Getting Started with AWS
Web Application Hosting for Microsoft Windows
Getting Started with AWS: Web Application Hosting for Microsoft Windows

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.
# Table of Contents

Overview .................................................................................................................. 1
How Does AWS Help? ................................................................................................. 2
Web Application Hosting Architecture ....................................................................... 4

Getting Started ......................................................................................................... 6
Step 1: Sign Up for the Service .................................................................................. 7
Step 2: Install the Command Line Tools .................................................................. 7
Step 3: Create an Amazon S3 Bucket ................................................................... 7
Where You're At ........................................................................................................ 9
Step 4: Create a CloudFront Distribution ................................................................ 10
Where You're At ....................................................................................................... 13
Step 5: Create an Elastic Load Balancer ................................................................. 13
Where You're At ....................................................................................................... 17
Step 6: Create and Configure Your Amazon EC2 Security Group ....................... 18
Step 7: Create a Key Pair ......................................................................................... 19
Step 8: Launch Amazon EC2 Instances Using Auto Scaling .................................. 20
Where You're At ....................................................................................................... 22
Step 9: Create a CloudWatch Alarm ....................................................................... 23
Where You're At ....................................................................................................... 27
Step 10: Add Amazon RDS ...................................................................................... 27
Create a DB Security Group ..................................................................................... 28
Authorize Access ...................................................................................................... 28
Launch an Instance ................................................................................................. 29
Step 11: Deploy Your Application ........................................................................... 33
Connect to Your Amazon EC2 Instance ................................................................. 34
Deploy Sample Application ...................................................................................... 36
Modify the Application ............................................................................................. 37
Step 12: Create a Custom AMI ................................................................................ 40
Step 13: Launch New Environments Using AWS CloudFormation ....................... 41
Create an AWS CloudFormation Template .............................................................. 42
Modify an AWS CloudFormation Template ............................................................. 45
Create an AWS CloudFormation Stack .................................................................. 47
Step 14: Clean Up ................................................................................................... 50
Delete Your CloudWatch Alarm ............................................................................ 51
Delete Your Elastic Load Balancer ......................................................................... 51
Terminate Your Amazon EC2 Instances in Your Auto Scaling Group ................ 52
Terminate Your DB Instance ................................................................................... 54
Delete a Key Pair ..................................................................................................... 54
Delete an Amazon EC2 Security Group ................................................................. 55
Delete an Amazon CloudFront Distribution ......................................................... 55
Delete Objects and an Amazon S3 Bucket ............................................................. 55
Amazon Route 53 ................................................................................................. 57
Pricing ......................................................................................................................... 58
Amazon S3 Cost Breakdown ..................................................................................... 58
Amazon CloudFront Cost Breakdown ................................................................... 60
Amazon EC2 Cost Breakdown ............................................................................... 62
Amazon RDS Cost Breakdown .............................................................................. 65
Summing It All Up .................................................................................................. 67
Related Resources ................................................................................................. 68
Document History .................................................................................................... 70
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 .NET application using several key AWS services and following best practices. 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.
• Create a location to store static files for your web application.
• Create a content delivery network.
• Launch, connect, secure, and deploy a sample .NET web application 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 a sample .NET application.
• 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.

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

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

- **Getting Started with AWS** — Provides information about Amazon Web Services, with helpful links for learning more.
- **Getting Started with AWS Free Usage Tier** — Provides information about how to get started with the free usage tier.
- **Hosting Websites on Amazon S3** in the *Amazon Simple Storage Service Developer Guide* — Provides a walkthrough in just a few steps of a static website deployment that does not require running an application.
- **Getting Started with AWS CloudFormation** in the *AWS CloudFormation User Guide* — Helps you quickly get started using an AWS CloudFormation WordPress blog sample template without needing to figure out the order in which AWS services need to be provisioned or worry about the subtleties of how to make those dependencies work.
- **Getting Started with AWS Computing Basics for Windows** - 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 Windows platform and walks you through a deployment that uses this architecture.
- **Amazon Elastic Compute Cloud Microsoft Windows Guide** - Provides information that helps you get started using Amazon EC2 instances that run the Microsoft Windows Server operating system.

### 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.

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Amazon Web Services</th>
<th>Benefits</th>
</tr>
</thead>
</table>
| Servers need to be provisioned to handle peak capacity and the unused cycles are wasted at other times. | • Amazon Elastic Compute Cloud (EC2)  
• Amazon Elastic Load Balancing  
• Auto Scaling  
• Amazon CloudWatch | • 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. |
### How Does AWS Help?

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Amazon Web Services</th>
<th>Benefits</th>
</tr>
</thead>
</table>
| Need a content delivery network (CDN) to provide low-latency, high data transfer speeds so end users don’t experience unnecessary delays. | • Amazon CloudFront  
• Amazon Simple Storage Service (Amazon S3) | • 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. |

<table>
<thead>
<tr>
<th>Challenges</th>
<th>AWS Components</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need to provide security to protect application servers from outside malicious users.</td>
<td>Amazon Security Group</td>
<td>An Amazon Security Group lets you specify the protocols, ports, and source IP address ranges that are allowed to reach your Amazon EC2 instances.</td>
</tr>
</tbody>
</table>
### Web Application Hosting Architecture

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

![Web Application Hosting Architecture Diagram](image)

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:
• Sign up for AWS.
• Create an Amazon S3 bucket and store files in the bucket.
• Create an Amazon CloudFront distribution.
• Launch, connect, secure, and deploy a .NET sample application to an Amazon EC2 instance.
• Create a Custom AMI.
• Set up an Elastic Load Balancer to distribute traffic across your Amazon EC2 instances.
• Scale your fleet of instances automatically using Auto Scaling.
• Monitor your AWS resources using Amazon CloudWatch.
• Create a database instance and use it with a sample .NET application.
• Create an AWS CloudFormation template based on the resources you created.
• Clean up your AWS resources.

For more information on how to use Amazon Route 53 in this architecture, see Amazon Route 53 (p. 57).
Let's suppose you want to build an ASP.NET web application where customers can upload and browse photos. 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 deployment variations for web applications, but this walkthrough focuses on one example that follows best practices and uses many of the Amazon Web 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

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. The package has a README that includes instructions on how to install the command line tools. After you have installed the command line tools, try a couple of commands to make sure they work.

After you have installed the command line tools, you can start creating your AWS resources. First you are going to need a place to store your photos when you upload them to the website. Amazon Simple Storage Service is a perfect solution for storing static objects such as photos. Let's proceed to the next step to create our first Amazon S3 bucket.

Step 3: Create an Amazon S3 Bucket

We'll use Amazon Simple Storage Service (Amazon S3) to store our photos and other static objects. We'll make the objects fully public so you can see how Amazon S3 acts as an image server in the cloud. And because Amazon S3 storage can be redundant across multiple data centers, it provides very high levels of durability for only a few cents per gigabyte per month.

Every object in Amazon S3 is stored in a bucket. Before we can store data in Amazon S3 we must create a bucket.

Note

You are not charged for creating a bucket; you are only charged for storing objects in the bucket and for transferring objects in and out of the bucket. The charges you will incur working through this example are minimal. For more information, go to Amazon S3 Pricing.

In this step we will create an Amazon S3 bucket for static content and enable logging to hold application data.
To create a bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. On the Amazon S3 console, click Create Bucket.

3. Enter a bucket name in the Bucket Name field.

   The bucket name you choose must be unique across all existing bucket names in Amazon S3. One way to help ensure uniqueness is to prefix your bucket names with the name of your organization. For this example, we’ll use webapplication; however, you should choose a unique name.

   Bucket names must comply with the following requirements. Bucket names:
   - Can contain lowercase letters, numbers, underscores (_), and dashes (-)
   - Must start with a number or letter
   - Must be between 3 and 255 characters long
   - Must not be formatted as an IP address (e.g., 265.255.5.4), and do not use periods.

   In some AWS regions, there might be additional restrictions on bucket names. If you intend to access objects by using a URL, we recommend that you create bucket names that are DNS-compliant. For more information, see the Amazon Simple Storage Service Console User Guide.

   Note
   After you create a bucket, you cannot change its name. In addition, the bucket name is visible in the URL that points to the objects stored in the bucket. Ensure that the bucket name you choose is appropriate.

4. In the Region box, select a region.

   By default, Amazon S3 creates buckets in the US Standard region. You can choose a region to optimize latency, minimize costs, or address regulatory requirements. Objects stored in a region never leave that region unless you explicitly transfer them to another region. For more information about regions, see Introduction to Amazon S3 in the Amazon Simple Storage Service Console User Guide.

   Click Set Up Logging. Log files will be created and stored in your Amazon S3 bucket. The log files will store application data.

5. In the Create a Bucket — Set Up Logging, do the following:
   - Select the Enabled check box.
   - In the Target Bucket box, select webappapplication.
   - In the Target Prefix box, type webapp-logging.
6. When the settings are as you want them, click **Create**.

When Amazon S3 successfully creates your bucket, the console displays the properties of your empty bucket.

After this step, you would normally upload your content and set permissions. However, for this exercise we will do so after we deploy our web application. For more information about Amazon S3, see [Amazon Simple Storage Service (S3) Documentation](https://docs.aws.amazon.com/AmazonS3/latest/userguide/).

## Where You're At

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

Now that you have finished creating your bucket, you need to think about how you deliver the content. Let's suppose you have people viewing your website from other parts of the world. The experience for these people may be unsatisfactory if they experience a lot of latency. **Amazon CloudFront** is a good solution for the latency problem. Let's move on to the next step to create an Amazon CloudFront distribution.
Amazon CloudFront is a content delivery service from Amazon Web Services that helps you improve the performance, reliability, and availability of your websites and applications. The content you deliver with Amazon CloudFront will be stored on 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 has the lowest latency to the user who requests it. Amazon CloudFront is a good choice if you have users that are globally distributed, expect high volume traffic, or even expect at least one hit per day. In this example, the origin is the Amazon S3 bucket where you store static content. Let's create a CloudFront distribution.

**Note**
This step is optional. You can skip this step and still deploy your web application at the end of this tutorial.

To create a distribution with an Amazon S3 origin, we will use the AWS Management Console.

**To create an Amazon CloudFront distribution**

1. Open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Make sure that **US East (N. Virginia)** is selected in the region selector of the navigation bar.
3. In the CloudFront console click **Create Distribution**.
4. On the first page of the Create Distribution wizard, accept the default selection, **Download**, and click **Continue**.
5. On the Create Distribution page under **Origin Settings**, select the Amazon S3 bucket that you created earlier.
6. For the value of Origin ID, accept the default value.

7. Under Default Cache Behavior Settings, accept the default values, and CloudFront will:
   • Forward all requests that use the CloudFront URL for your distribution (for example, http://d111111abcdef8.cloudfront.net/index.html) to the Amazon S3 bucket that you specified in Step 4.
   • Allow end users to use either HTTP or HTTPS to access your objects.
   • Cache your objects at CloudFront edge locations for 24 hours.
   • Exclude query string parameters, if any, when forwarding requests for objects to your origin.

   For more information about cache behavior options, go to Cache Behaviors in the Amazon CloudFront Developer Guide.

8. Under Distribution Details, specify the following details:

   a. If your distribution will use a CNAME, for CNAMEs, enter the CNAME alias you want to associate with this distribution. You must own the domain name and have created the CNAME record with your registrar. For this example, leave CNAMEs blank.
b. If you want to use a default root object with your distribution, for **Default Root Object** enter the default root object to associate with the distribution. For example, `index.html`. For this example, leave this field blank.

c. If you want to enable logging, for **Logging** select **On**, and then from **Bucket for Logs**, select the Amazon S3 bucket to which you want to save your logs. For this example, we select `webapplication.s3.amazonaws.com`, however you should select the bucket that you created. Type the log file prefix in the **Log Prefix** field. For this example, we type `webapp-logging/`.

d. In **Comments**, you can enter any comments you want to include about the distribution. We will leave this blank for this example.

e. For the **Distribution Status**, select **Enabled** if you want the distribution to accept end-user requests for content as soon as it is deployed. Otherwise, if you prefer to enable the distribution later, choose **Disabled**. For this example, select **Enabled**.

9. If you are satisfied with the distribution settings, click **Create Distribution**.

After creating the distribution, it might take a few minutes to deploy. The distribution's current status is displayed in the console under **Status**. **InProgress** indicates that the distribution is not yet fully deployed.

**Note**

Even if the distribution's status is **Deployed**, you still must enable the distribution for use before end users can retrieve content.
When your distribution is deployed, you are ready to reference your content with your new Amazon CloudFront domain name or CNAME.

For more information about Amazon CloudFront, see Amazon CloudFront Documentation. Since the deployment step can take a few minutes to complete, we'll move on to the next step and check our progress later on.

**Where You're At**

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

While we are waiting for the Amazon CloudFront distribution to be deployed, 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 5: 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 13: Launch New Environments Using AWS CloudFormation (p. 41) 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/](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**.
   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](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/).
   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.

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

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.
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**.
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.
Step 6: 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 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**, and then click **Create Security Group**.
6. In the Create Security Group dialog box, type webappsecuritygroup in the Security group name field, and a description of your choice in the Description field.
7. On the Inbound tab, click **Add Rule**, and select **HTTP** from the Type list.
8. Select **Custom IP** from the Source field, and enter the name of the security group that's associated with your load balancer, for example, amazon-elb/amazon-elb-sg. 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 **RDP** from the **Type** list to connect to your Amazon EC2 instances. Select **Anywhere** from the **Source** list.

![Create Security Group](image)

**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 your key pair.

### Step 7: 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 8: 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 Windows AMI running Microsoft Windows Server 2008 and Microsoft Internet Information Services (IIS). We use an t1.micro instance type, and use our security group and our key pair we created in the previous steps. In this example, the 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, US East (N. Virginia).

Note

The AMI that is used in this example is part of the AWS Free Usage Tier. If you are eligible for the free usage tier, then you will not be charged for launching the Amazon EC2 instance. If you are not eligible for the AWS Free Usage Tier, 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-c082e3a9 --instance-type t1.micro --group webappsecuritygroup --key mykeypair

Auto Scaling returns the following:

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

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-xxxxxxxx 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 9: 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 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.
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.
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 the Auto Scaling Group drop-down menu.
   d. Select MyScaleUpPolicy (Add 1 instance) from the 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 Microsoft Windows

Step 9: Create a CloudWatch Alarm

1. Type an email address in the Email(s) box.
2. Click Add Action.
3. Click Continue.
4. In the Review page, click 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 10: Add Amazon RDS

**Topics**

- Create a DB Security Group (p. 28)
- Authorize Access (p. 28)
- Launch an Instance (p. 29)

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 11:
Deploy Your Application (p. 33).

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
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
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/](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**.

<table>
<thead>
<tr>
<th>For this parameter...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Model</td>
<td>Keep the default: <em>general-public-license</em>.</td>
</tr>
<tr>
<td>DB Engine Version</td>
<td>Keep the default: <em>5.5.27 (default)</em>.</td>
</tr>
<tr>
<td>DB Instance Class</td>
<td>Select <strong>db.m1.small</strong>. The DB Instance class defines the CPU and memory capacity of your DB instance.</td>
</tr>
<tr>
<td>Multi-AZ Deployment</td>
<td>Choose <strong>Yes</strong>. Although the Multi-AZ deployment is more expensive, it is a best practice.</td>
</tr>
</tbody>
</table>
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 mydbinstance 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 Password text box.

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

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

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.

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.
7. Launching can take a few minutes to complete. When you see the notice that your instance is being created, click 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.
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 deploy your sample web application.

Step 11: Deploy Your Application

Topics

• Connect to Your Amazon EC2 Instance (p. 34)
• Deploy Sample Application (p. 36)
• Modify the Application (p. 37)

Now that you've created all of your AWS resources, it's time to deploy your application to your Amazon EC2 instance. In this step, you'll use a sample application that you can download, modify, and deploy to your Amazon EC2 instance. Amazon provides an AWS Toolkit for Visual Studio to help you build and deploy your .NET applications. For information, go to http://aws.amazon.com/visualstudio/.
Connect to Your Amazon EC2 Instance

To connect to a Windows instance, you must retrieve the initial administrator password first, and then use it with Remote Desktop. You’ll need the contents of the private key file that you created (e.g., mykeypair.pem) in Step 7: Create a Key Pair (p. 19).

**Note**
It can take up to 30 minutes to get the original password from the time you launched your Amazon EC2 instance.

**To connect to your Windows instance**

1. Retrieve the auto-generated administrator password:
   a. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
   b. Make sure that **US East (N. Virginia)** is selected in the region selector of the navigation bar.
   c. Locate the instance on the **Instances** page.
   d. Right-click the instance and select **Get Windows Password**.
   
   The **Retrieve Default Windows Administrator Password** dialog box is displayed (it might take a few minutes after the instance is launched before the password is available).

   e. Click **Browse** to locate the private key file.

   f. Click **Decrypt Password**.

   The console displays the default administrator password for the instance.

   g. Save the password. You will need it to connect to the instance.

   **Note**
   This step only works for a new Amazon EC2 instance. Once you change the Administrator password or create a custom AMI, you need to remember the password.

2. In the EC2 console, make a note of the public DNS name of the instance. You will need it in the following steps.

3. Connect to the instance using Remote Desktop:
   a. Start the Remote Desktop application (e.g., from the **Start** menu, point to **All Programs > Accessories**, and then click **Remote Desktop Connection**).

   **Note**
   Most modern Windows operating systems from Windows XP onward already include the Remote Desktop application. If you’re using an old version of Windows, you can download the Remote Desktop application from the [Microsoft website](https://www.microsoft.com).

   b. Click **Show Options** and ensure that you are not specifying a domain.

   c. Enter the public DNS name of the instance and click **Connect**.

   d. Log in using **Administrator** as the username and the administrator password you got in the previous task as the password.

   **Note**
   You can copy and paste the password using Ctrl+C and Ctrl+V.

You’re now connected to your instance. You can work with it the same way you would work with any Windows server.
Caution
After you connect to any new Windows instance you've just launched, we recommend you change the Windows administrator password from the default value.

We will be updating this site; however start by making certain that you can connect to the server with Internet Information Services (IIS) Manager.

To connect to the server with IIS Manager

1. From your Amazon EC2 instance, click Start.
2. Click Administrative Tools.
3. Click Internet Information Services (IIS) Manager.
4. Expand the localhost node.
5. Expand the Sites node.
6. Right-click Default Web Site.
7. Select Manage Web Site and click Start.
8. In the Load Balancers page of the EC2 console, copy the public DNS address of your load balancer. Then paste it into your browser's address bar to make certain that you can connect to the site.

Now that we know our web server is working correctly, let's move on to the next step to deploy a sample application to our Amazon EC2 instance.
Deploy Sample Application

In this topic we will download a sample ASP.NET application, grant write access to the appropriate folders and files, and configure IIS to point to the new application on your Amazon EC2 instance.

To download the sample application

1. In the Amazon EC2 instance, create a folder for your sample application in your root directory called ImageGallery, `c:\ImageGallery`.
2. Add the website location for the BaseImageGallery application to trusted sites on your Amazon EC2 instance
   a. In the Amazon EC2 instance, run Internet Explorer from the Start menu.
   b. Click the gear icon in top right corner.
   c. Click Internet Options.
   d. Click the Security tab.
   e. Click Trusted Sites and then click Sites.
   g. Click the Advanced tab.
   h. Scroll down to Security.
   i. Un-check Do not save encrypted pages to disk and click OK.
3. Download the application.
      
      **Note**
      This is a case-sensitive URL. This zip file is a .NET application; however we are not going to delve into the details of .NET. Instead we’ll “paste and modify” the application.

   b. When the File Download dialog box appears, click Save As. Save the file to `c:\ImageGallery`.
      
      **Note**
      If you receive an error that Internet Explorer is unable to download the file, close the error dialog and try downloading the application again inside the same browser window.

   c. Paste the contents of the ImageGallery folder (found inside the zip file) into this directory, so that individual files sit directly under C:\ImageGallery.
To modify the physical path for your default website

1. Open IIS Manager on your Amazon EC2 instance and navigate to the Default Web Site again.
2. Right-click Default Web Site.
3. Select Manage Web Site and click Advanced Settings.
4. In the Physical Path box, type c:\ImageGallery and click OK.

Modify the Application

Now we are ready to modify the application so that our images are published to Amazon S3 automatically and we use Amazon RDS to index our content stored in Amazon S3. We will modify the code from the web server so we don’t expose the webmaster’s S3 credentials to end users.

Now we’ll modify the application in the following ways:

- Update your web.config file
- Modify your code to point to your Amazon CloudFront distribution or your Amazon S3 bucket
- Modify your code to point to your Amazon RDS file

To update your web.config file

- In the c:\ImageGallery folder on your Amazon EC2 instance, open the web.config file and find the appSettings section. Specify your AWS credentials using one of the methods described in Using Credentials in an Application.

For more information about managing AWS credentials, see Best Practices for Managing AWS Access Keys.

To modify code to point to an Amazon CloudFront distribution or an Amazon S3 bucket

1. In the c:\ImageGallery folder in your Amazon EC2 instance, open default.aspx.cs and replace http://your-non-dotted-bucket-name.s3.amazonaws.com/ with the name of your Amazon CloudFront distribution or your Amazon S3 bucket. For instance, in Step 4: Create a CloudFront Distribution (p. 10), our Amazon CloudFront distribution is d1j6fvxrgjehco.cloudfront.net.
2. A few lines below that, replace your-non-dotted-bucket-name with the name of your Amazon S3 bucket. For instance, in Step 3: Create an Amazon S3 Bucket (p. 7), our Amazon S3 bucket is webapplication.

Note
Make sure you do not forget the trailing slash for your Amazon S3 bucket or your Amazon CloudFront distribution.

```csharp
class _Default : System.Web.UI.Page
{
    public string PhotoGalleryBaseUrl = "http://d1j6fvxrgjehco.cloudfront.net/";
    ...
    private const string ImageBucketName = "webapplication";
}
```

If you skipped creating the CloudFront distribution, then you would enter your Amazon S3 bucket as in the following example:

```csharp
class _Default : System.Web.UI.Page
{
    public string PhotoGalleryBaseUrl = "http://webapplication.s3.amazonaws.com/";
    ...
    private const string ImageBucketName = "webapplication";
}
```

Note
This code publishes images to your Amazon S3 bucket and sets the permissions on each file to be public, that is readable by everyone.

To modify code to point to your Amazon RDS database

1. In the c:\ImageGallery folder in your Amazon EC2 instance, open default.aspx.cs if it's not open already. Replace the text for the following variables with the actual values. These values will be used to build your connection string using MySQL Connector/.NET. Many of these can be found on the DB Instances page of the RDS console at https://console.aws.amazon.com/rds/

   - Database endpoint
   - Database master user ID
   - Database master password
   - Database name
public partial class _Default : System.Web.UI.Page
{
    private string dbinstance = "mydbinstance.cgwxy4t1e0xb.us-east-1.rds.amazonaws.com";
    private string userid = "awsuser";
    private string password = "mypassword";
    private string database = "mydb";
}

2. Save and close the file.

Now it's time to deploy your sample website.

To restart your website

1. In the IIS Manager, navigate to Default Web Site.
2. Right-click Default Web Site.
3. Select Manage Web Site and then click Restart.
4. Find the DNS address of the load balancer.
   a. In the AWS Management Console, click the EC2 tab.
   b. In the navigation pane, click Load Balancers.
   c. In the list of load balancers, click MyLB and note the public DNS in the Description tab.
d. Type the DNS address of the load balancer in your web browser and verify that your application still works.

e. You will not see any images on the site because there are none currently in the Amazon RDS database. Upload images less than or equal to 4 MB to the website and check that they display correctly.

Note
If you have trouble displaying the images, you may not be connecting to your Amazon RDS database; make sure you typed your Amazon EC2 security group correctly when you authorized access for database security group. If you still have trouble, try re-authorizing your Amazon EC2 security group.

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

Step 12: 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 14: Clean Up (p. 50).

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.

The Create Image dialog box opens.
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 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.

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 13: Launch New Environments Using AWS CloudFormation**

**Topics**

- Create an AWS CloudFormation Template (p. 42)
- Modify an AWS CloudFormation Template (p. 45)
- 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 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 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 14: Clean Up (p. 50) 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.

```
{
    "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": []
                    }
                ]
            }
        },
        "distd18k4jybr69gw2cloudfrontnet": {
            "Type": "AWS::CloudFront::Distribution",
            "Properties": {
                "DistributionConfig": {
                    "S3Origin": {
                        "DNSName": "webapplication.s3.amazonaws.com"
                    },
                    "Enabled": "true",
                    "Logging": {
                        "Bucket": "webapplication.s3.amazonaws.com",
                        "Prefix": "webapp-logging/"
                    }
                }
            }
        }
    }
}
```
Getting Started with AWS Web Application Hosting for Microsoft Windows
Create an AWS CloudFormation Template

```json
"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-c082e3a9",
    "InstanceType": "t1.micro",
    "KeyName": "mykeypair",
    "SecurityGroups": [
      {
        "Ref": "sgwebappsecuritygroup"
      }
    ]
  }
},

"aspMyScaleUpPolicy": {
  "Type": "AWS::AutoScaling::ScalingPolicy",
  "Properties": {
    "AdjustmentType": "ChangeInCapacity",
    "AutoScalingGroupName": { "Ref": "asgMyAutoScalingGroup" },
    "Cooldown": "300",
    "ScalingAdjustment": "1"
  }
},

"cwCPUAlarmHigh": {
  "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"
  }
}
```
Getting Started with AWS Web Application Hosting for Microsoft Windows

Create an AWS CloudFormation Template

```
"AlarmActions": [ { "Ref": "aspMyScaleUpPolicy" } ],
"Dimensions": [ {
    "Name": "AutoScalingGroupName",
    "Value": { "Ref": "asgMyAutoScalingGroup" }
}
],
"ComparisonOperator": "GreaterThanThreshold"
},
"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": "07:00-07:30",
    "PreferredMaintenanceWindow": "sat:04:00-sat:04:30",
    "MultiAZ": "true",
    "DBSecurityGroups": [ {
        "Ref": "dbsgmydbsecuritygroup"
    } ]
},
"s3webapplication": { 
    "Type": "AWS::S3::Bucket"
},
"sgwebappsecuritygroup": { 
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": { 
        "GroupDescription": "for web app",
        "SecurityGroupIngress": [ {
            "IpProtocol": "tcp",
            "FromPort": "80",
            "ToPort": "80",
            "SourceSecurityGroupName": "amazon-elb-sg",
            "SourceSecurityGroupOwnerId": "amazon-elb"
        }, {
            "IpProtocol": "tcp",
            "FromPort": "3389",
            "ToPort": "3389",
            "CidrIp": "0.0.0.0/0"
        }
    ]
}
},
"dbsgmydbsecuritygroup": {
```

44
"Type": "AWS::RDS::DBSecurityGroup",
"Properties": {
  "GroupDescription": "security group for my web app",
  "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 multiple 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 an AWS 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 format of the CloudDistribution group to include DistributionConfig and the Logging portion.

3. Update the Min Size, Max Size, and Desired Capacity in the Auto Scaling group to 2.
4. Update the Image ID in the Launch Configuration group to the custom AMI that you created in Step 12: Create a Custom AMI (p. 40).

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

5. Update the following parameters in the Database group.

- Update DBName to mydb.
- Update MasterUserPassword to the master password you specified in Step 10: Add Amazon RDS (p. 27).
"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 14: Clean Up (p. 50).

**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](http://aws.amazon.com/pricing).

### To create an AWS CloudFormation stack

1. Launch the Create Stack wizard.
   b. Make sure **US East (N. Virginia)** is selected in the region selector in the top 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 **Provide a Template URL** and type 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.

3. Review your settings. You can make changes to the settings by clicking the edit link for a specific step in the process.

4. Click **Create Stack**.
5. Click **Close** to close the confirmation window.

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**.

**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.

---

**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.
AWS CloudFormation creates a new endpoint for your Amazon RDS database. You'll need to update the Amazon RDS endpoint on each instance in your default.aspx.cs file just as you did in Modify the Application (p. 37). You can connect to your Amazon EC2 instances using the new Administrator password that you created when you first connected to your instance. You'll also notice that you don't see the images that you stored in your original Amazon RDS database because you are now pointing to a new database.

To restore your original images you can create a backup of your database and then restore it from snapshot. For more information about backing up your Amazon RDS database, see Backing Up and Restoring DB Instances in the Amazon Relational Database Service User Guide. Now let's clean up all of your AWS resources. Deleting your AWS CloudFormation stack will delete all of the AWS resources it just created.

To delete an AWS CloudFormation stack

1. At the top of the AWS Management Console, click the AWS CloudFormation tab.
2. Click the stack you want to delete, and click Delete Stack.

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

Step 14: Clean Up

Topics

- Delete Your CloudWatch Alarm (p. 51)
- Delete Your Elastic Load Balancer (p. 51)
- Terminate Your Amazon EC2 Instances in Your Auto Scaling Group (p. 52)
- Terminate Your DB Instance (p. 54)
- Delete a Key Pair (p. 54)
- Delete an Amazon EC2 Security Group (p. 55)
- Delete an Amazon CloudFront Distribution (p. 55)
- Delete Objects and an Amazon S3 Bucket (p. 55)
Delete Your CloudWatch Alarm

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

To delete your alarm

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

Delete Your Elastic Load Balancer

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

To delete your load balancer

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

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

**Caution**

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

## Terminate Your Amazon EC2 Instances in Your Auto Scaling Group

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

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

### To remove the Amazon EC2 instance from the Auto Scaling group

1. Open a command prompt window: From a Windows computer, click **Start**. In the Search box, type `cmd`, and then press **Enter**.

2. You'll use the `as-update-auto-scaling-group` command to update the Auto Scaling group that we created earlier. At the command prompt, type the following, and then press **Enter**:

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

   Auto Scaling returns the following:

   ```
   OK-Updated AutoScalingGroup
   ```

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

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

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

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

   ```
   52
   ```
Getting Started with AWS Web Application Hosting for Microsoft Windows

Terminate Your Amazon EC2 Instances in Your Auto Scaling Group

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

1. 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 in the navigation bar.
3. In the left navigation pane, click **DB Instances**.
4. Locate the DB Instance in your list of DB Instances.
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.

   ![Delete DB Instance](image)

   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 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 an Amazon CloudFront Distribution

To disable a CloudFront distribution

1. Open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Select a CloudFront distribution and click **Disable**.
   
   A confirmation dialog box appears.
3. Click **Yes, Disable**.

   Amazon CloudFront disables the distribution.

To delete a CloudFront distribution

1. Select a CloudFront distribution and click **Delete**.
   
   A confirmation dialog box appears.
2. Click **Yes, Delete**.

   Amazon CloudFront deletes the distribution.

Delete Objects and an Amazon S3 Bucket

Before you can delete an Amazon S3 bucket, all objects within the bucket must be deleted.

You should also ensure that logging for your Amazon S3 bucket is disabled; otherwise, logs might be immediately written to your bucket after you delete your bucket's objects.

To disable logging

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** pane, right-click your bucket and then click **Properties**.
3. In the **Properties** pane, click **Logging**.
4. Clear the **Enabled** check box.

To delete an object

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** pane, click the bucket where the objects are stored.
3. In the list of objects, right-click the object that you want to delete and then click **Delete**.
A dialog box shows the actions you can take on the selected object(s).

**Tip**
You can use the `SHIFT` and `CTRL` keys to select multiple objects and perform the same action on them simultaneously.

4. In the confirmation message that appears, click **Yes, Delete**.

To delete a bucket, you must first delete all of the objects in it.

**To delete a bucket**

1. Continuing from the previous procedure, right-click the bucket you want to delete, and then click **Delete**.
2. In the confirmation message that appears, click **Yes, Delete**.
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 domains 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).
The AWS Simple Monthly Calculator estimates your monthly bill. It provides a per-service cost breakdown, as well as an aggregate monthly estimate. You can also use the calculator to see an estimate and breakdown of costs for common solutions. This topic walks you through an example of using the AWS Simple Monthly Calculator to estimate your monthly bill.

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

### Amazon S3 Cost Breakdown

The following table shows the characteristics for Amazon S3 we have identified for this web application hosting architecture.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
</table>
| Storage        | 0.001 GB/month | 20 JPEG @ 50 KB = 1 MB  
Total of 1.0 MB for 20 objects = 0.001 GB |
We will plan to update the objects twice a month. (PUT requests include from any location including Amazon EC2) 1 PUT request x 20 objects = 20 requests

We will transfer the objects 10 times per month to each of the 10 Amazon CloudFront edge locations. (GET requests include from any location including Amazon EC2) 10 GET requests x 10 CloudFront Nodes x 20 objects = 2000 requests

If the average size object is 50KB, and we make about 2000 requests per month, than the average data transfer is approximately 0.1 GB.

The following image shows the cost breakdown for Amazon S3 in the AWS Simple Monthly Calculator.

The total monthly cost is the sum of the cost for storing the objects, updating the objects, and transferring the objects to Amazon CloudFront.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioned Storage</td>
<td>Storage rate \times Storage Amount (GB)</td>
<td>$0.125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$0.00</td>
</tr>
</tbody>
</table>

The following image shows the cost breakdown for Amazon S3 in the AWS Simple Monthly Calculator.
We use the AWS Simple Monthly calculator to estimate this. With the calculator, the total cost for Amazon S3 is $0.02.

### Amazon CloudFront Cost Breakdown

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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Distribution</td>
<td>50% US</td>
<td>Distribution of traffic across regions</td>
</tr>
<tr>
<td></td>
<td>25% EU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10% HK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15% JP</td>
<td></td>
</tr>
<tr>
<td>Request Type</td>
<td>HTTP</td>
<td>Type of requests that customers make to the cached locations</td>
</tr>
<tr>
<td>Data Transfer Out</td>
<td>30.0 GB/month</td>
<td>1.0 MB x 30.5 days x 1,000 hits/day</td>
</tr>
</tbody>
</table>

The following image shows the cost breakdown for Amazon CloudFront in the AWS Simple Monthly Calculator.
The total monthly cost is the sum of the data transfer out plus the requests costs for each of the regions.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transfer Out for US</td>
<td>Monthly Volume (GB) x Traffic Distribution (%) x Data Out Rate</td>
<td>30.00 x 0.50 x 0.12 = $1.80</td>
</tr>
<tr>
<td>Data Transfer Out for EU</td>
<td>Monthly Volume (GB) x Traffic Distribution (%) x Data Out Rate</td>
<td>30.00 x 0.25 x 0.12 = $0.90</td>
</tr>
<tr>
<td>Data Transfer Out for HK/Singapore</td>
<td>Monthly Volume (GB) x Traffic Distribution (%) x Data Out Rate</td>
<td>30.00 x 0.10 x 0.19 = $0.57</td>
</tr>
</tbody>
</table>
### Amazon EC2 Cost Breakdown

We use the AWS Simple Monthly calculator to obtain this estimate. According to the calculator, the total cost for CloudFront is $4.70.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transfer Out for JP</td>
<td>Monthly Volume (GB) x Traffic Distribution (%) x Data Out Rate</td>
<td>$0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requests for US</td>
<td>Request Rate x (Monthly Volume/Object Size (per 10,000 requests))</td>
<td>$0.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requests for EU</td>
<td>Request Rate x (Monthly Volume/Object Size (per 10,000 requests))</td>
<td>$0.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requests for HK/Singapore</td>
<td>Request Rate x (Monthly Volume/Object Size (per 10,000 requests))</td>
<td>$0.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Amazon EC2 Cost Breakdown

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uptime</strong></td>
<td>24 hrs/day</td>
<td>Assuming there is an average of 30.5 days in a month, the instance runs 732 hours/month</td>
</tr>
<tr>
<td><strong>Machine Characteristics</strong></td>
<td>t1.micro instance</td>
<td>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 <a href="http://aws.amazon.com/ec2/instance-types/">http://aws.amazon.com/ec2/instance-types/</a></td>
</tr>
<tr>
<td><strong>Additional Storage</strong></td>
<td>1 EBS Volume</td>
<td>The AMI is EBS-backed, the volume will have 30 GB provisioned storage, and 100 I/O requests made to the volume per second.</td>
</tr>
<tr>
<td></td>
<td>Storage: 30 GB/Month</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100 IOPS</td>
<td></td>
</tr>
<tr>
<td><strong>Data Transfer</strong></td>
<td>Data In: 0.005 GB/day</td>
<td>There are approximately 1,000 hits per day and each response is about 50 KB and each request is about 5 KB.</td>
</tr>
<tr>
<td></td>
<td>Data Out: 0.05 GB/day</td>
<td></td>
</tr>
<tr>
<td><strong>Instance Scale</strong></td>
<td>2</td>
<td>On average in a given day, there are 2 instances running.</td>
</tr>
<tr>
<td><strong>Elastic Load Balancing</strong></td>
<td>Hourly usage: 732 hrs/month</td>
<td>ELB is used 24 hrs/day, 7 days/week</td>
</tr>
<tr>
<td></td>
<td>Data processed: 1.525 GB/month</td>
<td>ELB processes a total of 0.055 GB/day (data in + data out)</td>
</tr>
<tr>
<td><strong>Detailed Monitoring</strong></td>
<td>$3.50 per instance per month</td>
<td>We have setup detailed monitoring for our Amazon EC2 instances.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Cost</td>
<td>Instance cost per hour	Number of instances x Uptime in hours</td>
<td>$0.02 x 2 x 732</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= $29.28</td>
</tr>
<tr>
<td>Additional Storage</td>
<td>Storage rate x Storage Amount (GB) + (I/O requests rate x seconds per month x Request rate (per 1M requests))</td>
<td>$0.10 x 30 x 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ (2.6M x $0.01)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= $29.35</td>
</tr>
<tr>
<td>Elastic Load Balancing</td>
<td>Hours used x Hourly rate + (Data processed (GB) x Process rate)</td>
<td>732 x $0.025</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 1.6775 x $0.008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= $18.31</td>
</tr>
</tbody>
</table>
We use the AWS Simple Monthly calculator to estimate this. With the calculator, the total cost for Amazon EC2 is $83.94.

## Amazon RDS Cost Breakdown

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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uptime</td>
<td>24 hrs/day</td>
<td>24 Assuming there is an average of 30.5 days in a month, the instance runs 732 hours/month</td>
</tr>
<tr>
<td>Database Characteristics</td>
<td>Small Amazon RDS instance</td>
<td>1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, Moderate I/O Capacity</td>
</tr>
<tr>
<td>Provisioned Storage</td>
<td>5 GB/month</td>
<td>Amazon provides 5 GB to 1 TB of associated storage capacity for your primary data set.</td>
</tr>
<tr>
<td>Requests</td>
<td>2M IOP/month</td>
<td>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.</td>
</tr>
<tr>
<td>Deployment Type</td>
<td>Multi-AZ</td>
<td>We will run our database instance across multiple Availability Zones.</td>
</tr>
<tr>
<td>Additional Backup Storage</td>
<td>none</td>
<td>We’ll use up to the provisioned amount which is 5 GB.</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>Data in: 0 GB, Data out: 0 GB</td>
<td>There is no data transfer from RDS to the Internet.</td>
</tr>
<tr>
<td>Database Instance Scale</td>
<td>1</td>
<td>We need one database instance.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Cost</td>
<td>Instance cost per hour x Number of instances x Uptime in hours</td>
<td>$0.153 x 1 x 732 = $112.00</td>
</tr>
<tr>
<td>Provisioned Storage</td>
<td>Storage rate x Storage Amount (GB)</td>
<td>$0.20 x 5 = $1.00</td>
</tr>
<tr>
<td>I/O Requests</td>
<td>I/O rate x Number of requests (millions)</td>
<td>$0.10 x ~2 = $0.22</td>
</tr>
</tbody>
</table>

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 S3, Amazon CloudFront, 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, Amazon S3, and Amazon RDS. For Amazon EC2, we have 0.05 GB per day which is approximately 1.525 GB per month. For Amazon S3, we have 0.1 GB per month, and for Amazon RDS we did not have any AWS Transfered out. Since up to 1 GB per month of data transferred out is free, we are left with a total of 0.625 GB per month.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Data Transfer</td>
<td>(Data in (GB) X Data In Rate) + (Data out (GB) X Data Out Rate)</td>
<td>$0.08</td>
</tr>
</tbody>
</table>

The following image shows an example of your monthly estimate.

The total cost of this web application is estimated at $165.37 per month including the free usage tier discount.
The following table lists related resources that you'll find useful as you work with AWS services.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Products and Services</td>
<td>A comprehensive list of products and services AWS offers.</td>
</tr>
<tr>
<td>Documentation</td>
<td>Official documentation for each AWS product including service introductions, service features, and API references, and other useful information.</td>
</tr>
<tr>
<td>AWS Architecture Center</td>
<td>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.</td>
</tr>
<tr>
<td>AWS Economics Center</td>
<td>Provides access to information, tools, and resources to compare the costs of Amazon Web Services with IT infrastructure alternatives.</td>
</tr>
<tr>
<td>AWS Cloud Computing Whitepapers</td>
<td>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.</td>
</tr>
<tr>
<td>Videos and Webinars</td>
<td>Previously recorded webinars and videos about products, architecture, security, and more.</td>
</tr>
<tr>
<td>Discussion Forums</td>
<td>A community-based forum for developers to discuss technical questions related to Amazon Web Services.</td>
</tr>
<tr>
<td>AWS Support Center</td>
<td>The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and AWS Premium Support. (subscription required).</td>
</tr>
<tr>
<td>AWS Premium Support Information</td>
<td>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.</td>
</tr>
<tr>
<td>Resource</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Form for questions related to your AWS account: Contact Us</td>
<td>This form is <em>only</em> for account questions. For technical questions, use the Discussion Forums.</td>
</tr>
<tr>
<td>Conditions of Use</td>
<td>Detailed information about the copyright and trademark usage at Amazon.com and other topics.</td>
</tr>
</tbody>
</table>
### Document History

This document history is associated with the 2011-09-30 release of Getting Started. This guide was last updated on May 18, 2012.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New content</td>
<td>Created new document</td>
<td>30 September 2011</td>
</tr>
<tr>
<td>Added new section</td>
<td>Added new section to talk about AWS Identity and Account Management</td>
<td>24 October 2011</td>
</tr>
<tr>
<td>Added new Windows AMI</td>
<td>Added new Windows AMI for Auto Scaling</td>
<td>26 October 2011</td>
</tr>
<tr>
<td>Changed Windows AMI</td>
<td>Changed Windows AMI to a t1.micro for Auto Scaling and Pricing example</td>
<td>09 November 2011</td>
</tr>
<tr>
<td>Added new section</td>
<td>Changed Windows AMI to a newer Windows AMI</td>
<td>25 November 2011</td>
</tr>
<tr>
<td>Added new section</td>
<td>Updated content for creating a CloudFront distribution.</td>
<td>18 May 2012</td>
</tr>
</tbody>
</table>