
Getting Started with AWS

Web Application Hosting for Linux



Amazon Web Services

Getting Started with AWS: Web Application Hosting for Linux

Amazon Web Services

Copyright © 2014 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, Cloudfront, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Overview	1
Getting Started	6
Step 1: Sign Up for the Service	6
Step 2: Install the Command Line Tools	7
Step 3: Create an Elastic Load Balancer	8
Step 4: Create and Configure Your Amazon EC2 Security Group	13
Step 5: Create a Key Pair	14
Step 6: Launch Amazon EC2 Instances Using Auto Scaling	15
Step 7: Create a CloudWatch Alarm	17
Step 8: Add Amazon RDS	21
Create a DB Security Group	22
Authorize Access	22
Launch an Instance	23
Step 9: Deploy Your Application	27
Connecting to your Amazon EC2 Instance from Your Web Browser Using the MindTerm SSH Client	28
Connect to Your Amazon EC2 Instance from Windows Using PuTTY	29
Connecting to Your Amazon EC2 Instance from a Linux/UNIX Machine Using a Standalone SSH Client	33
Configure the Amazon EC2 Instance	34
Step 10: Create a Custom AMI	41
Step 11: Launch New Environments Using AWS CloudFormation	42
Create an AWS CloudFormation Template	42
Modify a CloudFormation Template	46
Create an AWS CloudFormation Stack	47
Step 12: Clean Up	51
Terminate Your Amazon EC2 Instances in Your Auto Scaling Group	52
Terminate Your DB Instance	53
Delete Your CloudWatch Alarm	54
Delete Your Elastic Load Balancer	55
Delete a Key Pair	55
Delete an Amazon EC2 Security Group	56
Delete Your Custom AMI	56
Amazon Route 53	57
Amazon CloudFront	58
Pricing	59
Amazon EC2 Cost Breakdown	59
Amazon RDS Cost Breakdown	61
Summing It All Up	63
Related Resources	65
Document History	67

Overview

If you purchase hardware to run your website, you might find that highly available and scalable web hosting can be a complex and expensive proposition. Your website would likely experience dense peak traffic periods and significant fluctuations in traffic patterns. This would result in low utilization rates of your expensive hardware, and you could incur high operating costs to maintain mostly idle hardware. Amazon Web Services (AWS) provides the reliable, scalable, secure, and high performance infrastructure required for the most demanding web applications. AWS enables an elastic, scale-out and scale-in infrastructure model that matches IT costs with real-time shifts in customer traffic patterns.

This guide will help you use AWS to create scalable, robust web applications that handle sophisticated demands and workloads. We'll review an example architecture of a web application hosted on AWS, and we'll walk through the process of deploying a sample Drupal application using several key AWS services and following best practices. (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 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.
- Create a database instance and use it with Drupal.
- Create a template for the AWS resources you created.
- 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.

Getting Started with AWS Web Application Hosting for Linux

How Does AWS Help?

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.
- [Amazon Elastic Compute Cloud Getting Started Guide](#) - Walks you through launching and connecting to an Amazon EC2 Linux instance. For information about configuring software to run on instances (e.g., MySQL, Tomcat, Python), go to [Amazon Machine Images \(AMI\)](#) in the *Amazon Elastic Compute Cloud User Guide*.
- [Getting Started with AWS Computing Basics for Linux](#) - Introduces you to several key AWS services and components—what these services are, why they are important, and how to use them. The guide also provides a simple example architecture on a Linux platform and walks you through a deployment that uses this architecture. You will also learn how to install MySQL server and configure a database on an EC2 instance.

How Does AWS Help?

If you are responsible for running a web application then there are a variety of infrastructure and architecture issues that you face 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.

Challenges	Amazon Web Services	Benefits
Servers need to be provisioned to handle peak capacity and the unused cycles are wasted at other times.	<ul style="list-style-type: none">• Amazon Elastic Compute Cloud (EC2)• Amazon Elastic Load Balancing• Auto Scaling• Amazon CloudWatch	<ul style="list-style-type: none">• Amazon EC2 runs the web server and application servers.• Elastic Load Balancing supports health checks on hosts, distribution of traffic to Amazon EC2 instances across multiple Availability Zones, and the dynamic addition and removal of Amazon EC2 hosts from the load-balancing rotation.• Auto Scaling creates capacity groups of servers that can grow or shrink on demand.• Amazon CloudWatch reports metrics data for Amazon EC2 instances, and the metrics it gathers are used by Auto Scaling.

**Getting Started with AWS Web Application Hosting for
Linux
How Does AWS Help?**

Challenges	Amazon Web Services	Benefits
Need a content delivery network (CDN) to provide low-latency, high data transfer speeds so end users don't experience unnecessary delays.	<ul style="list-style-type: none">• Amazon CloudFront• Amazon Simple Storage Service (Amazon S3)	<ul style="list-style-type: none">• Amazon CloudFront speeds up the loading of streaming or downloaded static content by caching the content via a local edge cache at a location with the lowest latency.• Amazon S3 stores data backups from the relational database, web, and application servers, and for Amazon CloudFront distribution.
Applications may require a database, file system, or access to raw block-level storage.	Amazon Elastic Block Store (Amazon EBS)	Amazon EBS provides a persistent file system for web and application servers.
Maintaining a database can be expensive and time-consuming.	Amazon Relational Database Service (Amazon RDS)	Amazon RDS provides cost-efficient and resizable capacity while managing time-consuming database administration tasks.
Developers and businesses need a reliable and cost-effective way to route end users to Internet applications.	Amazon Route 53	Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost effective way to route end users to Internet applications by translating human readable names like www.example.com into the numeric IP addresses like 192.0.2.1 that computers use to connect to each other.
Need to plan the order in which Amazon Web Services will be provisioned, keeping in mind dependencies among the services.	AWS CloudFormation	AWS CloudFormation gives developers and systems administrators an easy way to create a collection of related AWS resources and provision them in an orderly and predictable fashion.

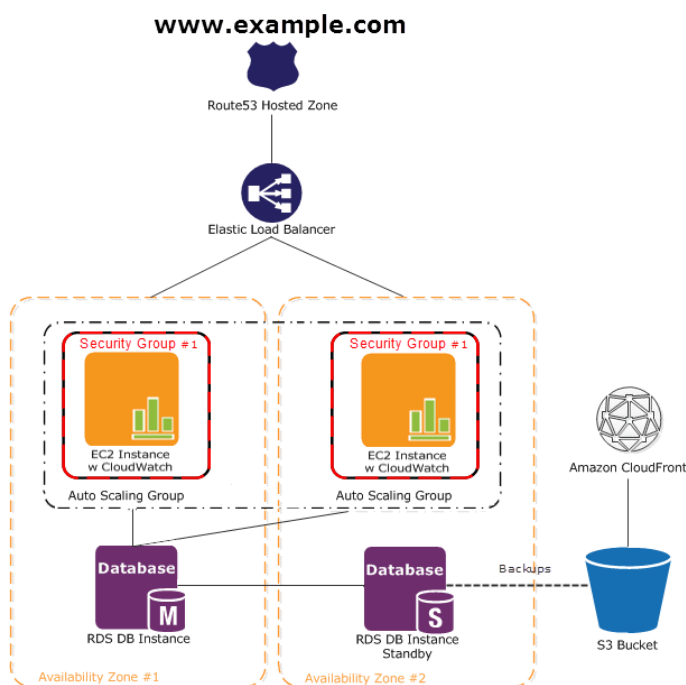
Challenges	AWS Components	Benefits
Need to provide security to protect application servers from outside malicious users.	Amazon Security Group	An Amazon Security Group lets you specify the protocols, ports, and source IP address ranges that are allowed to reach your Amazon EC2 instances.

**Getting Started with AWS Web Application Hosting for
Linux
Web Application Hosting Architecture**

Challenges	AWS Components	Benefits
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.

Web Application Hosting Architecture

The following diagram shows an example architecture of a web application using the AWS resources mentioned in the previous section.



In this diagram, Amazon EC2 instances run the application and web server and belong to an Amazon EC2 Security Group. The Amazon EC2 Security Group acts as an exterior firewall for the Amazon EC2 instances. An Auto Scaling group is used to maintain a fleet of Amazon EC2 instances that can handle the presented load. This Auto Scaling group spans over multiple Availability Zones to protect against potential failures if an Availability Zone becomes unavailable. 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 based on load, then the Elastic Load Balancer will automatically adjust accordingly. The database tier consists of Amazon RDS database instances, including master and local slave, located in multiple Availability Zones for failover protection. Amazon RDS provides automated backups to Amazon S3. Amazon S3 stores backups and static content. Since the consumers of this application may be globally distributed or a large number may visit the site at one time, high volume static content is edge cached using Amazon CloudFront for better performance. Amazon Route 53 can be used to provide secure and reliable routing to your infrastructure that uses Amazon Web Services.

For a step-by-step walkthrough of how to build out this architecture, see [Getting Started \(p. 6\)](#). This walkthrough will teach you how to do the following:

Getting Started with AWS Web Application Hosting for Linux

Web Application Hosting Architecture

- Sign up for AWS.
- Launch, connect, secure, and deploy Drupal to an Amazon EC2 instance.
- Create a Custom AML.
- 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.
- Create a database instance and use it with Drupal.
- Create an AWS CloudFormation template based on the resources you created.
- Clean up your AWS resources.

For more information on how to use Amazon CloudFront in this architecture, see [Amazon CloudFront \(p. 58\)](#). For more information on how to use Amazon Route 53 in this architecture, see [Amazon Route 53 \(p. 57\)](#).

Getting Started

Topics

- [Step 1: Sign Up for the Service \(p. 6\)](#)
- [Step 2: Install the Command Line Tools \(p. 7\)](#)
- [Step 3: Create an Elastic Load Balancer \(p. 8\)](#)
- [Step 4: Create and Configure Your Amazon EC2 Security Group \(p. 13\)](#)
- [Step 5: Create a Key Pair \(p. 14\)](#)
- [Step 6: Launch Amazon EC2 Instances Using Auto Scaling \(p. 15\)](#)
- [Step 7: Create a CloudWatch Alarm \(p. 17\)](#)
- [Step 8: Add Amazon RDS \(p. 21\)](#)
- [Step 9: Deploy Your Application \(p. 27\)](#)
- [Step 10: Create a Custom AMI \(p. 41\)](#)
- [Step 11: Launch New Environments Using AWS CloudFormation \(p. 42\)](#)
- [Step 12: Clean Up \(p. 51\)](#)

Let's suppose you want to build a content management system (CMS) application. You want to leverage the reliable, scalable, secure and high performance infrastructure that AWS offers. 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, but this walkthrough shows you one example that follows best practices and uses many of the AWS services so you can see how the services work together. Let's begin!

Note

In this example, we are going through the steps in a specific order to minimize the time for billable services. However, when you deploy your application you will likely start by launching your Amazon EC2 instance, configuring your application and database, creating a custom AMI, and then scaling your application.

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.

```
Command Name      Description
-----
as-create-auto-scaling-group  Create a new Auto Scaling group.
as-create-launch-config      Creates a new launch configuration.
as-create-or-update-tags      Create or update tags.
as-delete-auto-scaling-group  Deletes the specified Auto Scaling group.
as-delete-launch-config       Deletes the specified launch configuration.
as-delete-notification-configuration  Deletes the specified notification configuration.
as-delete-policy              Deletes the specified policy.
as-delete-scheduled-action    Deletes the specified scheduled action.
as-delete-tags                Delete the specified tags
as-describe-adjustment-types   Describes all policy adjustment types.
as-describe-auto-scaling-groups  Describes the specified Auto Scaling groups.
as-describe-auto-scaling-instances  Describes the specified Auto Scaling instances.
as-describe-auto-scaling-notification-types  Describes all Auto Scaling notification types.
as-describe-launch-configs      Describes the specified launch configurations.
as-describe-metric-collection-types  Describes all metric colle... metric granularity types.
as-describe-notification-configurations  Describes all notification...given Auto Scaling groups.
as-describe-policies            Describes the specified policies.
as-describe-process-types       Describes all Auto Scaling process types.
as-describe-scaling-activities   Describes a set of activit...ties belonging to a group.
as-describe-scheduled-actions    Describes the specified scheduled actions.
as-describe-tags                Describes tags
as-describe-termination-policy-types  Describes all Auto Scaling termination policy types.
as-disable-metrics-collection    Disables collection of Auto Scaling group metrics.
as-enable-metrics-collection     Enables collection of Auto Scaling group metrics.
as-execute-policy                Executes the specified policy.
as-put-notification-configuration  Creates or replaces notifi...or the Auto Scaling group.
as-put-scaling-policy             Creates or updates an Auto Scaling policy.
as-put-scheduled-update-group-action  Creates or updates a scheduled update group action.
as-resume-processes              Resumes all suspended Auto... given Auto Scaling group.
as-set-desired-capacity          Sets the desired capacity of the Auto Scaling group.
as-set-instance-health           Sets the health of the instance.
as-suspend-processes            Suspends all Auto Scaling ... given Auto Scaling group.
as-terminate-instance-in-auto-scaling-group  Terminates a given instance.
as-update-auto-scaling-group      Updates the specified Auto Scaling group.
help                             Prints the version of the CLI tool and the API.
version

For help on a specific command, type '<commandname> --help'

user ~ %_
```

After you have installed the command line tools, you can start creating your AWS resources. You are ready to start thinking about launching your Amazon EC2 instances. Even though for the purposes of this tutorial, you only have one Amazon EC2 instance up and running, you'll want to have multiple Amazon EC2 instances running across multiple Availability Zones eventually. This way if one Availability Zone goes down, the traffic will be rerouted to another Availability Zone. To prepare for the eventuality of maintaining multiple Amazon EC2 instances, we'll go ahead and create our Elastic Load Balancer resource. In the AWS CloudFormation step, we can scale out to make use of our Elastic Load Balancer. Let's move on to the next step to create our [Elastic Load Balancer](#).

Step 3: Create an Elastic Load Balancer

Elastic Load Balancing is a cost-effective and easy-to-use web service to help you improve the availability and scalability of your application. It makes it easy for you to distribute application loads between two or more Amazon EC2 instances. Elastic Load Balancing enables availability through redundancy and supports traffic growth of your application.

Elastic Load Balancing lets you automatically distribute and balance the incoming application traffic among all the instances you are running. The service also makes it easy to add new instances when you need to increase the capacity of your application. You can dynamically register or deregister instances from the load balancer as the capacity requirements of your application change with time.

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, see the [Elastic Load Balancing](#) details page.

In this step, we will create a load balancer for an HTTP service. We'll specify that the load balancer listens on port 80 and distributes traffic to port 80 on the instances.

Note

We'll go ahead and create our Elastic Load Balancer resource so that in the future when you have multiple instances running, your traffic will be load balanced between your instances. Elastic Load Balancing is a small cost relative to instance hours. In [Step 11: Launch New Environments Using AWS CloudFormation \(p. 42\)](#) we'll use AWS CloudFormation to create a template for our resources and add instances to our Auto Scaling Group.

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, select **US East (N. Virginia)** from 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. For this example, type **myLB**.

Getting Started with AWS Web Application Hosting for Linux
Step 3: Create an Elastic Load Balancer

Create Load Balancer

1. Define Load Balancer 2. Configure Health Check 3. Add EC2 Instances 4. Review

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name to identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured the load balancer with a standard web server on port 80.

Load Balancer name:

Create LB Inside:

Create an internal load balancer: ☐ (what's this?)

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

- f. Leave **Create LB Inside** set to **EC2-Classic**.
 - g. Leave the **Listener Configuration** set to the default values.
 - h. Click **Continue**.
2. Configure the health check for your load balancer. 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. For this example, leave **Ping Protocol** set to its default value of **HTTP**. When you deploy your application in the future, you can specify **HTTPS**. For information on using HTTPS with Elastic Load Balancing, see [Elastic Load Balancing Security Features](#) in *Elastic Load Balancing Developer Guide*.
 - b. For this example, 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 port value, your Amazon EC2 instances must accept incoming traffic on the port that you specified for the health check. You can set a different port value other than 80, and you can come back and set this value at a later time. However, for this example, set it to 80.

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

Elastic Load Balancing sends health check queries to the path you specify in **Ping Path**. 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 only responds to the health check. This helps differentiate between traffic that is hitting your site and responses to the load balancer.

Getting Started with AWS Web Application Hosting for Linux
Step 3: Create an Elastic Load Balancer

- d. Set the **Healthy Threshold** to **2**. Leave the rest of the **Advanced Details** set to their default values.

The screenshot shows the 'Create Load Balancer' console interface. At the top, there's a progress bar with four steps: '1. Define Load Balancer', '2. Configure Health Check' (which is the active step, highlighted with an orange underline), '3. Add EC2 Instances', and '4. Review'. Below the progress bar, the main heading is 'Configure Health Check'. A descriptive paragraph states: 'Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to healthy instances. If an instance fails the health check, it is automatically removed from the load balancer. Configure the health check based on your specific needs.' Below this, there are three input fields: 'Ping Protocol' is a dropdown menu set to 'HTTP'; 'Ping Port' is a text box containing '80'; and 'Ping Path' is a text box containing '/'. Further down, there's a section titled 'Advanced Details'. It contains four rows of settings, each with an information icon (i) to its left: 'Response Timeout' is a text box with '5' followed by 'seconds'; 'Health Check Interval' is a text box with '30' followed by 'seconds'; 'Unhealthy Threshold' is a dropdown menu set to '2'; and 'Healthy Threshold' is a dropdown menu set to '2'.

- e. Click **Continue**.
3. On the **Add Instances to Load Balancer** page, click **Continue**.
4. Review your settings. You can make changes to the settings by clicking the **Edit** link for a specific step in the process.

Getting Started with AWS Web Application Hosting for Linux
Step 3: Create an Elastic Load Balancer

Create Load Balancer

1. Define Load Balancer2. Configure Health Check3. Add EC2 Instances4. Review

Review

Please review the load balancer details before continuing

▼ Define Load Balancer

Edit load balancer definition

Load Balancer name: MyLB

Scheme: internet-facing

Port Configuration: 80 (HTTP) forwarding to 80 (HTTP)

▼ Configure Health Check

Edit health check

Ping Target: HTTP:80/

Timeout: 5 seconds

Interval: 30 seconds

Unhealthy Threshold: 2

Healthy Threshold: 2

▼ Add EC2 Instances

Edit instances

Cross-Zone Load Balancing: Enabled

Connection Draining: Enabled, 300 seconds

Instances:

BackCreate

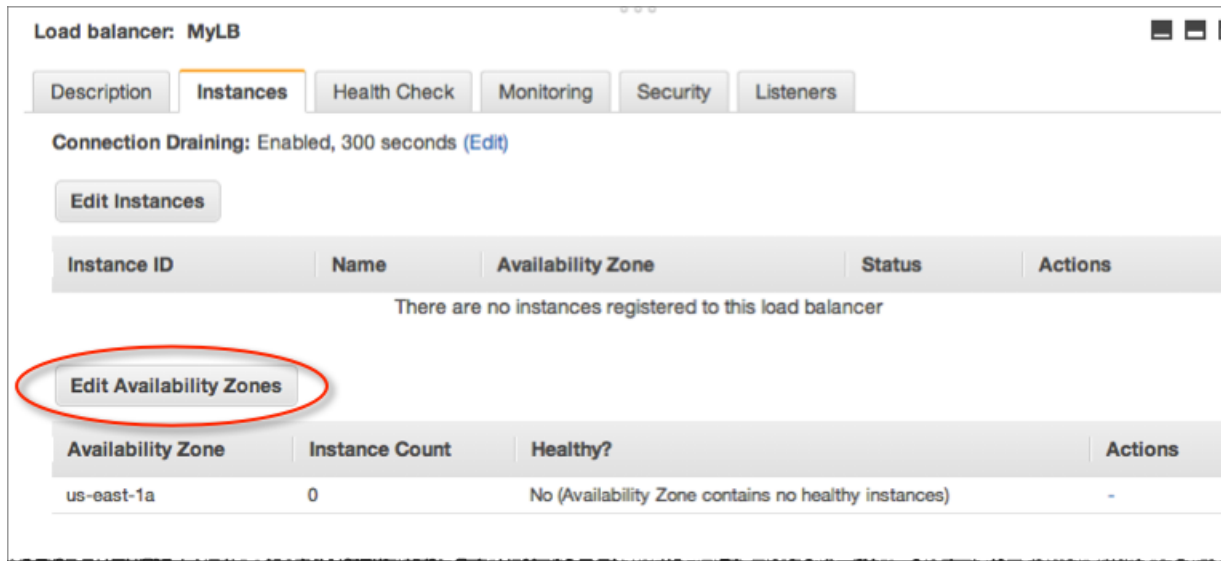
5. Click **Create**. On the **Create Load Balancer** confirmation page, click **Close**.

Your new load balancer now appears in the list.

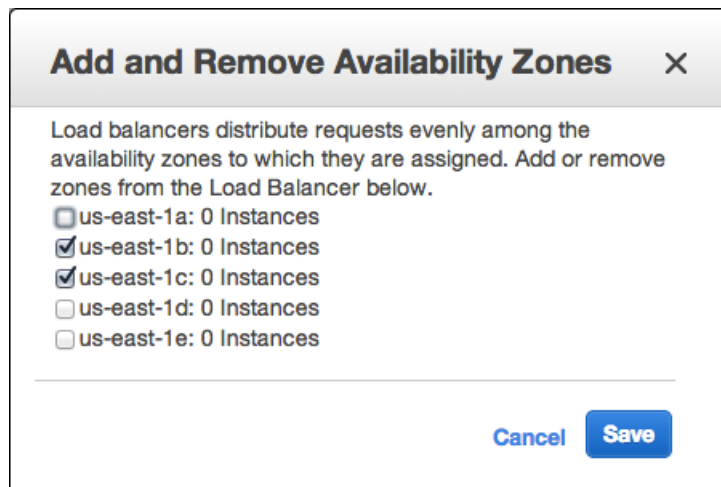
As a best practice, you should have sufficient instances across Availability Zones to survive the loss of any one Availability Zone. In the next step, you'll ensure that your load balancer points to multiple Availability Zones.

6. Add an Availability Zone.
 - a. In the **Load Balancers** list, click **MyLB**.
 - b. Click the **Instances** tab.
 - c. Click **Edit Availability Zones**.

Getting Started with AWS Web Application Hosting for Linux Where You're At



- 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**.



The Availability Zones column for the load balancer now shows the Availability Zones you selected.

Where You're At

Here's where you are at in building your architecture.



Let's move on to the next topic to create your Amazon EC2 security group. You will need to create an Amazon EC2 security group in order to open up ports on your instance. Your security group is essentially acting as a firewall.

Step 4: Create and Configure Your Amazon EC2 Security Group

An Amazon EC2 security group acts as a firewall that controls the traffic allowed into a group of instances. When you launch an Amazon EC2 instance, you can assign it to one or more security groups. For each security group, you add rules that govern the allowed inbound traffic to instances in the group. All other inbound traffic is discarded. You can modify rules for a security group at any time. The new rules are automatically enforced for all existing and future instances in the group.

In this step, we will do the following:

- Create an Amazon EC2 security group
- Configure an Amazon EC2 security group

To create and configure your security group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Make sure **US East (N. Virginia)** is selected in the region selector of the navigation bar.

Note

For this purposes of this walkthrough, we will always use the US East (N. Virginia) region. However when you deploy your application, select the region that is closest to you.

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 the name to configure your instance's security group rules.
5. In the navigation pane, click **Security Groups**, then click **Create Security Group**.
6. In the **Create Security Group** dialog box, type `webappsecuritygroup` in the **Security group name** box, and a description of your choice in the **Description** box.
7. On the **Inbound** tab, click **Add Rule**, and select **HTTP** from the **Type** list.
8. Select **Custom IP** from the **Source** list, and enter the name of the security group that's associated with your load balancer, for example, `amazon-elb/amazon-elb-sg`. When you select this source, this means that only traffic that comes through the Elastic Load Balancer can connect to your Amazon EC2 instance.
9. Click **Add Rule**.
10. Select **SSH** from the **Type** list to connect to your Amazon EC2 instances. Select **Anywhere** from the **Source** list.

Getting Started with AWS Web Application Hosting for Linux

Step 5: Create a Key Pair

Create Security Group

Security group name: webappsecuritygroup

Description: My security group

VPC: No VPC

Security group rules:

Inbound

Type	Protocol	Port Range	Source
HTTP	TCP	80	Custom IP: amazon-elb/amazon-
SSH	TCP	22	Anywhere: 0.0.0.0/0

Add Rule

Cancel Create

Important

The security group settings are configured to allow access from everywhere: 0.0.0.0/0. This is not good practice, and it is only for the purposes of this exercise that we are setting it up this way. Best practice should be to set rules that restrict access to only those computers or networks that require access to this service. The number after the "/" indicates a range of addresses.

11. Click **Create**.

Your Amazon EC2 security group is not yet enforced. We will enforce this when we create our Auto Scaling group. However, you can also apply an Amazon EC2 security group to an Amazon EC2 instance. For more information, see [Using Security Groups](#) in the *Amazon Elastic Compute Cloud User Guide*.

Now that we have created our Amazon EC2 security group, we will need a way to access our Amazon EC2 instance to deploy our application. Public AMI instances use a public/private key pair to login rather than a password. Let's move on to the next section to create our key pair.

Step 5: Create a Key Pair

You can create your key pair so that you can connect to your Amazon EC2 instances. Public AMI instances use a public/private key pair to log in rather than a password. The public key half of this pair is embedded in your instance, allowing you to use the private key to log in securely without a password. After you create your own AMIs, you can choose other mechanisms to securely log in to your new instances. In this step we will use AWS Management Console to create a key pair.

To generate a key pair

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the top navigation bar, in the region selector, click **US East (N. Virginia)**.
3. In the left navigation pane, under **Network and Security**, click **Key Pairs**.
4. Click **Create Key Pair**.

5. Type `mykeypair` in the new **Key pair name** box, and then click **Yes**.
6. Download the private key file, which is named `mykeypair.pem`, and keep it in a safe place. You will need it to access any instances that you launch with this key pair.

Important

If you lose the key pair, you cannot connect to your Amazon EC2 instances.

For more information about key pairs, see [Amazon EC2 Key Pairs](#) in the *Amazon Elastic Compute Cloud User Guide*.

Since your traffic may vary, you want AWS to scale the number instances appropriately. To do this you'll want to use [Auto Scaling](#) to create an Auto Scaling group. Let's move on to the next step to create our Auto Scaling group and associate our Auto Scaling group with our Elastic Load Balancer.

Step 6: Launch Amazon EC2 Instances Using Auto Scaling

Auto Scaling is designed to launch or terminate Amazon EC2 instances automatically based on user-defined policies, schedules, and alarms. You can use Auto Scaling to maintain a fleet of Amazon EC2 instances that can handle any presented load. As its 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 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. You can also use Auto Scaling to 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—distinct physical locations for the hosted Amazon EC2 instances—so that if an Availability Zone becomes unavailable, Auto Scaling will automatically redistribute applications to 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, we will set up the basic infrastructure that must be in place to get Auto Scaling started for most applications. We'll set up an Amazon EC2 application to be load-balanced and auto-scaled with a minimum number of one instance and maximum number of one instance so you are only charged for one instance. However, when you create your actual website you should follow the best practice of having sufficient instances across Availability Zones to survive the loss of any one Availability Zone. Additionally, increase your maximum number of instances to be greater than your minimum to make use of the Auto Scaling feature. You can also specify the maximum number of instances to control your fleet size. Auto Scaling in this example is configured to scale out by one when there is a change in capacity. We define the policy in this topic and then create a CloudWatch alarm in the next section to take action on the policy when the average CPU usage exceeds a threshold of 60 percent for 10 minutes. Auto Scaling and Amazon CloudWatch work together to launch or terminate instances based on the policies you create. To save time, we will create just one policy, however, you can create more policies, such as a scale-in policy.

If you haven't already installed the Auto Scaling command line tools, you need to do that now at [Using the Command Line Tools](#) in the *Auto Scaling DeveloperGuide*. We will use the command line tools to set up Auto Scaling.

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

1. Open a command prompt window. In Microsoft Windows, start the Command Prompt application (from the **Start** menu, click **Programs**, click **Accessories**, and then click **Command Prompt**).
2. Use the Auto Scaling `as-create-launch-config` command. In this example, we use a publicly available Linux AMI running a content management system (CMS). We use a `t1.micro` instance type, and use the security group and the key pair we created in the previous steps. In this example, the

Getting Started with AWS Web Application Hosting for Linux

Step 6: Launch Amazon EC2 Instances Using Auto Scaling

key pair file is located in the directory in which we are creating our Auto Scaling group. We will not specify a region because we want to use the default region.

Note

You will be charged for launching one Amazon EC2 instance. The charges in this example are minimal. For more information about Amazon EC2 pricing, see the [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) details page.

```
PROMPT>as-create-launch-config MyLC --image-id ami-7813e011 --instance-type t1.micro --group webappsecuritygroup --key mykeypair
```

Auto Scaling returns output similar to the following example output:

```
OK-Created launch config
```

Note

You can copy and paste the commands from the document into the command line window. To paste the contents in the command line window, use right-click. If you have trouble getting the commands to work, make sure the command was pasted correctly.

3. Use the Auto Scaling `as-create-auto-scaling-group` command. In this example, we use two Availability Zones. This 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.

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

Auto Scaling returns the following:

```
OK-Created AutoScalingGroup
```

4. Use the Auto Scaling `as-put-scaling-policy` command to create a policy to enlarge your fleet of instances.

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

Auto Scaling returns output similar to the following example output:

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

Note

To save time, we only created a scale-out policy. However, you typically would want to create a scale-in policy as well. Auto Scaling decreases the number of instances when your application doesn't need the resources, saving you money. To create a scale-in policy, change the policy name and change the adjustment from 1 to -1.

5. Verify that your Auto Scaling group exists by using the `as-describe-auto-scaling-groups` command.

Getting Started with AWS Web Application Hosting for Linux Where You're At

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

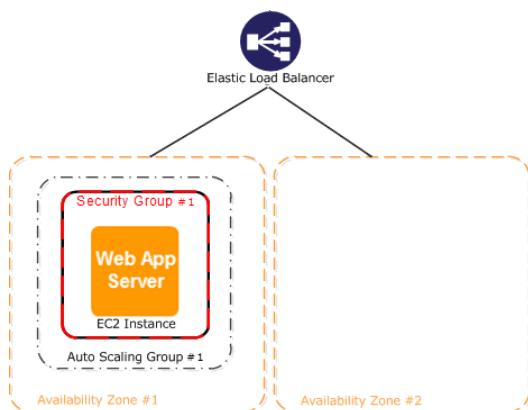
Auto Scaling returns the following:

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		
1	1	1			
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG
INSTANCE	i-xxxxxxx	us-east-1c	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](#). 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. 52).

Where You're At

Here's where you are at 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, we'll create an Amazon CloudWatch alarm so we can track the Auto Scaling policy you just created.

Step 7: Create a CloudWatch Alarm

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

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 also publish your own metrics directly to Amazon CloudWatch.

Amazon CloudWatch alarms help you implement decisions more easily by enabling you to send notifications or automatically make changes to the resources you are monitoring, based on rules that you define. For

Getting Started with AWS Web Application Hosting for Linux

Step 7: Create a CloudWatch Alarm

example, you can create alarms that initiate Auto Scaling and Amazon Simple Notification Service (Amazon SNS) actions on your behalf.

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 remains below a certain threshold level on your Amazon EC2 instances. You can then create an automated procedure to ensure that you always have the right number of instances to match the amount of traffic you have. You can also 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 email you right away when your application slows down, to go back and discover that a particular database was being overloaded, and later to watch response times come back up to speed. For more information about creating CloudWatch alarms, go to [Creating CloudWatch Alarms](#) in the *Amazon CloudWatch Developer Guide*.

In the previous task, we created an Auto Scaling policy to scale out the number of instances. In this task, you need to associate that Auto Scaling policy with an alarm action to make changes to your resources. This topic walks you through how to create a CloudWatch alarm to alert the application when this threshold is breached. To save time during this walkthrough, we'll just create one alarm; however, you can apply the same procedure create other alarms. For example, you could create another alarm to scale in your instances. 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, make sure **US East (N. Virginia)** is selected in the region selector.
 - c. In the navigation pane, select **Alarm** under **Alarms**.
 - d. Click **Create Alarm**.
 - e. On the **Select Metric** page of the **Create Alarm Wizard**, select **EC2: Aggregated by Auto Scaling Group** from the **Viewing** drop-down menu.

Create Alarm Wizard Cancel

SELECT METRIC DEFINE ALARM CONFIGURE ACTIONS REVIEW

Set an alarm for any of your CloudWatch metrics. Your alarm will react automatically when a metric reaches your specified threshold. Available actions include sending Amazon SNS notifications and executing Auto Scaling policies. To get started, select a metric. Then preview it, select a statistic and sampling period, and click **Continue**.

Statistic: **Average** Period: **5 Minutes**

CPUUtilization (Percent)

Important: select a sample period. A shorter period allows a more sensitive alarm. A longer period smooths out brief spikes.

Viewing: **EC2: Aggregated by Auto Scaling Group** Search 1 to 28 of 28 Metrics

myAutoScalingGroup3	DiskReadOps
MyAutoScalingGroup3	DiskWriteBytes
MyAutoScalingGroup3	DiskWriteOps
MyAutoScalingGroup3	NetworkIn
MyAutoScalingGroup3	NetworkOut
MyAutoScalingGroup	CPUUtilization
MyAutoScalingGroup	DiskReadBytes
MyAutoScalingGroup	DiskReadOps
MyAutoScalingGroup	DiskWriteBytes
MyAutoScalingGroup	DiskWriteOps
MyAutoScalingGroup	NetworkIn
MyAutoScalingGroup	NetworkOut

Continue

**Getting Started with AWS Web Application Hosting for
Linux
Step 7: Create a CloudWatch Alarm**

- f. Click the **MyAutoScalingGroup/CPU Utilization** row.
 - g. Click **Continue**.
 2. Define the alarm.
 - a. On the **Define Alarm** page of the **Create Alarm** wizard, type **MyHighCPUAlarm** in the **Name** box.
 - b. Type a description in the **Description** box.
 - c. In the **Define Alarm Threshold** section, select **>** and type **60** in the first box and **10** in the minutes box for this example. For your application, you can do some load testing to see what value makes the most sense for your application.

The screenshot shows the 'Create Alarm Wizard' window with the 'DEFINE ALARM' step selected. The progress bar at the top shows four steps: SELECT METRIC, DEFINE ALARM (active), CONFIGURE ACTIONS, and REVIEW. Below the progress bar, a message says: 'Provide the details and threshold for your alarm. Use the graph below to help set the appropriate threshold.'

Identify Your Alarm
Assign your alarm a name and description.

Name: MyHighCPUAlarm
Description: Alarm triggered when CPU utilization is high

Define Alarm Threshold
Alarms have three states: ALARM, OK, and INSUFFICIENT DATA. The state of your alarm changes according to a threshold you specify. First, define the criterion for entering the ALARM state. Later, you can specify an action to be taken when your alarm enters any of the three states.

This alarm will enter the ALARM state when CPUUtilization is **>** **60** for **10** minutes.

A line graph titled 'CPUUtilization (Percent)' shows a blue line representing CPU utilization over time. The y-axis ranges from 0 to 100 in increments of 25. The x-axis shows dates and times: 9/22 19:00, 9/22 20:00, 9/22 21:00, 9/22 22:00, 9/22 23:00, and 9/23 00:00. A horizontal red line is drawn at the 60% mark. The blue line remains below 60% until approximately 23:00 on 9/22, where it spikes sharply to nearly 100% and then drops back down.

At the bottom left is a '< Back' button, and at the bottom right is a 'Continue >' button.

- d. Click **Continue**.
 3. Define your actions.
 - a. On the **Configure Actions** page of the **Create Alarm** wizard, select **Alarm** from the **When Alarm state is** drop-down menu.
 - b. Select **Auto Scaling Policy** from the **Take Action** drop-down menu.
 - c. Select **MyAutoScalingGroup** from the **Auto Scaling Group** drop-down menu.
 - d. Select **MyScaleUpPolicy (Add 1 instance)** from the **Policy** drop-down menu.
 - e. Click **Add Action**.
 - f. Select **Alarm** from the **When Alarm state is** drop-down menu.
 - g. Select **Send Notification** from the **Take Action** drop-down menu.
 - h. For topic, select **Create New Email Topic**. Then type a topic name in the **Topic** box.

Getting Started with AWS Web Application Hosting for Linux

Step 7: Create a CloudWatch Alarm

Create Alarm Wizard Cancel

SELECT METRIC DEFINE ALARM **CONFIGURE ACTIONS** REVIEW

Define what actions are taken when your 'MyHighCPUAlarm' alarm changes.

You can define multiple actions for a single alarm. For example, you may want to scale out your fleet and send an email to your pager when this alarm enters the ALARM state, and then send another all-clear email when it returns to the OK state.

Define Your Actions

Actions define what steps you want to automate when the alarm state changes. For example, you can send a message using email via the [Simple Notification Service \(SNS\)](#). You can also execute an [Auto Scaling Policy](#), if you have one configured ([learn about policies](#)).

When Alarm state is	Take action	Action details	
ALARM	Auto Scaling Policy	Auto Scaling Group: MyAutoScalingGroup Policy: MyScaleUpPolicy (Add 1)	REMOVE
ALARM	Send Notification	Topic: MyHighCPUAlarmTopic Email(s): <input type="text"/>	ADD ACTION

A topic is a communication channel that can be reused across Send Notification actions. Please enter a new topic name and a list of comma-separated email addresses.

[< Back](#) [Continue](#)

- i. Type an email address in the **Email(s)** box.
 - j. Click **Add Action**.
 - k. Click **Continue**.
4. In the **Review** page, click **Create Alarm**.

Create Alarm Wizard Cancel

SELECT METRIC DEFINE ALARM CONFIGURE ACTIONS **REVIEW**

If you want to make any changes to this alarm, click **Back** or select a step on the right to edit.

Alarm Definition Edit Definition

Name: MyHighCPUAlarm
Description: Alarm triggered when CPU utilization is high
In ALARM state when: the value is > 60 for 10 minutes

Metric Edit Metric

Namespace: AWS/EC2
MetricName: CPUUtilization
AutoScalingGroupName: MyAutoScalingGroup
Period / Statistic: 5 Minutes / Average

Alarm Actions Edit Actions

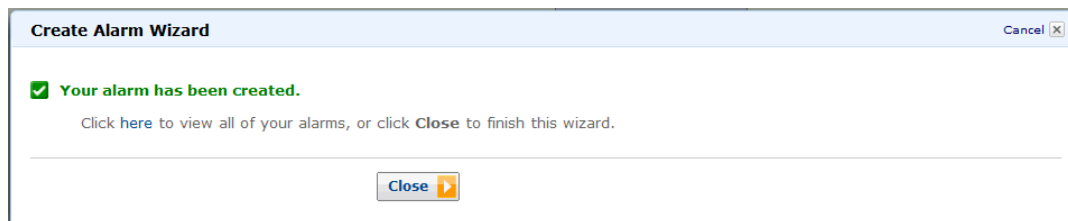
Actions:

When alarm state is "ALARM "
Action Type: Auto Scaling Policy
Action: Use policy MyScaleUpPolicy (Add 1 instance) for group MyAutoScalingGroup

When alarm state is "ALARM "
Action Type: Send Notification to New Topic
Action: Notify topic: MyHighCPUAlarmTopic (janedoe@example.com)

[< Back](#) [Create Alarm](#)

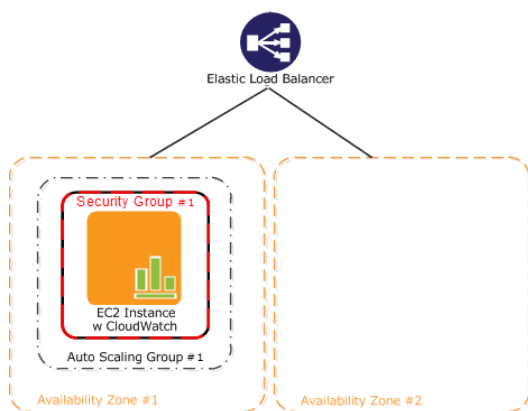
5. Click **Close**.



Your new alarm now appears in the list. When you create your MyScaleDownPolicy, you can create another alarm using the same steps.

Where You're At

Here's where you are at in building your architecture.



Next, let's add a database to the web application. Amazon provides a couple of database options, but for this example, we'll use [Amazon Relational Database Service \(Amazon RDS\)](#) because it's easy to operate and relieves us from the database administrative overhead.

Step 8: Add Amazon RDS

Topics

- [Create a DB Security Group \(p. 22\)](#)
- [Authorize Access \(p. 22\)](#)
- [Launch an Instance \(p. 23\)](#)

Now we are ready to add Amazon Relational Database (Amazon RDS) to our architecture. In this step we will launch a Multi-AZ RDS instance. When you create or modify your DB Instance to run as a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. Updates to your DB Instance are synchronously replicated across Availability Zones to the standby in order to keep both in sync and protect your latest database updates against DB Instance failure. During certain types of planned maintenance, or in the unlikely event of DB Instance failure or Availability Zone failure, Amazon RDS will automatically fail over to the standby so that you can resume database writes and reads as soon as the standby is promoted. Since the name record for your DB Instance remains the same, your application can resume database operation without the need for manual administrative intervention. With Multi-AZ deployments, replication is transparent: you do not interact directly with the standby, and it cannot be used to serve read traffic.

Important

The DB Instance you're about to launch will be live (and not running in a sandbox). You will incur the standard Amazon RDS usage fees for the instance until you terminate it. The total charges will be minimal (typically less than a dollar) if you complete the exercise described here in one sitting and terminate your DB Instance when you are finished. For more information about Amazon RDS usage rates, go to the [Amazon RDS product page](#).

Note

This is an optional step, so if you would like to skip this step, you can continue on to [Step 9: Deploy Your Application \(p. 27\)](#).

To set up your Amazon RDS database you need to do the following:

- Create a DB security group
- Authorize your DB instance
- Launch a DB instance

Create a DB Security Group

To create a DB Security Group, you need to provide a name and a description.

To create a DB Security Group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Make sure **US East (N. Virginia)** is selected in the region selector of the navigation bar.
3. In the left navigation pane, click **Security Groups**.
4. Click the **Create DB Security Group** button.
5. Type the name of the new DB security group. For this example, type **mydbsecuritygroup**.
6. Type a description for the new DB Security Group in the **Description** text box.
7. Click **Yes, Create**.

Now you're ready to authorize access to the Amazon EC2 security group.

Authorize Access

Now you will need to grant your Amazon EC2 security group access to your DB Security Group.

To authorize your Amazon EC2 security group for access to your DB Security Group

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Make sure **US East (N. Virginia)** is selected in the region selector of the navigation bar.
3. In the left navigation pane, click **Security Groups**.
4. Select **mydbsecuritygroup**.
5. In the lower pane, select **EC2 Security Group** for the **Connection Type**.
6. Your AWS Account ID appears in the right half of the lower pane. If you want to authorize a different AWS ID to use this DB Security Group, select **Another account**, and then type your ID in the **AWS Account ID** box.

Note

Make sure to remove the hyphens when you type your account ID.

7. For the **EC2 Security Group Name**, select **webappsecuritygroup**.

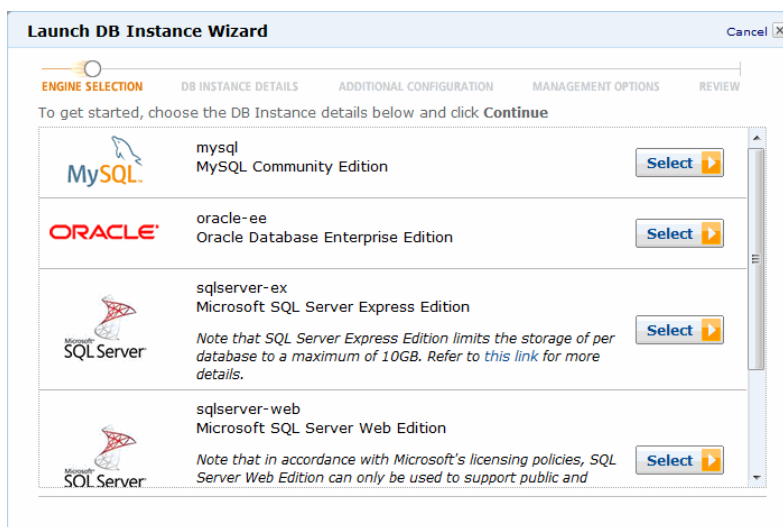
8. Click **Authorize**.

The authorization will take a few moments. After the security group has been authorized, the **Status** column in the top pane will say **authorized**. Move on to the next step to launch your first Amazon RDS database.

Launch an Instance

To launch an instance

1. Start the launch wizard:
 - a. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
 - b. Make sure **US East (N. Virginia)** is selected in the region selector in the navigation bar.
 - c. In the left navigation pane, click **DB instances**.
 - d. In the **Amazon RDS Console Dashboard**, click **Launch a DB Instance**.



2. Click **Select** next to the **MySQL Community Edition**.
3. On the **DB Instance Details** page, specify your DB instance details as shown in the following table. Then click **Continue**.

For this parameter...	Do this
License Model	Keep the default: general-public-license .
DB Engine Version	Keep the default: 5.5.27 (default) .
DB Instance Class	Select db.m1.small . The DB Instance class defines the CPU and memory capacity of your DB instance.
Multi-AZ Deployment	Choose Yes . Although the Multi-AZ deployment is more expensive, it is a best practice.

Getting Started with AWS Web Application Hosting for Linux

Launch an Instance

For this parameter...	Do this
Auto Minor Version Upgrade	Keep the default setting of Yes for this example. The Auto Minor Version Upgrade option enables your DB Instance to receive minor engine version upgrades automatically when they become available.
Allocated Storage	You can specify how much storage in gigabytes you want initially allocated for your DB Instance. For this example, type 5.
Use Provisioned IOPS	Leave the check box unselected. This option turns on Provisioned IOPS (I/O operations per second), a high-performance storage option in RDS that is optimized for I/O-intensive, transactional (OLTP) database workloads. For more information about high performance storage, see Provisioned IOPS .
DB Instance Identifier	The DB Instance is a name for your DB Instance that is unique for your account in a Region. Type <code>mydbinstance</code> in the DB Instance Identifier text box.
Master Username	Type a name for your master user in the Master Username text box. You use the master user name to log on to your DB Instance with all database privileges.
Master Password	Type a password for your master user in the Master User Password text box.

Important

You must specify a password containing from 4 to 16 alphanumeric characters only.

Launch DB Instance Wizard [Cancel]

ENGINE SELECTION **DB INSTANCE DETAILS** ADDITIONAL CONFIGURATION MANAGEMENT OPTIONS REVIEW

To get started, choose a DB engine below and click **Continue**

DB Engine: mysql
License Model: General Public License
DB Engine Version: MySQL 5.5.27 (default)
DB Instance Class: db.m1.small
Multi-AZ Deployment: Yes
Auto Minor Version Upgrade: ☒ Yes ☐ No

Provide the details for your RDS Database Instance.

Allocated Storage: 5 GB (Minimum: 5 GB, Maximum: 3072 GB) Higher allocated storage may improve IOPS performance.
Use Provisioned IOPS: ☐
DB Instance Identifier: mydbinstance (e.g. mydbinstance)
Master Username: awsuser (e.g. awsuser)
Master Password: (e.g. mypassword)

< Back **Continue** >

4. Provide additional configuration information for your DB Instance:
 - a. Type `mydb` into the **Database Name** text box.

Getting Started with AWS Web Application Hosting for Linux

Launch an Instance

You provide a database name so that Amazon RDS will create a default database on your new DB Instance. If you skip this step, Amazon RDS will not create a database on your DB Instance.

- b. Select **mydbsecuritygroup** in the **DB Security Groups** box.

Launch DB Instance Wizard Cancel

ENGINE SELECTION DB INSTANCE DETAILS **ADDITIONAL CONFIGURATION** MANAGEMENT OPTIONS REVIEW

Provide the optional additional configuration details below.

Database Name: (e.g. mydb)

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Database Port:

Choose a VPC: Only VPCs with a DB Subnet Group(s) are allowed

Availability Zone: MultiAZ deployment selected

Option Group:

If you have custom DB Parameter Groups or DB Security Groups you would like to associate with this DB Instance, select them below, otherwise proceed with default settings.

Parameter Group:

DB Security Group(s):

[< Back](#) [Continue >](#)

- c. Accept the default values for the rest of the parameters available on this page, and then click **Continue**.

5. Use the **Management Options** page to specify backup and maintenance options for your DB Instance. For this example, accept the default values, and then click **Continue**.

Launch DB Instance Wizard Cancel

ENGINE SELECTION DB INSTANCE DETAILS **ADDITIONAL CONFIGURATION** **MANAGEMENT OPTIONS** REVIEW

Enabled Automatic Backups: ☒ Yes ☐ No

The number of days for which automated backups are retained.

Please note that automated backups are currently **supported for InnoDB storage engine only**. If you are using MyISAM, refer to details [here](#).

Backup Retention Period: days

The daily time range during which automated backups are created if automated backups are enabled

Backup Window: ☐ Select Window ☒ No Preference

The weekly time range (in UTC) during which system maintenance can occur.

Maintenance Window: ☐ Select Window ☒ No Preference

[< Back](#) [Continue >](#)

6. Review the options for your DB Instance. If you need to correct any options, click **Back** to return to previous pages and make corrections. When you're ready, click **Launch DB Instance** to launch your new DB Instance.

Getting Started with AWS Web Application Hosting for Linux

Launch an Instance

Launch DB Instance Wizard Cancel

ENGINE SELECTION DB INSTANCE DETAILS ADDITIONAL CONFIGURATION MANAGEMENT OPTIONS REVIEW

Please review the information below, then click **Launch DB Instance**.

Engine: mysql
Engine Version: MySQL 5.5.27
License Model: general-public-license
Auto Minor Ver. Upgrade: Yes
DB Instance Class: db.m1.small
Multi-AZ Deployment: Yes
Allocated Storage: 5
Provisioned IOPS: default
DB Instance Identifier: mydbinstance
Master User Name: awsuser
Master User Password: *****

Database Name: mydb
Database Port: 3306
Availability Zone: Using a Multi-AZ Deployment disables this preference.
Option Group: default:mysql-5-5
DB Parameter Group: default:mysql5.5
DB Security Group(s): mydbsecuritygroup
DB Subnet Group:
Publicly Accessible: Only applicable with VPC

Backup Retention Period: 1
Backup Window: No Preference
Maintenance Window: No Preference

< Back Launch DB Instance

7. Launching can take a few minutes to complete. When you see the notice that your instance is being created, click **Close**.

Launch DB Instance Wizard Cancel

☒ **Your DB instance is being created.**
Note: Your DB instances may take a few minutes to launch.
> [View your DB instances on the DB Instances page](#)

Other RDS Features

DB Security Groups
You will be unable to connect to your database instance unless you have previously authorized access on your chosen DB Security Group. The DB Security Groups page allows you to authorize and revoke access based on CIDR/IP or EC2/VPC Security Group.
> [Go to the DB Security Groups Page](#)

DB Snapshots
Create a backup of your DB Instance by taking a snapshot.
> [Go to the DB Snapshots Page](#)

Related AWS Services

Amazon ElastiCache
Add a managed Memcached-compatible in-memory cache to speed up your database access.
> [Click here to learn more and launch your Cache Cluster](#)

Close

Your DB instance appears in the list on this page with the status of **creating** until your DB Instance is created and ready for use.

Getting Started with AWS Web Application Hosting for Linux

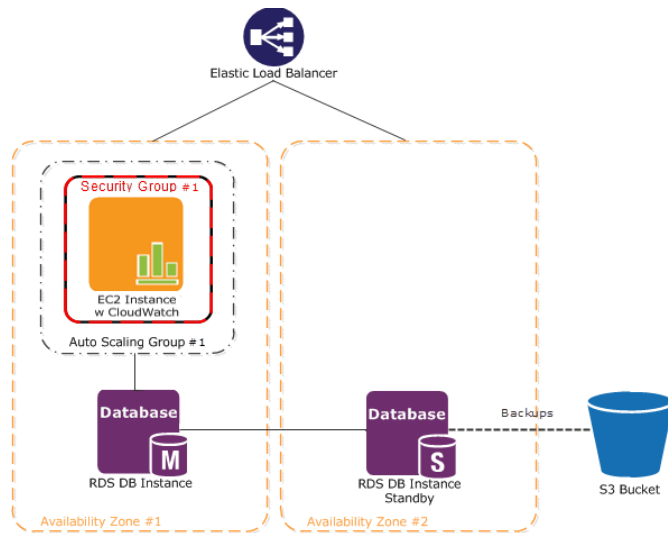
Step 9: Deploy Your Application

Launch DB Instance		Show Monitoring		Instance Actions ▾					
Filter: All Instances ▾		Search DB Instances...		X				Viewing 9 of 9 DB Instances	
<input type="checkbox"/>	DB Instance	VPC ▾	Multi-AZ ▾	Class ▾	Status ▾	Storage	IOPS	Security Groups	Engine ▾ Zone ▾
		mydbinstance	Yes	db.m1.small	creating	5 GB		mydbsecuritygroup (active)	mysql us-east-1a

After your DB instance changes to the **available** state, you're billed for each hour or partial hour that you keep the DB Instance running (even if the DB Instance is idle).

Where You're At

Here's where you are at in building your architecture.



Now that you have launched your Amazon RDS database, you're ready to deploy your sample web application.

Step 9: Deploy Your Application

Topics

- [Connecting to your Amazon EC2 Instance from Your Web Browser Using the MindTerm SSH Client \(p. 28\)](#)
- [Connect to Your Amazon EC2 Instance from Windows Using PuTTY \(p. 29\)](#)
- [Connecting to Your Amazon EC2 Instance from a Linux/UNIX Machine Using a Standalone SSH Client \(p. 33\)](#)
- [Configure the Amazon EC2 Instance \(p. 34\)](#)

Now that we've created all of our AWS resources, it's time to deploy our application to our Amazon EC2 instance. In this step, we'll first connect to the Amazon EC2 instance, and then we'll configure the instance by using a sample application that 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. 28)
2. [Connect Using the Java SSH Client](#) (p. 28)

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 Java 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**.

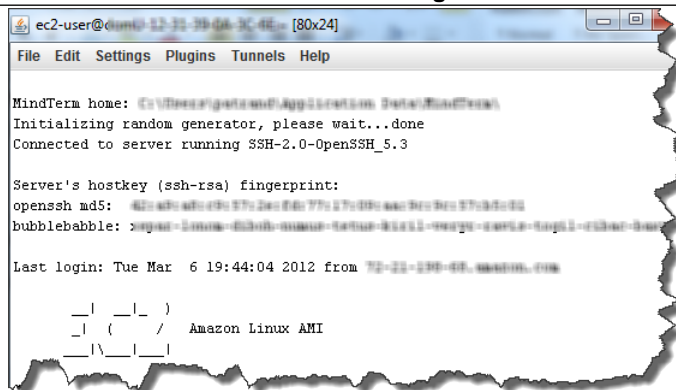
Note

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 **Stored 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.

Getting Started with AWS Web Application Hosting for Linux

Connect to Your Amazon EC2 Instance from Windows Using PuTTY



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 Windows Using PuTTY

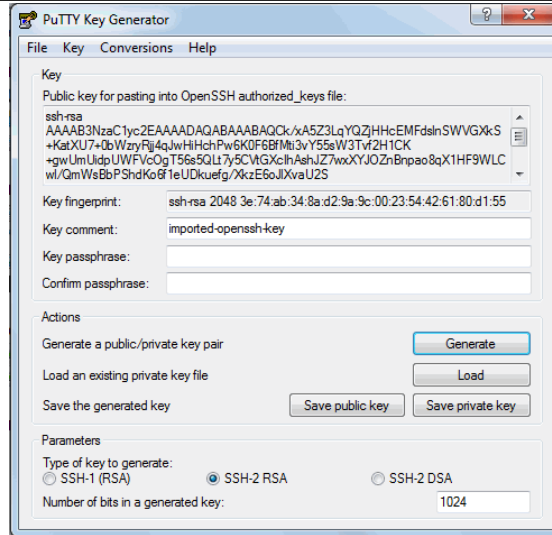
If you are running Windows from your local machine, you will need to install PuTTY and PuTTYGen since SSH is not built in. To connect to a Linux instance, you must retrieve the initial administrator password first, and then use it with Microsoft Remote Desktop. You'll need the contents of the private key file that you created (e.g., `mykeypair.pem`) in [Step 5: Create a Key Pair \(p. 14\)](#).

To connect to your Amazon EC2 instance from a Windows machine

1. Install PuTTY and PuTTYGen.
 - Download and install PuTTY. 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 5: Create a Key Pair \(p. 14\)](#)).
 - a. Launch PuTTYGen and select **Import Key** from the **Conversions** menu.
 - b. Browse for `mykeypair.pem`, and import the key.

Getting Started with AWS Web Application Hosting for Linux

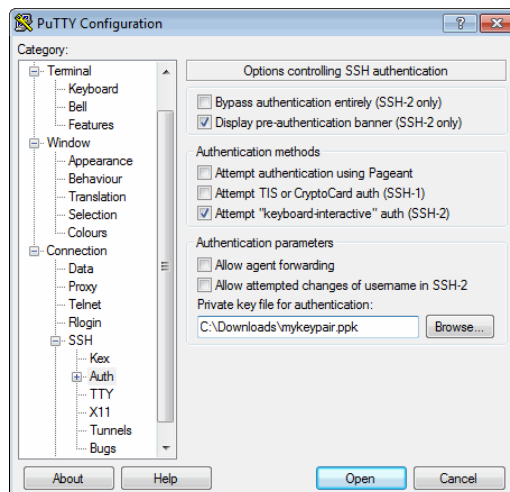
Connect to Your Amazon EC2 Instance from Windows Using PuTTY



- c. Click **Save private key**. Ignore the dialog box that asks if you want to do this without a passphrase. Save the key as `mykeypair.ppk`.
- d. Close PuTTYGen.

3. Configure the SSH settings.

- a. Launch PuTTY, expand the **SSH** node, and click **Auth**.
- b. Enter the location for `mykeypair.ppk`.

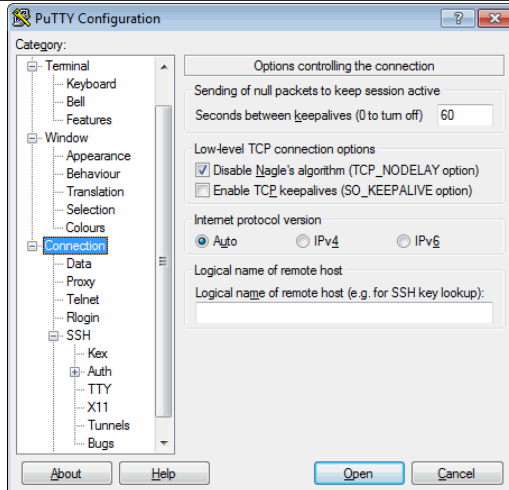


4. Modify the keepalive.

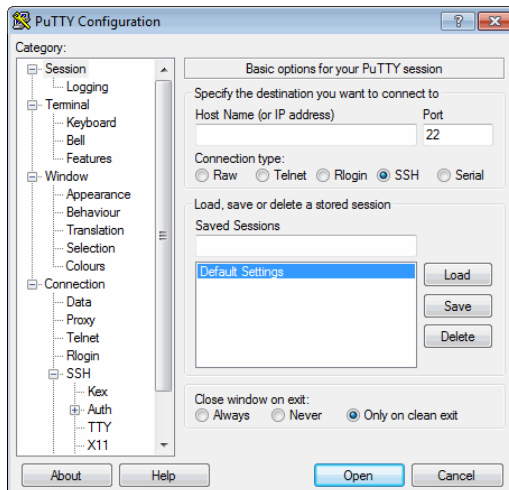
- a. In the PuTTY Configuration window, in the **Category** pane, click **Connection**.
- b. Type **60** in the **Seconds between keepalives (0 to turn off)** box. If you don't change this value, your session will time out.

Getting Started with AWS Web Application Hosting for Linux

Connect to Your Amazon EC2 Instance from Windows Using PuTTY



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



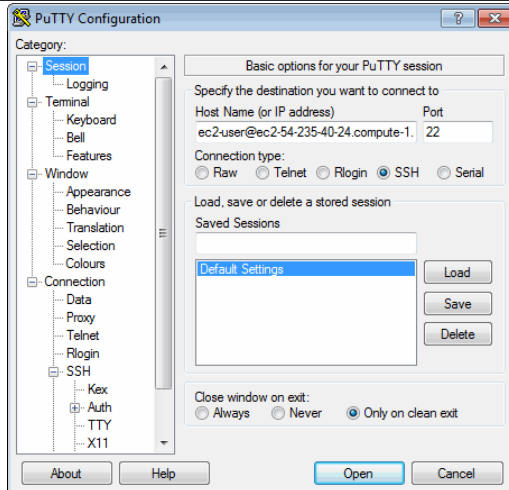
6. Type the DNS address of your Amazon EC2 instance.
 - a. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
 - b. Make sure **US East (N. Virginia)** is selected in the region selector of the navigation bar.
 - c. In the navigation pane, click **Instances**.
 - d. Select your running instance and note the public DNS address in the bottom pane.
 - e. In the PuTTY Configuration window, click **Sessions** in the **Category** pane, and in the **Host Name (or IP address)** box, type `<ec2-user@DNS address of your Amazon EC2 instance>`.

Note

We put `ec2-user` in front of the DNS name because the username for the AMI is `ec2-user`.

Getting Started with AWS Web Application Hosting for Linux

Connect to Your Amazon EC2 Instance from Windows Using PuTTY

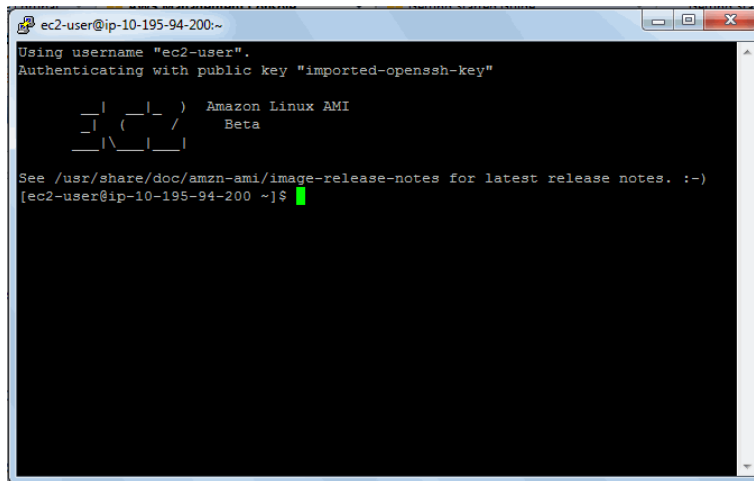


- f. Click **Open**. When the PuTTY Security Alert dialog box appears, click **Yes** to confirm that the fingerprint is OK. The SSH PuTTY window will launch.

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.

Your screen should look similar to the following screen.



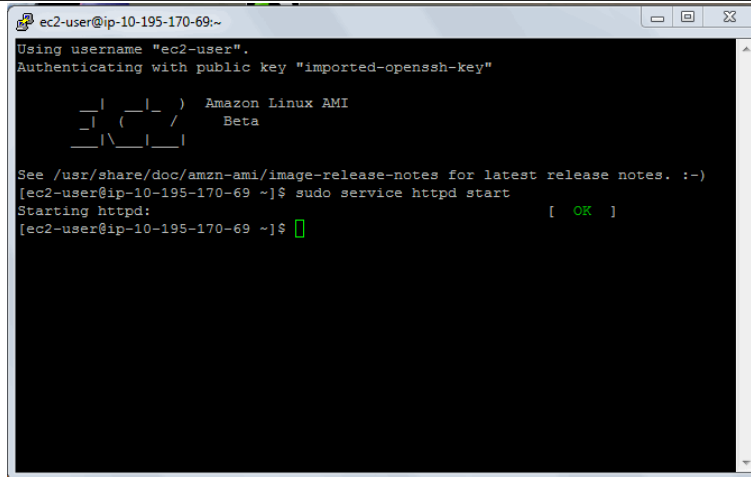
- g. Use the `sudo service httpd start` command to start the web server.

```
sudo service httpd start
```

Your screen should look similar to the following screen.

Getting Started with AWS Web Application Hosting for Linux

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

A screenshot of a terminal window showing an SSH connection to an Amazon Linux AMI instance. The terminal title is 'ec2-user@ip-10-195-170-69:~'. The prompt shows the user is 'ec2-user' and the host is 'ip-10-195-170-69'. The terminal displays the Amazon Linux AMI logo and version 'Beta'. It shows the command 'sudo service httpd start' being executed, which returns '[OK]'. The prompt then changes to '[ec2-user@ip-10-195-170-69 ~]\$'.

Now that you have successfully logged into your AMI, you are ready to configure your AMI. For instructions on how to configure your AMI, see [Configure the Amazon EC2 Instance \(p. 34\)](#).

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 machine.

Note

Most Linux and UNIX machines include an 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 5: Create a Key Pair \(p. 14\)](#).
2. Use the `chmod` command to make sure your private key file isn't publicly viewable. For example, for `mykeypair.pem`, you would enter:

```
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. Make sure **US East (N. Virginia)** is selected in the region selector of the navigation bar.
5. In the left navigation pane, click **Instances**.
6. Select your instance, and then click **Connect**.
7. 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.
8. Connect to your instance using the instance's public DNS name. For example, if the key file is `mykeypair.pem` and the instance's DNS name is `ec2-107-20-66-228.compute-1.amazonaws.com`, use the following command.

```
ssh -i mykeypair.pem ec2-user@ec2-107-20-66-228.compute-1.amazonaws.com
```

Note

We use **ec2-user** as the username in this exercise for this AMI.

You'll see a response like the following.

```
The authenticity of host 'ec2-107-20-66-228.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
```

9. Enter **yes**.

You'll see a response like the following.

```
Warning: Permanently added 'ec2-107-20-66-228.compute-1.amazonaws.com' (RSA)
to the list of known hosts.
```

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

```
sudo service httpd start
```

You'll see a response like the following.

```
Starting httpd [OK]
```

Now that you have successfully logged into your AMI, you are ready to configure your AMI. For instructions on how to configure your AMI, see [Configure the Amazon EC2 Instance \(p. 34\)](#).

Configure the Amazon EC2 Instance

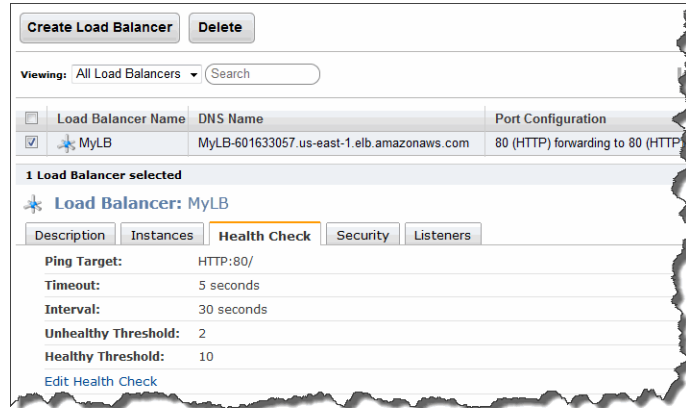
In this topic we will configure the running AMI. First, let's configure the health check for the load balancer so that we can connect to the instance through our load balancer. We will temporarily change the health check to point to the `install.php` script until our instance is configured. After the instance is configured we will point the health check back to `root`.

To configure the health check to point to the install script

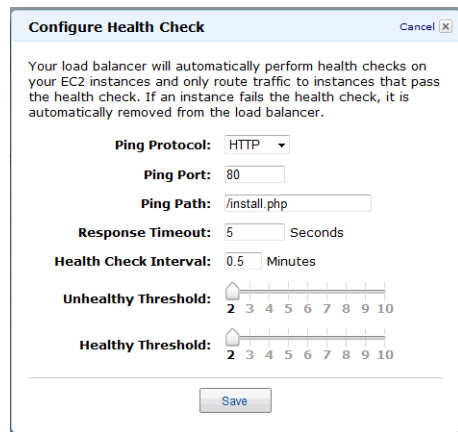
1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Make sure **US East (N. Virginia)** is selected in the region selector of the navigation bar.
3. In the left navigation pane, click **Load Balancers**.
4. Click your load balancer and click the **Health Check** tab.

Getting Started with AWS Web Application Hosting for Linux

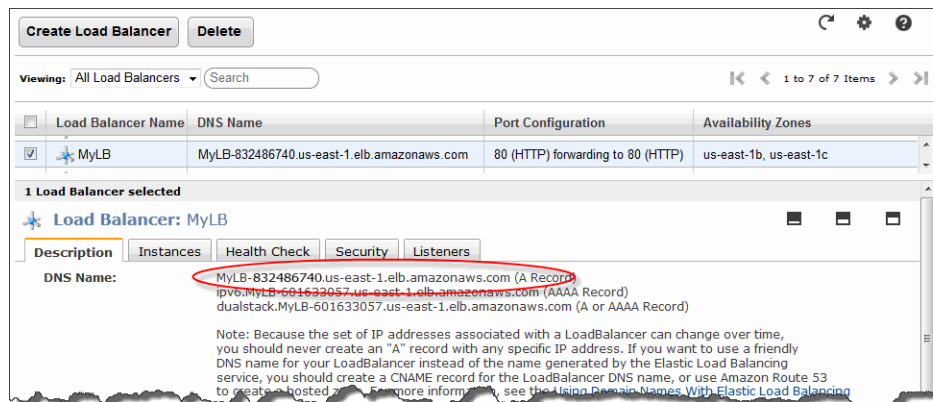
Configure the Amazon EC2 Instance



5. Click **Edit Health Check**.
6. In the **Configure Health Check** dialog box, type `/install.php` in the **Ping Path** box. Then click **Save**.



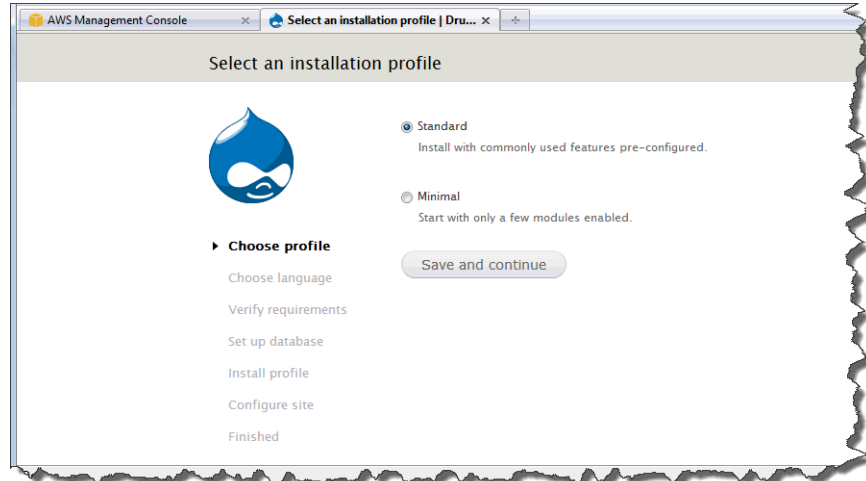
Configure the application



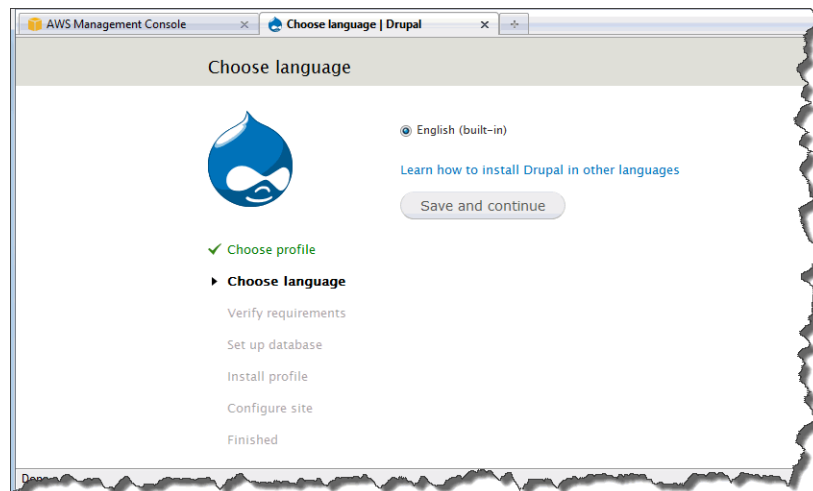
1. With your load balancer still selected, click the **Description** tab. Make a note of the public **DNS Name**.
2. Open your web browser, and type the public DNS address of your load balancer in the address bar.
3. On the **Choose profile** page, click **Standard** and **Save and continue**.

Getting Started with AWS Web Application Hosting for Linux

Configure the Amazon EC2 Instance



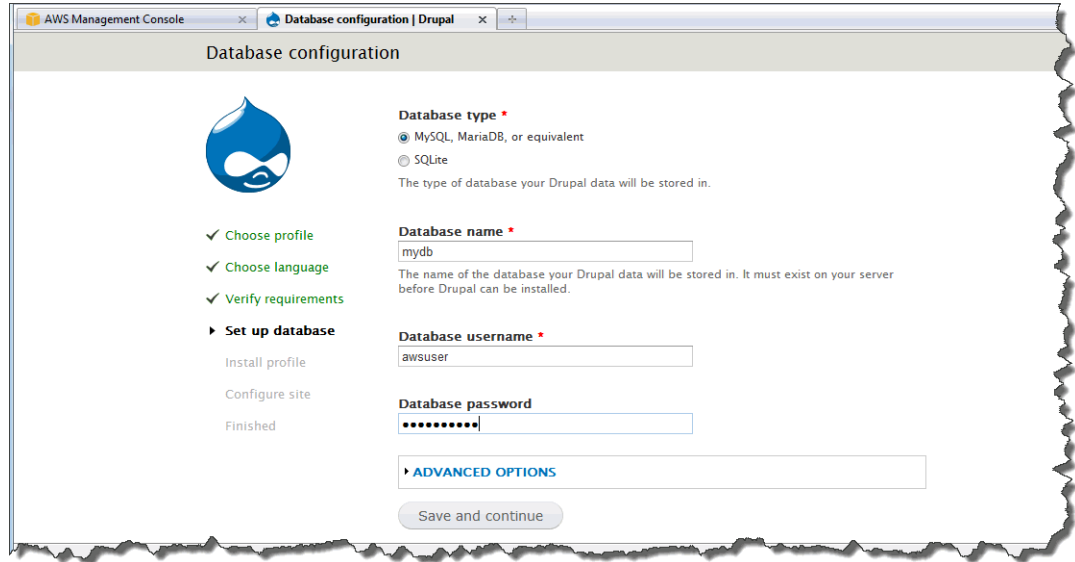
4. On the **Choose language** page, click **English** and **Save and continue**.



5. On the **Set up database** page enter the following information.
- Click **MySQL, MariaDB, or equivalent** for database type.
 - In the **Database name** box, type the name of your database. In our example, we use **mydb**.
 - In the **Database username** box, type the username for your database. In our example, we use **awsuser**.
 - In the **Database password** box, type the password for your database. In our example, we use **mypassword**.


Getting Started with AWS Web Application Hosting for Linux

Configure the Amazon EC2 Instance



AWS Management Console Database configuration | Drupal

Database configuration



- ✓ Choose profile
- ✓ Choose language
- ✓ Verify requirements
- ▶ Set up database
 - Install profile
 - Configure site
 - Finished

Database type *

☒ MySQL, MariaDB, or equivalent

☐ SQLite

The type of database your Drupal data will be stored in.

Database name *

The name of the database your Drupal data will be stored in. It must exist on your server before Drupal can be installed.

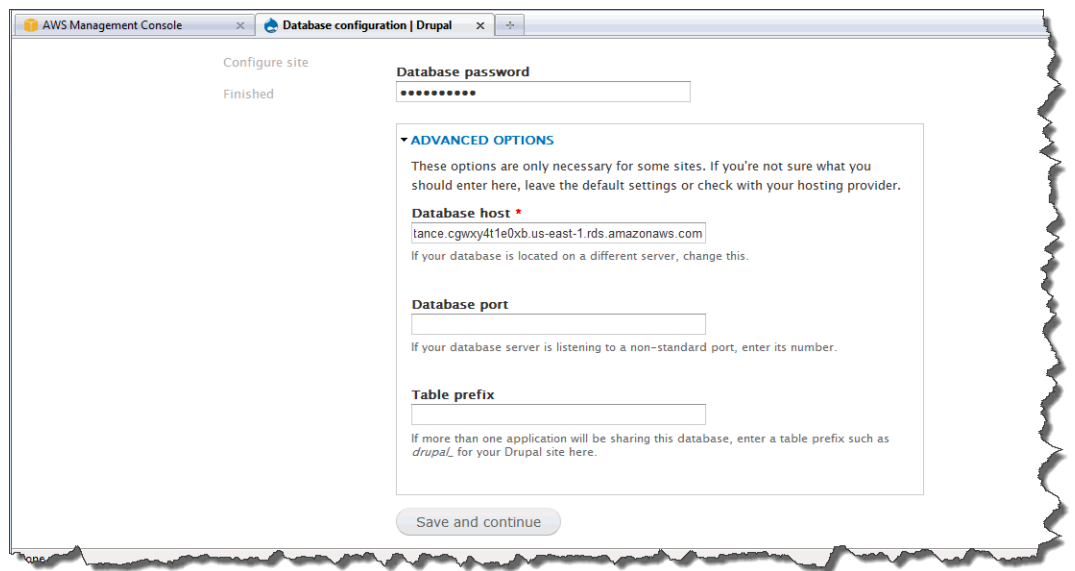
Database username *

Database password

▶ **ADVANCED OPTIONS**

Save and continue

- e. Click **ADVANCED OPTIONS**.
- f. In the **Database host** box, type the Amazon RDS endpoint.



AWS Management Console Database configuration | Drupal

Configure site

Finished

Database password

▼ **ADVANCED OPTIONS**

These options are only necessary for some sites. If you're not sure what you should enter here, leave the default settings or check with your hosting provider.

Database host *

If your database is located on a different server, change this.

Database port

If your database server is listening to a non-standard port, enter its number.

Table prefix

If more than one application will be sharing this database, enter a table prefix such as *drupal_* for your Drupal site here.

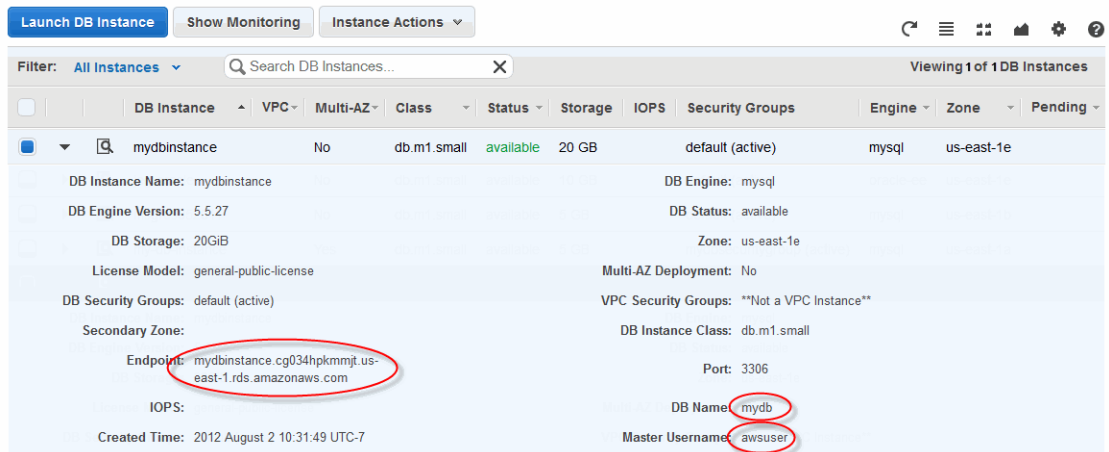
Save and continue

Note

You can find the Amazon RDS endpoint on the Amazon RDS console on the My DB Instances page as shown in the following image.

Getting Started with AWS Web Application Hosting for Linux

Configure the Amazon EC2 Instance



Note

Make sure that your database instance is up and running before proceeding to the next step. The status should say **available** as shown in the above diagram.

- g. Click **Save and continue**. The **Configure site** page appears.
6. On the **Configure site** page, enter the following information.
- a. In the **Site name** box, type the DNS address of the load balancer as you did at the beginning of this procedure.
 - b. In the **Site e-mail address** box, type an email address.
 - c. In the **Username** box, type a username.
 - d. In the **Password** box, type a password.
 - e. In the **Confirm password** box, re-type the password.

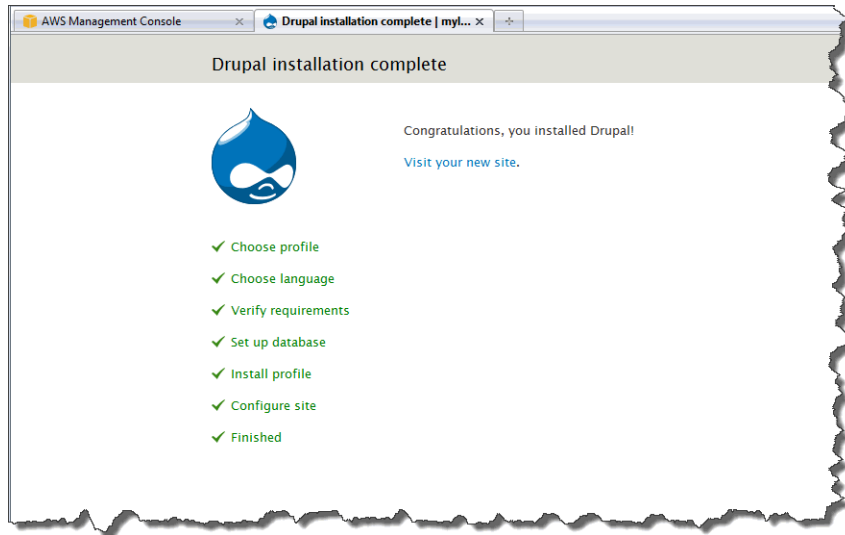
The screenshot shows the 'Configure site | Drupal' page. On the left, a sidebar lists steps: Choose profile, Choose language, Verify requirements, Set up database, Install profile, and Configure site (which is highlighted and marked as 'Finished'). The main content area has two sections: 'SITE INFORMATION' and 'SITE MAINTENANCE ACCOUNT'. The 'SITE INFORMATION' section has fields for 'Site name' (filled with 'mylb-832486740.us-east-1.elb.amazonaws.com') and 'Site e-mail address'. The 'SITE MAINTENANCE ACCOUNT' section has fields for 'Username', 'E-mail address', 'Password', and 'Confirm password'. There is also a 'Password strength' indicator.

- f. Click **Save and continue**.

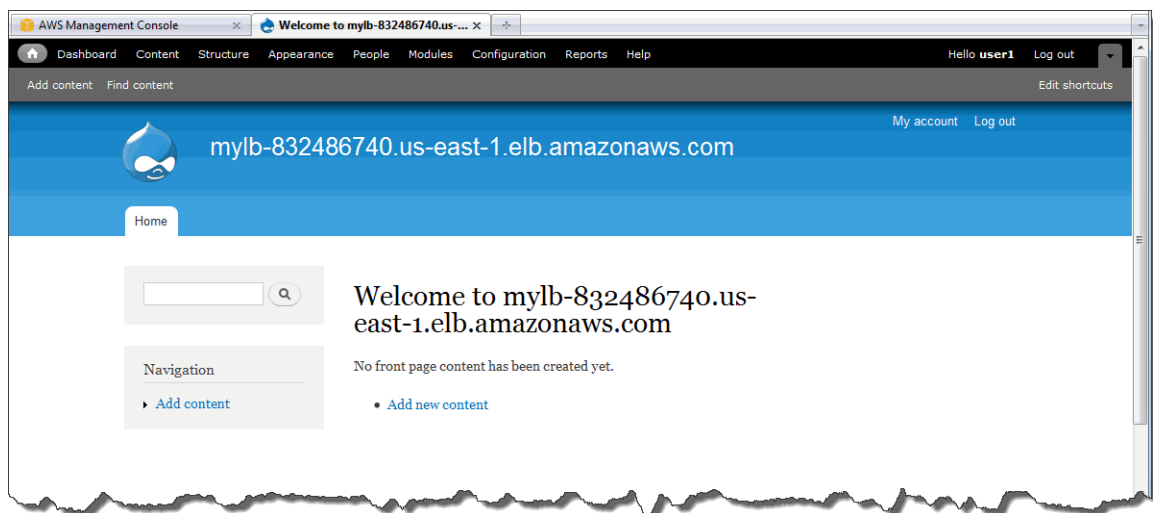
Getting Started with AWS Web Application Hosting for Linux

Configure the Amazon EC2 Instance

The installation is complete.



7. Click **Visit your new site**. Your new site appears.



8. Click **Add new content** to add one new article to your new site.

Now that our new site is created, we can backup your database.

To backup your database

1. At the prompt, type the following command to change to the home directory.

```
cd
```

2. At the prompt, type the following command to create a new folder called backups.

```
mkdir backups
```

3. At the prompt, type the following command to back up your existing database.

Getting Started with AWS Web Application Hosting for Linux

Configure the Amazon EC2 Instance

```
mysqldump -u awsuser -pmypassword mydb --host=your Amazon RDS endpoint > backups/backup.sql
```

Make sure to replace the your Amazon RDS endpoint with *your Amazon RDS endpoint* you noted in the previous step.

4. Verify the backup exists.
 - a. At the prompt, type the following command to change to the backups folder.

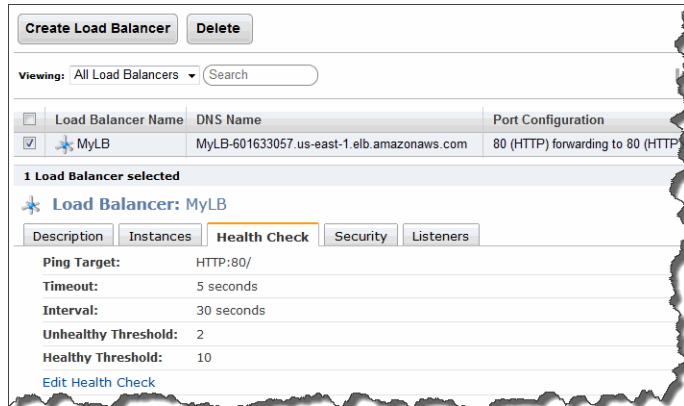
```
cd backups
```

- b. At the prompt, type the following command to list the contents of the directory.

```
ls -al
```

To configure the health check to point to the root document

1. In the AWS Management Console, click the Amazon EC2 tab.
2. Click **Instances** in the **Navigation** pane.
3. In the **Navigation** pane select US East (N. Virginia) from the **Region** drop-down menu.
4. In the **Load Balancers** pane, click your load balancer and click the **Health Check** tab.



5.
 - a. Click **Edit Health Check**.
 - b. In the **Configure Health Check** dialog box, type / in the **Ping Path** box.

Getting Started with AWS Web Application Hosting for Linux

Step 10: Create a Custom AMI

Configure Health Check Cancel

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer.

Ping Protocol: HTTP

Ping Port: 80

Ping Path: /

Response Timeout: 5 Seconds

Health Check Interval: 0.5 Minutes

Unhealthy Threshold: 2

Healthy Threshold: 2

Save

Congratulations! You just successfully deployed your web application using Amazon Web Services! Now, in the future if we decide we want to launch more instances we don't want to have to customize each one. Let's move on to the step to create a custom AMI with all our configuration changes.

Step 10: 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 using AWS CloudFormation. This is an optional step. If you prefer to finish the tutorial now, you can skip ahead to clean up your AWS resources in [Step 12: Clean Up \(p. 51\)](#).

To create an AMI from a running Amazon EBS-backed 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. Click **Instances** in the navigation pane.
4. On the **Instances** page, right-click your running instance and select **Create Image**.
5. Fill in a unique image name and an optional description of the image (up to 255 characters), and click **Create Image**.

Tip

If you're familiar with Amazon EC2 instance store-backed AMIs, the image name replaces the manifest name (e.g., `s3_bucket/something_of_your_choice.manifest.xml`), which uniquely identifies each Amazon Amazon EC2 instance store-backed AMI.

Amazon EC2 powers down the instance, takes images of any volumes that were attached, creates and registers the AMI, and then reboots the instance.

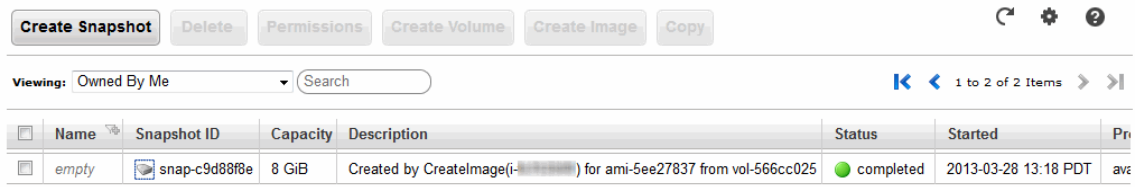
6. Go to the **AMIs** page and view the AMI's status. While the new AMI is being created, its status is *pending*.

It takes a few minutes for the whole process to finish.

7. Once your new AMI's status is *available*, go to the **Snapshots** page and view the new snapshot that was created for the new AMI. Any instance you launch from the new AMI uses this snapshot for its root device volume. You could update your Auto Scaling group with the new AMI, however we will do this as part of the AWS CloudFormation step.

Getting Started with AWS Web Application Hosting for Linux

Step 11: Launch New Environments Using AWS CloudFormation



The screenshot shows the AWS Management Console interface. At the top, there are buttons for 'Create Snapshot', 'Delete', 'Permissions', 'Create Volume', 'Create Image', and 'Copy'. Below these buttons, there is a 'Viewing:' dropdown set to 'Owned By Me' and a search bar. To the right, there are navigation icons and a page indicator '1 to 2 of 2 Items'. Below this is a table with columns: Name, Snapshot ID, Capacity, Description, Status, Started, and Progress. The table contains one row with the name 'empty', Snapshot ID 'snap-c9d88f8e', Capacity '8 GiB', Description 'Created by CreateImage(i-...) for ami-5ee27837 from vol-566cc025', Status 'completed', and Started '2013-03-28 13:18 PDT'.

	Name	Snapshot ID	Capacity	Description	Status	Started	Progress
<input type="checkbox"/>	empty	snap-c9d88f8e	8 GiB	Created by CreateImage(i-...) for ami-5ee27837 from vol-566cc025	completed	2013-03-28 13:18 PDT	100%

We've taken a lot of steps so far to create all of our AWS resources, deploy our application, and customize our AMI. Wouldn't it be great if we could save all of this information and launch new environments without having to manually configure everything again? We can! [AWS CloudFormation](#) is a way to launch environments easily. That is, when you launch an AWS CloudFormation environment, you are able to launch specific AMIs with particular key pairs, on pre-defined instance sizes, and behind load balancers. And if any portion of your environment fails to launch, the environment rolls itself back, terminating all the pieces along the way. Let's move on to the next topic to begin using AWS CloudFormation.

Step 11: Launch New Environments Using AWS CloudFormation

Topics

- [Create an AWS CloudFormation Template \(p. 42\)](#)
- [Modify a CloudFormation Template \(p. 46\)](#)
- [Create an AWS CloudFormation Stack \(p. 47\)](#)

You can use AWS CloudFormation to create and provision AWS infrastructure deployments predictably and repeatedly. Use AWS CloudFormation to build highly reliable, highly scalable, cost-effective applications without worrying about creating and configuring the underlying AWS infrastructure. AWS CloudFormation consists of template files you use to create and delete collections of resources as a single unit (an AWS CloudFormation stack). Using AWS CloudFormation you can leverage other services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), Amazon Simple Notification Service (Amazon SNS), Elastic Load Balancing, and Auto Scaling.

In this example, we'll use the CloudFormer tool to generate a template based on the AWS resources we just created. CloudFormer is intended to create a starting point for your template. After you create the template, you'll customize the template to launch a new environment with multiple instances spanning multiple Availability Zones to enable a fault-tolerant architecture.

This is an optional step. If you want to skip this step, you can move on to [Step 12: Clean Up \(p. 51\)](#) to begin deleting your resources.

Create an AWS CloudFormation Template

First, you'll need to create a template based on the resources you just created. You'll use a tool called CloudFormer that collects information about all your running resources and creates a template. CloudFormer is a prototype that helps you get started. You'll then make some tweaks to the template before you create your new stack. Visit the [AWS Forums](#) to learn more and to run the tool.

After generating the template and making a few tweaks, you may have something that looks like the following.

**Getting Started with AWS Web Application Hosting for
Linux
Create an AWS CloudFormation Template**

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "elbMyLB": {
      "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
      "Properties": {
        "AvailabilityZones": [
          "us-east-1b",
          "us-east-1c"
        ],
        "HealthCheck": {
          "HealthyThreshold": "2",
          "Interval": "30",
          "Target": "HTTP:80/",
          "Timeout": "5",
          "UnhealthyThreshold": "2"
        },
        "Listeners": [
          {
            "InstancePort": "80",
            "LoadBalancerPort": "80",
            "Protocol": "HTTP",
            "PolicyNames": [
            ]
          }
        ]
      }
    },
    "asgMyAutoScalingGroup": {
      "Type": "AWS::AutoScaling::AutoScalingGroup",
      "Properties": {
        "AvailabilityZones": [
          "us-east-1b",
          "us-east-1c"
        ],
        "Cooldown": "300",
        "DesiredCapacity": "1",
        "MaxSize": "1",
        "MinSize": "1",
        "LaunchConfigurationName": {
          "Ref": "lcMyLC"
        },
        "LoadBalancerNames": [
          {
            "Ref": "elbMyLB"
          }
        ]
      }
    },
    "lcMyLC": {
      "Type": "AWS::AutoScaling::LaunchConfiguration",
      "Properties": {
        "ImageId": "ami-45b77f2c",

```

**Getting Started with AWS Web Application Hosting for
Linux
Create an AWS CloudFormation Template**

```
        "InstanceType": "t1.micro",
        "KeyName": "mykeypair",
        "SecurityGroups": [
            {
                "Ref": "sgwebappsecuritygroup"
            }
        ]
    }
},
"aspMyScaleUpPolicy" : {
    "Type" : "AWS::AutoScaling::ScalingPolicy",
    "Properties" : {
        "AdjustmentType" : "ChangeInCapacity",
        "AutoScalingGroupName" : { "Ref" : "asgMyAutoScalingGroup" },
        "Cooldown" : "300",
        "ScalingAdjustment" : "1"
    }
},
"cwCPULAlarmHigh": {
    "Type": "AWS::CloudWatch::Alarm",
    "Properties": {
        "AlarmDescription": "Scale-up if CPU > 60% for 10 minutes",
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/EC2",
        "Statistic": "Average",
        "Period": "300",
        "EvaluationPeriods": "2",
        "Threshold": "60",
        "AlarmActions": [ { "Ref": "aspMyScaleUpPolicy" } ],
        "Dimensions": [
            {
                "Name": "AutoScalingGroupName",
                "Value": { "Ref": "asgMyAutoScalingGroup" }
            }
        ],
        "ComparisonOperator": "GreaterThanOrEqualToThreshold"
    }
},
"rdsmydbinstance": {
    "Type": "AWS::RDS::DBInstance",
    "Properties": {
        "AllocatedStorage": "5",
        "BackupRetentionPeriod": "1",
        "DBInstanceClass": "db.m1.small",
        "DBName": "MyDatabase",
        "DBParameterGroupName": "default.mysql5.1",
        "Engine": "mysql",
        "EngineVersion": "5.1.57",
        "MasterUsername": "awsuser",
        "MasterUserPassword": "awsuser",
        "Port": "3306",
        "PreferredBackupWindow": "10:00-10:30",
        "PreferredMaintenanceWindow": "sun:05:00-sun:05:30",
        "MultiAZ": "true",
        "DBSecurityGroups": [
```


Getting Started with AWS Web Application Hosting for Linux

Create an AWS CloudFormation Template

```
        {
          "Ref": "dbsgmydbsecuritygroup"
        }
      ]
    }
  },
  "sgwebappsecuritygroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
      "GroupDescription": "this is a security group for demo",
      "SecurityGroupIngress": [
        {
          "IpProtocol": "tcp",
          "FromPort": "80",
          "ToPort": "80",
          "SourceSecurityGroupName": "amazon-elb-sg",
          "SourceSecurityGroupOwnerId": "amazon-elb"
        },
        {
          "IpProtocol": "tcp",
          "FromPort": "22",
          "ToPort": "22",
          "CidrIp": "0.0.0.0/0"
        }
      ]
    }
  },
  "dbsgmydbsecuritygroup": {
    "Type": "AWS::RDS::DBSecurityGroup",
    "Properties": {
      "GroupDescription": "my database security group",
      "DBSecurityGroupIngress": [
        {
          "EC2SecurityGroupName": {
            "Ref": "sgwebappsecuritygroup"
          },
          "EC2SecurityGroupOwnerId": "123456789012"
        }
      ]
    }
  },
  "Description": ""
}
```

You'll want to make a couple of changes to this template before you launch your new environment. In this tutorial, you only launched one Amazon EC2 instance. However, it's a best practice to launch multiple instances across multiple Availability Zones; you'll want to update your Auto Scaling group to launch more instances. You'll also want to launch a new environment with your custom AMI. Finally, You'll update your database information to include your database name and password.

Modify a CloudFormation Template

Now that the template has been created, let's modify it so that you can launch a new environment with the custom AMI so that you will have multiple instances spanned across multiple Availability Zones.

To launch a new stack with a modified template

1. Open the template you created using CloudFormer.
2. Update the **Min Size**, **Max Size**, and **Desired Capacity** in the **Auto Scaling** group to 2.

```
"asgMyAutoScalingGroup": {
  "Type": "AWS::AutoScaling::AutoScalingGroup",
  "Properties": {
    "AvailabilityZones": [
      "us-east-1b",
      "us-east-1c"
    ],
    "Cooldown": "300",
    "DesiredCapacity": "2",
    "MaxSize": "2",
    "MinSize": "2",
    "LaunchConfigurationName": {
      "Ref": "lcMyLC"
    },
    "LoadBalancerNames": [
      {
        "Ref": "elbMyLB"
      }
    ]
  }
},
```

3. Update the **Image ID** in the **Launch Configuration** group to the custom AMI that you created in [Step 10: Create a Custom AMI \(p. 41\)](#).

Note

Your AMI ID will be different than the one you see below.

Add the **UserData** information as you see below so the web server will startup on bootup.

Note

Make sure to put the comma right after the `]` and just before **UserData**.

```
"lcMyLC": {
  "Type": "AWS::AutoScaling::LaunchConfiguration",
  "Properties": {
    "ImageId": "ami-91b270f8",
    "InstanceType": "t1.micro",
    "KeyName": "mykeypair",
    "SecurityGroups": [
      {
        "Ref": "sgwebappsecuritygroup"
      }
    ],
    "UserData": "..."
  }
},
```

Getting Started with AWS Web Application Hosting for Linux

Create an AWS CloudFormation Stack

```
    "UserData"      : { "Fn::Base64" : { "Fn::Join" : ["", [
      "#!/bin/bash -v\n",
      "sed -i 's/AllowOverride None/AllowOverride All/g' /etc/httpd/conf/ht
      tpd.conf\n",
      "service httpd start\n"
    ]}}}
  }
},
```

4. Update the following parameters in the **Database** group.

- Update **DBName** to **awsuser**.
- Update **MasterUserPassword** to **mypassword**.

```
"mydbinstance": {
  "Type": "AWS::RDS::DBInstance",
  "Properties": {
    "AllocatedStorage": "5",
    "BackupRetentionPeriod": "1",
    "DBInstanceClass": "db.m1.small",
    "DBName": "mydb",
    "DBParameterGroupName": "default.mysql5.1",
    "Engine": "mysql",
    "EngineVersion": "5.1.57",
    "MasterUsername": "awsuser",
    "MasterUserPassword": "mypassword",
    "Port": "3306",
    "PreferredBackupWindow": "08:30-09:00",
    "PreferredMaintenanceWindow": "fri:03:30-fri:04:00",
    "MultiAZ": "true",
    "DBSecurityGroups": [
      {
        "Ref": "dbsgmydbsecuritygroup"
      }
    ]
  }
},
```

Now that you have modified the template, let's move on to the next step to launch your new environment using your template.

Create an AWS CloudFormation Stack

Now that you have modified your AWS CloudFormation template, let's create a new stack to launch your new environment. Before you launch your new stack, you can verify that it works by cleaning up all your AWS resources *except* for your key pair and custom AMI. For instructions on how to clean up your resources see [Step 12: Clean Up \(p. 51\)](#).

Getting Started with AWS Web Application Hosting for Linux

Create an AWS CloudFormation Stack

Note

AWS CloudFormation is a free service. However, you are charged for the AWS resources you include in your stacks at the current rates for each. For more information about AWS pricing, go to the detail page for each product on <http://aws.amazon.com/pricing>.

To create an AWS CloudFormation stack

1. Launch the Create Stack wizard.
 - a. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation/>.
 - b. Make sure **US East (N. Virginia)** is selected in the region selector of the navigation bar.
 - c. Click **Create Stack**.
2. Select a template.
 - a. On the **SELECT TEMPLATE** page of the **Create Stack** wizard, type a stack name in the **Stack Name** box.
 - b. Click **Upload a Template URL** and type in the location where you saved your modified template.
 - c. Click **Show Advanced Options**.
 - d. Select **Create a new SNS topic** from the **Amazon SNS Topic** drop-down menu. You will receive email notifications when resources are created and when they are deleted.
 - e. Type **CRM** in the **New topic name** box.
 - f. Type your email address in the **Email** box and accept the rest of the default values.
 - g. Click **Continue**.

The screenshot shows the 'Create Stack' wizard in the AWS CloudFormation console. The 'SELECT TEMPLATE' step is active, indicated by a progress bar at the top. The wizard includes a 'Stack Name' field with the value 'MyCloudFormationStack'. Under the 'Template' section, the 'Upload a Template File' option is selected, with a text box showing 'C:\MyFolder\MyModifie' and a 'Browse...' button. The 'Provide a Template URL' option is also visible. Below this, the 'Show Advanced Options' checkbox is checked. The 'Notifications (optional)' section shows 'Amazon SNS Topic Create a new SNS topic' selected, with 'New topic name' set to 'CRM' and 'Email' set to 'inedoe@example.com'. The 'Creation Timeout (minutes)' is set to 'none', and 'Rollback on Failure' is set to 'Yes'. A 'Continue' button is at the bottom right.

3. In the Add Tags page, click **Continue**.
4. Review your settings. You can make changes to the settings by clicking the edit link for a specific step in the process.

Getting Started with AWS Web Application Hosting for Linux

Create an AWS CloudFormation Stack

Create Stack [Cancel X]

SELECT TEMPLATE SPECIFY PARAMETERS ADD TAGS **REVIEW**

Please review the information below, then click Continue to create the stack.

Stack Information [Edit Stack]

Stack Name: MyCloudFormationStack

Stack Description:

Template: https://cf-templates-1.s3.amazonaws.com/2013087G9Y-cloudformer.template

IAM Acknowledgement: false

Estimated Cost: Cost

Notification [Edit Notification]

Notification: CRM

Creation Timeout (minutes): none

Rollback on Failure: true

< Back [Continue >]

5. Click **Continue**.

A confirmation window opens.

Create Stack [Cancel X]

Your stack is being created. You can select the stack and, using the tabs shown below, you can see the current state and track the progress of the stack creation. When your stack is created, it will have a status of CREATE_COMPLETE. The Events tab will show you what is currently being created and will also contain any error messages if there are problems creating your stack.

1 Stack selected

Stack: USEast1

Description Outputs Resources **Events** Template Parameters

Stack Events

Time	Type
2011-02-16 09:21 PST	AWS::Stack
2011-02-16 09:21 PST	AWS::AutoScaling::AutoScalingGroup
2011-02-16 09:20 PST	AWS::AutoScaling::AutoScalingGroup
2011-02-16 09:20 PST	AWS::AutoScaling::LaunchConfiguration
2011-02-16 09:20 PST	AWS::AutoScaling::LaunchConfiguration

[Close >]

6. Click **Close**.

The confirmation window closes, returning you to the **CloudFormation** page. Your new AWS CloudFormation template appears in the list with the status set to **CREATE_IN_PROGRESS**.

CloudFormation Stacks (Showing 5 of 5)				
[Create Stack] [Update Stack] [Delete Stack] Viewing: Active [Show/Hide] [Refresh]				
	Name	Created	Status	Description
<input checked="" type="checkbox"/>	MyCloudFormationStack	2013-03-28 14:24:37 UTC-7	CREATE_IN_PROGRESS	

Note

Your stack will take a few minutes to create. Make sure to click **Refresh** in the Stacks page to see when the template has successfully been created.

After your stack has been created, you can verify that it all works.

Getting Started with AWS Web Application Hosting for Linux

Create an AWS CloudFormation Stack

To verify your AWS CloudFormation stack works

1. Connect to one of your newly created Amazon EC2 instances as you did in [Connecting to Your Amazon EC2 Instance from a Linux/UNIX Machine Using a Standalone SSH Client \(p. 33\)](#) or [Connect to Your Amazon EC2 Instance from Windows Using PuTTY \(p. 29\)](#).
2. Switch the database to the Amazon Relational Database Service (Amazon RDS) database
 - a. Navigate to the Amazon RDS console and get your new Amazon RDS endpoint that AWS CloudFormation created. Follow the same process you used in [Step 9: Deploy Your Application \(p. 27\)](#).
 - b. In the SSH window, at the prompt type the following command.

```
cd
```

- c. At the prompt type the following command.

```
mysql -u awsuser -pmypassword --database=mydb --host=your Amazon RDS endpoint < backups/backup.sql
```

Make sure to replace your Amazon RDS endpoint with your Amazon RDS endpoint. This is the endpoint that you retrieved in the previous step.

- d. At the prompt type the following command.

```
cd /var/www/html/sites/default
```

- e. At the prompt, type the following command to list the contents of the directory.

```
ls
```

- f. At the prompt, type the following command to open the settings.php file.

```
vi settings.php
```

- g. Use the **PgDn** key to navigate to a section that looks similar to the following.

```
$databases = array (
  'default' =>
    array (
      'default' =>
        array (
          'database' => 'mydb',
          'username' => 'awsuser',
          'password' => 'mypassword',
          'host' => 'mydbinstance.cgwxy4tle0xb.us-east-1.rds.amazonaws.com',

          'port' => '',
          'driver' => 'mysql',
          'prefix' => '',
        ),
    ),
);
```

```
    ),  
  );
```

- h. Press the **i** key to enter Insert mode.
- i. Replace `mydbinstance.cgwxy4t1e0xb.us-east-1.rds.amazonaws.com` with your new Amazon RDS endpoint.

Note

Right-click to paste the contents if you prefer copy and paste.

```
$databases = array (  
  'default' =>  
    array (  
      'default' =>  
        array (  
          'database' => 'mydb',  
          'username' => 'awsuser',  
          'password' => 'mypassword',  
          'host' => 'mycloudformationstack-rdsmydbinstance-1276gxy2eians.cg  
wxy4t1e0xb.us-east-1.rds.amazonaws.com',  
          'port' => '',  
          'driver' => 'mysql',  
          'prefix' => '',  
        ),  
      ),  
    ),  
  );
```

- j. Press **Esc** and then **:wq** to save the file and quit.
- k. Repeat the same steps for the other Amazon EC2 instance.
- l. Verify that it all works by navigating to the Amazon EC2 console and getting the DNS name for the new load balancer your Amazon CloudFormation stack created, just as you did in [Step 9: Deploy Your Application \(p. 27\)](#).

Where You're At

Congratulations! You have just launched your new environment using the AWS resources you created in this tutorial. Your Elastic Load Balancer is now pointing to both of your Amazon EC2 instances across multiple Availability Zones.

To delete an AWS CloudFormation stack

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation/>.
2. Click the stack you want to delete, and click **Delete Stack**.

Step 12: Clean Up

Topics

- [Terminate Your Amazon EC2 Instances in Your Auto Scaling Group \(p. 52\)](#)
- [Terminate Your DB Instance \(p. 53\)](#)

Getting Started with AWS Web Application Hosting for Linux

Terminate Your Amazon EC2 Instances in Your Auto Scaling Group

- [Delete Your CloudWatch Alarm \(p. 54\)](#)
- [Delete Your Elastic Load Balancer \(p. 55\)](#)
- [Delete a Key Pair \(p. 55\)](#)
- [Delete an Amazon EC2 Security Group \(p. 56\)](#)
- [Delete Your Custom AMI \(p. 56\)](#)

Congratulations! You have just deployed your web application. Now, to prevent accruing any further charges, let's terminate our environments and clean our resources.

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

Getting Started with AWS Web Application Hosting for Linux

Terminate Your DB Instance

```
INSTANCE    i-xxxxxxx    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 DB Instance

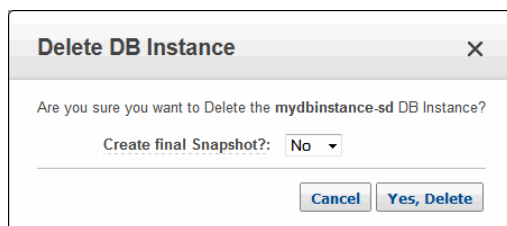
To terminate your DB Instance

- Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
- Make sure **US East (N. Virginia)** is selected in the region selector in the navigation bar.
- In the left navigation pane, click **DB Instances**.
- Locate the DB Instance in your list of DB Instances.

Getting Started with AWS Web Application Hosting for Linux

Delete Your CloudWatch Alarm

5. Select the check box next to the DB Instance, and then choose **Delete** from the **Instance Actions** drop-down list at the top of the page.
6. Select **No** in the **Create final snapshot?** drop-down list.



The dialog box is titled "Delete DB Instance" with a close button (X) in the top right corner. It contains the text "Are you sure you want to Delete the mydbinstance-sd DB Instance?". Below this text is a label "Create final Snapshot?:" followed by a dropdown menu currently set to "No". At the bottom of the dialog are two buttons: "Cancel" and "Yes, Delete".

If this weren't an exercise, you might create a final snapshot before you deleted the DB Instance so that you could restore the DB Instance later.

Note

Creating a final snapshot incurs additional storage fees.

7. Click **Yes, Delete**.

Amazon RDS begins terminating the instance. As soon as the DB Instance status changes to **deleted**, you stop incurring charges for that DB Instance.

To delete your DB Security Group

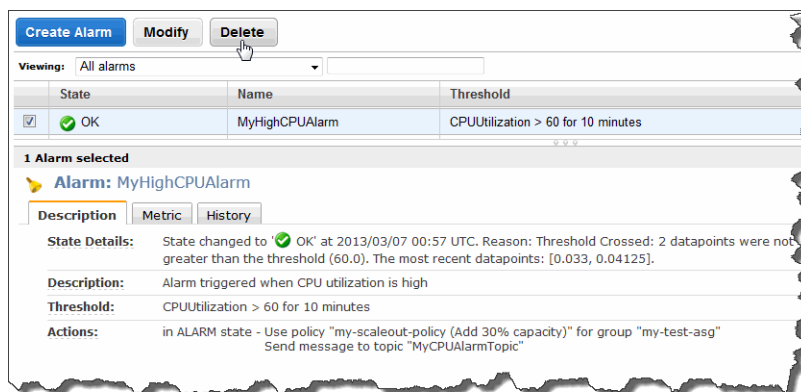
1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. Click **DB Security Groups** in the navigation pane on the left.
3. Select a DB Security Group and click **Delete DB Security Group**.
4. Click **Yes, Delete**.

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**.



The screenshot shows the Amazon CloudWatch console's "Alarms" page. At the top are buttons for "Create Alarm", "Modify", and "Delete". Below these is a "Viewing:" dropdown set to "All alarms". A table lists alarms with columns for "State", "Name", and "Threshold". One alarm, "MyHighCPUALarm", is selected with a checkmark in the "State" column. Below the table, it says "1 Alarm selected". The details for "Alarm: MyHighCPUALarm" are shown, including tabs for "Description", "Metric", and "History". The "Description" tab is active, showing "State Details", "Description", "Threshold", and "Actions".

State	Name	Threshold
<input checked="" type="checkbox"/> OK	MyHighCPUALarm	CPUUtilization > 60 for 10 minutes

1 Alarm selected

Alarm: MyHighCPUALarm

Description | Metric | History

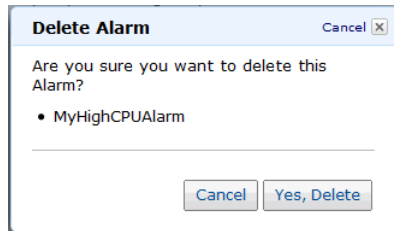
State Details: State changed to 'OK' at 2013/03/07 00:57 UTC. Reason: Threshold Crossed: 2 datapoints were not greater than the threshold (60.0). The most recent datapoints: [0.033, 0.04125].

Description: Alarm triggered when CPU utilization is high

Threshold: CPUUtilization > 60 for 10 minutes

Actions: in ALARM state - Use policy "my-scaleout-policy (Add 30% capacity)" for group "my-test-asg" Send message to topic "MyCPUALarmTopic"

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



Delete Your Elastic Load Balancer

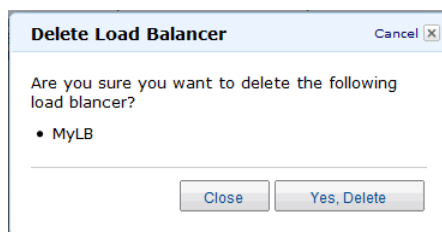
As soon as your load balancer becomes available, AWS bills you for each hour or partial hour that you keep the load balancer running. After you've decided that you no longer need the load balancer, you can delete it.

To delete your load balancer

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



- 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.

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**.

Delete Your Custom AMI

To delete an AMI and a snapshot

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Make sure **US East (N. Virginia)** is selected in the region selector in the navigation bar.
3. In the left navigation pane, click **AMIs**.
4. Right-click your AMI, and select **Deregister**. When prompted, click **Continue**.

The image is deregistered, which means it is deleted and can no longer be launched.

5. Go to the **Snapshots** page, right-click the snapshot, and select **Delete Snapshot**. When prompted, click **Yes, Delete**.

The snapshot is deleted.

Amazon Route 53

Amazon Route 53 is a scalable Domain Name System (DNS) web service. It provides secure and reliable routing to your infrastructure that uses Amazon Web Services (AWS) products, such as Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing, or Amazon Simple Storage Service (Amazon S3). You can also use Amazon Route 53 to route users to your infrastructure outside of AWS.

Amazon Route 53 automatically routes queries to the nearest DNS server in a global network of DNS servers, resulting in low latency. It is an authoritative DNS service, meaning it translates friendly domain names like `www.example.com` into IP addresses like `192.0.2.1`.

You can manage your DNS records through the Amazon Route 53 API, or set account-level user and access management through the Identity and Access Management (IAM) API. Like other AWS products, there are no contracts or minimum commitments for using Amazon Route 53—you pay only for the domains you configure and the number of queries that the service answers. For more information about Amazon Route 53 pricing, see [Amazon Route 53 Pricing](#).

The following procedure explains the high-level steps you need to take to use Route 53 for this example. For instructions on how to do steps one through four, go to [Amazon Route 53 Getting Started Guide](#). For information on how to create an alias in the last step, go to [How to Create an Alias Record Set](#) in the *Amazon Route 53 Developer Guide*.

To use Amazon Route 53

1. Create a hosted zone for `example.com`.
2. Create a new DNS record for your static content (e.g., `static.example.com`) that points to your CloudFront distribution (e.g., `d18k4jybr69gw2.cloudfront.net`).
3. Create a new DNS record for your website (e.g., `www.example.com`) that points to your Elastic Load Balancer CNAME.
4. Confirm your requests are complete.
5. Update the registrar's name server records.
6. Create an alias for your load balancer which responds to queries for `example.com` and `www.example.com`. You use the hosted zone ID for the load balancer (e.g., `Z3DZXE0Q79N41H`).

Amazon CloudFront

[Amazon CloudFront](#) is a content delivery service from Amazon Web Services that helps you improve the performance, reliability, and availability of your web sites and applications. The content you deliver with CloudFront will be stored on a server referred to as an *origin server*. Amazon CloudFront works by distributing your web content (such as images, video, and so on) using a network of edge locations around the world. Your content is served from your configured Amazon S3 bucket or custom origin, to the edge location that is closest to the user who requests it. In this example, to make use of Amazon CloudFront, we would store our static content (images, html, etc.) in an Amazon S3 bucket, and then create a CloudFront distribution from our S3 bucket. Once our CloudFront distribution is created, we simply update the code to point our static content to our CloudFront distribution. For more information on how to do this, go to [Start Using CloudFront with Amazon S3](#) in the *Amazon CloudFront Getting Started Guide*.

Pricing

Topics

- [Amazon EC2 Cost Breakdown](#) (p. 59)
- [Amazon RDS Cost Breakdown](#) (p. 61)
- [Summing It All Up](#) (p. 63)

The [AWS Simple Monthly Calculator](#) estimates your monthly bill. It provides per service cost breakdown as well as an aggregate monthly estimate. You can also use the tool to see an estimation and breakdown of costs for common solutions. This topic walks you through an example of how to use the AWS Simple Monthly Calculator to estimate your monthly bill for the sample web application we just created.

Note

AWS pricing you see in this documentation is current as of the last update to this document. You should go to [AWS Service Pricing Overview](#) for the latest pricing.

Amazon EC2 Cost Breakdown

The following table shows the characteristics for Amazon EC2 we have identified for this web application hosting architecture.

Characteristic	Metric	Description
Uptime	24 hrs/day	Assuming there is an average of 30.5 days in a month, the instance runs 732 hours/month
Machine Characteristics	ti.micro instance	613 MB of memory, up to 2 ECUs (for short periodic bursts), EBS storage only, 32-bit or 64-bit platform For a list of instance types, go to http://aws.amazon.com/ec2/instance-types/ .

Getting Started with AWS Web Application Hosting for Linux

Amazon EC2 Cost Breakdown

Characteristic	Metric	Description
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 made to the volume per second.
Data Transfer	Data In: 0.005 GB/day Data Out: 0.05 GB/day	There are approximately 1,000 hits per day and each response is about 50 KB and each request is about 5 KB.
Instance Scale	2	On average in a given day, there are 2 instances running.
Elastic Load Balancing	Hourly usage: 732 hrs/month Data processed: 1.525 GB/month	ELB is used 24 hrs/day, 7 days/week ELB processes a total of 0.055 GB/day (data in + data out)
Detailed Monitoring	\$3.50 per instance per month	We have setup detailed monitoring for our Amazon EC2 instances.

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

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
	2	24 Hours/Day	Micro	Linux	Default	<input checked="" type="checkbox"/>
<input type="checkbox"/> EBS-Optimized						

+ Compute: Amazon EC2 Reserved Instances:

Description	Instances	Usage	Instance Type	Operating System	Offering and Term	Tenancy	Detailed Monitoring
	0	0 Hours/Month	Small	Linux	Medium Utilization	Default	<input type="checkbox"/>
<input type="checkbox"/> EBS-Optimized							
3 yr term							

+ Storage: Amazon EBS Volumes:

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

Elastic IP:

Number of Additional Elastic IPs:

Elastic IP Non-attached Time: Hours/Month

Number of Elastic IP Remaps: Per Month

Data Transfer:

Inter-Region Data Transfer Out: GB/Month

Data Transfer Out: GB/Day

Data Transfer In: GB/Day

Intra-Region Data Transfer: GB/Month

Public IP/Elastic IP Data Transfer: GB/Month

Elastic Load Balancing:

Number of Elastic LBs:

Total Data Processed by all ELBs: GB/Day

The total monthly cost is the sum of the cost of the running instances, EBS volumes and I/O requests, elastic load balancers, the data processed by the elastic load balancers, and Amazon CloudWatch metrics.

**Getting Started with AWS Web Application Hosting for
Linux
Amazon RDS Cost Breakdown**

Variable	Formula	Calculation
Instance Cost	Instance cost per hour Number of instances x Uptime in hours -----	\$0.02 2 x 732 ----- \$29.28
Additional Storage	Storage rate X Storage Amount (GB) + (I/O requests rate x seconds per month x Request rate(per 1M requests)) -----	\$0.10 X 10 + (100 x ~2.6M x \$0.10)/1M ----- \$27.35
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
Amazon CloudWatch	Number of instances x Detailed Monitoring Rate -----	2 x \$3.50 ----- \$7.00

We use the AWS Simple Monthly calculator to estimate this. With the calculator, the total cost for Amazon EC2 is \$81.94.

Amazon RDS Cost Breakdown

The following table shows the characteristics for Amazon RDS we have identified for this web application hosting architecture.

Characteristic	Metric	Description
Uptime	24 hrs/day	24 Assuming there is an average of 30.5 days in a month, the instance runs 732 hours/month
Database Characteristics	Small Amazon RDS instance	1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, Moderate I/O Capacity

Getting Started with AWS Web Application Hosting for Linux

Amazon RDS Cost Breakdown

Characteristic	Metric	Description
Provisioned Storage	5 GB/month	Amazon provides 5 GB to 1 TB of associated storage capacity for your primary data set.
Requests	2M IOP/month	We have 1,000 hits per day at a rate of 5 IOP per hit on site. Assume there are 30.5 days in a month on average. This is a total of 152,500 I/O requests per month or 1M (rounded to the nearest million), but since the write I/O request will double since data is also replicated to the standby instance, we have a total of 2M.
Deployment Type	Multi-AZ	We will run our database instance across multiple Availability Zones.
Additional Backup Storage	none	We'll use up to the provisioned amount which is 5 GB.
Data Transfer	Data in: 0 GB Data out: 0 GB	There is no data transfer from RDS to the Internet.
Database Instance Scale	1	We need one database instance.

The following image shows the cost breakdown for Amazon RDS in the AWS Simple Monthly Calculator.

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 RDS is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. [Clear Form](#)

+ Amazon RDS On-demand DB Instances:

Description	DB Instances	Usage	DB Engine and License	Class and Deployment	Storage	IOPS
	1	24 Hours/Day	MySQL	Small Multi-AZ	Provisioned 5 GB	01

+ Additional Backup Storage (Free backup storage up to 100% of provisioned Storage):

Backup Storage: 0 GB-month of Storage

+ Amazon RDS Reserved DB Instances:

Description	DB Instances	Usage	DB Engine and License	Class and Deployment	Offering and Term	Storage	IOPS
	0	0 Hours/Month	MySQL	Small Standard (Single-AZ) o	Medium Utilization 3 yr term	Standard 20 GB	0

Data Transfer:

Data Transfer Out: 0 GB/Month

Data Transfer In: 0 GB/Month

Intra-Region Data Transfer: 0 GB/Month

Because we do not have data transfer in or out or backup storage, the total monthly cost is the sum of the cost of the running instances, provisioned storage, and I/O requests.

**Getting Started with AWS Web Application Hosting for
Linux
Summing It All Up**

Variable	Formula	Calculation
Instance Cost	Instance cost per hour	\$0.153
	Number of instances	1
	x Uptime in hours	x 732
	-----	-----
		\$112.00
Provisioned Storage	Storage rate	\$0.20
	x Storage Amount (GB)	x 5
	-----	-----
		\$1.00
I/O Requests	I/O rate	\$0.10
	x Number of requests (millions)	x ~2
	-----	-----
		\$0.22

We use the AWS Simple Monthly calculator to estimate this. With the calculator, the total cost for Amazon RDS is \$113.52.

Summing It All Up

To calculate the total cost for this scenario, we add the cost for Amazon EC2, Amazon RDS, and the AWS Data Transfer Out, and subtract any discount that falls into the AWS Free Usage Tier.

The total AWS Transfer Out is an aggregate Data Transfer Out usage across Amazon EC2, and Amazon RDS. For Amazon EC2, we have 0.05 GB per day which is approximately 1.525 GB per month. For Amazon RDS we did not have any AWS Transferred out. Since 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)	0.1525 X \$0.00
	+ (Data out (GB) X Data Out Rate)	+ (0.525) X \$0.12
	-----	-----
		\$0.06

The following image shows an example of your monthly estimate.

Getting Started with AWS Web Application Hosting for Linux


Summing It All Up

Services

Estimate of your Monthly Bill (\$ 160.95)

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)

Compute:	\$	36.28	\$	81.94
Intra-Region Data Transfer:	\$	0.00		
EBS Volumes:	\$	1.00		
EBS IOPS:	\$	26.35		
EBS Snapshots:	\$	0.00		
Reserved Instances (One-time Fee):	\$	0.00		
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		

☒ Amazon RDS Service (US-East)

			\$	113.52
AWS Data Transfer In			\$	0.00
AWS Data Transfer Out			\$	0.06
AWS Support (Basic)				\$ 0.00

Free Tier Discount:

			\$	-34.58
--	--	--	----	--------

Total One-Time Payment:

			\$	0.00
--	--	--	----	------

Total Monthly Payment:

			\$	160.95
--	--	--	----	--------

The total cost of this web application is estimated at \$160.95 per month including the free tier discount.

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 <i>only</i> 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 2011-09-30 release of Getting Started. This guide was last updated on March 8, 2012.

Change	Description	Release Date
New content	Created new document	30 September 2011
Added new section	Added new section to talk about AWS Identity and Account Management	24 October 2011
Changed application and Linux AMI	Modified example to demonstrate how to set up Drupal on an Amazon Linux AMI	23 November 2011
Added new section	Added section for connecting to Amazon EC2 using the MindTerm client	8 March 2012