Getting Started with AWS

Computing Basics for Linux

Getting Started with AWS: Computing Basics for Linux

Overview 1
Getting Started 7
Step 1: Sign Up for the Service 8
Step 2: Install the Command Line Tools 8
Step 3: Find a Suitable AMI 9
Step 4: Launch an Instance 10
Step 5: Deploy Your Application 12
    Connecting to your Amazon EC2 Instance from Your Web Browser Using the MindTerm SSH Client 12
    Connect to Your Amazon EC2 Instance from a Windows Computer Using PuTTY 14
    Connecting to Your Amazon EC2 Instance from a Linux/UNIX Machine Using a Standalone SSH Client 17
    Configure the Amazon EC2 Instance 18
Step 6: Create a Custom AMI 24
Step 7: Create an Elastic Load Balancer 25
Step 8: Update Your Amazon EC2 Security Group 31
Step 9: Launch Amazon EC2 Instances Using Auto Scaling 32
Step 10: Create a CloudWatch Alarm 36
Step 11: Clean Up 43
    Delete Your CloudWatch Alarm 44
    Delete Your Elastic Load Balancer 44
    Terminate Your Amazon EC2 Instances in Your Auto Scaling Group 45
    Terminate Your Instance 47
    Delete a Key Pair 47
    Delete an Amazon EC2 Security Group 47
Pricing 48
Amazon EC2 Cost Breakdown 48
Summing It All Up 51
How To Further Save Costs 52
Related Resources 55
Document History 57

Overview

When you deploy any type of application, you typically need to do the following:

- Set up a computer to run your application.
- Secure your application and resources.
- Set up your network for users to access your application.
- Scale your application.
- Monitor your application and resources.
- Ensure that your application is fault-tolerant.

This guide introduces you to several key AWS services and components that help address these basic needs. In this guide, you will learn more about what these key services are, why they are important in deploying a web application, and how to use them.

To help you learn about the key AWS services, we'll review an example architecture of a web application hosted on AWS, and we'll walk through the process of deploying Drupal. (Drupal is an open-source content management system.) You can adapt this sample to your specific needs if you want. By the end of this walkthrough, you should be able to do the following:

- Sign up for AWS.
- Launch, connect, secure, and deploy Drupal (including a MySQL database) to a computer in the cloud.
- Create a custom template of a computer containing the hardware, software, and configuration you need.
- Set up a load balancer to distribute traffic across multiple computers in the cloud.
- Scale your fleet of computers in the cloud.
- Monitor the health of your application and computers.
- Clean up your AWS resources.

For a deeper understanding of AWS best practices and the various options that AWS provides, we recommend that you read *Web Application Hosting: Best Practices* at AWS Cloud Computing Whitepapers.

If you are looking for a quicker and easier way to deploy your web applications, you can use an application management service. AWS application management services help you leverage other AWS services without having to manage each of them separately and manually:

- AWS Elastic Beanstalk lets you focus on the code while the service manages the rest.

- AWS OpsWorks gives you the flexibility to define your own software stack and deploy, operate, and automate a variety of applications and architectures.

For additional information about deployment and resource management on AWS, go to Deployment and Management on AWS.

If this guide is not exactly what you are looking for, you may want to check out the following documents:

- Getting Started with AWS — Provides information about Amazon Web Services, with helpful links for learning more.
- Getting Started with AWS Free Usage Tier — Provides information about how to get started with the free usage tier.
- Hosting Websites on Amazon S3 in the *Amazon Simple Storage Service Developer Guide* — Provides a walkthrough in just a few steps of a static website deployment that does not require running an application.
- Getting Started with AWS CloudFormation in the *AWS CloudFormation User Guide* — Helps you quickly get started using an AWS CloudFormation WordPress blog sample template without needing to figure out the order in which AWS services need to be provisioned or worry about the subtleties of how to make those dependencies work.
- Getting Started with AWS Web Application Hosting for Linux — Provides a more in-depth walkthrough that uses more services, such as Amazon Relational Database Service (Amazon RDS) and Amazon Route 53.
- Amazon Elastic Compute Cloud Getting Started Guide — Provides information that helps you get started using Amazon EC2 instances.

Introduction to AWS

If you are responsible for running a web application, you face a variety of infrastructure and architecture issues for which AWS can give you easy, seamless, and cost-effective solutions. This section provides a list of Amazon Web Services and components, and it explains the value they add in meeting the challenges you'll face in this example solution. We break this down in to the following sections: computing resources, security, monitoring, networking, and fault-tolerance.

Computing Resources

When you deploy an on-premises solution, you need to buy a computer with an operating system, software, and hardware that match your needs. When you deploy your solution on Amazon Web Services, you select an Amazon Machine Image (AMI) and then use it to deploy a virtual server known as an Amazon Elastic Compute Cloud (EC2) instance. An AMI is a template that contains a software configuration (e.g., operating system, application server, and applications). For example, an AMI might contain all the software to act as a web server (e.g., Linux, Apache, and your website). A large selection of public AMIs is available from Amazon and the Amazon EC2 community. You can find an AMI that most closely matches your needs and then customize it. You can save this customized configuration to another AMI, which you can use to launch new Amazon EC2 instances whenever you need them.

Storage can be an integral part of an Amazon EC2 instance, or it can be an independent component whose lifetime is managed separately from the lifetime of the instance. There are AMIs for each storage strategy, and you will need to decide which type you want to use. When you launch your Amazon EC2 instances, you can store your root device data on Amazon Elastic Block Store (Amazon EBS) or the local instance store. Amazon Elastic Block Store (Amazon EBS) is a durable, block-level storage volume that you can attach to a single Amazon EC2 running instance. Amazon EBS volumes behave like raw, unformatted, external block devices you can attach. They persist independently from the running life of an Amazon EC2 instance. Alternatively, the local instance store is a temporary storage volume and

persists only during the life of the instance. You might use Amazon EBS-backed instances for web or database servers that keep state locally and require the data to be available even if the associated instance crashes. You might use Amazon instance-store backed instances to manage traffic on large web sites where each instance is a clone. This is an inexpensive way to launch instances where data is not stored to the root device. To summarize the two key differences between these AMIs:

- You can stop and restart an Amazon EBS-backed instance, but you can only run or terminate an Amazon EC2 instance store-backed instance.
- By default, any data on the instance store is lost if the instance fails or terminates. Data on Amazon EBS-backed instances is stored on an Amazon EBS volume, so no data is lost if the instance is terminated.

For more information about the differences between instance store-backed and Amazon EBS-backed instances, go to Basics of Amazon EBS-Backed AMIs and Instances in the *Amazon Elastic Compute Cloud User Guide*.

Security

When you buy a new computer, you create a user name and password, which you then use to log in. In AWS, you use a public/private key pair to sign in to your Amazon EC2 instance. The public key is embedded in your instance, and you use the private key to sign in securely without a password. When connecting to Amazon Linux instances, you initially sign in with a username of "ec2-user" or "root," depending on the AMI you are using.

When you deploy your application, you'll want to secure your system. For an on-premises deployment, you would normally specify the ports and the protocols in which users can access your application. In AWS, you do the same thing. AWS has security groups that act like inbound network firewalls so you can decide who can connect to your Amazon EC2 instances over which ports.

Scaling

You may find that your application traffic varies during the day. For example, from 9 a.m. to 5 p.m., you may experience peak traffic; for the rest of the day, traffic may be much slower. As traffic levels change, it would be useful to continually adjust the number of computers running your application to changes in traffic. Auto Scaling can automatically launch and terminate instances on your behalf according to the policies that you set. If you have defined a baseline AMI, Auto Scaling launches new instances with the exact same configuration. Auto Scaling can also send you notifications when it adds or removes instances.

Monitoring

You need to stay aware of the current performance and state of your resources. If your resources are not in the appropriate state, can't handle the traffic load, or are sitting idle, you need to be alerted so you can take appropriate action. Amazon CloudWatch monitors AWS cloud resources and the applications you run on AWS. You can collect and track metrics, analyze the data, and react immediately to keep your applications and business running smoothly. You can use information from Amazon CloudWatch to take action on the policies that you set using Auto Scaling. For example, you can create an alarm to notify you if your CPU utilization exceeds 95%. If the threshold is exceeded, Amazon CloudWatch sends an alarm, and Auto Scaling takes action according to the policy that you set. In this example, Auto Scaling can launch a new instance to handle the increased load. Similarly, you could set an alarm that notifies you if your CPU utilization falls below a certain threshold. In that case, Auto Scaling could terminate an instance, saving you money.

You can monitor the status of your instances by viewing status checks and scheduled events for your instances. Automated status checks performed by Amazon EC2 detect whether specific issues are

affecting your instances. The status check information, together with the data provided by Amazon CloudWatch, gives you detailed operational visibility into each of your instances.

You can also see the status of specific events scheduled for your instances. Scheduled events provide information about upcoming activities, such as rebooting or terminating an instance, that are planned for your instances, along with the scheduled start and end times of each event. To learn more about instance status, go to Monitoring the Status of Your Instances in the *Amazon Elastic Compute Cloud User Guide*.

Networking

If you require multiple computers to host your web application, you need to balance the traffic across those computers. Elastic Load Balancing provides this service in the same way that an on-premises load balancer does. You can associate a load balancer with an Auto Scaling group. As instances are launched and terminated, the load balancer automatically directs traffic to the running instances. Elastic Load Balancing also performs health checks on each instance. If an instance is not responding, the load balancer can automatically redirect traffic to the healthy instances.

AWS assigns a URL to your AWS resources, such as your Elastic Load Balancer and your Amazon EC2 instances; however, you may want a URL that is more specific and easy to remember, such as www.example.com. To do so, you need to purchase a domain name from a domain registrar. After you purchase your domain name, you can use Amazon Route 53 to map your domain name to your AWS deployment.

You may want to provision a private, isolated network. You can use Amazon Virtual Private Cloud (Amazon VPC) to provision a private, isolated section of the Amazon Web Services (AWS) cloud where you can launch AWS resources in a virtual network that you define. For example, if you are hosting a multitier web application, you may want to customize the network configuration so that your web servers are public facing and your database and application servers are in a private-facing subnet with no Internet access. The application servers and databases can't be directly accessed from the Internet, but they can still access the Internet over a NAT instance so they can, for example, download patches.

You can control access between the servers and subnets by using inbound and outbound packet filtering provided by network access control lists and security groups. Some other cases where you may want to use Amazon VPC include:

- Hosting scalable web applications in the AWS cloud that are connected to your data center
- Extending your corporate network into the cloud
- Disaster recovery

For information on how to get started using Amazon VPC, go to Get Started with Amazon VPC in the *Amazon Virtual Private Cloud Getting Started Guide*.

Fault Tolerance

To make your web application fault-tolerant, you need to consider deploying your computers in different physical locations. It can be expensive to maintain hardware in different physical locations for an on-premises solution. AWS offers resources in different Availability Zones and regions. Availability Zones are analogous to data centers. You can have multiple instances running in different Availability Zones so that if one Availability Zone becomes unavailable (e.g., due to a natural disaster), then all traffic would be routed to another Availability Zone. There are multiple Availability Zones in each region.

It's even more advantageous to spread your instances across Regions. If a region, including all of its Availability Zones, becomes completely unavailable, your traffic is routed to another region.

Summary

The following table summarizes the key challenges to developing a simple web application and the AWS services that address these challenges.

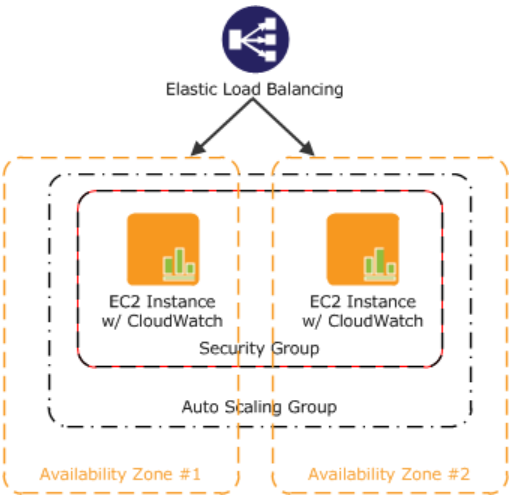
Challenge	Amazon Web Services	Benefit
Need computers to run your application.	Amazon Elastic Compute Cloud (EC2)	Amazon EC2 runs the web server and application servers.
Incoming traffic needs to be evenly distributed across computers to maximize performance.	Elastic Load Balancing	Elastic Load Balancing supports health checks on hosts, distribution of traffic to Amazon EC2 instances across multiple Availability Zones, and dynamic addition and removal of Amazon EC2 hosts from the load-balancing rotation.
Servers need to be provisioned to handle peak capacity, but the unused cycles are wasted at other times.	Auto Scaling	Auto Scaling creates capacity groups of servers that can grow or shrink on demand.
Servers need to be monitored for performance and state	Amazon CloudWatch	Amazon CloudWatch reports metrics data for Amazon EC2 instances, and the metrics it gathers are used by Auto Scaling.
Applications may require persistent storage.	Amazon Elastic Block Store (Amazon EBS)	Amazon EBS provides a persistent file system for web and application servers.

The following table summarizes additional challenges to developing a simple web application and the AWS components that address these challenges.

Challenge	AWS Component	Benefit
Need a secure mechanism to connect to the computer.	Amazon Key Pair	A key pair is a security credential similar to a password, which you use to securely connect to your instance after the instance is running.
Need to provide security to protect application servers from outside malicious users.	Amazon Security Group	An Amazon Security Group gives you control over the protocols, ports, and source IP address ranges that are allowed to reach your Amazon EC2 instances.
Need to design with failover in mind.	Availability Zones	Availability Zones are distinct locations engineered to be insulated from failures in other Availability Zones. Each Availability Zone provides inexpensive, low-latency network connectivity to other Availability Zones in the same region.

Sample Architecture

The following diagram shows an example architecture that uses the AWS resources mentioned in the previous section.



As an example, we'll walk through a deployment of a simple web application. If you're doing something else, you can adapt this example architecture to your specific situation. In this diagram, Amazon EC2 instances in a security group run the application and web server. The Amazon EC2 Security Group acts as an exterior firewall for the Amazon EC2 instances. An Auto Scaling group maintains a fleet of Amazon EC2 instances that can be automatically added to or removed in order to handle the presented load. This Auto Scaling group spans two Availability Zones to protect against potential failures in either Availability Zone. To ensure that traffic is distributed evenly among the Amazon EC2 instances, an Elastic Load Balancer is associated with the Auto Scaling group. If the Auto Scaling group launches or terminates instances to respond to load changes, the Elastic Load Balancer automatically adjusts accordingly.

For a step-by-step walkthrough of how to build out this architecture, see Getting Started (p. 7). This walkthrough will teach you how to do the following:

- Sign up for AWS.
- Launch, connect, and deploy Drupal to an Amazon EC2 instance.
- Create a Custom AMI.
- Set up an Elastic Load Balancer to distribute traffic across your Amazon EC2 instances.
- Scale your fleet of instances automatically using Auto Scaling.
- Monitor your AWS resources using Amazon CloudWatch.
- Clean up your AWS resources.

Getting Started

Topics

- Step 1: Sign Up for the Service (p. 8)
- Step 2: Install the Command Line Tools (p. 8)
- Step 3: Find a Suitable AMI (p. 9)
- Step 4: Launch an Instance (p. 10)
- Step 5: Deploy Your Application (p. 12)
- Step 6: Create a Custom AMI (p. 24)
- Step 7: Create an Elastic Load Balancer (p. 25)
- Step 8: Update Your Amazon EC2 Security Group (p. 31)
- Step 9: Launch Amazon EC2 Instances Using Auto Scaling (p. 32)
- Step 10: Create a CloudWatch Alarm (p. 36)
- Step 11: Clean Up (p. 43)

Let's suppose you want to deploy Drupal, an open-source content management system (CMS). It's easy to get started, and for most of the tasks we can use the AWS Management Console. In this topic, we'll walk through a series of steps to deploy your web application to AWS. There are many different ways you can go about deploying your web application. The approach that this walkthrough takes follows best practices and uses several of the core services so you can see how they work together.

Before you begin deploying Drupal using AWS, you'll need to sign up for an AWS account and install the Auto Scaling command line tools. Signing up for AWS gives you access to all of the services; however, you are charged only for what you use.

First, you'll find a suitable AMI that meets your hardware and software needs. You'll use this AMI to launch an Amazon EC2 instance. When launching your Amazon EC2 instance, you'll create a new key pair and a security group. The security group sets rules for who can access the Amazon EC2 instance, and the key pair is necessary for connecting to your Amazon EC2 instance.

With your instance running and secured, you will finish installing the required software and then configure the Drupal application. To simplify launching new Amazon EC2 instances that are already configured, you'll create a custom AMI that will become your new baseline.

You'll then create an Elastic Load Balancer to distribute the traffic load across multiple instances and then update your security group to allow HTTP traffic from only your load balancer instead of from everyone. You create your Elastic Load Balancer before you launch your instances so that you can associate your

Auto Scaling group with your Elastic Load Balancer. That way, your load balancer can automatically stop routing traffic to any terminated instances, and it can start routing traffic to any newly launched instances.

At this point, you'll use Auto Scaling to launch your Amazon EC2 instances. You'll create an Auto Scaling policy that tells Auto Scaling when to increment or decrement the number of instances in your group.

Finally, you'll create a CloudWatch alarm that monitors the instances in your Auto Scaling group and tells the Auto Scaling group when to take action on that policy.

Because this is a sample deployment, you may want to terminate all the AWS resources that you have created. As soon as you terminate an AWS resource, you stop accruing charges for that resource.

Step 1: Sign Up for the Service

If you don't already have an AWS account, you'll need to get one. Your AWS account gives you access to all services, but you will be charged only for the resources that you use. For this example walkthrough, the charges will be minimal.

To sign up for AWS

1. Go to http://aws.amazon.com and click **Sign Up**.
2. Follow the on-screen instructions.

AWS notifies you by email when your account is active and available for you to use.

You use your AWS account to deploy and manage resources within AWS. If you give other people access to your resources, you will probably want to control who has access and what they can do. AWS Identity and Access Management (IAM) is a web service that controls access to your resources by other people. In IAM, you create users, which other people can use to obtain access and permissions that you define. For more information about IAM, go to Using IAM.

Step 2: Install the Command Line Tools

We'll need to install some command line tools for Auto Scaling. Do this first to minimize your usage of billable services.

To install the Auto Scaling command line tools to your local computer, go to Using the Command Line Tools in the *Auto Scaling Developer Guide*. After you have installed the command line tools, try a couple of commands to make sure they work. For example, try typing the **as-cmd** command at the prompt.

```
PROMPT>as-cmd
```

This command returns a list of all the Auto Scaling commands and their descriptions. You should see something similar to the following illustration.

After you have installed the command line tools, you can start creating your AWS resources. Move on to Step 3: Find a Suitable AMI (p. 9) to learn how to find a suitable AMI. You will use this AMI to launch your Amazon EC2 instance. It will also serve as a baseline for creating your own custom AMI.

Step 3: Find a Suitable AMI

An Amazon Machine Image (AMI) contains all information necessary to launch instances of your software. For example, an AMI might contain all the software needed to act as a web server (e.g., Linux, Apache, and your website). We'll use one of these AMIs for this walkthrough. You can launch one or more Amazon EC2 instances from an AMI, and all the instances are exactly alike.

A large selection of AMIs is available from Amazon and the Amazon EC2 community. For more information, go to AWS Marketplace.

You can use the AWS Management Console (at http://console.aws.amazon.com) to search for AMIs that meet specific criteria and then launch instances of those AMIs. For example, you can view the AMIs that Amazon has provided, AMIs the EC2 community has provided, or AMIs that use a specific operating system.

In this task, you will use an Amazon Linux AMI that has Apache, MySQL, PHP, and Drupal installed. You can use this AMI as a baseline, customize it, and then, in a later task, create your own custom AMI.

To find a suitable AMI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Make sure that **US East (N. Virginia)** is selected in the region selector of the navigation bar.
3. In the navigation pane, click **AMIs**.
4. In the **Filter** lists, select **Public images**, then **Amazon images**, then **Amazon Linux**. This limits the display to AMIs that are provided by Amazon Web Services. In the text box, type **`drupal`**.

5. Select an AMI that already has Drupal installed and then click **Launch**.

You will use this AMI as a baseline. Clicking **Launch** starts the launch wizard, which configures your instance and then launches it.

Step 4: Launch an Instance

You are now ready to launch an Amazon EC2 instance using the AMI that you selected in the previous step. Launching an instance involves the following tasks:

- Configure the instance.
- Create a key pair.
- Create a security group.
- Launch the instance.

In the previous step, you selected an AMI and clicked **Launch**, which displays the launch wizard. However, EC2 provides other ways to launch an instance. If you click **Instances** in the left navigation pane and then click **Launch Instance**, the launch wizard appears.

Because we already selected an AMI in the previous step, the wizard appears on the second step, **Choose an Instance Type**.

Important
The instance you're about to launch will be live. You will incur the standard Amazon EC2 usage fees for the instance until you terminate it in the last task in this tutorial. If you complete this walkthrough in one session, the total charges will be minimal (typically less than a dollar). For more information about Amazon EC2 usage rates, go to the Amazon EC2 product page.

To launch an Amazon EC2 instance

1. On the **Choose an Instance Type** page, you can select the hardware configuration and size of the instance to launch. In this case, leave the default selection and click **Next: Configure Instance Details**.
2. On the **Configure Instance Details** page, change the following settings as necessary, leave the other default settings, and then click **Next: Add Storage**:

   - **Network**: Your account may support the EC2-Classic and EC2-VPC platforms, or EC2-VPC only. If your account supports EC2-Classic, select **Launch into EC2-Classic**, and select **us-east-1b** from the **Availability Zone** list. If your account supports EC2-VPC only, select your default VPC, and select a default subnet in the **us-east-1b** Availability Zone from the **Subnet** list.

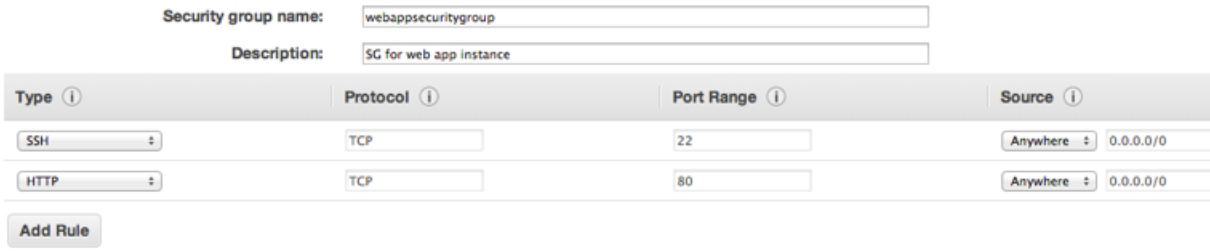
   On the next pages of the wizard, click **Next** until you get to the **Configure Security Group** page, then go to the next step.
3. Create a security group:

   A security group defines firewall rules for your instances. These rules specify which incoming network traffic should be delivered to your instance (e.g., accept web traffic on port 80). All other traffic is ignored. You can modify rules for a group at any time. The new rules are automatically enforced for all running instances. For more information about security groups, go to Using Security Groups in the *Amazon Elastic Compute Cloud (Amazon EC2)*.

   Caution
   By default, the launch wizard creates a security group that enables *all* IP addresses to access your instance over SSH. This is acceptable for the short exercise in this tutorial, but

it's not secure for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your instance.

a. In the **Security group name** field, clear the default security group name and type **`webappsecuritygroup`**.
b. In the **Description** field, you can clear the default description, and type a description of your choice.
c. Click **Add Rule**, and select **HTTP** from the **Type** list.



d. Click **Review and Launch**.

The security group is created and assigned an ID (e.g., sg-48996e20). Your instance will be launched into this new security group.

4. Review your settings and click **Launch**. You'll be prompted to select or create a key pair. In this exercise, we'll create a new key pair in the next step.
5. Create a key pair:

   a. Amazon EC2 instances created from a public AMI use a public/private key pair, rather than a password, for signing in. The public key is embedded in your instance. You use the private key to sign in securely without a password. After you create your own AMIs, you can choose other mechanisms to securely log in to your new instances.

      Select **Create a new key pair**, and in the **Key pair name** box, type **`mykeypair`**. This will be the name of the private key file associated with the pair (with a `.pem` extension).

   b. Click **Download Key Pair**.

      You're prompted to save the private key from the key pair to your system.

   c. Save the private key in a safe place on your system, and record the location where you saved it.

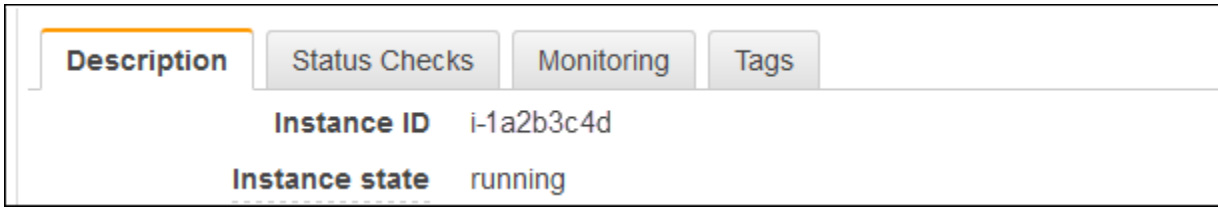
      Important
      You need the key pair file to be able to connect to your Amazon EC2 instance. You can't download the key pair file again, so if you lose it, you will not be able to connect to your instance.

   d. Select the acknowledgment check box, and click **Launch Instances**.

6. When a confirmation message appears, click **View Instances**. It takes a short time for an instance to launch. While the instance is launching, its status will be shown as *pending*.

   After a short period, your instance's status switches to *running*. To manually refresh the display at any time, you can click Refresh. When your instance's status is *running*, you can connect to your instance and deploy your application.
7. Record the public DNS name for your instance:

- Select the running instance, and note the public DNS address in the bottom pane. You will need it for the next task.



Step 5: Deploy Your Application

Topics

- Connecting to your Amazon EC2 Instance from Your Web Browser Using the MindTerm SSH Client (p. 12)
- Connect to Your Amazon EC2 Instance from a Windows Computer Using PuTTY (p. 14)
- Connecting to Your Amazon EC2 Instance from a Linux/UNIX Machine Using a Standalone SSH Client (p. 17)
- Configure the Amazon EC2 Instance (p. 18)

Now that you've launched your Amazon EC2 instance, it's time to connect to it and deploy your application. In this step, you'll first connect to your Amazon EC2 instance, and then you'll deploy Drupal, which is already available on the Linux AMI.

Connecting to your Amazon EC2 Instance from Your Web Browser Using the MindTerm SSH Client

The steps to connect to a Linux/UNIX instance using your browser are as follows:

1. Install and Enable Java on Your Browser (p. 12)
2. Connect Using the MindTerm (SSH) Client (p. 13)

Install and Enable Java on Your Browser

To connect to your instance from the Amazon Elastic Compute Cloud (Amazon EC2) console, you must have Java installed and enabled in your browser. To install and enable Java, follow the steps Oracle provides below or contact your IT administrator to install and enable Java on your web browser.

> **Note**
> On a Windows or Mac client, you must run your Web browser with administrator credentials. For Linux, additional steps may be required if you are not logged in as root.

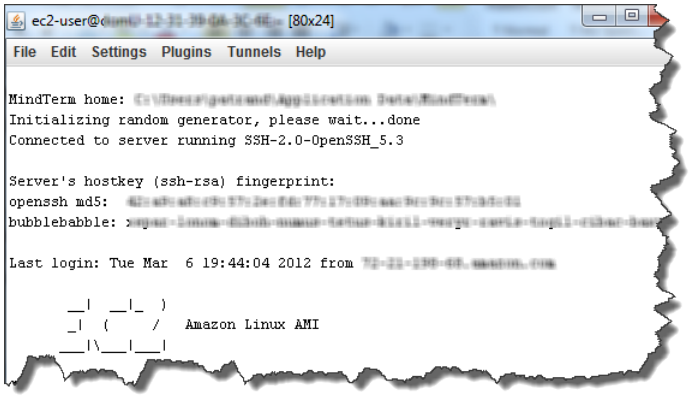
1. Install Java (see http://java.com/en/download/help/index_installing.xml).
2. Enable Java in your web browser (see http://java.com/en/download/help/enable_browser.xml).

Connect Using the MindTerm (SSH) Client

To connect to your instance through a web browser

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, click **Instances**.
3. Select your instance, and then click **Connect**.
4. Click **A Java SSH client directly from my browser (Java required)**. AWS automatically detects the public DNS address of your instance and the key pair name you launched the instance with.
5. In **User name**, enter the user name to log in to your instance. For this example, enter **`ec2-user`**.

   Note
   For an Amazon Linux instance, the default user name is ec2-user. For Ubuntu, the default user name is ubuntu. Some AMIs allow you to log in as root.
6. The **Key name** field is automatically populated for you.
7. In **Private key path**, enter the fully qualified path to your .pem private key file.
8. For **Save key location**, click **Store in browser cache** to store the key location in your browser cache so the key location is detected in subsequent browser sessions, until you clear your browser's cache.
9. Click **Launch SSH Client**.
10. If necessary, click **Yes** to trust the certificate.
11. Click **Run** to run the MindTerm client.
12. If you accept the license agreement, click **Accept**.
13. If this is your first time running MindTerm, a series of dialog boxes will ask you to confirm setup for your home directory and other settings.
14. Confirm settings for MindTerm setup.
15. A screen similar to the following opens and you are connected to your instance.



If you have trouble connecting using MindTerm, check the following:

- Make sure you installed Java and enabled it in your browser.
- Make sure you are using the correct user name.
- Make sure you have typed the correct key pair and path to your private key pair.
- Make sure you have configured your security group to allow you to connect to your instance. .
- If you still continue to experience issues, check the AWS Forums for a possible solution.

16. Use the `sudo service httpd start` command to start the web server.

```
sudo service httpd start
```

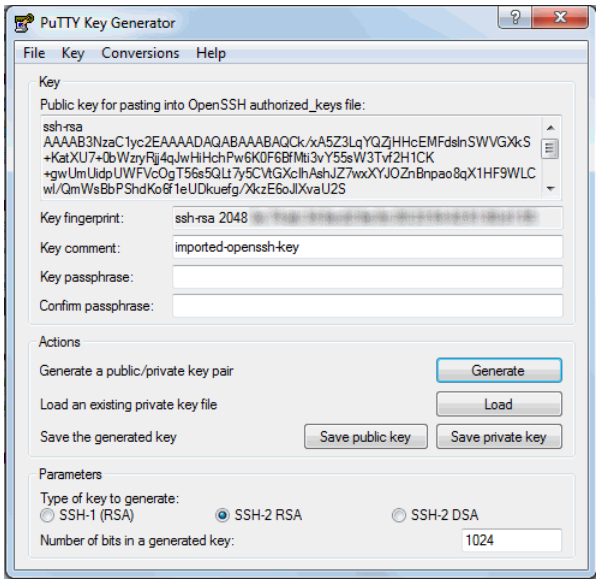
Connect to Your Amazon EC2 Instance from a Windows Computer Using PuTTY

If you are running Windows from your local machine, Secure Shell (SSH) is not built in, so you will need to install PuTTY and PuTTYGen. You'll need the contents of the private key file that you created (e.g., `mykeypair.pem`) in Step 4: Launch an Instance (p. 10).

To connect to your Amazon EC2 instance from a Windows machine

1. Download and install PuTTY and PuTTYGen. A search on "download Putty" on Google returns a list of download sites. Be certain that you install both PuTTY and PuTTYGen, because you will need both of them.
2. Convert the key pair using PuTTYGen. For information on key pairs, see Step 4: Launch an Instance (p. 10).

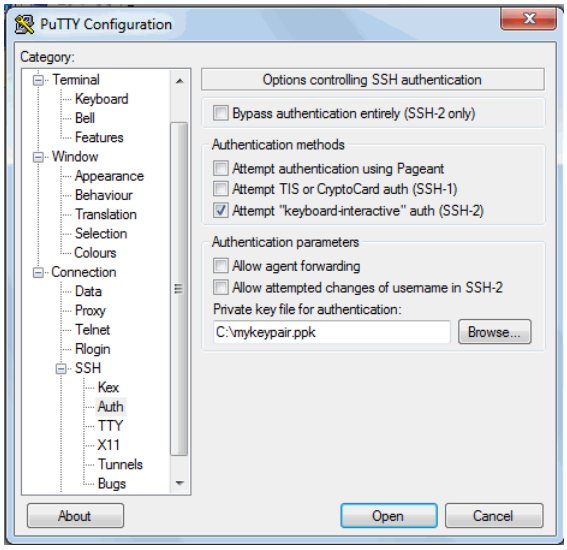
   a. Launch PuTTYGen. On the **Conversions** menu, click **Import Key**.
   b. Browse for **`mykeypair.pem`**, and then click **Open**.



   c. Click **Save private key**. In the message that asks if you want to continue without a passphrase, click **Yes**. Save the key as **`mykeypair.ppk`**.
   d. Close PuTTYGen.

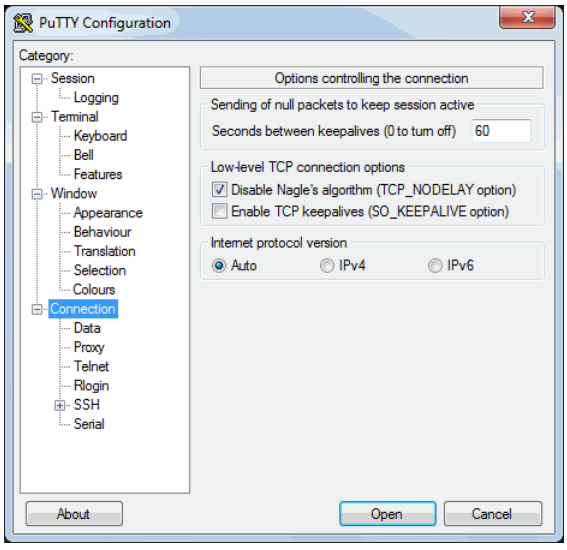
3. Configure the SSH settings.

   a. Start PuTTY, expand the **SSH** node, and then click **Auth**.
   b. In the **Private key file for authentication** box, enter the location for `mykeypair.ppk`.



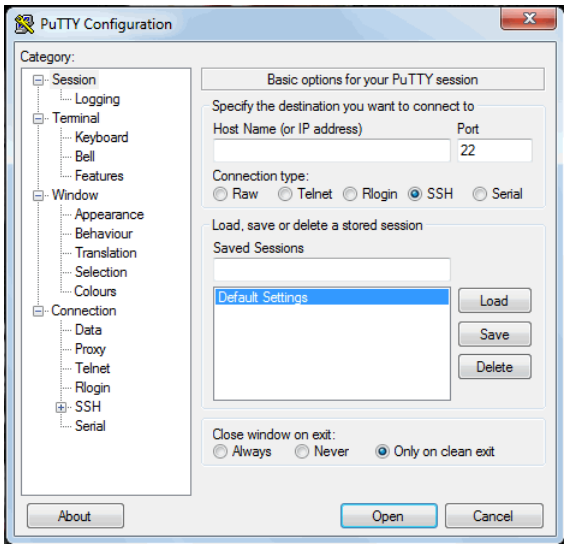
4. Modify the keepalive.

   a. In the PuTTY Configuration window, in the **Category** pane, click **Connection**.
   b. In the **Seconds between keepalives (0 to turn off)** box, type **`60`** . If you don't change this value, your session will time out.



5. Save the session settings.

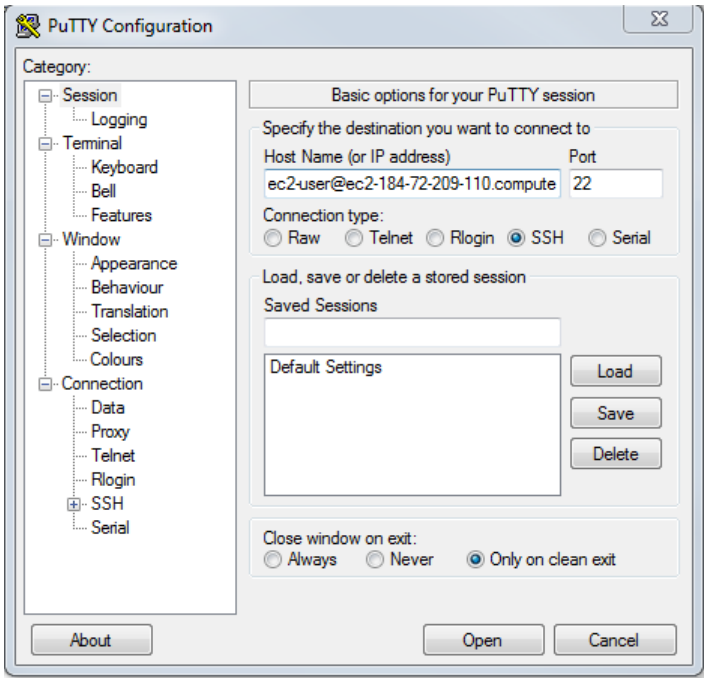
   a. In the **PuTTY Configuration** window, in the **Category** pane, click **Session**.
   b. In the **Load, save, or delete a stored session** box, click **Default Settings**, and click **Save**.



6. Type the DNS address of the Amazon EC2 instance that you retrieved in the previous task.

   a. In the PuTTY Configuration window, in the **Category** pane, click **Sessions**. In the **Host Name (or IP address)** box, type **`ec2-user`**@<*DNS address of your Amazon EC2 instance*>.

      Note
      The user name for the AMI is **ec2-user**.

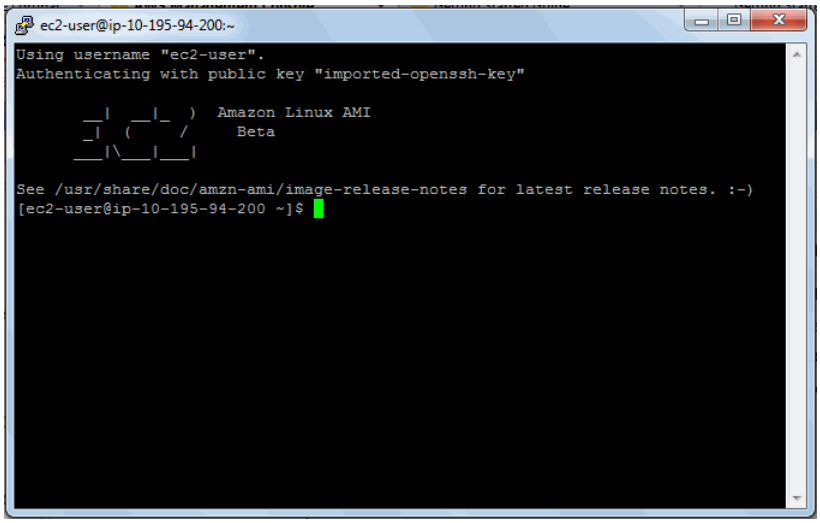


   b. Click **Open**. When the **PuTTY Security Alert** dialog box appears, click **Yes** to confirm that the fingerprint is OK. The **SSH PuTTY** window will open.

      Note
      The SSH fingerprint will eventually appear in the system log. You can use the SSH fingerprint as a comparison to protect against a man-in-the-middle attack. For more

information, go to Connecting Using PuTTY SSH in the *Amazon Elastic Compute Cloud User Guide*.

Your screen should look similar to this:



Now that you have successfully signed in to your instance, you are ready to configure it. For instructions on configuring your instance, see Configure the Amazon EC2 Instance (p. 18).

Connecting to Your Amazon EC2 Instance from a Linux/UNIX Machine Using a Standalone SSH Client

Use the `ssh` command to connect to your Linux/UNIX instance from a Linux/UNIX computer.

Note
Most Linux and UNIX computers include a Secure Shell (SSH) client by default. If yours doesn't, the OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, go to http://www.openssh.org.

To use SSH to connect

1. In a command line shell, change directories to the location of the private key file that you created in Step 4: Launch an Instance (p. 10).
2. Use the `chmod` command to ensure that your private key file isn't publicly viewable. For example, for `mykeypair.pem`, you would enter the following:

```
chmod 400 mykeypair.pem
```

3. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
4. In the top navigation bar, select **US East (N. Virginia)** in the region selector.
5. In the left navigation pane, click **Instances**.
6. Select your instance, and then click **Connect**.

7. Click **A standalone SSH client**. AWS automatically detects the public DNS address of your instance and the key pair name you launched the instance with.
8. Connect to your instance by using the public DNS name of the instance. For example, if the key file is `mykeypair.pem` and the instance's DNS name is ec2-184-72-209-110.compute-1.amazonaws.com, use the following command.

```
ssh -i mykeypair.pem ec2-user@ec2-184-72-209-110.compute-1.amazonaws.com
```

Note
We use **`ec2-user`** as the user name in this exercise for this AMI.

You'll see a response like the following.

```
The authenticity of host 'ec2-184-72-209-110.compute-1.amazonaws.com
(10.254.142.33)'
can't be established.
RSA key fingerprint is 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00.
Are you sure you want to continue connecting (yes/no)? yes
```

Note
The SSH fingerprint will eventually show up in the system log. You can use the SSH fingerprint as a comparison to protect against a man in the middle attack. For more information, go to Connect to Linux/UNIX Instances from Linux/UNIX with SSH in the *Amazon Elastic Compute Cloud User Guide*.

9. Enter **`yes`**.

You'll see a response like the following.

```
Warning: Permanently added 'ec2-184-72-209-110.compute-1.amazonaws.com'
(RSA)
   to the list of known hosts.
```

Now that you have successfully logged into your instance, you are ready to configure your instance. For instructions on how to configure your instance, see Configure the Amazon EC2 Instance (p. 18).

Configure the Amazon EC2 Instance

In this topic, we will configure the running instance. You will do the following tasks:

- Set permissions on the settings file
- Install MySQL Server.
- Start the web server and MySQL.
- Configure a database.
- Configure the application.

To simplify this tutorial, we are creating a database that will run locally on one Amazon EC2 instance. You will configure the Drupal application to use this Amazon EC2 instance for your database; all other Amazon EC2 instances will connect to this instance to access the database.

If you are going to use more than one Amazon EC2 instance, you will usually set up your database on a different server from the one that is running your application. That way, the information is stored in one

location, and all instances have access to the same data instead of a local running database that may be out of sync.

Creating a separate database is beyond the scope of this document. For more information about setting up Amazon RDS for your web application, go to Step 8: Add Amazon RDS inside *Getting Started with AWS Web Application Hosting for Linux*.

To set permissions on the settings file

- On your Amazon EC2 instance, at a command prompt, use the following command to set permissions:

```
sudo chmod 666 /var/www/html/sites/default/settings.php
```

To install a MySQL Server

- On your Amazon EC2 instance, at a command prompt, use the following command to install MySQL Server:

```
sudo yum install mysql-server
```

  When you are prompted, type 'y'.

To start the web server and MySQL

1. On your Amazon EC2 instance, at a command prompt, start the web server, and then configure it to start up automatically on reboot:

```
sudo chkconfig httpd on
```

```
sudo service httpd start
```

   You'll see a response like the following.

```
Starting httpd [OK]
```

2. Start mysql, and configure it to start up automatically on reboot.

```
sudo chkconfig mysqld on
```

```
sudo service mysqld start
```

   You'll see a response like the following.

```
Starting mysqld [OK]
```

To configure a database

1. On your Amazon EC2 instance, update the password for the 'root' user. In this example, you'll use the password 'root'.

```
mysqladmin -u root password root
```

2. Create a database. In this example, you'll use the database name 'mydb'.

```
mysqladmin -u root -p create mydb
```

When you are prompted for a password, type 'root'.

3. Sign in and set the access database rights.

```
mysql -u root -p
```

When you are prompted for a password, type 'root'.

4. At the MySQL prompt, set the permissions by using the following command. Replace <your public EC2 DNS address> with the public DNS address of the Amazon EC2 instance, which you recorded in Step 4: Launch an Instance (p. 10).

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, LOCK
TABLES, CREATE TEMPORARY TABLES ON mydb.* TO 'awsuser'@'<your public EC2
DNS address>' IDENTIFIED BY 'mypassword';
```

If successful, MySQL will reply with the following:

```
Query OK, 0 rows affected
```

5. At the MySQL prompt, set the permissions by using the following command.

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, LOCK
TABLES, CREATE TEMPORARY TABLES ON mydb.* TO 'awsuser'@'%' IDENTIFIED BY
'mypassword';
```

If successful, MySQL will reply with the following:

```
Query OK, 0 rows affected
```

6. To activate the new permissions, enter the following command:

```
FLUSH PRIVILEGES;
```

If successful, MySQL will reply with the following:

```
Query OK, 0 rows affected
```

7. Exit the MySQL prompt by typing the following:

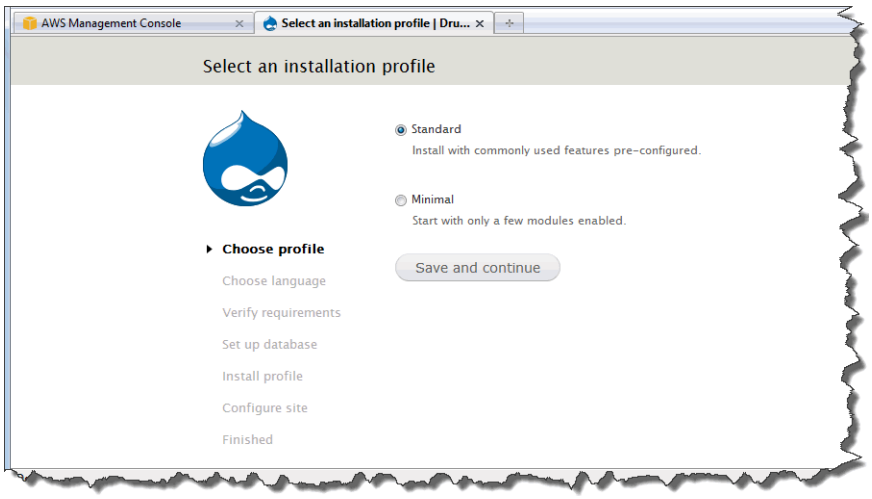
```
exit
```

The server responds with the following:

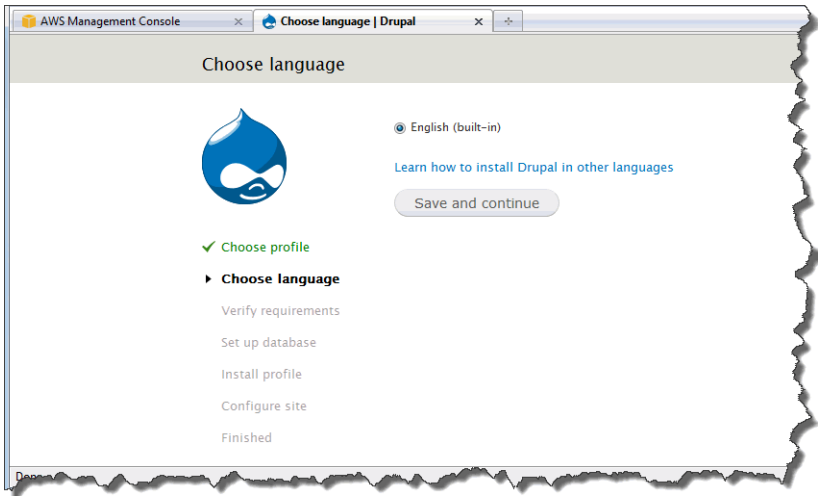
```
Bye
```

To configure the application

1. Open your web browser. In the Address box, type the public DNS address of the Amazon EC2 instance, which you recorded in Step 4: Launch an Instance (p. 10). The **Choose profile** page appears in the Drupal installation wizard.
2. On the **Choose profile** page, click **Standard**, and then click **Save and continue**.



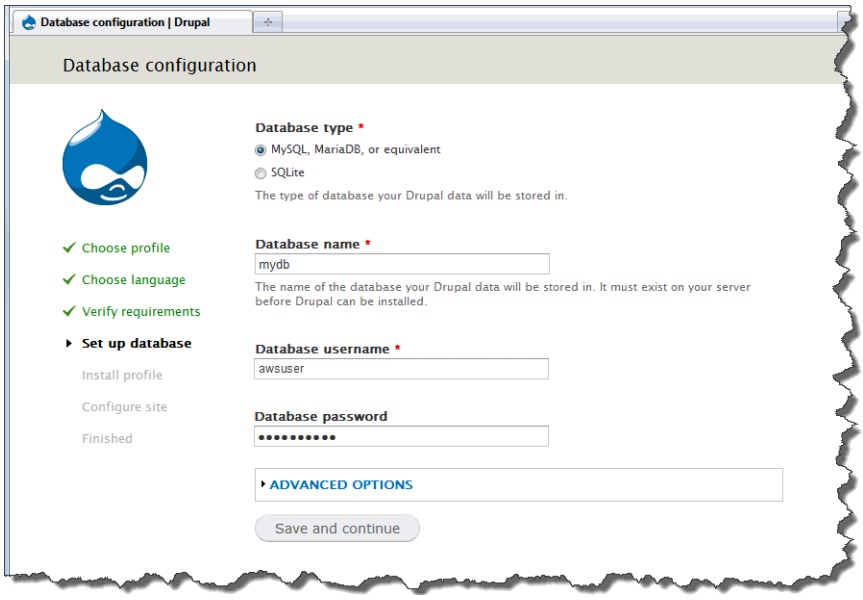
3. On the **Choose language** page, click **English**, and then click **Save and continue**. The **Set up database** page appears.



4. On the **Set up database** page, enter the following information.
   a. Under **Database type**, click **MySQL, MariaDB, or equivalent**.
   b. In the **Database name** box, type the name of your database. In this example, you'll use **`mydb`**.

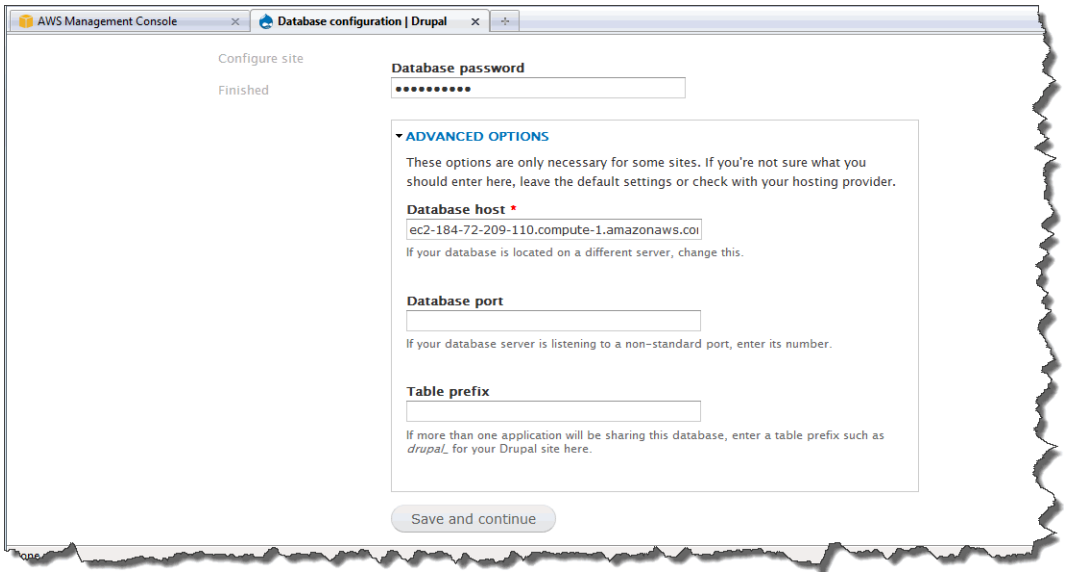
c. In the **Database username** box, type the user name for your database. In this example, you'll use **`awsuser`**.

d. In the **Database password** box, type the password for your database. In this example, you'll use **`mypassword`**.



e. Click **Advanced Options**.

f. In the **Database host** box, type the public DNS address of your Amazon EC2 instance, which you recorded in Step 4: Launch an Instance (p. 10).



g. Click **Save and continue**.

5. On the **Configure site** page, enter the following information.

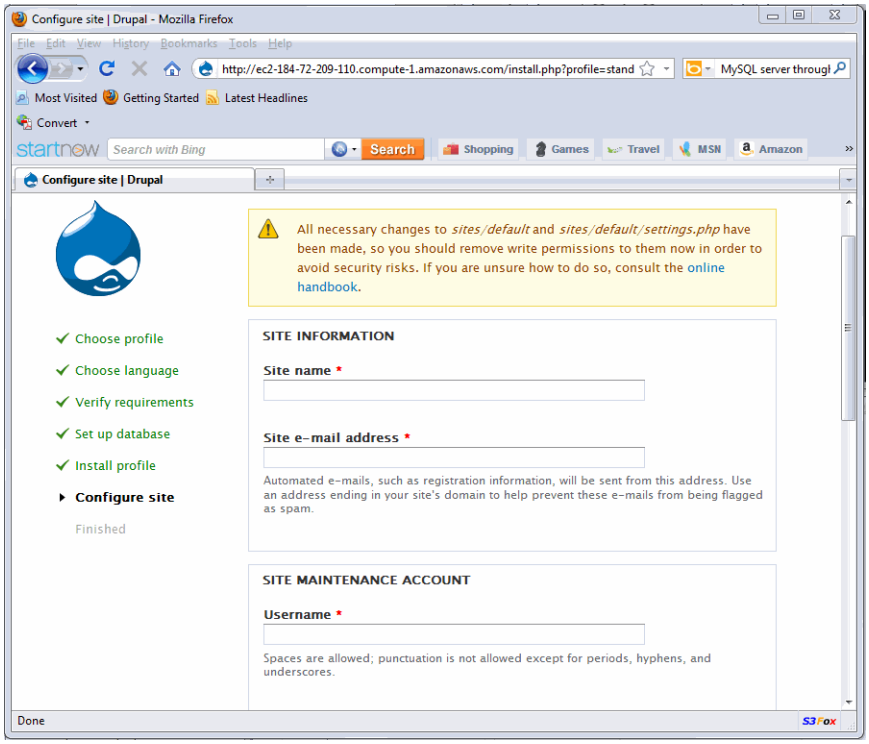
a. In the **Site name** box, type a name for your site.

b. In the **Site e-mail address** box, type your email address.

c. In the **Username** box, type a user name.

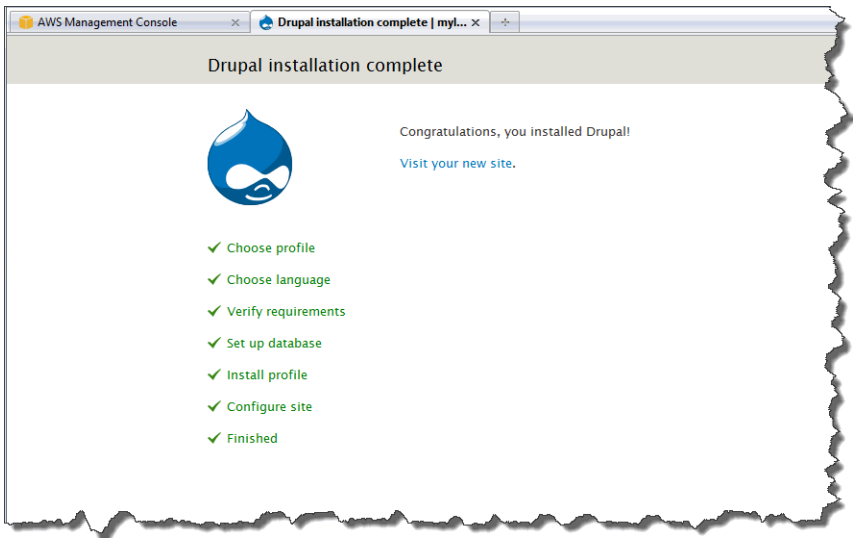
d. In the **Password** box, type a password that corresponds to the user name.

e. In the **Confirm password** box, retype the password.

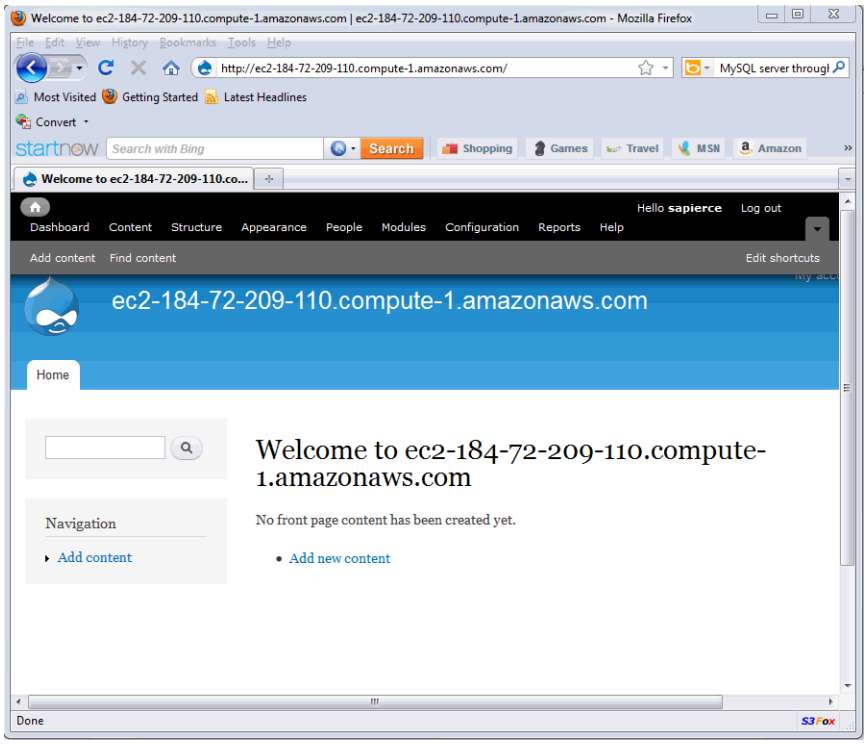


f. Click **Save and continue**.

The installation is complete.



6. Click **Visit your new site**.



7. To add a new article to your site, click **Add new content**.

Congratulations! You have successfully deployed your web application with Amazon Web Services. In the future, if you decide that you want to launch more instances, you won't want to customize each one. Let's create a custom AMI that incorporates all the configuration changes we've made.

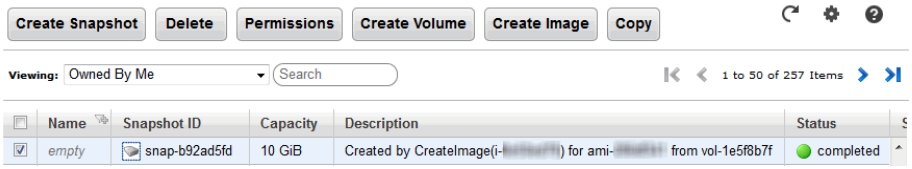
Step 6: Create a Custom AMI

Now that we have customized our Amazon EC2 instance, we can save this Amazon Machine Image (AMI) and launch future environments with this saved configuration.

To create an AMI from a running Amazon EC2 instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Make sure that **US East (N. Virginia)** is selected in the region selector of the navigation bar.
3. In the navigation pane, click **Instances**.
4. On the **Instances** page, right-click your running instance, and then click **Create Image**.
5. In the **Create Image** dialog box, fill in a unique image name and an optional description of the image (up to 255 characters), and then click **Create Image**.

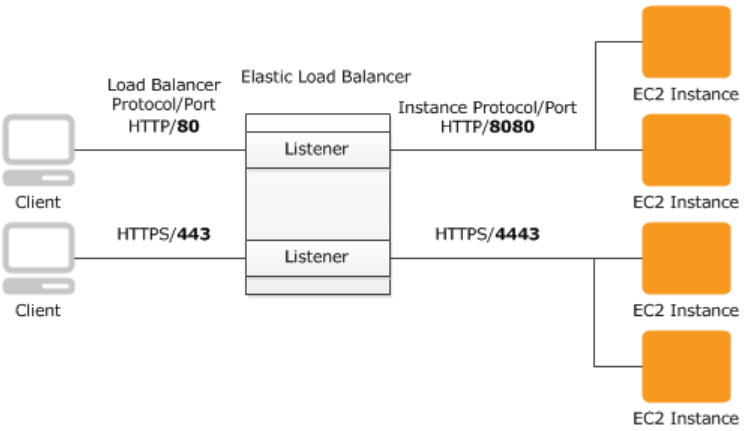
   Amazon EC2 terminates the instance, takes images of any volumes that were attached, creates and registers the AMI, and then relaunches the instance.
6. The **Create Image** dialog shows the AMI ID. Make a note of it; you will need it in a later task.
7. To view the status of the new AMI, click **AMIs** in the navigation pane. While the new AMI is being created, its status is *pending*. It takes a few minutes for the whole process to finish.
8. When the status of your AMI changes to *available*, go to the **Snapshots** page by clicking **Snapshots** in the navigation pane. View the new snapshot that was created for the AMI. Any instance that you launch from the new AMI uses this snapshot as its root device volume.



Eventually, you'll probably want to have multiple Amazon EC2 instances running across multiple Availability Zones. If one Availability Zone becomes unavailable, the traffic will be rerouted to another Availability Zone. An Elastic Load Balancer will enhance the availability of your application, whether all of your instances are in the same Availability Zone or in multiple Availability Zones. To create an Elastic Load Balancer, move on to Step 7: Create an Elastic Load Balancer (p. 25).

Step 7: Create an Elastic Load Balancer

Elastic Load Balancing automatically distributes and balances the incoming application traffic among all the instances you are running, improving the availability and scalability of your application. The service also makes it easy to add new instances or remove underused instances when you need to increase or decrease the capacity of your application. The following diagram shows how the load balancer works. In this diagram, the load balancer contains two listeners. The first listener accepts traffic on port 80 using HTTP and forwards these requests to the Amazon EC2 instances using HTTP on port 8080. The other listener accepts traffic on port 443 using HTTPS and forwards these requests to the Amazon EC2 instances using HTTPS on port 4443.



You can specify the protocol and port for both the client and the Amazon EC2 instances. In this step, we will create a load balancer for an HTTP service. We'll specify that the load balancer listen on port 80 for incoming traffic from clients and then distribute traffic on port 80 to the instances.

As soon as your load balancer becomes available, you're billed for each hour or partial hour that you keep the load balancer running. For more information about Elastic Load Balancing pricing, see the Elastic Load Balancing details page.

For more information about elastic load balancers, go to the Elastic Load Balancing Documentation.

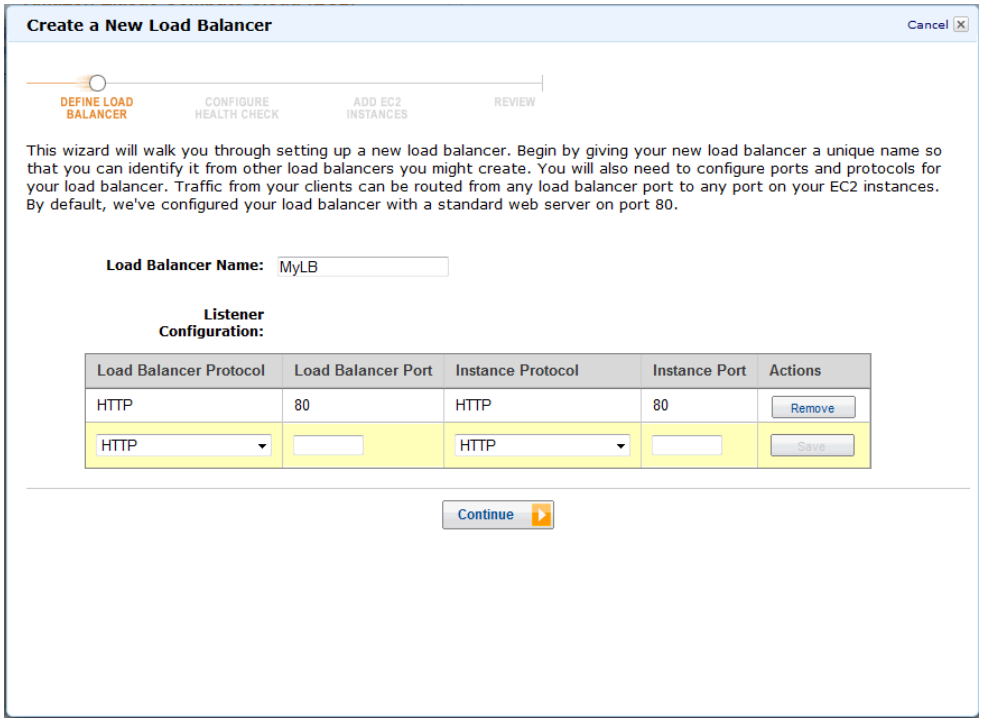
To create a load balancer

1. Define a load balancer:
   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   b. In the top navigation bar, click **US East (N. Virginia)** in the region selector.

c. In the left navigation pane, click **Load Balancers**.

d. Click **Create Load Balancer**.

e. In the **Create a New Load Balancer** wizard, on the **Define Load Balancer** page, enter a name for your load balancer. In this example, type **`MyLB`**.



f. Leave the **Listener Configuration** set to the default value for this example. The Load Balancer Port and Protocol specifies the port and protocol that the load balancer will use to listen for traffic from clients. The Instance Protocol and Port specifies the port and protocol the load balancer will use to route traffic to the instances. For example, if you want the load balancer to forward traffic to the instances using port 8080, you can specify that here.

> **Note**
> After you configure the listener information, you cannot change it. If you want to update this information, you will need to create a new load balancer.

g. Click **Continue**.

2. Configure the health check:

   Elastic Load Balancing routinely checks the health of each load-balanced Amazon EC2 instance. If Elastic Load Balancing finds an unhealthy instance, it stops sending traffic to the instance and reroutes traffic to healthy instances.

   a. On the **Configure Health Check** page under **Configuration Options**, do the following:

   - Leave **Ping Protocol** set to its default value of **HTTP**. In the future, if you want to use a more secure protocol for the load balancer to send ping requests to your instances, you can use HTTPS and specify a different port. For information on using HTTPS with Elastic Load Balancing, see Elastic Load Balancing Developer Guide in *Elastic Load Balancing Developer Guide*.
   - Leave **Ping Port** set to its default value of 80.

Elastic Load Balancing uses the ping port to send health check queries to your Amazon EC2 instances.

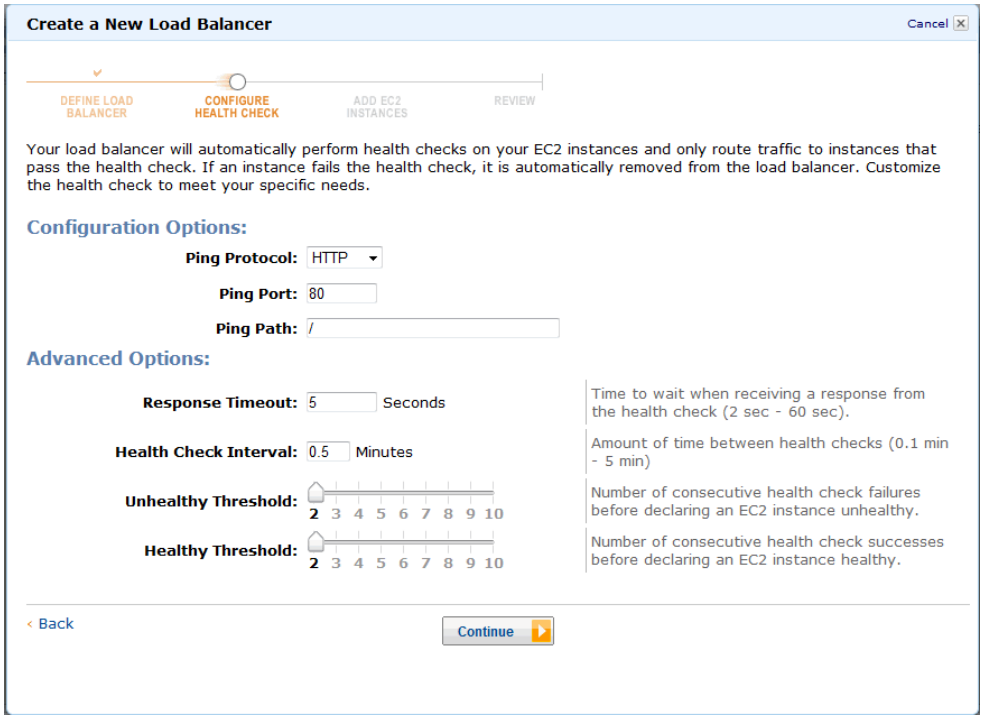
Note
If you specify a ping port value, your Amazon EC2 instances must accept incoming traffic on the port that you specify. You can set a port value other than 80, and you can change this value at any time.

- In the **Ping Path** box, replace the default value with a single forward slash ("/").

  Elastic Load Balancing sends health check queries to the ping path you specify. This example uses a single forward slash so that Elastic Load Balancing sends the query to your HTTP server's default home page, whether that default page is named `index.html`, `default.html`, or a different name. When you deploy your application, consider creating a special lightweight file that responds only to the health check. Doing so helps differentiate between traffic that is hitting your site and responses to the load balancer.

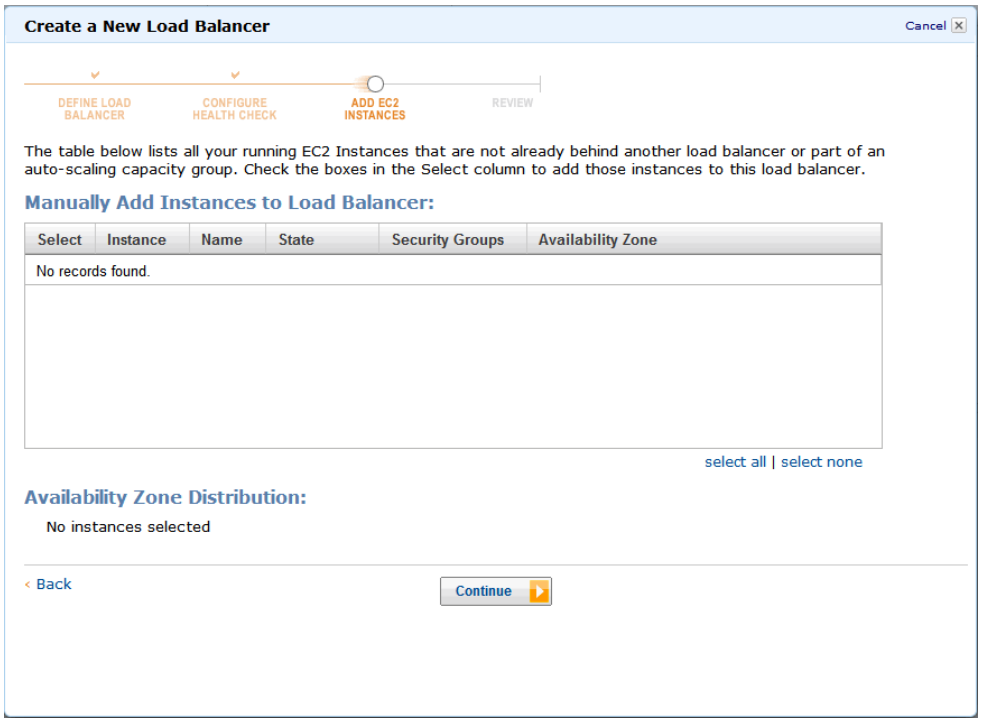
b. Under **Advanced Options**, set the **Healthy Threshold** to **2**. Accept the default values on the other options.

   Typically, the default value of 10 is fine for a healthy threshold. To expedite this tutorial, we specify 2 so you don't have to wait as long to see healthy instances.

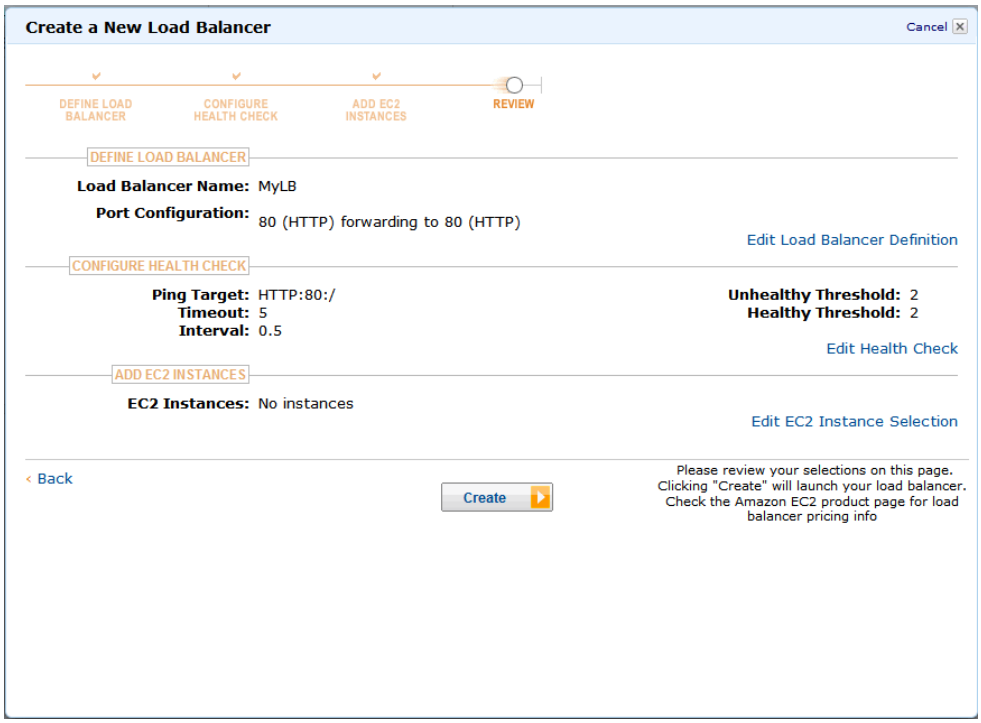


c. Click **Continue**.

3. On the **Add EC2 Instances** page, click **Continue**.



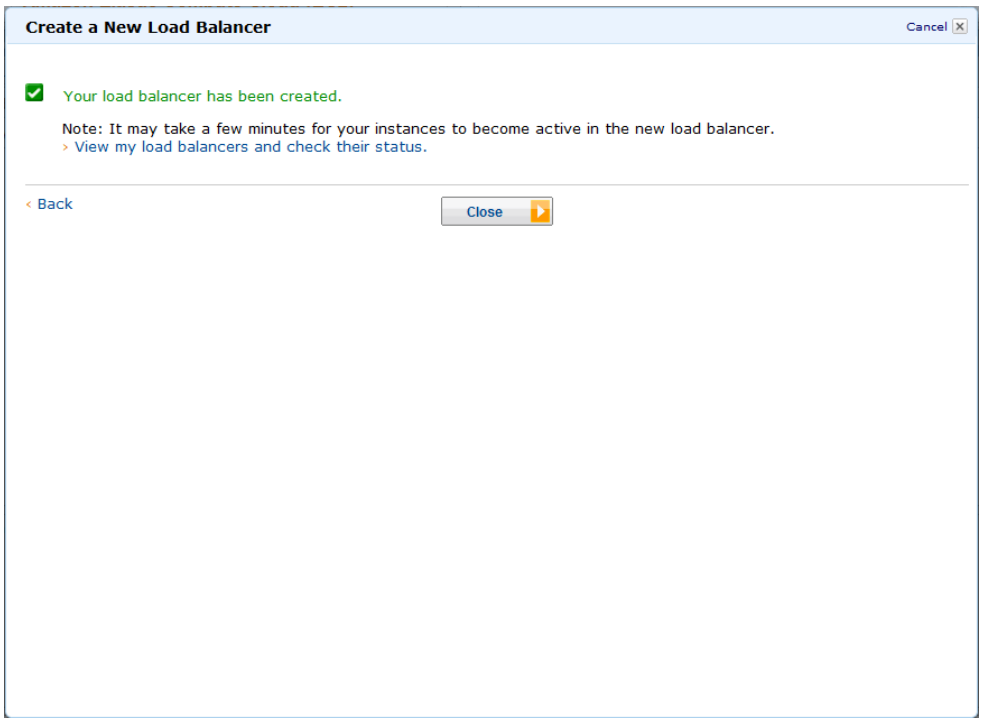
4. Review your settings. To make changes to the settings, click the **Edit** link for a specific step in the process.



Important

After you create a load balancer, you can modify any of the settings except for **Load Balancer Name** and **Port Configuration**. To rename a load balancer or change its port configuration, create a replacement load balancer.

5. Click **Create**.
6. On the **Confirmation** page, click **Close**.



Your new load balancer now appears in the list.

As a best practice, you should have sufficient instances across Availability Zones to survive the loss of any one Availability Zone. Therefore, we will ensure that our load balancer points to multiple Availability Zones in the next step.

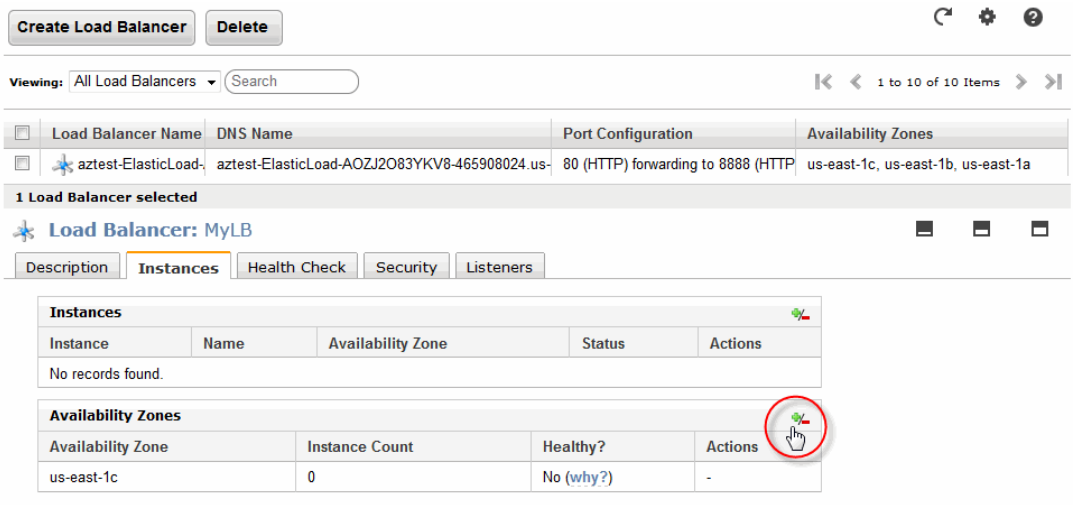
7. Record the public DNS address:

   a. In the **Load Balancers** pane, click **MyLB**.
   b. Click the **Description** tab.

   c. Write down the public DNS address. You will need it later in this tutorial.

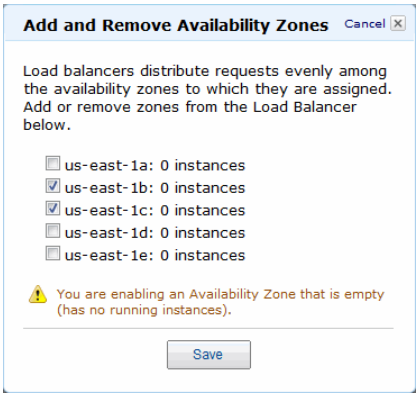
8. Add an Availability Zone:

a. In the list of load balancers, click **MyLB**.
b. Click the **Instances** tab.
c. Click the plus icon.



d. In the **Add and Remove Availability Zones** dialog box do the following:

- Click **us-east-1b: 0 instances**.
- Click **us-east-1c: 0 instances**.
- Click **Save**.



In a later task, you will launch instances in these two Availability Zones by using Auto Scaling. You'll see that the Availability Zones column for the load balancer is updated for both Availability Zones.

Where You're At

Here's where you are in building your architecture.

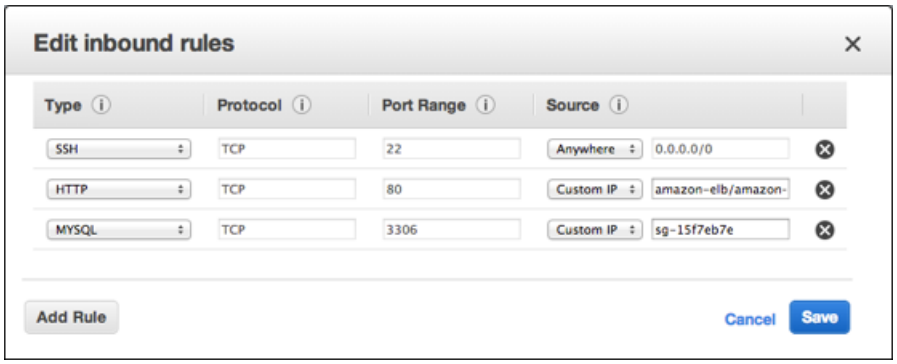
In Step 4: Launch an Instance (p. 10), you set a security group to allow all traffic to connect to your Amazon EC2 instance via port 80 (HTTP). Now that you have created an Elastic Load Balancer, you can update your security group to allow only incoming HTTP traffic from your Elastic Load Balancer. Move on Step 8: Update Your Amazon EC2 Security Group (p. 31).

Step 8: Update Your Amazon EC2 Security Group

In Step 4: Launch an Instance (p. 10), we created a security group that enabled HTTP over port 80. The security group allows all traffic to access the Amazon EC2 instance directly over HTTP/80. Because you created an Elastic Load Balancer, a more secure method is to allow only the load balancer to access your Amazon EC2 instance. In addition, because we launched two new instances with our Auto Scaling group, we want all the instances to access the information from one database so that the information presented to the user will stay in sync. To do so, we need to set up a new rule so that the new instances can query the database on the original instance by using MySQL. In this task, you will update your security group to allow only the load balancer to access your Amazon EC2 instance over HTTP/80 and allow only the instances inside the webappsecuritygroup to accept inbound traffic over 3306/MySQL. There are several ways you can setup your database, including setting up a dedicated database server or using Amazon RDS. Setting up a database is beyond the scope of this document. For more information about setting up Amazon RDS for your web application, go to Step 8: Add Amazon RDS inside *Getting Started with AWS Web Application Hosting for Linux*.

To configure your security group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the top navigation bar, click **US East (N. Virginia)** in the region selector.
3. In the left navigation pane, click **Load Balancers**.
4. Select the load balancer that you created earlier, and click the **Security** tab. In the **Source Security Group** field, copy or write down the name of the security group that's associated with the load balancer. You will need this name to update your instance's security group rules.
5. In the left navigation pane, click **Security Groups**.
6. On the **Security Groups** page, click the security group `webappsecuritygroup` that you created in the previous procedure. If you cannot see your security group, you may need to select **All security groups** from the filter list.
7. Click the **Inbound** tab, and click **Edit**.
8. In the row that displays port 80 (HTTP), select **Custom IP** from the **Source** field, and enter the name of the security group that's associated with your load balancer, for example, **`amazon-elb/amazon-elb-sg`**.
9. Click **Add Rule**.
10. Select **MYSQL** from the **Type** list.
11. In the **Source** field, type **`webappsecuritygroup`**. Select the security group ID for the webappsecuritygroup when it appears.



12. Click **Save**. The rules for this security group will be enforced when the instances that use these rules are launched.

Now that you have configured your Amazon EC2 security group, you can move on to Step 9: Launch Amazon EC2 Instances Using Auto Scaling (p. 32). Auto Scaling can adjust the number of running instances as traffic levels change.

Step 9: Launch Amazon EC2 Instances Using Auto Scaling

Auto Scaling launches and terminates Amazon EC2 instances automatically according to user-defined policies, schedules, and alarms. You can use Auto Scaling to maintain a fleet of Amazon EC2 instances that can adjust to any presented load. You can also use Auto Scaling to bring up multiple instances in a group at one time.

As the name implies, Auto Scaling responds automatically to changing conditions. All you need to do is specify how it should respond to those changes. For example, you can instruct Auto Scaling to launch an additional instance whenever CPU usage on one or more existing instances exceeds 60 percent for ten minutes, or you could tell Auto Scaling to terminate half of your website's instances over the weekend, when you expect traffic to be low.

Auto Scaling can ensure that the instances in your fleet are performing optimally so that your applications continue to run efficiently. Auto Scaling groups can even work across multiple Availability Zones, so that if an Availability Zone becomes unavailable, Auto Scaling will automatically redistribute traffic to applications in a different Availability Zone. With Auto Scaling, you can ensure that you always have at least one healthy instance running. For more information, see Auto Scaling.

In this example, you will set up the basic infrastructure that must be in place to get Auto Scaling started for most applications. You'll do the following:

- Create a launch configuration.
- Create an Auto Scaling group.
- Create a policy for your Auto Scaling group.

For the purposes of this tutorial, we'll set up an application running on Amazon EC2 to be load-balanced and auto-scaled with a minimum number of two instances and maximum number of two instances. By setting the minimum and maximum number to be the same, you can ensure that you always have the desired number of instances even if one instance fails. When you create your actual website, as a best practice you should launch sufficient instances across Availability Zones to survive the loss of any one

Availability Zone. Additionally, the maximum number of instances must be greater than the minimum to make use of the Auto Scaling feature.

You can control how big your fleet gets by specifying a maximum number of instances. In this example, Auto Scaling is configured to add one instance when there is an increase in load. We will define the policy in this topic, and in the next section we will create a CloudWatch alarm to take action on the policy when the average NetworkOut exceeds a threshold of 6,000,000 bytes for 5 minutes. Auto Scaling and Amazon CloudWatch work together to launch or terminate instances according to the policies you create. To save time, we will create just one policy; however, you can create more policies, such as a policy to terminate instances when load decreases.

If you haven't already installed the Auto Scaling command line tools, you need to do that now. For information go to, Using the Command Line Tools in the *Auto Scaling Developer Guide*. You will use the command line tools to set up Auto Scaling.

To set up an auto-scaled, load-balanced application running on Amazon EC2

1. Open a command prompt window: From a local Windows computer, click **Start**. In the **Search** box, type `cmd` and then press Enter.
2. The launch configuration is a template for the instances you launch in your Auto Scaling group. To define the launch configuration for this example, we will use the `as-create-launch-config` command. The following parameters define your launch configuration.

   - *image-id* is the AMI ID. Use the custom AMI ID that you created in Step 6: Create a Custom AMI (p. 24).
   - *instance-type* contains basic information, such as operating system, memory, and local storage, about the instance that you will launch. For this example, use the same instance type that you used when your first launched your instance.
   - *key* is the key pair used to connect to your instances. Use the same key pair that you created when you first launched your instance.
   - *group* is the security group where you defined the access rules for your instance. Use the same security group that you created when you first launched your instance.

     Note
     If you launched your instance and created your security group in a VPC, you will need to specify the security group's ID in the command, and not its name.
   - *monitoring-disabled* specifies that you want to use basic monitoring instead of detailed monitoring. By default, detailed monitoring is enabled. For more information about basic and detailed monitoring, go to Amazon CloudWatch.

   We will not specify a region, because we want to use the default region. At the command prompt, type the following, and then press Enter:

```
PROMPT>as-create-launch-config MyLC --image-id ami-95ce1afc --instance-type
 t1.micro --group webappsecuritygroup --key mykeypair --monitoring-disabled
```

   Auto Scaling returns output similar to the following example:

```
OK-Created launch config
```

   Note
   You can copy the commands from this document and paste them in the Command Prompt window. To paste the contents in the command line window, right-click in the Command Prompt window, and then click Paste. If you have trouble getting the commands to work, ensure that the command was entered correctly.

You have now created your launch configuration.

3. To create an Auto Scaling group in which you can launch multiple Amazon EC2 instances, you will use the `as-create-auto-scaling-group` command. Use the following parameters to define your Auto Scaling group.

   - *launch-configuration* is the name of the launch configuration that you created in the previous step.
   - *availability-zones* specifies the Availability Zones where the Amazon EC2 instances in the Auto Scaling group will be launched. In this example, you will specify two Availability Zones.

     Specifying multiple Availability Zones is a good practice for building fault-tolerant applications. If one Availability Zone experiences an outage, traffic will be routed to another Availability Zone. The number of instances that are launched in the Auto Scaling group will be evenly distributed across the Availability Zones.
   - *min-size and max-size* set the minimum and maximum number of Amazon EC2 instances in the Auto Scaling group. By setting the minimum and maximum number to be the same, you can fix the number of instances in your group. In this example, set both the minimum and maximum number to 2.
   - *load-balancer* is the name of the load balancer that is used to route traffic to the Auto Scaling group.

   At the command prompt, type the following, and then press Enter.

   ```
   PROMPT>as-create-auto-scaling-group MyAutoScalingGroup --launch-configuration
    MyLC --availability-zones us-east-1b, us-east-1c  --min-size 2 --max-size
    2 --load-balancers MyLB
   ```

   Note
   If you do not have permission to launch instances in us-east-1b, then try us-east-1d.

   Auto Scaling returns the following:

   ```
   OK-Created AutoScalingGroup
   ```

4. To create a policy to enlarge your fleet of instances, use the Auto Scaling `as-put-scaling-policy` command. This policy applies to your Auto Scaling group you created in the previous step. Use the following parameters when defining your Auto Scaling policy.

   - *auto-scaling-group* is the name of the Auto Scaling group that you want to apply the policy to. Use the Auto Scaling group name that you created in the previous step.
   - *adjustment* is the number of instances you want to increment or decrement. For this example, use 1.
   - *type* is the type of policy you want to create. For this example, use ChangeInCapacity to change the fleet size of your instances.
   - *cooldown* is the time, in seconds, after an action before Auto Scaling should evaluate conditions again.

   At the command prompt, type the following, and then press Enter:

```
PROMPT>as-put-scaling-policy MyScaleUpPolicy --auto-scaling-group MyAutoScal
ingGroup  --adjustment=1 --type ChangeInCapacity --cooldown 300
```

Auto Scaling returns output similar to the following example:

```
POLICY-ARN arn:aws:autoscaling:us-east-1:012345678901:scalingPolicy:cbe7da4e-
5d00-4882-900a-2f8113431e30:autoScalingGroupName/MyAutoScalingGroup:policy
Name/MyScaleUpPolicy
```

Note
To save time, we created only a policy to add an instance. In most cases, you would also create a policy to terminate one or more instances when traffic declines. Auto Scaling can decrease the number of instances when your application doesn't need the resources, saving you money. To create a policy for terminating an instance, start from the policy you just created, change the policy name, and then change the value of adjustment from 1 to -1. You use **"--adjustment=-1"** on a Windows machine.
At the command prompt, type the following, and then press Enter:

```
PROMPT>as-put-scaling-policy MyScaleDownPolicy --auto-scaling-group
MyAutoScalingGroup --adjustment=-1 --type ChangeInCapacity  --cooldown
 300
```

5. To verify that your Auto Scaling group exists, we'll use the `as-describe-auto-scaling-groups` command. At the command prompt, type the following, and then press Enter:

```
PROMPT>as-describe-auto-scaling-groups MyAutoScalingGroup --headers
```

Auto Scaling returns the following:

```
AUTO-SCALING-GROUP  GROUP-NAME          LAUNCH-CONFIG  AVAILABILITY-ZONES
    MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY
AUTO-SCALING-GROUP  MyAutoScalingGroup  MyLC           us-east-1b,us-east-
1c  2         2         2
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
INSTANCE  i-xxxxxxxx   us-east-1c         InService  Healthy  MyLC
INSTANCE  i-xxxxxxxx   us-east-1b         InService  Healthy  MyLC
```

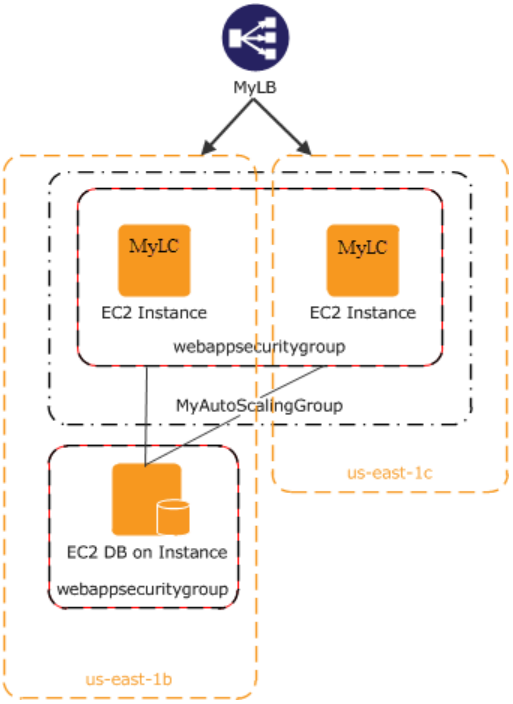
Your Amazon EC2 application has been launched as an auto-scaled and load-balanced application.

For more information about Auto Scaling, see the Auto Scaling Documentation.

Caution
You will continue to incur costs as long as your Amazon EC2 instances are running. If at any time you want to terminate these instances, see Terminate Your Amazon EC2 Instances in Your Auto Scaling Group (p. 45).

Where You're At

Here's where you are in building your architecture.



Now that you have created your Auto Scaling group and your Amazon EC2 instance is up and running, you'll want a way to monitor the health of your instance. In the next step, you'll create an Amazon CloudWatch alarm to track the Auto Scaling policy you just created.

Step 10: Create a CloudWatch Alarm

Amazon CloudWatch is a web service that enables you to monitor, manage, and publish various metrics and to configure alarm actions based on those metrics.

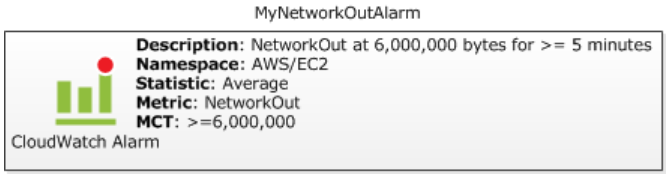
With Amazon CloudWatch, you can collect, analyze, and view system and application metrics so that you can make operational and business decisions quickly and confidently. Amazon CloudWatch automatically collects metrics about your AWS resources, such as the performance of your Amazon EC2 instances. You can publish your own metrics directly to Amazon CloudWatch.

You can use Amazon CloudWatch to diagnose problems by looking at system performance before and after a problem occurs. Amazon CloudWatch helps you identify the cause and verify your fix by tracking performance in real time. For example, you can set up Amazon CloudWatch to send you email right away when your application slows down, so you can go back and discover, for example, that a particular database was being overloaded. When you have fixed the problem, you can use Amazon CloudWatch to watch response times return to normal. For more information about creating CloudWatch alarms, go to Creating CloudWatch Alarms in the Amazon CloudWatch Developer Guide.

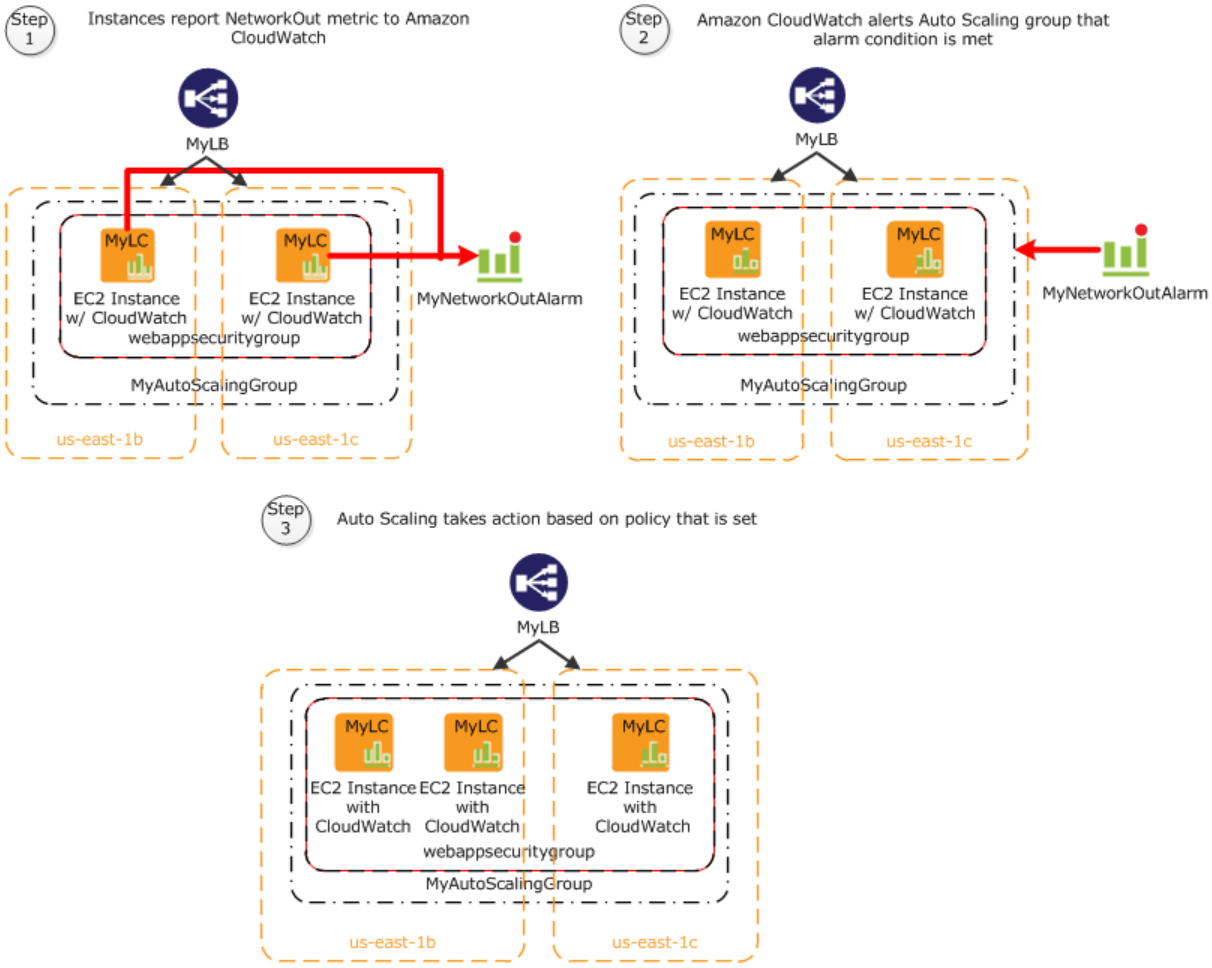
A common use for Amazon CloudWatch is to keep your applications and services healthy and running efficiently. For example, you can use it to discover that your website runs best when network traffic to your Amazon EC2 instances remains below a certain threshold. You can then create an Auto Scaling policy to ensure that you always have the right number of instances to match the amount of traffic you have.

In the previous task, we created an Auto Scaling policy to add to the number of running instances. In this task, we'll associate that policy with an alarm action. When the alarm is triggered, Auto Scaling is notified and makes the appropriate changes to your resources.

You'll create an alarm with the following characteristics:



The following diagram demonstrates how Amazon CloudWatch and Auto Scaling work together. The Amazon EC2 instance reports its NetworkOut metric to Amazon CloudWatch. Amazon CloudWatch fires an alarm if the specified threshold has been exceeded and reports this to the Auto Scaling Group. The Auto Scaling group then takes action based on the policy that is set.

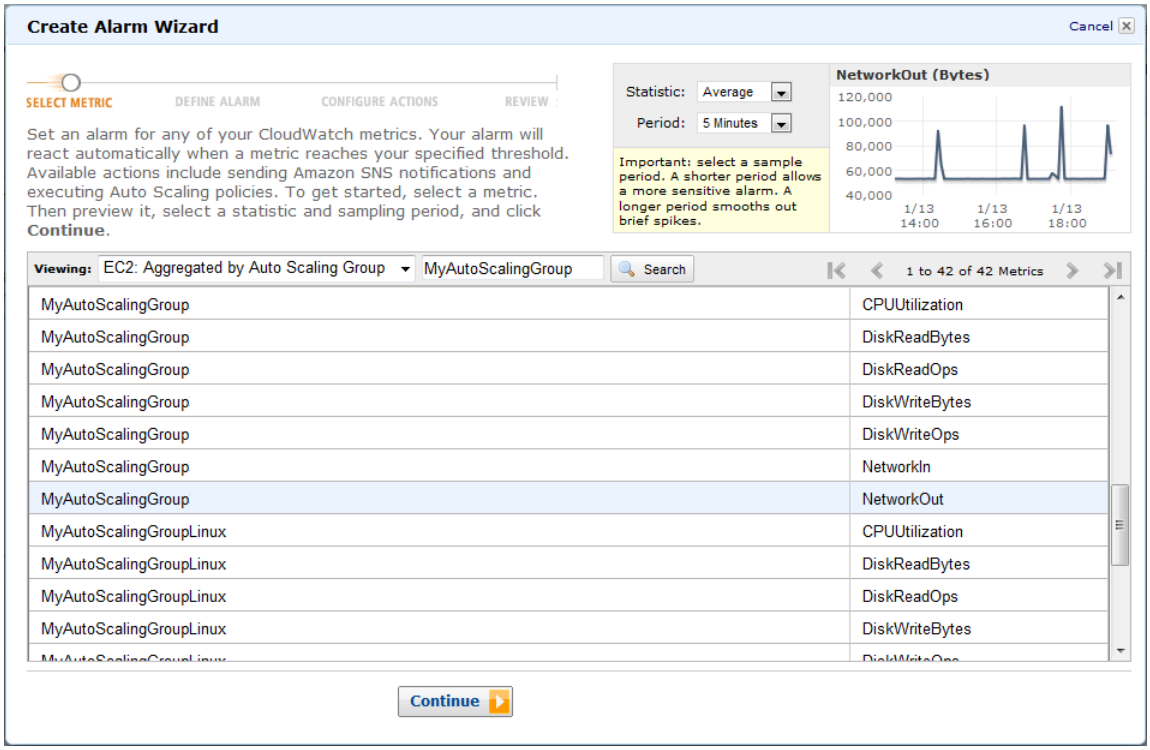


This topic walks you through creating a CloudWatch alarm to alert the application when the threshold is exceeded. To save time, we'll create just one alarm; however, you can apply the same procedure to create other alarms. For example, you could create another alarm to notify Auto Scaling that it needs to terminate an instance. For more information about Amazon CloudWatch, see the Amazon CloudWatch details page.

To create an Amazon CloudWatch alarm

1. Select a metric for your alarm:

a. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
b. In the top navigation bar, click **US East (N. Virginia)** in the region selector.
c. In the left navigation pane, click **Alarm**.
d. In the details pane, click **Create Alarm**.
e. In the **Create Alarm Wizard**, on the **Select Metric** page, in the **Viewing** list, select **EC2: Aggregated by Auto Scaling Group**.



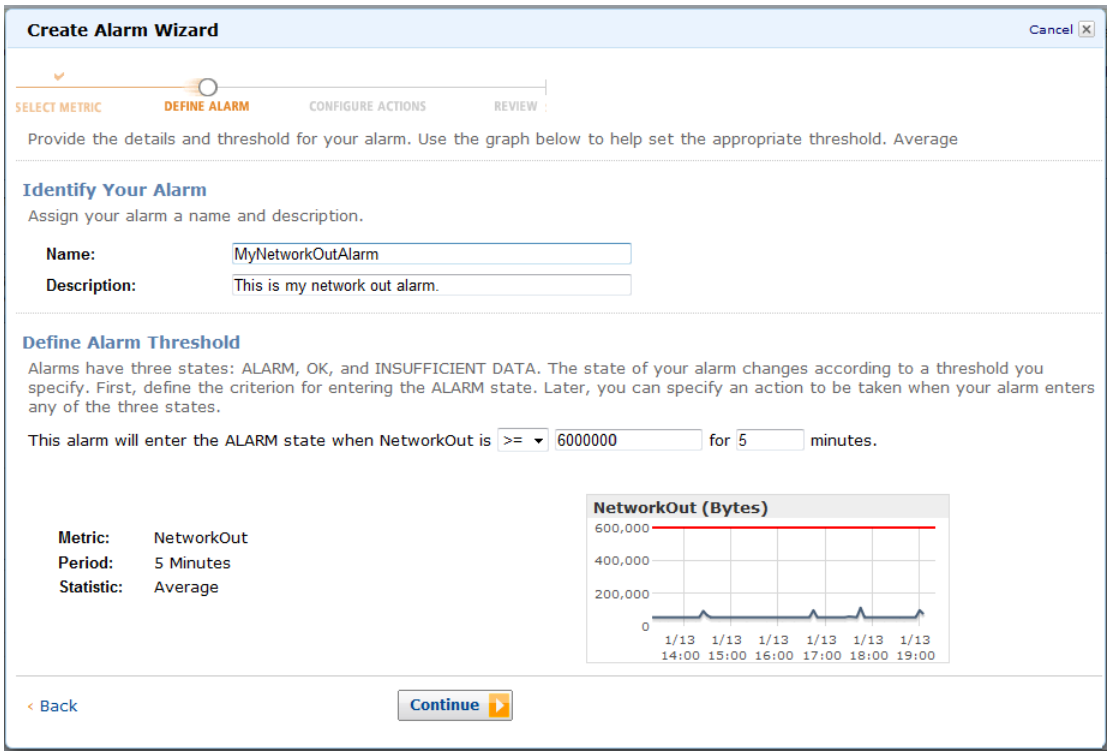
f. Click the **MyAutoScalingGroup/NetworkOut** row, and then click **Continue**.

> **Note**
> It can take up to 15 minutes for the Auto Scaling group to appear in the list. If you do not see your Auto Scaling group, wait up to 15 minutes, and then try again.

2. Define the alarm:

   On the **Define Alarm** page of the **Create Alarm** wizard, do the following, and then click **Continue**:

   - In the **Name** box, type **`MyNetworkOutAlarm`**.
   - In the **Description** box, type a description.
   - In the **Define Alarm Threshold** section, click **>=**, type **`6000000`** in the first box and **5** in the minutes box. For your own application, you can do some load testing to see what values make the most sense.



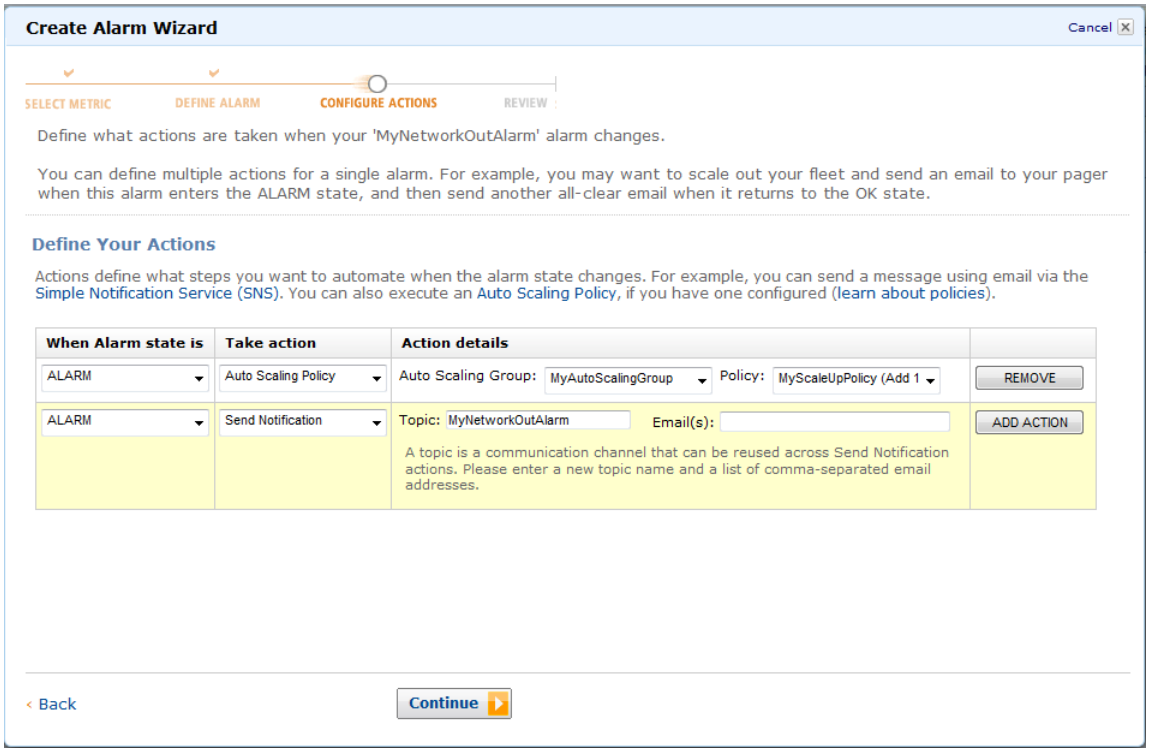
3. Define your actions:

   a. On the **Configure Actions** page of the **Create Alarm** wizard, do the following, and then click **Add Action**.

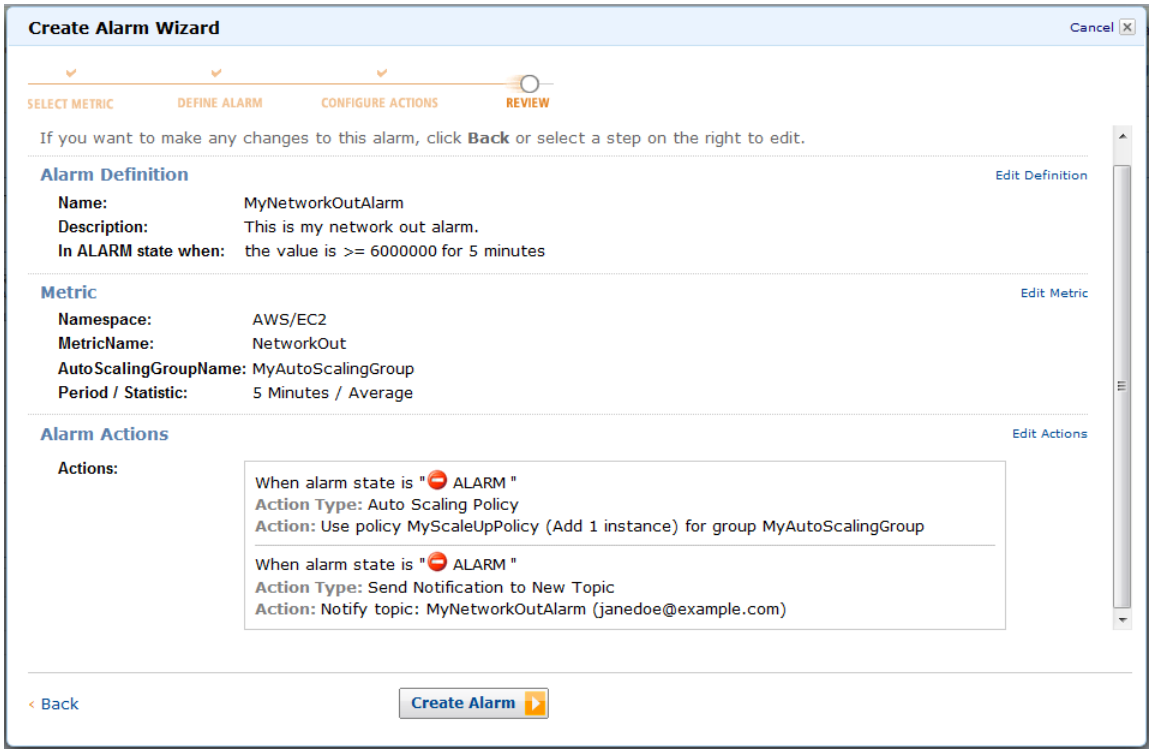
      - Under **When Alarm state is**, click **ALARM**.
      - Under the **Take Action** list, click **Auto Scaling Policy**.
      - In the **Auto Scaling Group** list, click **MyAutoScalingGroup**.
      - In the **Policy** list, click **MyScaleUpPolicy (Add 1 instance)**.

   b. Do the following, and then click **Continue**.

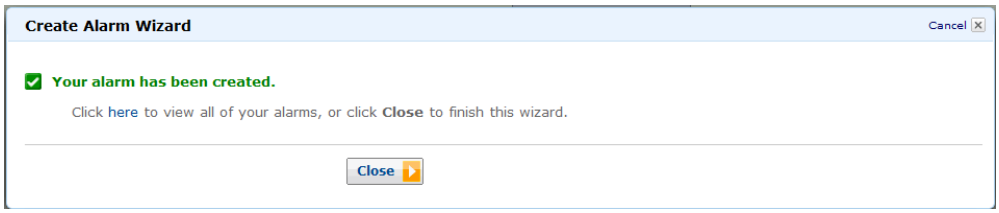
      - In the new row that is created, under **When Alarm state is**, click **ALARM**.
      - Under the **Take Action** list, click **Send Notification**.
      - In the **Topic** box, click **Create New Email Topic** and then type a topic name.
      - In the **Email(s)** box, type an email address where notifications will be sent.



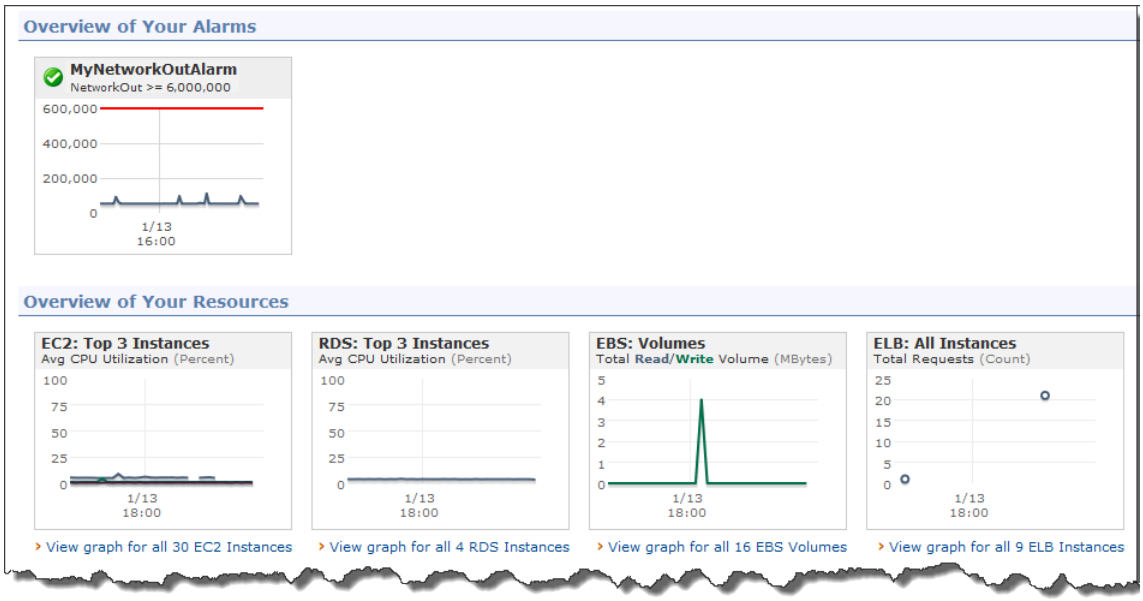
4. On the **Review** page, review the settings. If everything is all right, click **Create Alarm**.



5. On the confirmation page, click **Close**.



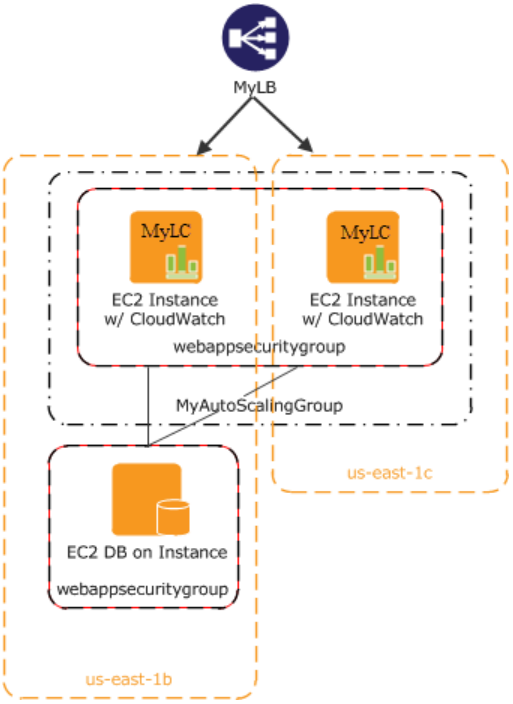
On the dashboard page of the Amazon CloudWatch console, your new alarm now appears in the list.



If you create a MyScaleDownPolicy, you can create another alarm using the same steps.

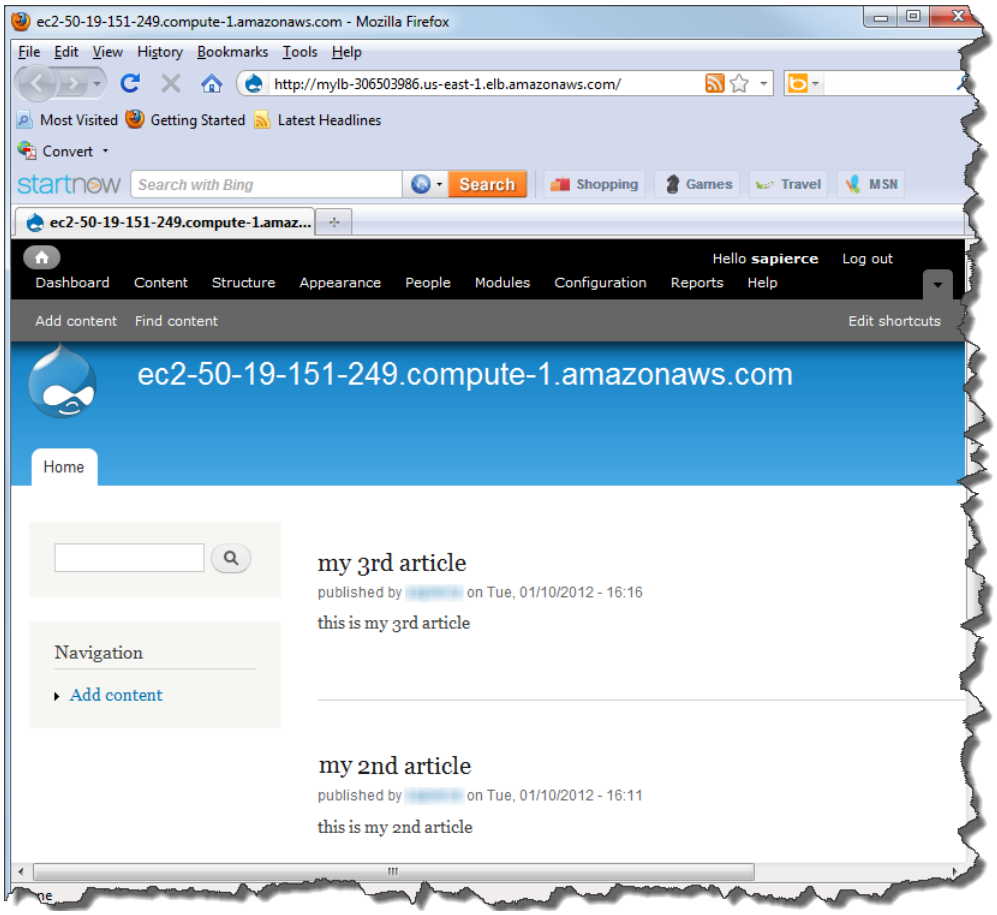
Where You're At

Here's where you are in building your architecture.



Congratulations! You have successfully deployed your web application to EC2 using the some of the essential building blocks of AWS. To verify that everything is working as it should, do the following:

1. Refresh your browser. You should no longer be able to access your website, because you updated your security group to allow access only through your Elastic Load Balancer.
2. Type the public DNS address of your Elastic Load Balancer which you recorded in Step 7: Create an Elastic Load Balancer (p. 25), to verify you can see your application.



In this tutorial, you learned how to deploy your web application by using the following AWS products:

- Amazon EC2 to run your application
- Elastic Load Balancing to load balance traffic across your running instances
- Auto Scaling to automatically add and terminate instances according to policies that you set
- Amazon CloudWatch to monitor your instances and to notify you when thresholds you specify are exceeded

When you have a better understanding of the AWS services and how you want to use them, there is an easier way you can deploy your application. AWS CloudFormation helps you deploy resources in AWS without worrying about the order in which AWS services need to be provisioned or the subtleties of how to make those dependencies work. To learn how to build sample template using the services we used in this tutorial, go to Auto Scaling Group with LoadBalancer, Auto Scaling Policies, and CloudWatch Alarms in the *AWS CloudFormation User Guide*.

If you are finished using your AWS resources, you can terminate them so that you are no longer billed. Move on to Step 11: Clean Up (p. 43).

Step 11: Clean Up

Topics

- Delete Your CloudWatch Alarm (p. 44)
- Delete Your Elastic Load Balancer (p. 44)
- Terminate Your Amazon EC2 Instances in Your Auto Scaling Group (p. 45)
- Terminate Your Instance (p. 47)
- Delete a Key Pair (p. 47)
- Delete an Amazon EC2 Security Group (p. 47)

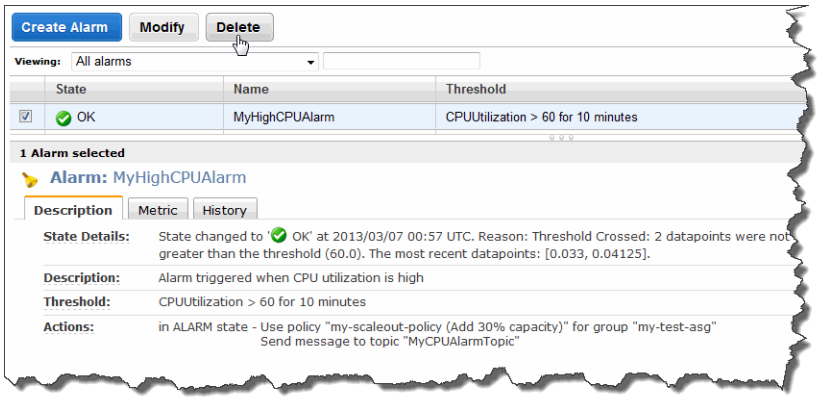
Congratulations! You have just deployed your web application. To prevent accruing any further charges, terminate your environments and clean up your resources.

Delete Your CloudWatch Alarm

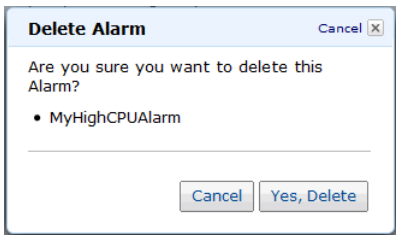
After you've decided that you no longer need the alarm, you can delete it.

To delete your alarm

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the top navigation bar, click **US East (N. Virginia)** in the region selector.
3. In the left navigation pane, click **Alarms**.
4. Select the check box next to the alarm that you want to delete, and then click **Delete**.



5. When a confirmation message appears, click **Yes, Delete**.



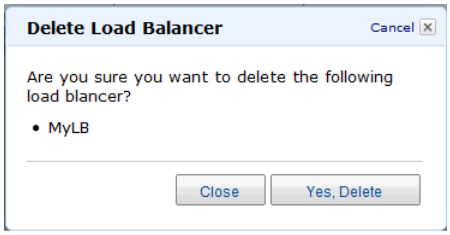
Delete Your Elastic Load Balancer

As soon as your load balancer becomes available, AWS bills you for each hour or partial hour that you keep the load balancer running. After you've decided that you no longer need the load balancer, you can delete it.

To delete your load balancer

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the top navigation bar, click **US East (N. Virginia)** in the region selector.
3. In the left navigation pane, click **Load Balancers**.
4. Select the check box next to the load balancer you want to delete and then click **Delete**.

5. When a confirmation message appears, click **Yes, Delete**.



Elastic Load Balancing deletes the load balancer. As soon as the load balancer is deleted, you stop incurring charges for that load balancer.

Caution
Even after you delete a load balancer, the Amazon EC2 instances associated with the load balancer continue to run. You will continue to incur charges on the Amazon EC2 instances while they are running.

Terminate Your Amazon EC2 Instances in Your Auto Scaling Group

In this section you will first remove the Amazon EC2 instance, then delete the Auto Scaling group, and finally delete the launch configuration.

You must terminate all Amazon EC2 instances in an Auto Scaling group before you can delete the group. A simple way to terminate all instances in a group is to update the group so that both the minimum size and maximum size are set to zero.

To remove the Amazon EC2 instance from the Auto Scaling group

1. Open a command prompt window: From a Windows computer, click **Start**. In the Search box, type `cmd`, and then press **Enter**.
2. You'll use the `as-update-auto-scaling-group` command to update the Auto Scaling group that we created earlier. At the command prompt, type the following, and then press **Enter**:

```
PROMPT>as-update-auto-scaling-group MyAutoScalingGroup --min-size 0 --max-size 0
```

Auto Scaling returns the following:

```
OK-Updated AutoScalingGroup
```

3. Now you'll use the `as-describe-auto-scaling-groups` command to verify that Auto Scaling has removed the instance from `MyAutoScalingGroup`.

   It can take a few minutes for the instance to terminate, so you might have to check the status more than once. At the command prompt, type the following, and then press **Enter**:

```
PROMPT>as-describe-auto-scaling-groups MyAutoScalingGroup --headers
```

   If the instance termination is still in progress, Auto Scaling returns information similar to the following. (Your value for `INSTANCE-ID` will differ):

```
AUTO-SCALING-GROUP  GROUP-NAME          LAUNCH-CONFIG  AVAILABILITY-ZONES
    LOAD-BALANCERS  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY
AUTO-SCALING-GROUP  MyAutoScalingGroup  MyLC           us-east-1b,us-east-
1c  MyLB            0         0         0
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
INSTANCE  i-xxxxxxxx   us-east-1c         InService  Healthy  MyLC
```

   Note
   You can also click **Instances** in the Amazon EC2 console to view the status of your instances.

When no instances exist in `MyAutoScalingGroup`, you can delete the group.

To delete the Auto Scaling group

- At the command prompt, type the following, and then press **Enter**:

```
PROMPT>as-delete-auto-scaling-group MyAutoScalingGroup
```

  To confirm the deletion, type `Y`, and then press **Enter**.

```
Are you sure you want to delete this MyAutoScalingGroup? [Ny]
```

  Auto Scaling returns the following:

```
OK-Deleted MyAutoScalingGroup
```

All that remains now is to delete the launch configuration you created for this Auto Scaling group.

To delete the launch configuration

- At the command prompt, type the following, and then press **Enter**:

```
PROMPT>as-delete-launch-config MyLC
```

  To confirm the deletion, type `Y` and then press **Enter**.

```
Are you sure you want to delete this launch configuration? [Ny]
```

Auto Scaling returns the following:

```
OK-Deleted launch configuration
```

Terminate Your Instance

As soon as your instance starts to boot, AWS bills you for each hour or partial hour that you keep the instance running, even if the instance is idle. You can terminate the instance so you are no longer charged for it. Because this instance is not part of your Auto Scaling group, you'll need to terminate it manually.

To terminate an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the top navigation bar, click **US East (N. Virginia)** in the region selector.
3. In the left navigation pane, click **Instances**.
4. Right-click the instance, and then click **Terminate**.
5. When you are prompted for confirmation, click **Yes, Terminate**. As soon as the instance status changes to **shutting down** or **terminated**, you stop incurring charges for that instance.

Delete a Key Pair

This is an optional step. You are not charged for keeping a key pair, and you may want to reuse the key pair for later use.

To delete a key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the top navigation bar, click **US East (N. Virginia)** in the region selector.
3. In the left navigation pane, click **Key Pairs**.
4. Select the check box beside the key pair you want to delete, and then click **Delete**.
5. When a confirmation message appears, click **Yes**.

Delete an Amazon EC2 Security Group

To delete a security group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the top navigation bar, click **US East (N. Virginia)** in the region selector.
3. In the left navigation pane, click **Security Groups**.
4. In the details pane, under **Security Groups**, select a security group you want to delete, and then click **Delete**.
5. Click **Yes, Delete**.

Pricing

Topics

- Amazon EC2 Cost Breakdown (p. 48)
- Summing It All Up (p. 51)
- How To Further Save Costs (p. 52)

The AWS Simple Monthly Calculator estimates your monthly bill. It provides a per-service cost breakdown, as well as an aggregate monthly estimate. You can also use the calculator to see an estimate and breakdown of costs for common solutions. This topic walks you through an example of using the AWS Simple Monthly Calculator to estimate your monthly bill.

> **Note**
> AWS pricing you see in this documentation is current at the time of publication. For the latest pricing information, go to AWS Service Pricing Overview. For more information on how AWS pricing works, go to How AWS Pricing Works.

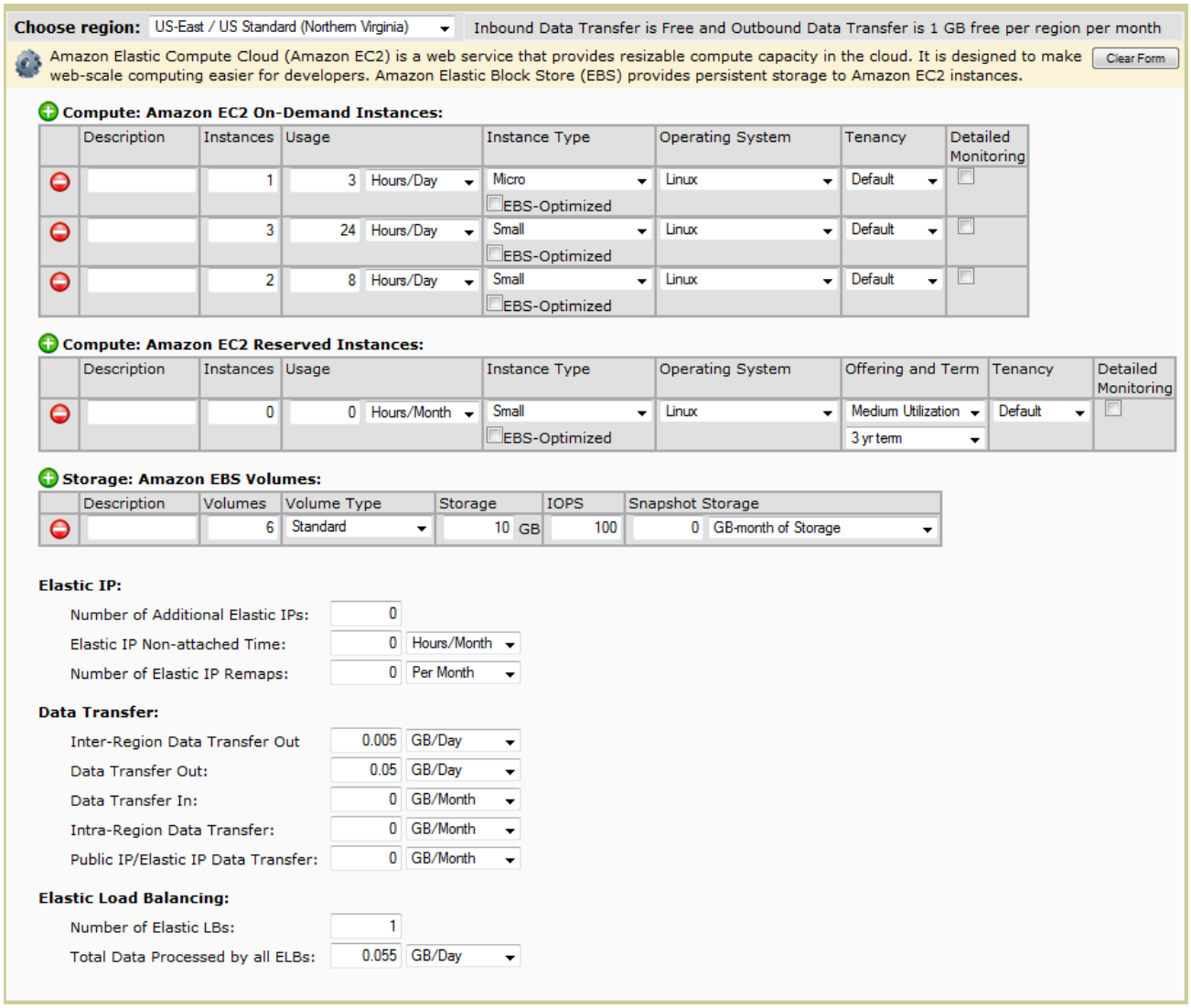
Amazon EC2 Cost Breakdown

The following table shows the characteristics for Amazon EC2 we have identified for this web application architecture. In this example, we'll assume that you've moved into full production and you need between three and six instances. Three instances run all the time, two additional instances are required to handle peak traffic times, and another instance handles nightly backups.

Characteristic	Metric	Description
Clock Hours of Server Time	3 instances running 24 hours/day 2 instances running 8 hours/day 1 instances running 3 hours/day	Assuming an average of 30.5 days in a month, the full-time instances run 732 hours/month, the peak traffic instances run 244 hours/month, and the backup instances run 91.5 hours/month

Characteristic	Metric	Description
Machine Characteristics	1 ti.micro instance 5 m1.small instances	Micro - 613 MB of memory, up to 2 EC2 Compute Units (for short periodic bursts), Elastic Block Store (EBS) storage only, 32-bit or 64-bit platform Small- 1.7 GB of memory, 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit), 160 GB of local instance storage, 32-bit platform For a list of instance types, go to http://aws.amazon.com/ec2/instance-types/.
Additional Storage	1 EBS Volume Storage: 10 GB/Month 100 IOPS	The AMI is EBS-backed. The volume will have 10 GB provisioned storage, and 100 I/O requests per second made to the volume.
Data Transfer	Data In: 0.005 GB/day Data Out: 0.05 GB/day	There are approximately 1,000 hits per day. Each response is about 50 KB, and each request is about 5 KB.
Instance Scale	Between 3 and 6 instances	You need 3 instances running all the time, another two to handle peak traffic, and another to handle nightly backups.
Elastic Load Balancing	Hourly usage: 732 hrs/month Data processed: 1.525 GB/month	Elastic Load Balancing is used 24 hrs/day, 7 days/week Elastic Load Balancing processes a total of 0.055 GB/day (data in + data out)

The following image from the AWS Simple Monthly Calculator shows the cost breakdown for Amazon EC2.



Choose region: US-East / US Standard (Northern Virginia) — Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon Elastic Block Store (EBS) provides persistent storage to Amazon EC2 instances. Clear Form

Compute: Amazon EC2 On-Demand Instances:

Description	Instances	Usage	Instance Type	Operating System	Tenancy	Detailed Monitoring
	1	3 Hours/Day	Micro, EBS-Optimized	Linux	Default	
	3	24 Hours/Day	Small, EBS-Optimized	Linux	Default	
	2	8 Hours/Day	Small, EBS-Optimized	Linux	Default	

Compute: Amazon EC2 Reserved Instances:

Description	Instances	Usage	Instance Type	Operating System	Offering and Term	Tenancy	Detailed Monitoring
	0	0 Hours/Month	Small, EBS-Optimized	Linux	Medium Utilization, 3 yr term	Default	

Storage: Amazon EBS Volumes:

Description	Volumes	Volume Type	Storage	IOPS	Snapshot Storage
	6	Standard	10 GB	100	0 GB-month of Storage

Elastic IP:

Number of Additional Elastic IPs:	0	
Elastic IP Non-attached Time:	0	Hours/Month
Number of Elastic IP Remaps:	0	Per Month

Data Transfer:

Inter-Region Data Transfer Out	0.005	GB/Day
Data Transfer Out:	0.05	GB/Day
Data Transfer In:	0	GB/Month
Intra-Region Data Transfer:	0	GB/Month
Public IP/Elastic IP Data Transfer:	0	GB/Month

Elastic Load Balancing:

Number of Elastic LBs:	1	
Total Data Processed by all ELBs:	0.055	GB/Day

The total monthly cost is the sum of the cost of the running instances, Amazon Elastic Block Store volumes and I/O requests, Elastic Load Balancer, and the data processed by the Elastic Load Balancers. Because we used basic monitoring and only one metric and alarm for our Amazon EC2 instances, there is no additional charge for Amazon CloudWatch monitoring.

Variable	Formula	Calculation
Instance Cost	Instance cost per hour	$0.060
	Number of instances	3
	x Clock hours of server time	x 732
	----------	-------
		$131.76

Variable	Formula	Calculation
Instance Cost	Instance cost per hour Number of instances x Clock hours of server time ----------	$0.060 2 x 244 ------- $29.28
Instance Cost	Instance cost per hour Number of instances x Clock hours of server time ----------	$0.02 1 x 91.5 ------- $1.83
Additional Storage	Storage rate x Storage Amount (GB) + (I/O requests rate x seconds per month x Request rate(per 1M requests)) x Number of Volumes _____	$0.10 X 10 + (100 x ~2.6M x $0.10)/1M x 6 ---------- $164.11
Elastic Load Balancing	Hours used x Hourly rate + (Data processed (GB) x Process rate) ----------	732 x $0.025 + 1.6775 x $0.008 ---------- $18.31
Total Cost Per Month		$345.29

Move on to Summing It All Up (p. 51) to view a summary of the total charges including AWS Data Transfer Out and the Free Usage Tier discounts.

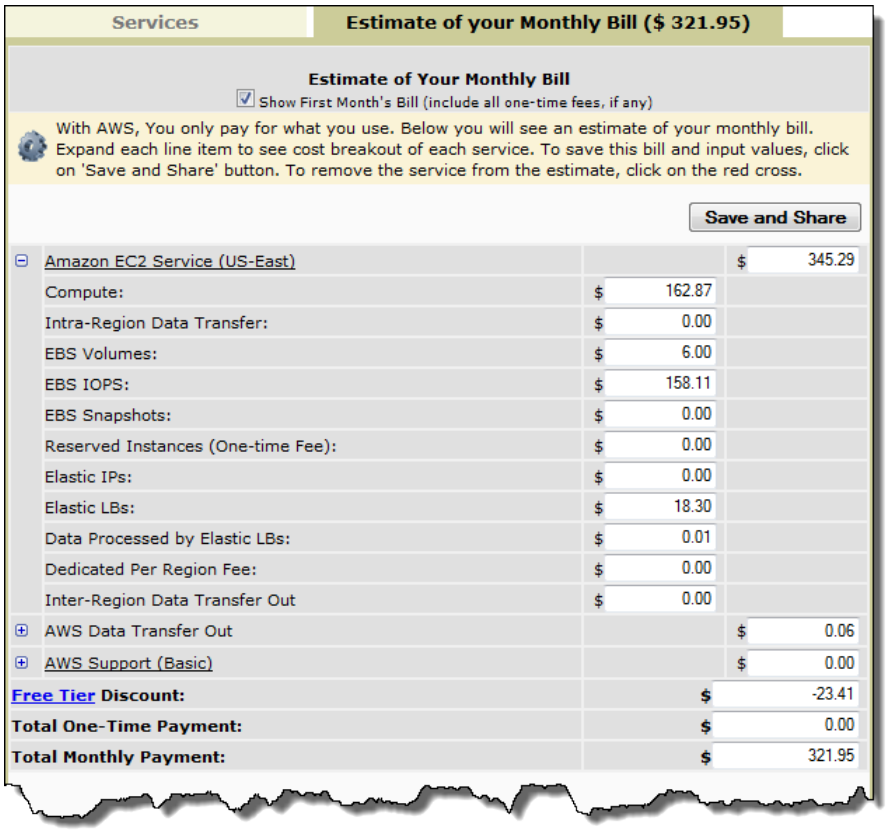
Summing It All Up

To calculate the total cost for this example, we add the cost for Amazon EC2 instances and the AWS Data Transfer Out and then subtract any discount that falls into the AWS free usage tier. To learn more about the free usage tier and to find out if you are eligible, go to Getting Started with AWS Free Usage Tier.

The total AWS Transfer Out is an aggregate Data Transfer Out usage across all Amazon EC2 instances. For Amazon EC2, we have 0.05 GB per day, which is approximately 1.525 GB per month. Because up to 1 GB per month of data transferred out is free, we are left with a total of 0.525 GB per month.

Variable	Formula	Calculation
AWS Data Transfer	(Data in (GB) X Data In Rate) + (Data out (GB) X Data Out Rate) ----------	0.1525 X $0.00 + (0.525) X $0.12 ---------- $0.06

The following image shows an example of your monthly estimate.



According to the calculator, the total cost for Amazon EC2 is $321.95.

How To Further Save Costs

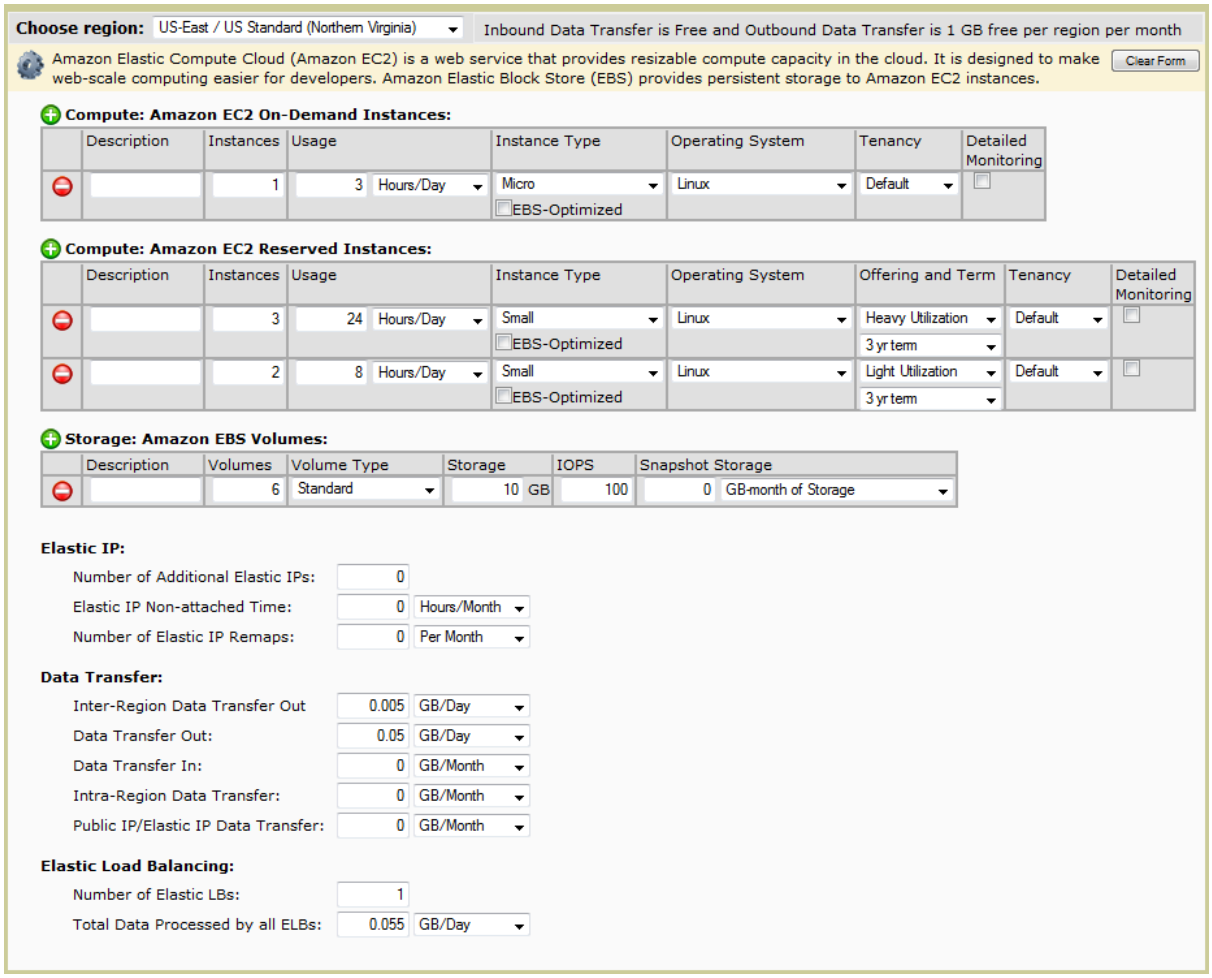
In the example deployment we have been discussing, we used On-Demand Instances for all six of our instances. With On-Demand Instances, you are charged only from the time you launch an instance until the time you terminate it. If you plan to be running your instances for a long time, you can save more money by reserving them.

To obtain Reserved Instances, you make a low, one-time payment for each instance you want to reserve. In return, you receive a significant discount on the hourly usage charge. If you know approximately how heavily your Amazon EC2 instances will be used when they are running, you can save even more by opting for Heavy, Medium, or Light Utilization Reserved Instances. With Heavy Utilization, you pay a

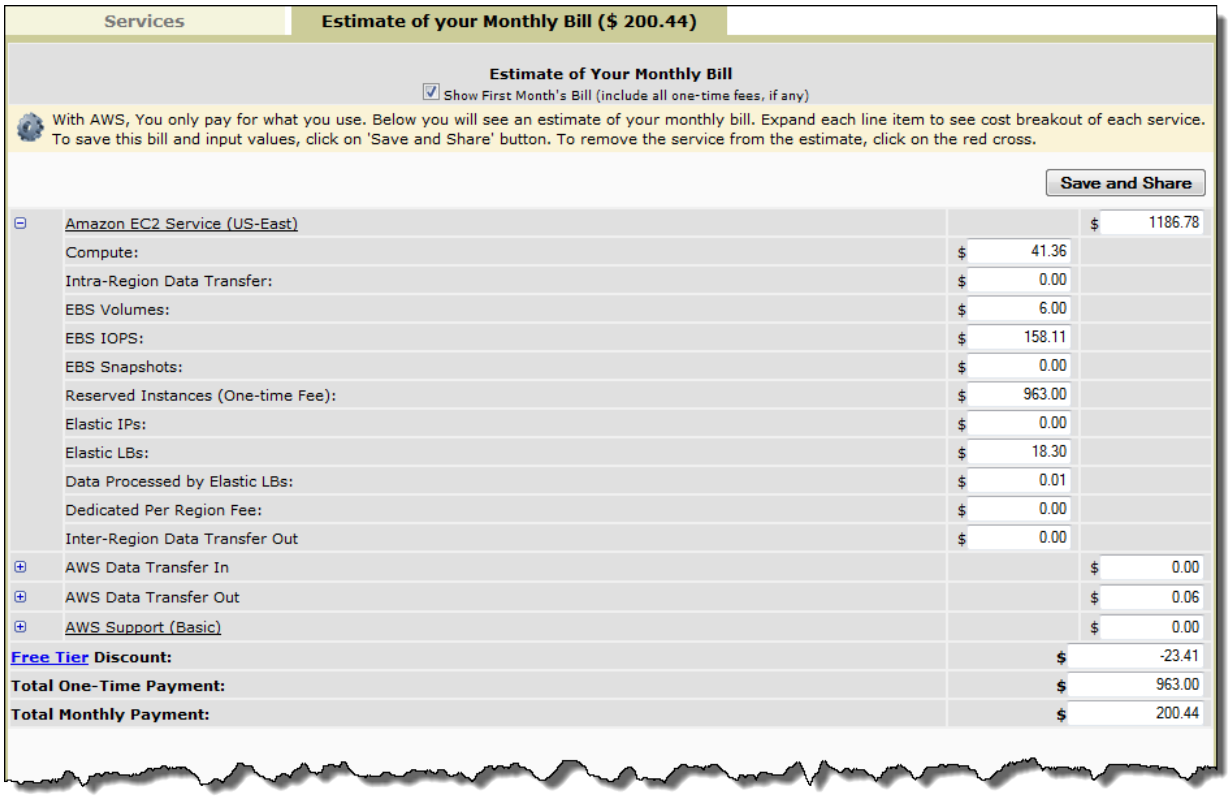
higher upfront fee, but your hourly usage rate is the lower than that for Medium and Light Utilization Reserved Instances. Light Utilization has the lowest upfront free, but your hourly rate is higher than that for Medium and Heavy Utilization Instances. In the previous example, three of the instances are running all the time. This is an ideal candidate for Heavy Utilization Reserved Instances. Two instances run only during peak traffic, about one third of the time. These instances are ideal candidates for Light Utilization Reserved Instances. Because the instance that performs the nightly backups runs only a few hours a day, you can run it as an On-Demand Instance.

Reserved Instances can be obtained on 1-year or 3-year terms. The 3-year term can offer additional savings over the 1-year term. For more information about reserved instances, go to Amazon EC2 Reserved Instances. You can see the cost comparison with On-Demand versus Reserved Instances over a three-year period in the following table.

Using the same characteristics and metrics in the above example, let's update the calculator to enter the Heavy and Light Utilization as in the following diagram.



The total monthly cost is calculated the same way as the previous example, except that there is an additional one-time fee for Reserved Instances. The total cost is shown in the following diagram.



Services | Estimate of your Monthly Bill ($ 200.44)

Estimate of Your Monthly Bill

Show First Month's Bill (include all one-time fees, if any)

With AWS, You only pay for what you use. Below you will see an estimate of your monthly bill. Expand each line item to see cost breakout of each service. To save this bill and input values, click on 'Save and Share' button. To remove the service from the estimate, click on the red cross.

Save and Share

⊖	Amazon EC2 Service (US-East)		$ 1186.78
	Compute:	$ 41.36	
	Intra-Region Data Transfer:	$ 0.00	
	EBS Volumes:	$ 6.00	
	EBS IOPS:	$ 158.11	
	EBS Snapshots:	$ 0.00	
	Reserved Instances (One-time Fee):	$ 963.00	
	Elastic IPs:	$ 0.00	
	Elastic LBs:	$ 18.30	
	Data Processed by Elastic LBs:	$ 0.01	
	Dedicated Per Region Fee:	$ 0.00	
	Inter-Region Data Transfer Out	$ 0.00	
⊕	AWS Data Transfer In		$ 0.00
⊕	AWS Data Transfer Out		$ 0.06
⊕	AWS Support (Basic)		$ 0.00
Free Tier Discount:			$ -23.41
Total One-Time Payment:			$ 963.00
Total Monthly Payment:			$ 200.44

The following table compares the total costs for using a mix of Heavy and Light Utilization Reserved Instances with those for On-Demand Instances.

Instance	Monthly Cost	One-time Fee	Total Cost (3 years)
6 On-Demand Instances	$345.29	n/a	$12430.44
1 On-Demand Instance 3 Heavy Utilization Reserved Instances 2 Light Utilization Reserved Instances	$200.44	$963.00	$8178.84

As you can see from the table, by using a mix of Heavy and Light Utilization Reserved Instances in this example, you can save approximately 30%. For more information on how AWS pricing works, go to the http://media.amazonwebservices.com/AWS_Pricing_Overview.pdf whitepaper.

Another way you can save money is by using Spot Instances. Spot Instances are unused Amazon EC2 capacity that you bid for. Instances are charged at the Spot Price, which is set by Amazon EC2 and fluctuates periodically depending on the supply of, and demand for, Spot Instance capacity. If your maximum bid exceeds the current Spot Price, your bid request is fulfilled, and your instances will run until either you choose to terminate them or the Spot Price increases above your maximum bid, whichever is sooner. To learn more about Spot Instances, go to http://aws.amazon.com/ec2/spot-instances.

Related Resources

The following table lists related resources that you'll find useful as you work with AWS services.

Resource	Description
AWS Products and Services	A comprehensive list of products and services AWS offers.
Documentation	Official documentation for each AWS product including service introductions, service features, and API references, and other useful information.
AWS Architecture Center	Provides the necessary guidance and best practices to build highly scalable and reliable applications in the AWS cloud. These resources help you understand the AWS platform, its services and features. They also provide architectural guidance for design and implementation of systems that run on the AWS infrastructure.
AWS Economics Center	Provides access to information, tools, and resources to compare the costs of Amazon Web Services with IT infrastructure alternatives.
AWS Cloud Computing Whitepapers	Features a comprehensive list of technical AWS whitepapers covering topics such as architecture, security, and economics. These whitepapers have been authored either by the Amazon team or by AWS customers or solution providers.
Videos and Webinars	Previously recorded webinars and videos about products, architecture, security, and more.
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. (subscription required).
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.

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.

Document History

This document history is associated with the release of Getting Started with AWS Computing Basics for Linux. This guide was last updated on May 06, 2014.

Change	Description	Release Date
New content	Created new document s	29 February 2012
Added new section	Added section for connecting to Amazon EC2 using the MindTerm client	8 March 2012