Getting Started with AWS

Deploying a Web Application

Getting Started with AWS: Deploying a Web Application

Table of Contents

Deploying a Web App Using Elastic Beanstalk

Using AWS, you can develop web apps quickly and then deploy them to a cloud environment that scales on demand. And with several AWS deployment services to choose from, you can create a deployment solution that gives you the right mix of automation and control.

In this tutorial, we'll assume that you're working on a new web app that isn't ready for production yet, but in the meantime you plan to deploy a small placeholder app that collects contact information from site visitors who sign up to hear more. The signup app will help you reach potential customers—people who might become early adopters or take part in a private beta test.

Here's a quick introduction to AWS Elastic Beanstalk and the other technologies we'll be using. (To dive right into the hands-on part of the tutorial, skip ahead to the next section.)

AWS Elastic Beanstalk

Elastic Beanstalk is a high-level deployment tool that helps you get an app from your desktop to the web in a matter of minutes. Elastic Beanstalk handles the details of your hosting environment—capacity provisioning, load balancing, scaling, and application health monitoring—so you don't have to.

Elastic Beanstalk supports apps developed in Java, PHP, .NET, Node.js, Python, and Ruby, as well as different container types for each language. A container defines the infrastructure and software stack to be used for a given environment. When you deploy your app, Elastic Beanstalk provisions one or more AWS resources, such as EC2 instances. The software stack that runs on your EC2 instances depends on the container type. For example, Elastic Beanstalk supports two container types for Node.js: a 32-bit Amazon Linux image and a 64-bit Amazon Linux image. Each runs a software stack tailored to hosting a Node.js app.

You can interact with Elastic Beanstalk by using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or eb, a high-level CLI designed specifically for Elastic Beanstalk. For this tutorial, we'll use the AWS Management Console.

The Signup App

In this tutorial, we'll deploy an example app that lets customers submit contact information and express interest in a preview of a hypothetical web app that you're developing.

The app is built on Node.js, a platform that uses server-side JavaScript to build network applications. Node.js consists of a library and a runtime. The runtime is provided by the V8 JavaScript Engine.

Node.js is designed around a non-blocking, event-driven I/O model, which makes it useful for creating highly scalable web servers. Our app employs two external Node modules: Express, a web application framework, and Jade, a Node.js template engine that can be used to create HTML documents.

AWS provides a Node.js SDK, which helps take the complexity out of coding by providing JavaScript objects for AWS. We've used the Node.js SDK to build our sample application. To learn more about the Node.js SDK, see AWS SDK for JavaScript in Node.js.

To make our app look good, we use Bootstrap, a mobile-first front-end framework that started as a Twitter project.

Amazon DynamoDB

We'll use Amazon DynamoDB, a NoSQL database service, to store the contact information that users submit. DynamoDB is a schema-less database, so you need to specify only a primary key attribute. We'll use an `email` field as a key for our app.

Amazon Simple Notification Service

We want to be notified when customers submit a form, so we'll use Amazon Simple Notification Service (Amazon SNS), a push messaging service that can deliver notifications over various protocols. For our app, we'll push notifications to an email address.

Setting Up

Before you start this tutorial, complete the following tasks.

Tasks

- Sign Up for AWS (p. 3)
- Download the Code for the App (p. 3)
- Create an IAM Role (p. 4)

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS and you can start using them immediately. You are charged only for the services that you use.

If you created your AWS account less than 12 months ago, you can get started with AWS for free. For more information, see AWS Free Tier.

If you have an AWS account already, skip to the next step. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

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

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Download the Code for the App

You can download the code for the app from the AWS Labs GitHub repository. Open eb-node-express-signup repo and then click **Download ZIP**.

You're going to make a few changes to the code as you complete the tutorial, so you need to unzip `eb-node-express-signup-master.zip`. The code for the app is stored in the `eb-node-express-signup-master` directory.

Create an IAM Role

Next you need to create an IAM role that grants your app permission to publish Amazon SNS notifications and put items into your DynamoDB table.

To create an IAM role with the required policy

1. Open the AWS Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam.
2. In the navigation pane, click **Policies**.
3. Click **Create Policy** to launch the Create Policy Wizard.
4. On the **Create Policy** page, next to **Create Your Own Policy**, click **Select**.
5. On the **Review Policy** page, enter a policy name that will help you identify the policy later. For example, **`ElasticBeanstalkStartupPolicy`**.
6. Open the `iam_policy.json` file from the `eb-node-express-signup-master` directory and copy its contents. Paste the contents into the **Policy Document** box.
7. (Optional) Click **Validate Policy**.
8. When you are ready, click **Create Policy**.
9. In the navigation pane, click **Roles**.
10. Click **Create New Role** to launch the Create Role Wizard.
11. On the **Set Role Name** page, in the **Role Name** box, enter a role name. Click **Next Step**.
12. On the **Select Role Type** page, next to **Amazon EC2**, click **Select** to allow EC2 instances to call AWS services on your behalf.
13. On the **Attach Policy** page, next to **Filter**, click **Policy Type**, and then click **Customer Managed Policies**.
14. Next to the name of the policy that you created earlier, select the check box. Click **Next Step**.
15. On the **Review** page, verify the information that you entered, and then click **Create Role**.

Step 1: Create a DynamoDB Table

Our signup app uses a DynamoDB table to store the contact information that users submit.

To create a DynamoDB table

1. Open the DynamoDB console.
2. In the menu bar, select the region in which to deploy your app. For this tutorial, select **US West (Oregon)**.
3. Add the region that you selected to the configuration file for the app as follows:

   a. Open the `app_config.json` file in a text editor.

   b. The value for "AWS_REGION" is an empty string to start.

   ```
   "AWS_REGION": "",
   ```

   Insert the API name for the US West (Oregon) region as follows.

   ```
   "AWS_REGION": "us-west-2",
   ```

   c. Save your edits to the file. You can leave the file open, as we'll make an additional change to it shortly.

4. Back in Dynamo DB, click **Create Table**.
5. In the **Table Name** box, enter a name for the table, such as *`my-signup-table`*.
6. For the **Primary Key Type**, select **Hash**.
7. For the **Hash Attribute Name**, select **String**. In the corresponding box, enter **`email`**, and then click **Continue**.
8. We don't need additional indexes, so on the **Add Indexes** page, click **Continue**.
9. For our sample app, we can use the minimum provisioned throughput capacity for our table. On the **Provisioned Throughput Capacity** page, verify that the **Help me calculate how much throughput capacity I need to provision** check box is clear and that the read and write capacity units are both set to "1". Click **Continue**.
10. Our sample app doesn't need any throughput alarms, so you can clear the **Use Basic Alarms** check box. Click **Continue**.
11. On the Review page, click **Create**. When the creation process is complete, the table **Status** is `ACTIVE`.

12. Add the table name to the configuration file for the app as follows:

    a. Open the `app_config.json` file in a text editor, if it's not already open.

    b. The value for "STARTUP_SIGNUP_TABLE" is an empty string to start.

    ```
    "STARTUP_SIGNUP_TABLE": "",
    ```

    Insert the name of your new database as follows.

    ```
    "STARTUP_SIGNUP_TABLE": "my-signup-table",
    ```

    c. Save your edits to the file. You can leave the file open, because we'll make one more change to it in the next step.

Step 2: Create an SNS Topic

Our signup app notifies you each time a user signs up. When the data from the signup form is written to the DynamoDB table, the app sends you an SNS notification. First, you need to create an SNS topic, which is a stream for notifications, and then you need to create a subscription that tells SNS where and how to send the notifications.

To set up Amazon SNS notifications

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. You must use the same region for the SNS topic that you did for the DynamoDB table. Ensure that **US West (Oregon)** is still selected, then click **Create New Topic**.
3. In the **Create New Topic** dialog box, enter **`my-sns-topic`**. as the topic name. You can leave the display name blank. Click **Create Topic**.
4. Add the unique identifier for the SNS topic to the configuration file for the app as follows:

   a. On the topic details page, copy the string from **Topic ARN**. Note that each Amazon Resource Name (ARN) has the following syntax:

   ```
   arn:aws:[service]:[region]:[account]:resourceType/resourcePath
   ```

   b. Open the `app_config.json` file in a text editor, if it's not already open.

   c. The value for "NEW_SIGNUP_TOPIC" is an empty string to start.

   ```
   "NEW_SIGNUP_TOPIC": "",
   ```

   Insert the name of your new SNS topic as follows.

   ```
   "NEW_SIGNUP_TOPIC": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",
   ```

   d. Your final configuration file should look similar to the following. Save your edits to the file and close the file.

   ```
   {
     "AWS_REGION": "us-west-2",
     "STARTUP_SIGNUP_TABLE": "my-signup-table",
   ```

```
  "NEW_SIGNUP_TOPIC": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
}
```

5. Click **Create Subscription**.
6. In the **Create Subscription** dialog box, select **Email** as the protocol. In the **Endpoint** box, enter an email address to receive the notification. (For this tutorial, enter your own email address, as you must respond to a confirmation message.) Click **Subscribe**. In the confirmation dialog box, click **Close**.
7. Go to your email app. You'll receive a message titled "AWS Notification — Subscription Confirmation." Open the message and click the link to confirm your subscription. To verify that your subscription has been confirmed, refresh the page in the SNS console.

Step 3: Deploy the App Using AWS Elastic Beanstalk

You can easily deploy the signup app using Elastic Beanstalk. You upload an app version (for example, a `.zip` file) to Elastic Beanstalk, and then provide some information about the app. Elastic Beanstalk launches an environment and provisions the AWS resources needed to run your code (such as an Amazon EC2 instance). After your environment is launched, you can manage your environment and deploy new app versions.

Tasks

Prepare the Source Bundle

Elastic Beanstalk requires that your app be bundled as a single `.zip` or `.war` file. A bundle can't include a top-level folder, so you must compress the individual app files, rather than compressing the directory that contains them.

To prepare the source bundle

1. Open the app folder (`eb-node-express-signup-master`) in a tool like Windows Explorer or Mac OS X Finder.
2. Select all the items in the folder, including the subfolders. Do not select the top-level folder.
3. Right-click the selected items and select the option to compress them, such as **Send to > Compressed (zipped) Folder** (Windows) or **Compress Items** (Finder).

For more information about compressing files using a variety of tools, see Creating an Application Source Bundle in the *AWS Elastic Beanstalk Developer Guide*.

Create an App

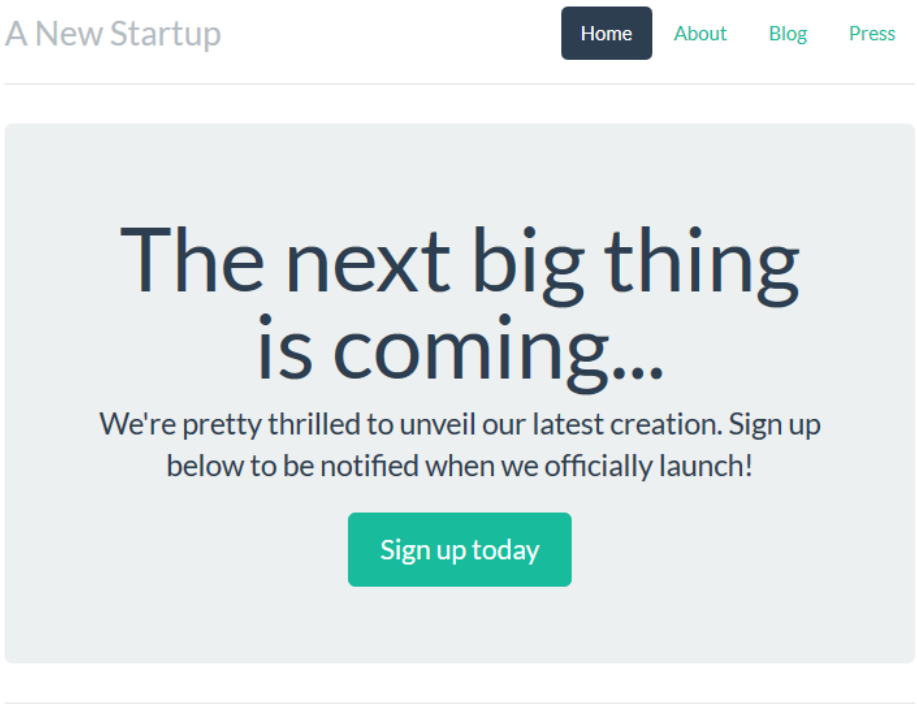
Create an Elastic Beanstalk application and deploy the app version to a new environment.

To create an application and environment

1. Open the Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.
2. Click **Create New Application**.
3. On the **Application Information** page, enter a name for the app. For this tutorial, let's call the app `my-startup-app`. Click **Next**.
4. On the **New Environment** page, click **Create web server**.
5. On the **Environment Type** page, set **Predefined configuration** to **Node.js**, and then click **Next**.
6. On the **Application Version** page, select **Upload your own**, click **Browse**, and then select the source bundle that you created. Click **Next**.
7. On the **Environment Information** page, enter an environment name and a unique environment URL. Click **Check availability** to ensure that your URL is available. Click **Next**.
8. On the **Additional Resources** page, click **Next**.
9. On the **Configuration Details** page, use the default values for all settings by clicking **Next**.
10. On the **Environment Tags** page, click **Next**.
11. On the **Permissions** page, set **Instance profile** to the role that you created in Create an IAM Role (p. 4). Don't change the value for **Service role**.
12. Click **Next**. Accept the prompt to create the default service role and assign it to your environment.

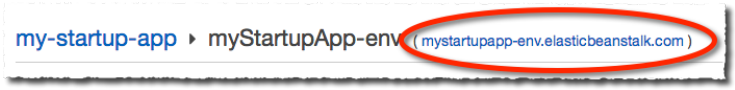
    Note
    Creating roles requires additional permissions. See Elastic Beanstalk User Policy in the Elastic Beanstalk Developer Guide for details.

13. On the **Review** page, verify your settings, and then click **Launch**. On the AWS Elastic Beanstalk dashboard, you can watch in real time as Elastic Beanstalk creates an environment and deploys the app. This process takes a few minutes.

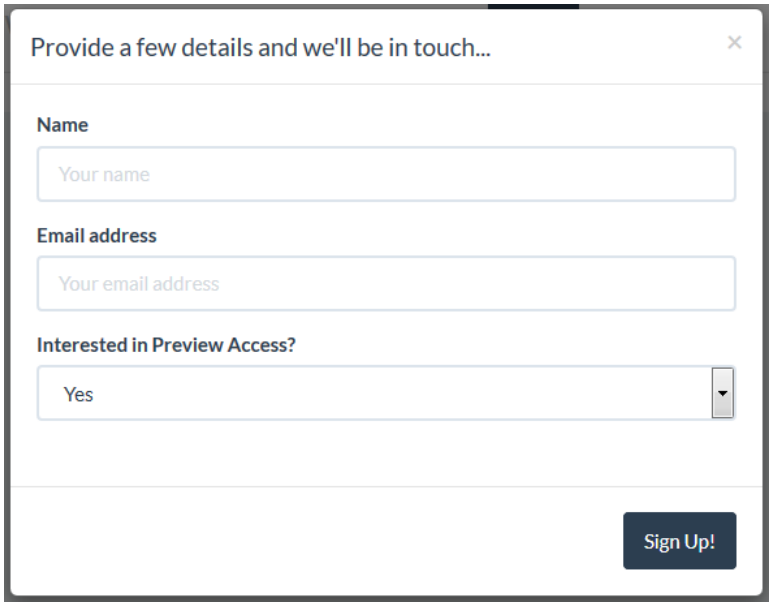


Test the App

When the deployment is finished and the environment health is listed as "Green", click the URL of the app.



You can test the signup app by filling out the form and verifying that you receive the notification.



Troubleshoot Deployment Issues

If you followed all the steps, clicked the URL, and got no app, there's a deployment problem. With our sample app, which runs on an nginx server, a deployment problem is likely to result in a "502 Bad Gateway" message. The "502" message is not very informative. To troubleshoot a deployment issue, you may need to use the logs that are provided by Elastic Beanstalk.

For example, let's say that, in the process of updating `app_config.json`, you accidentally leave off a quotation mark. Then, when you finish the deployment, you see the 502 error. How do you find and fix the problem?

Of course, you'd try to catch such an error in development. But if an error does get through to production, or you just want to update your app, Elastic Beanstalk makes it fast and easy to redeploy.

To troubleshooting a deployment issue

1. In the Elastic Beanstalk console, in the navigation pane for your environment, click **Logs**.
2. At the Logs page, click **Snapshot Logs**. Wait for your environment to update, and then click **View log file**. In the log file, you can see just what happened on the server side during deployment and runtime. There's a lot of material to sort through, and we're not going to cover the different sections of the log in this tutorial. But if you did indeed leave out a quotation mark in the config file and you scrolled through the log to `/var/log/nodejs/nodejs.log`, you'd find an error similar to this:

```
SyntaxError: Unexpected token u
    at Object.parse (native)
    at Object.<anonymous> (/var/app/current/server.js:23:15)
    at Module._compile (module.js:449:26)
    at Object.Module._extensions..js (module.js:467:10)
    at Module.load (module.js:356:32)
    at Function.Module._load (module.js:312:12)
    at Module.runMain (module.js:492:10)
   at process.startup.processNextTick.process._tickCallback (node.js:245:9)

undefined:2
  "AWS_REGION": us-west-2",^
```

In this case, the "Unexpected token u" message appears because the parser expected a quotation mark instead of a "u" in the string `us-west-2"`. Now that we've found the problem, we can fix it.

3. If you were actually troubleshooting this issue, you'd go back to the app code in your local environment and fix the missing quotation mark. Then you'd create a new `.zip` file to upload.
4. To redeploy the app, go to the Elastic Beanstalk Dashboard, click **Upload and Deploy**, and choose your updated `.zip` file.
5. Change the version label to something new. For example, if your first deployment was labeled "Archive", you can label this second deployment "Archive-2."
6. Click **Deploy**. Elastic Beanstalk will update the environment.
7. When the environment is updated and listed as "Green", use the app URL to test your app.

Step 4: Change the App Configuration (Optional)

Our web app uses an environment variable, `theme`, to control the CSS that is applied. The `server.js` file contains the following statement to make this environment variable available to the app:

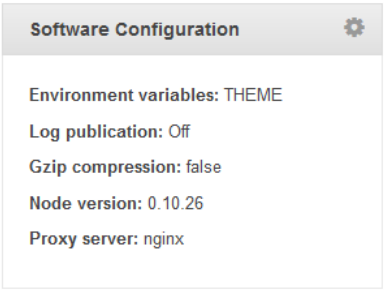
```
app.locals.theme = process.env.THEME;
```

Changing the setting of this environment variable changes the look of the app. For more information, see `public/static/bootstrap/css/theme` and `views/layout.jade` in the code for the app.

The easiest way to update this environment variable is to use Elastic Beanstalk to update the configuration for your environment.

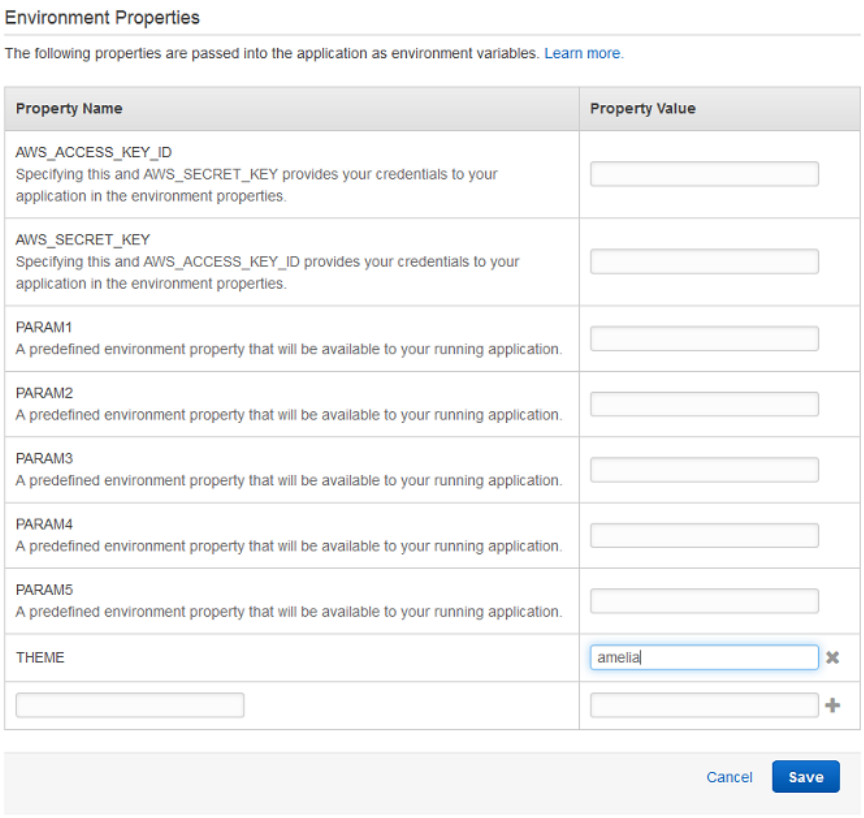
To change the app theme using Elastic Beanstalk

1. Open the Elastic Beanstalk console.
2. In the navigation pane for your environment, click **Configuration** and open **Software Configuration**.

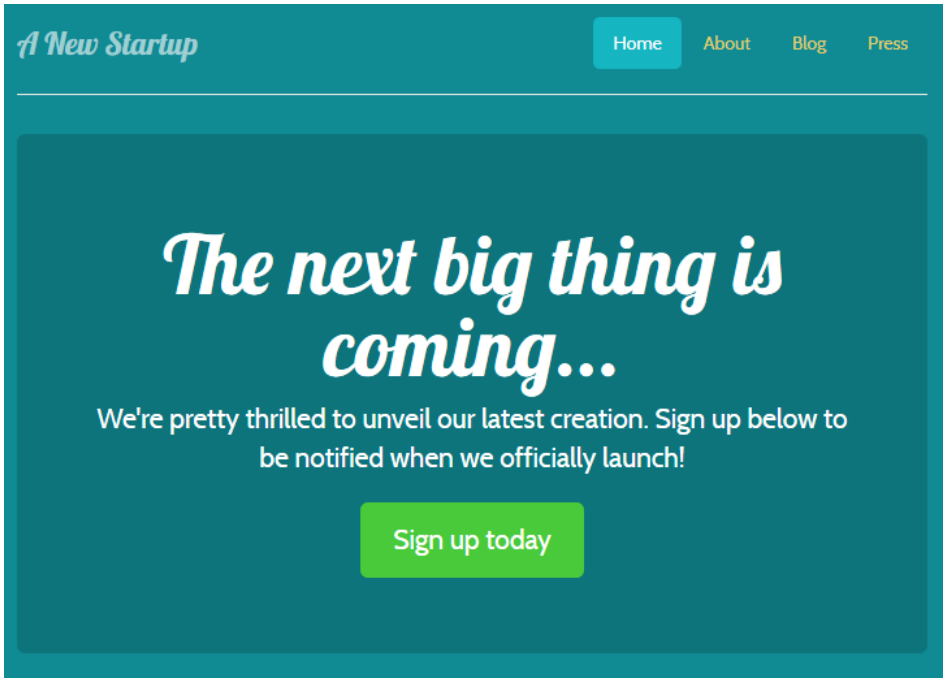


3. Under **Environment Properties**, locate the **`THEME`** environment variable. Change the value from the default (**`flatly`**) to one of the following values, and then click **Save**.

   - `amelia`
   - `default`
   - `slate`
   - `united`



4. After Elastic Beanstalk finishes updating the environment, click the URL for the app. The app has a new look.



Step 5: Clean Up

To prevent your account from accruing additional charges, you should clean up the apps and environments that you created for this tutorial using Elastic Beanstalk.

To delete AWS Elastic Beanstalk resources

1. Open the Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk.
2. Delete all the application versions as follows:
   a. Click the name of your app, and then click **Application Versions** in the dropdown menu.
   b. Select all application versions that you created for this tutorial, and then click **Delete**.
   c. When prompted for confirmation, click **Delete**.
   d. Click **Done**.
3. Terminate the environment as follows:
   a. Click **Environments** and then select your environment.
   b. Click **Actions** and then click **Terminate Environment**.
   c. When prompted for confirmation, click **Terminate**.
4. Click **Actions** and then click **Delete Application**. When prompted for confirmation, click **Delete**.

To delete the Amazon DynamoDB table

1. Open the DynamoDB console at https://console.aws.amazon.com/dynamodb/.
2. Select the table that you created in Step 1.
3. Click **Delete Table**.
4. Check **Delete this table** and then click **Delete**.

To delete the Amazon SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. Click **Topics**.
3. Select the topic that you created in Step 2.

4. Click **Actions**, and then click **Delete Topics**.
5. Click **Delete**.

More AWS Deployment Options

This tutorial focuses on the basic features of AWS Elastic Beanstalk. However, Elastic Beanstalk has many more features, and it's only one of the deployment solutions available with AWS. Alternatively, you can use AWS OpsWorks to deploy and manage applications, and if you want a do-it-yourself experience you can use Amazon Elastic Compute Cloud, AWS CloudFormation, Amazon CloudWatch, and Auto Scaling to build your own deployment framework. To learn more about choosing the right deployment solution, see Application Management for AWS.

To learn more about Elastic Beanstalk and other AWS deployment options, check out the following resources.

More about Elastic Beanstalk

- For complete documentation about managing apps with Elastic Beanstalk, see the AWS Elastic Beanstalk Developer Guide.
- To learn about the Elastic Beanstalk CLI, see Eb Command Line Interface.
- To learn how to deploy a Node.js app with eb, see Deploying AWS Elastic Beanstalk Applications in Node.js Using Eb and Git.
- To learn more about the components and architecture of an Elastic Beanstalk app, see How Does AWS Elastic Beanstalk Work?.
- By including a configuration file with your source bundle, you can customize your environment at the same time that you deploy your application. To learn more, see Customizing and Configuring AWS Elastic Beanstalk Environments.

More AWS Solutions

- To learn more about AWS OpsWorks, see AWS OpsWorks User Guide.
- To learn more about the individual services you'd typically use for a do-it-yourself deployment, see Amazon EC2 Getting Started Guide, AWS CloudFormation User Guide, Amazon CloudWatch Getting Started Guide, and Auto Scaling Getting Started Guide.
- The AWS Toolkit for Visual Studio includes a deployment tool. To learn more, see Standalone Deployment Tool.
- The AWS Toolkit for Eclipse also provides deployment options. To learn more, see Getting Started with the AWS Toolkit for Eclipse.

Related Resources

The following table lists some of the AWS resources that you'll find useful as you work with AWS.

Resource	Description
AWS Products & Services	Information about the products and services that AWS offers.
AWS Documentation	Official documentation for each AWS product, including service introductions, service features, and API reference.
AWS Discussion Forums	Community-based forums for discussing technical questions about Amazon Web Services.
Contact Us	A central contact point for account questions such as billing, events, and abuse. For technical questions, use the forums.
AWS Support Center	The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
AWS Support	The home page for AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
AWS Architecture Center	Provides the necessary guidance and best practices to build highly scalable and reliable applications in the AWS cloud. These resources help you understand the AWS platform, its services and features. They also provide architectural guidance for design and implementation of systems that run on the AWS infrastructure.
AWS Security Center	Provides information about security features and resources.
AWS Economics Center	Provides access to information, tools, and resources to compare the costs of Amazon Web Services with IT infrastructure alternatives.
AWS Technical Whitepapers	Provides technical whitepapers that cover topics such as architecture, security, and economics. These whitepapers have been written by the Amazon team, customers, and solution providers.

Resource	Description
AWS Blogs	Provides blog posts that cover new services and updates to existing services.
AWS Podcast	Provides podcasts that cover new services, existing services, and tips.