using port 3306, with the user name "sa" and the password "mypassword".

```
jdbc:mysql://mydbinstance.abcdefghijkl.us-east-1.rds.amazonaws.com:3306/em
ployees?user=sa&password=mypassword
```

5. Configure your AWS Elastic Beanstalk environment to pass the string to your AWS Elastic Beanstalk application as an environment property. For instructions on how to do this, go to Using Custom Environment Properties with AWS Elastic Beanstalk (p. 30).
6. Retrieve the JDBC connection string from the environment property passed to your server instance from AWS Elastic Beanstalk and use MySQL Connector/J to access your Amazon RDS database. The following code example shows how to retrieve the JDBC_CONNECTION_STRING custom environment property from a Java Server Page (JSP).

```
<p>
    The JDBC_CONNECTION_STRING environment variable is:
    <%= System.getProperty("JDBC_CONNECTION_STRING") %>
</p>
```

   For more information on getting started using the MySQL Connector/J to access your MySQL database, go to http://dev.mysql.com/doc/refman/5.0/en/connector-j.html.
7. Copy the MySQL Connector/J JAR file into your AWS Elastic Beanstalk application's WEB-INF/lib directory.
8. Deploy your application to AWS Elastic Beanstalk. For information on how to deploy your application using AWS Elastic Beanstalk and the AWS Management Console, see Getting Started Using AWS Elastic Beanstalk (p. 5). For information on how to deploy your application using Eclipse, go to Getting Started with AWS Elastic Beanstalk Deployment in Eclipse.