
Getting Started with AWS

Deploying a Web Application



Getting Started with AWS: Deploying a Web Application

Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Deploying a Web App	1
AWS Elastic Beanstalk	1
The Signup App	1
Amazon DynamoDB	2
Amazon Simple Notification Service	2
Prerequisites	3
Sign Up for AWS	3
Download the Code for the App	3
Create an IAM Policy and Role	3
Step 1: Create a DynamoDB Table	5
Step 2: Create an SNS Topic	6
Step 3: Deploy the App	7
Create an Elastic Beanstalk Environment	7
Create the Source Bundle	7
Deploy the Signup Application	8
Test the App	8
Troubleshoot Deployment Issues	9
Step 4: Change the App Configuration	11
Step 5: Clean Up	13
Delete AWS Elastic Beanstalk Resources	13
Delete the Amazon DynamoDB Table	13
Delete the Amazon SNS Resources	13
More AWS Deployment Options	14

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 \(p. 3\)](#).)

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, Go, PHP, .NET, Node.js, Python, and Ruby, with multiple configurations for each platform. A platform configuration defines the infrastructure and software stack to be used for a given environment. When you deploy your app, Elastic Beanstalk provisions a set of AWS resources that can include Amazon EC2 instances, alarms, a load balancer, security groups, and more. The software stack that runs your application depends on the platform configuration type. For example, Elastic Beanstalk supports 3 configurations for Python: Python 3.4, Python 2.7, and Python 2.6.

You can interact with Elastic Beanstalk by using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the Elastic Beanstalk CLI, 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.

Prerequisites

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 Policy and Role \(p. 3\)](#)

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 <https://aws.amazon.com/>, and then choose **Create an AWS Account**.

Note

This might be unavailable in your browser if you previously signed into the AWS Management Console. In that case, choose **Sign in to a different account**, and then choose **Create a new AWS account**.

2. Follow the online 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 choose **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 Policy and 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. You will apply the role to the EC2 instances that run your application when you create an AWS Elastic Beanstalk environment.

To create an IAM policy

1. Open the AWS Identity and Access Management (IAM) console at <https://console.aws.amazon.com/iam>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. Choose **JSON**.
5. 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.
6. Choose **Review policy**.
7. For **Name**, enter **gsg-signup-policy**.
8. Choose **Create policy**.

Create an IAM role and attach the policy to it.

To create an IAM role

1. In the navigation pane, choose **Roles**.
2. Choose **Create role**.
3. Under **Choose the service that will use this role**, choose **EC2** to allow EC2 instances to use the role to call AWS services on your behalf. Choose **Next**.
4. On the **Attach Policy** page, attach the following policies.
 - **gsg-signup-policy** – The policy that you created earlier.
 - **AWS Elastic Beanstalk WebTier** – Elastic Beanstalk provided role that allows the instances in your environment to upload logs to Amazon S3.

To locate policies quickly, type part of the policy name in the filter box. Select both policies and then choose **Next Step**.

5. For **Role name**, enter **gsg-signup-role**.
6. Choose **Create role**.

For more information on permissions, see <http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts-roles.html> in the *AWS Elastic Beanstalk Developer Guide*.

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 at <https://console.aws.amazon.com/dynamodb/home>.
2. In the menu bar, ensure that the region is set to **US West (Oregon)**.
3. Choose **Create table**.
4. For **Table name**, type **gsg-signup-table**.
5. For the **Primary key**, type **email**. Choose **Create**.

Add the table name and region to the configuration file for the app as follows:

To update the application configuration

1. In the `eb-node-express-signup-master` folder that you extracted from the sample archive, open the `app_config.json` file.
2. Change the value for `STARTUP_SIGNUP_TABLE` to **gsg-signup-table**:

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

3. Change the value of `AWS_REGION` to `us-west-2` for US West (Oregon) Region:

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

4. 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/v2/home>.
2. Choose **Create topic**.
3. For **Topic name**, type **gsg-signup-notifications**. Choose **Create topic**.
4. Choose **Create subscription**.
5. For **Protocol**, choose **Email**
6. For **Endpoint**, enter your email address.
7. Choose **Create Subscription**.
8. To confirm the subscription, Amazon SNS sends you an email named **AWS Notification — Subscription Confirmation**. Open the link in the email to confirm your subscription.

Add the unique identifier for the SNS topic to the configuration file for the app as follows.

To update the application configuration

1. 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:accountid:resourceType/resourcePath
```

2. Open the `app_config.json` file in a text editor, if it's not already open.
3. Insert the ARN of your SNS topic.

```
"NEW_SIGNUP_TOPIC": "arn:aws:sns:us-west-2:123456789012:gsg-signup-notifications",
```

4. 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": "gsg-signup-table",
  "NEW_SIGNUP_TOPIC": "arn:aws:sns:us-west-2:123456789012:gsg-signup-notifications"
}
```

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.

Tasks

- [Create an Elastic Beanstalk Environment \(p. 7\)](#)
- [Create the Source Bundle \(p. 7\)](#)
- [Deploy the Signup Application \(p. 8\)](#)
- [Test the App \(p. 8\)](#)
- [Troubleshoot Deployment Issues \(p. 9\)](#)

Create an Elastic Beanstalk Environment

Create an environment to run the sample application.

To launch an environment

1. Open the Elastic Beanstalk console using this preconfigured link:
console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=gsg-signup&environmentType=LoadBalanced
2. For **Platform**, choose **Node.js**.
3. For **Application code**, choose **Sample application**.
4. Choose **Review and launch**.
5. On the **Security** card, choose **Modify**.
6. For **IAM instance profile**, choose the role that you created earlier (**gsg-signup-role**).
7. Choose **Save**.
8. Choose **Create app**.

In just a few minutes, Elastic Beanstalk provisions the networking, storage, compute, and monitoring infrastructure required to run a scalable web application in AWS. Once the site is up and running, you can deploy a new version of your application code to the environment at any time.

Create the Source Bundle

Elastic Beanstalk requires that your app be bundled as a single ZIP or ZIP 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.

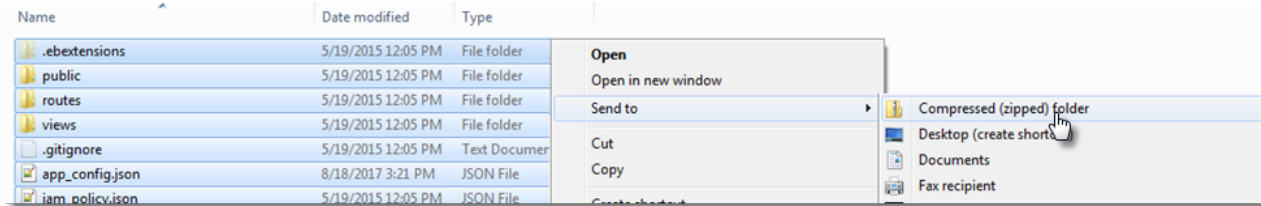
To create the source bundle in a file browser

1. Open the application folder `eb-node-express-signup-master`.
2. Select all the items in the folder, including the subfolders. Do not select the top-level folder.

Important

The signup application includes a hidden folder named `.ebextensions` that contains a configuration file that configures Node.js platform settings. You may need to configure your file browser to show hidden files to see this folder.

3. Right-choose the selected items and select the option to compress them, such as **Send to > Compressed (zipped) Folder** (Windows) or **Compress Items** (macOS).



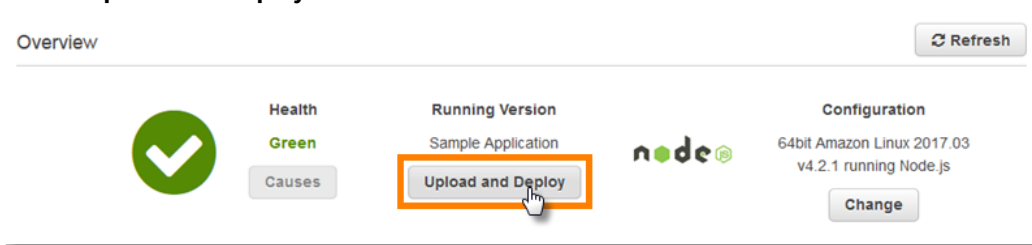
For more information about compressing files using a variety of tools, see [Creating an Application Source Bundle](#) in the *AWS Elastic Beanstalk Developer Guide*.

Deploy the Signup Application

Deploy the source bundle to the Elastic Beanstalk environment.

To deploy the application source bundle

1. Return to the environment that you created in the Elastic Beanstalk console.
2. Choose **Upload and Deploy**.



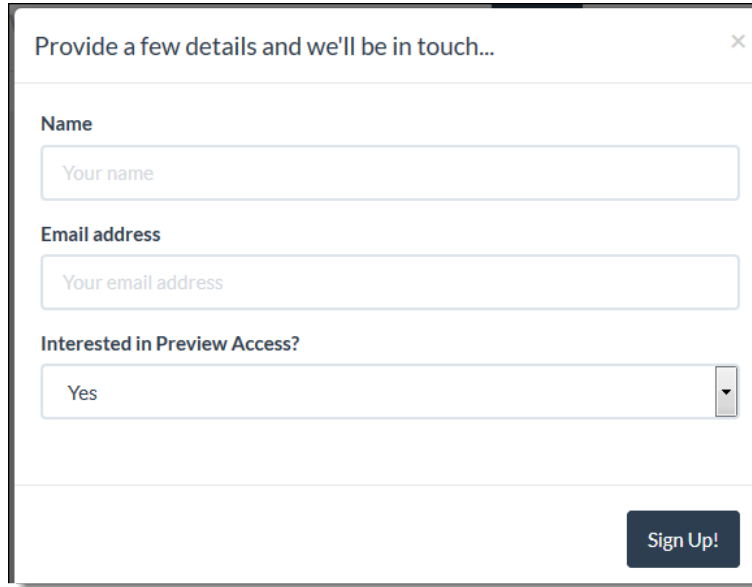
3. Choose **Browse** and upload the source bundle that you created in the previous section.
4. Choose **Deploy**.

Test the App

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

[All Applications](#) > [gsg-signup](#) > [GsgSignup-env](#) (Environment ID: e-x2faszmsyv, URL: [GsgSignup-env.us-west-2.elasticbeanstalk.com](#))

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



Provide a few details and we'll be in touch...

Name

Your name

Email address

Your email address

Interested in Preview Access?

Yes

Sign Up!

Troubleshoot Deployment Issues

If you followed all the steps, opened 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, choose **Logs**.
2. At the Logs page, choose **Snapshot Logs**. Wait for your environment to update, and then choose **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, choose **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. Choose **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

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:

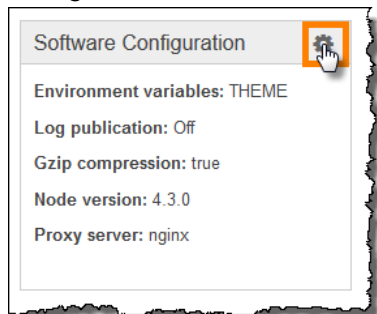
```
app.locals.theme = process.env.THEME;
```

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, choose **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 choose **Save**.
 - `amelia`
 - `default`
 - `slate`
 - `united`

Static Files

To improve performance, you can configure Apache or Nginx to serve static files from a set of directories inside your web application. [Learn more.](#)

Virtual Path (Example: /assets)	Directory (Example: /static/assets)
/static/	/public/static/ ✕
<input type="text"/>	<input type="text"/> +

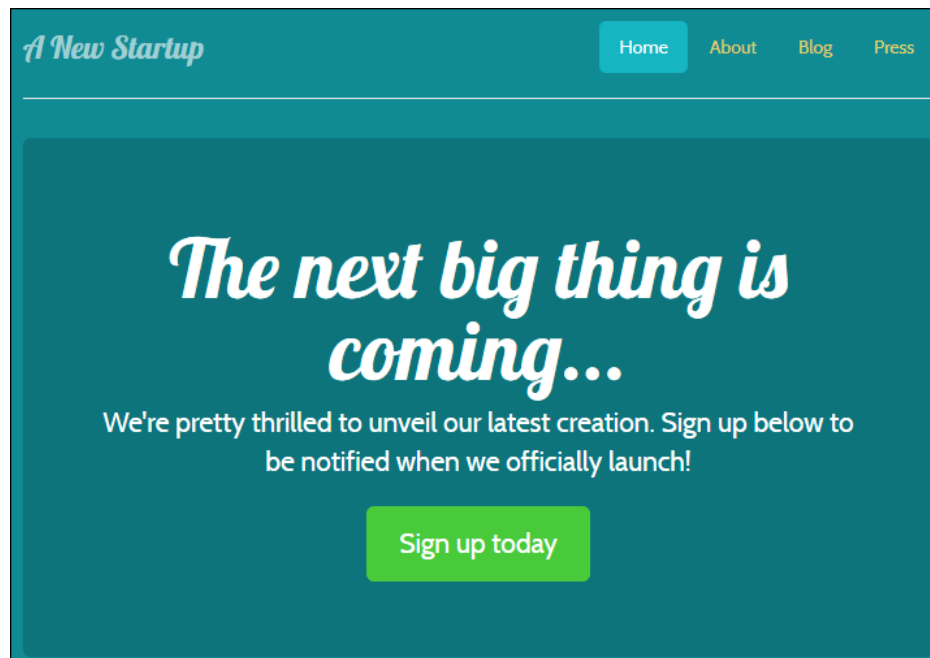
Environment Properties

The following properties are passed into the application as environment variables. [Learn more.](#)

Property Name	Property Value
THEME	amelia ✕
<input type="text"/>	<input type="text"/> +

Cancel Apply

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



Step 5: Clean Up

To prevent your account from accruing additional charges, delete the Elastic Beanstalk, DynamoDB and Amazon SNS resources that you created.

Topics

- [Delete AWS Elastic Beanstalk Resources \(p. 13\)](#)
- [Delete the Amazon DynamoDB Table \(p. 13\)](#)
- [Delete the Amazon SNS Resources \(p. 13\)](#)

Delete AWS Elastic Beanstalk Resources

To delete the AWS Elastic Beanstalk resources, perform the following procedure:

To delete AWS Elastic Beanstalk resources

1. Open the Elastic Beanstalk console at <https://console.aws.amazon.com/elasticbeanstalk>.
2. Choose **Actions** next to the application name (**gsg-signup**) and then choose **Delete application**.
3. Choose **Delete** to confirm.

Delete the Amazon DynamoDB Table

To delete the Amazon DynamoDB table, perform the following procedure:

To delete the Amazon DynamoDB table

1. Open the DynamoDB console at <https://console.aws.amazon.com/dynamodb/>.
2. Choose **Tables**
3. Choose your table (**gsg-signup-table**).
4. Choose **Actions** and then choose **Delete table**.
5. Choose **Delete** to confirm.

Delete the Amazon SNS Resources

To delete the Amazon SNS resources, perform the following procedures:

To delete the Amazon SNS subscription

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v2/home>.
2. In the left navigation pane, choose **Topics**.
3. Select the topic (**gsg-signup-notifications**), choose **Actions**, and choose **Delete Topics**.
4. Choose **Delete** to confirm.

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 Amazon EC2 Auto Scaling to build your own deployment framework.

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 [The Elastic Beanstalk Command Line Interface](#).
- To learn how to deploy a Node.js app with the EB CLI, see [Deploying an Express Application to Elastic Beanstalk](#).
- 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 [Advanced Environment Customization with Configuration Files \(.ebextensions\)](#).

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 User Guide](#), and [Amazon EC2 Auto Scaling User 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](#).