AWS for DevOps

Getting Started Guide

AWS for DevOps: Getting Started Guide

Table of Contents

Introducing the AWS for DevOps Walkthrough

Welcome to the AWS for DevOps Getting Started Guide. This guide contains information about AWS services for DevOps and a walkthrough you can use to experiment with those services.

What Is DevOps?

DevOps is a combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity, evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market. Good DevOps practices encourage software development engineers and operations professionals to work better together. This results in closer collaboration and communication, leading to shorter time-to-market, better code quality and maintenance, and more reliable releases. For more information, see What Is DevOps? on DevOps and AWS.

AWS Services for DevOps

This walkthrough shows you how to use the following AWS services:

- **AWS CodeCommit**, a fully managed source control service that makes it easy for you to host secure and highly scalable private Git repositories. For more information, see AWS CodeCommit.
- **AWS CodeDeploy**, which automates code deployments to any instance, including Amazon Elastic Compute Cloud (Amazon EC2) instances and on-premises servers. For more information, see AWS CodeDeploy.
- **AWS CodePipeline**, a continuous delivery service for fast and reliable application updates. For more information, see AWS CodePipeline.

This walkthrough also features these AWS services:

- **AWS Elastic Beanstalk**, an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on servers such as Apache, Nginx, Passenger, and IIS. If you want to experiment with web applications and services in this walkthrough, you can use Elastic Beanstalk to more easily and confidently deploy web applications and services to these types of servers. For more information, see AWS Elastic Beanstalk.

- **AWS OpsWorks**, a configuration management service that helps you configure and operate applications of all shapes and sizes using Chef. If you want to experiment with Chef in this walkthrough, you can use AWS OpsWorks to more easily and confidently deploy Chef cookbooks and applications. For more information, see AWS OpsWorks.
- **AWS CloudFormation**, which is an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. This walkthrough uses AWS CloudFormation to help you complete the setup steps more quickly and more easily and reliably clean up the resources when you're done. For more information, see AWS CloudFormation.
- **AWS Identity and Access Management (IAM)**, which enables you to control access to AWS services and resources. This walkthrough uses IAM to control access to AWS resources and the actions that you can perform with them. For more information, see AWS Identity and Access Management (IAM).

Understanding the Walkthrough

In this walkthrough, you will:

1. Use AWS CloudFormation to give users access to the required AWS services and the corresponding AWS resources and actions.
2. Create a source code repository in AWS CodeCommit and then use AWS CloudFormation to launch an Amazon EC2 instance that connects to the repository.
3. Download the source code you will deploy and then push it into the repository.
4. Use AWS CloudFormation to create the deployment target (an Amazon EC2 instance) and related AWS resources that are compatible with AWS CodeDeploy, Elastic Beanstalk, or AWS OpsWorks.
5. Use AWS CloudFormation to create and run a pipeline in AWS CodePipeline to automate continuous delivery of the repository's source code to the deployment target.
6. Verify the deployment's results on the deployment target.
7. Make a change to the source code and then push it into the repository, triggering an automatic redeployment to the deployment target.
8. Verify the deployed change on the deployment target.
9. Use AWS CloudFormation to clean up the resources you created for this walkthrough.

Important

Many of the steps in this walkthrough use AWS CloudFormation templates to create required AWS resources, which may result in charges to your AWS account. To see the estimated costs, in the AWS CloudFormation console's **Create stack** wizard, on the **Review** page, choose the **Cost** link.

After you've completed this walkthrough, you will use AWS CloudFormation to delete these AWS resources to avoid possible ongoing charges to your AWS account.

For more information about pricing related to the services used in this walkthrough, see the following:

- AWS CodeCommit Pricing
- AWS CodeDeploy Pricing
- AWS CodePipeline Pricing
- AWS Elastic Beanstalk Pricing
- AWS OpsWorks Pricing
- AWS CloudFormation Pricing
- AWS Identity and Access Management Pricing
- Amazon EC2 Pricing

Next Steps

To start the walkthrough, go to Step 1: Service Access Setup (p. 4).

Step 1: Set Up to Access Participating Services

In this step, you will set up user access to AWS services used in this walkthrough. Specifically, you will:

1. Create an AWS account and, optionally, an administrative IAM user in the account.
2. Use the root account or an administrative user in the account to create an IAM group and IAM user for use with this walkthrough.
3. Attach AWS service access permissions to the new group.
4. Add the new user to the new group.
5. Sign in to the AWS Management Console with the new user's credentials.

Start with Step 1.1: Create an Account (p. 4).

Topics

Step 1.1: Create an AWS Account

In this step, you will create an AWS account that you will use to complete the tasks in step 1.2. If you already have an account you want to use, sign in to the AWS Management Console with the account's root credentials. After you sign in to the console, go to Step 1.2: Create IAM Resources (p. 5).

> **Important**
> As a security best practice, you should use an administrative IAM user in the account instead of an AWS root account to complete the tasks in step 1.2. For more information, see Creating Your First IAM User and Administrative Group in the IAM User Guide. If you want to use an administrative user instead of an AWS root account, sign in to the AWS Management Console with the administrative user's credentials. (For more information, see User Sign-In Page in the IAM User Guide.) After you sign in to the console, go to Step 1.2: Create IAM Resources (p. 5).

To create an AWS account

1. Open http://aws.amazon.com.

2. Choose **Sign In to the Console**.
3. Complete the instructions to create an AWS account.
4. After your account has been created, sign in to the AWS Management Console with the account's root credentials. (For more information, see Root Account Sign-in Page in the IAM User Guide.)
5. Go to Step 1.2: Create IAM Resources (p. 5).

Step 1.2: Create IAM Resources

In this step, you will complete the following tasks:

- Create an IAM group and an IAM user specifically for use with this walkthrough.
- Attach participating AWS service access permissions to the new group.
- Add the new user to the new group.
- Sign in to the AWS Management Console with the new user's credentials.

Why should you create an IAM user? As an AWS security best practice, we do not recommend that you complete the tasks in this walkthrough while signed in as either an AWS root account or an administrative IAM user in the account. Instead, you should complete those tasks while signed in as an individual user associated with an IAM group.

Why should you create an IAM group? Groups are convenient: you can manage access permissions for groups, and you associate IAM users with groups instead of managing access permissions for individual users.

The following procedure uses an AWS CloudFormation template to complete this step's tasks more quickly. (To view the contents of the AWS CloudFormation template, see IAMSetup.template.) To learn how to accomplish these tasks in other ways, such as with the AWS Management Console or the AWS Command Line Interface (AWS CLI), see the following topics in the IAM User Guide:

- Creating IAM Groups
- Working with Policies
- Creating an IAM User in Your AWS Account
- Adding and Removing Users in an IAM Group

To create the IAM resources

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. In the AWS region selector, choose **US East (N. Virginia)**. (This walkthrough uses AWS services and resources in this AWS region.)
3. Choose **Create Stack**.
4. On the **Select Template** page, for **Specify an Amazon S3 template URL**, type the URL to the AWS CloudFormation template for this step: **`https://s3.amazonaws.com/aws-for-devops/cfn-templates/IAMSetup.template`**. Choose **Next**.
5. On the **Specify Details** page, for **Stack name**, type a stack name (for example, **`DevOpsIAMSetup`**). If you choose a different name, substitute it for **`DevOpsIAMSetup`** throughout this walkthrough.
6. The settings in the **Parameters** are used to:

   - Create an IAM group and an IAM user and then add the new user to the new group.
   - Attach to the new group a default set of access permissions for AWS CloudFormation, AWS CodeCommit, AWS CodeDeploy, AWS CodePipeline, Elastic Beanstalk, and AWS OpsWorks.

To accept these default settings, for **NewUserPassword**, type a password for the new user, and then skip to step 7 of this procedure.

Alternatively, you may want to experiment with these default settings by running this template multiple times to create IAM resources for this walkthrough. For example, later on you may want to attach access permissions to a newly created IAM group. Or you may not want to attach AWS OpsWorks access permissions to a group (or user), but you may change your mind later.

The following table shows which settings to choose in the **Parameters** area.

I want to create an IAM group and IAM user in my AWS account specifically for use with this walkthrough.	Leave **CreateGroupAndUser** set to the default value of **Yes**. For **NewUserPassword**, type a password for the new user to use to sign in to the AWS Management Console. (The user will be asked to change this password after initial sign-in.) **Note** If you have an existing group (or user) you want to use for this walkthrough, set **CreateGroupAndUser** to **No**.
I want to use an existing IAM group in my account for this walkthrough, and I already have at least one IAM user added to the group.	Set **ExistingGroup** to **Yes**. For **GroupName**, type the name of the group.
I want to use an existing IAM user in my account for this walkthrough.	Set **ExistingUser** to **Yes**. For **UserName**, type the name of the user.
I want to attach AWS CloudFormation administrative access permissions to the IAM groups (or users).	Leave **CloudFormation** set to the default value of **Yes**. **Note** Set **CloudFormation** to **No** only if the existing group (or user) you want to use for this walkthrough already has AWS CloudFormation administrative access permissions attached. For more information, see Controlling Access with AWS Identity and Access Management in the AWS CloudFormation User Guide.
I want to attach AWS CodeCommit full access permissions to the IAM groups (or users) I specified earlier.	Leave **CodeCommit** set to the default value of **Yes**. **Note** Set **CodeCommit** to **No** only if the existing group (or user) you want to use for this walkthrough already has AWS CodeCommit full access permissions attached. For more information, see the AWS CodeCommit User Access Permissions Reference in the AWS CodeCommit User Guide.

I want to attach AWS CodeDeploy default access permissions to the IAM groups (or users).	Leave **CodeDeploy** set to the default value of **Yes**. **Note** Set **CodeDeploy** to **No** only if the existing group (or user) you want to use for this walkthrough already has AWS CodeDeploy default access permissions attached, or if you do not want to deploy to AWS CodeDeploy deployment targets. For more information, see Step 1: Provision an IAM User in the AWS CodeDeploy User Guide.
I want to attach AWS CodePipeline full access permissions to the IAM groups (or users).	Leave **CodePipeline** set to the default value of **Yes**. **Note** Set **CodePipeline** to **No** only if the existing group (or user) you want to use for this walkthrough already has AWS CodePipeline full access permissions attached. For more information, see the AWS CodePipeline Access Permissions Reference in the AWS CodePipeline User Guide.
I want to attach Elastic Beanstalk full access permissions to the IAM groups (or users).	Leave **ElasticBeanstalk** set to the default value of **Yes**. **Note** Set **ElasticBeanstalk** to **No** only if the existing group (or user) you want to use for this walkthrough already has Elastic Beanstalk full access permissions attached, or if you do not want to deploy to Elastic Beanstalk deployment targets. For more information, see Controlling Access to Elastic Beanstalk in the AWS Elastic Beanstalk Developer Guide.
I want to attach AWS OpsWorks administrative access permissions to the IAM groups (or users).	Leave **OpsWorks** set to the default value of **Yes**. **Note** Set **OpsWorks** to **No** only if the existing group (or user) you want to use for this walkthrough already has AWS OpsWorks administrative access permissions attached, or if you do not want to deploy to AWS OpsWorks deployment targets. For more information, see Example Policies in the AWS OpsWorks User Guide.

Note
This walkthrough attaches very permissive access permissions to groups (or users). In production scenarios, as an AWS security best practice, you should limit these access permissions to only the AWS service actions and resources you need.

7. Choose **Next**.
8. On the **Options** page, choose **Next**. (You do not need to change anything on this page.)
9. On the **Review** page, select **I acknowledge that this template might cause AWS CloudFormation to create IAM resources**, and then choose **Create**.

   Note
   The steps in this walkthrough that instruct you to create AWS CloudFormation templates are very similar. If you forget how to create a template, use this topic as a refresher.
10. In the list of stacks, wait until **CREATE_COMPLETE** is displayed under **Status** for **DevOpsIAMSetup**.

    If you created a group and user, you can get information about them by choosing the corresponding **Physical ID** links on the **Resources** tab for the stack.
11. Sign out of the console, and then sign back in to the console with the new or existing user's credentials, and then go to Step 2: AWS CodeCommit Setup (p. 9).

Step 2: Set Up for AWS CodeCommit

In this step, you will set up access to AWS CodeCommit. Specifically, you will:

- Create a repository in AWS CodeCommit to store the source code to deploy.
- Launch an Amazon EC2 instance running Amazon Linux to connect to the AWS CodeCommit repository. (Although you could set up your local workstation to connect to the repository, we will show you how to launch an instance that is already set up.)
- Create an Amazon EC2 key pair, which you will use to log in to the newly launched instance.
- Confirm that you have successfully cloned the repository onto the instance.

Topics

Step 2.1: Create an AWS CodeCommit Repository

In this step, you will create an AWS CodeCommit repository that will store the source code. (Later, in Step 3: Download and Push the Source Code (p. 13), you will download the source code and then push it into this repository so that it can be deployed later in this walkthrough.)

To create the AWS CodeCommit repository

1. Open the AWS CodeCommit console at https://console.aws.amazon.com/codecommit/.
2. In the AWS region selector, choose **US East (N. Virginia)**. (This walkthrough uses AWS resources in this region only.)
3. On the **Dashboard** page, choose **Create new repository**. (If a welcome page appears instead of the **Dashboard** page, choose **Get started**.)

4. On the **Create new repository** page, for **Repository name**, type a repository name (for example, **`MyDemoRepo`**), and then choose **Create repository**. If you use a different name, substitute it for **`MyDemoRepo`** throughout this walkthrough.
5. On the **Dashboard** page, choose **MyDemoRepo**.
6. On the **Code: MyDemoRepo** page, choose **Clone URL**, and then choose **HTTPS**. Make a note of the URL that is displayed. You will need it for Step 2.3: Launch an Instance (p. 10).
7. Go to Step 2.2: Create a Key Pair (p. 10).

Step 2.2: Create an Amazon EC2 Key Pair

In this step, you will create an Amazon EC2 key pair that you will use to sign in to the instance that you will launch in Step 2.3: Launch an Instance (p. 10).

If you already have a key pair, make sure it was created in the US East (N. Virginia) region, and then go to Step 2.3: Launch an Instance (p. 10).

To create an Amazon EC2 key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the AWS region selector, choose **US East (N. Virginia)**.
3. Choose **Key Pairs**.
4. Choose **Create Key Pair**.
5. In the **Create Key Pair** dialog box, for **Key pair name**, type the name of your new key pair, and then choose **Create**.
6. When prompted, save the resulting `.pem` file to your local workstation.
7. Go to Step 2.3: Launch an Instance (p. 10).

Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository

Although you could set up your local workstation to connect to the repository you created in Step 2.1: Create a Repository (p. 9), in this step you will use an AWS CloudFormation template to launch an Amazon EC2 instance running Amazon Linux. (To view the contents of the AWS CloudFormation template, see CodeCommitInstance.template.)

This instance allows users to log in to it by using the SSH protocol. The instance has permission to take any AWS CodeCommit action for any AWS CodeCommit repository across the AWS account. AWS CloudFormation runs commands on the instance to set up the Git credential helper for AWS CodeCommit, clone the repository's contents onto the instance, and establish the user name and email address for all commits to the repository. (Later, in Step 2.4: Explore the Cloned Repository (p. 11), you will log in to the instance and confirm that you successfully cloned the repository onto the instance.)

To learn how to set up your local workstation to connect to AWS CodeCommit, see the following topics in the AWS CodeCommit User Guide:

- Setup for SSH Users Not Using the AWS CLI
- Setup Steps for HTTPS Connections to AWS CodeCommit Repositories on Linux, OS X, or Unix
- Setup Steps for HTTPS Connections to AWS CodeCommit Repositories on Windows
- Setup Steps for SSH Connections to AWS CodeCommit Repositories on Linux, OS X, or Unix
- Setup Steps for SSH Connections to AWS CodeCommit Repositories on Windows

To launch an Amazon Linux instance set up with AWS CodeCommit

1. Use the AWS CloudFormation console to create a stack in the US East (N. Virginia) region based on the following Amazon S3 template URL: **`https://s3.amazonaws.com/aws-for-devops/cfn-templates/CodeCommitInstance.template`**

   On the **Specify Details** page, for **Type**, type the Amazon EC2 instance type to launch (for example, **`t2.micro`**).

   For **KeyPair**, type the name of the key pair that you will use to log in to the instance (for example, the key pair you created in Step 2.2: Create an Amazon EC2 Key Pair (p. 10)).

   For **Tag**, type a tag for the instance to help you more easily identify it in places such as the Amazon EC2 console (for example, **`CodeCommitInstance`**).

   For **CloneURL**, type the HTTPS clone URL for the repository that you noted in Step 2.1: Create a Repository (p. 9).

   For **LocalRepo**, type the name of the subdirectory to create in the `/home/ec2-user` directory on the instance and then clone the repository into (for example, **`my-demo-repo`**). If you choose a different name, substitute it for **`my-demo-repo`** throughout this walkthrough.

   For **UserName**, type the user name you want to associate with all commits to the repository.

   For **UserEmail**, type the email address you want to associate with all commits to the repository.
2. When **CREATE_COMPLETE** is displayed for **Status** for the stack, you can explore the resources created by the stack. Otherwise, go to Step 2.4: Explore the Cloned Repository (p. 11).

To explore the Amazon EC2 resources created by the stack

1. In the AWS CloudFormation console, choose the **Resources** tab for the stack.
2. To view details about the newly launched instance, for **CodeCommitInstance**, choose the **Physical ID** link.
3. To view details about the newly created security group, on the **Description** tab in the Amazon EC2 console for the instance, choose the **Security groups** link.

To explore the IAM resources created by the stack

1. In the AWS CloudFormation console, choose the **Resources** tab for the stack.
2. To view the newly created IAM instance profile, for **CodeCommitInstanceProfileRole**, choose the **Physical ID** link.
3. Go to Step 2.4: Explore the Cloned Repository (p. 11).

Step 2.4: Explore the Cloned AWS CodeCommit Repository

In this step, you will log in to the newly launched instance and confirm that you successfully cloned the repository onto it.

To explore the cloned AWS CodeCommit repository on the Amazon Linux instance

1. To log in to the instance, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances. (To quickly get the public DNS for the instance, in the AWS

CloudFormation console, choose the **Resources** tab for the stack you created in Step 2.3: Launch an Instance (p. 10). For **CodeCommitInstance**, choose the **Physical ID** link. Then, in the Amazon EC2 console for the instance, find the **Public DNS** value.)

2. From the instance's command prompt, run the `cd /home/ec2-user/my-demo-repo` command or the `cd ~/my-demo-repo` command (or simply `cd my-demo-repo` because on login the command prompt defaults to the `/home/ec2-user` directory).
3. From the `my-demo-repo` directory, run the `ls -a` command. This lists the directory's contents, which should include the following:

```
. .. .git
```

4. From the `my-demo-repo` directory, you can run the `ls .git` command, which will produce the following output. These are Git-specific folders and files for the repository that should not be edited directly:

```
branches  config  descripton  HEAD  hooks  info  objects  refs
```

   There is not much more to explore yet, because you have not yet pushed any source code into it. You will do that in the next step.
5. Go to Step 3: Download and Push the Source Code (p. 13).

Step 3: Download and Push the Source Code

In this step, you will download the source code and then push it into the AWS CodeCommit repository so it can be deployed later in this walkthrough.

Choose the link that corresponds to the AWS service you want to use for this walkthrough:

Topics

Note

If you choose AWS CodeDeploy, follow these instructions in order: steps 3.1, 3.4, 4, 7.1, 8.1, 9.1, 10.1, 11.1, 11.2, 11.5, and 11.6.
If you choose AWS Elastic Beanstalk, follow these instructions in order: steps 3.2, 3.4, 5, 7.2, 8.2, 9.2, 10.2, 11.1, 11.3, 11.5, and 11.6.
If you choose AWS OpsWorks, follow these instructions in order: steps 3.3, 3.4, 6, 7.3, 8.3, 9.3, 10.3, 11.1, 11.4, 11.5, and 11.6.

Step 3.1: Download the Source Code for AWS CodeDeploy

In this step, you will download and prepare the source code that you will deploy to an AWS CodeDeploy deployment target as part of Step 7.1: AWS CodeDeploy Pipeline (p. 30).

If you want to download the source code for AWS Elastic Beanstalk, go to Step 3.2: Elastic Beanstalk Source Code (p. 15).

If you want to download the source code for AWS OpsWorks, go to Step 3.3: AWS OpsWorks Source Code (p. 16).

To download the source code for AWS CodeDeploy

1. You should already be logged in to the instance that you launched in Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository (p. 10). If not, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
2. From the command prompt on the instance, run the `pwd` command to confirm that you are in the `/home/ec2-user/my-demo-repo` directory (also referred to as the `~/my-demo-repo` directory). If a path other than `/home/ec2-user/my-demo-repo` appears in the output, run the `cd ~/my-demo-repo` command.
3. Download the source code for AWS CodeDeploy into the `~/my-demo-repo` directory, and then prepare the source code for deployment by running the following commands, one at a time.

```
wget https://s3.amazonaws.com/aws-codedeploy-us-east-
1/samples/latest/SampleApp_Linux.zip
```

```
unzip SampleApp_Linux.zip
```

```
rm SampleApp_Linux.zip
```

4. Explore the source code. Otherwise, go to Step 3.4: Push the Source Code (p. 17).

To explore the source code for AWS CodeDeploy

1. From the `~/my-demo-repo` directory, run these commands to display the contents of the extracted files in read-only mode.

```
less appspec.yml
```

```
less index.html
```

```
less scripts/install_dependencies
```

```
less scripts/start_server
```

```
less scripts/stop_server
```

To move up or down one line at a time, press the `k` or `j` key, respectively. To move up or down one page at a time, press `b` or the space bar, respectively. To exit, press the `q` key.

The `appspec.yml` file contains a set of AWS CodeDeploy deployment commands. These commands instruct AWS CodeDeploy to deploy the `index.html` file to the instance and to install and start a web server on the instance. For more information, see the AWS CodeDeploy AppSpec File Reference in the AWS CodeDeploy User Guide.
2. After you are finished exploring the contents, go to Step 3.4: Push the Source Code (p. 17).

Step 3.2: Download the Source Code for AWS Elastic Beanstalk

In this step, you will download and prepare the source code that you will deploy to an Elastic Beanstalk deployment target as part of Step 7.2: Elastic Beanstalk Pipeline (p. 32).

If you want to download the source code for AWS CodeDeploy, go to Step 3.1: AWS CodeDeploy Source Code (p. 13).

If you want to download the source code for AWS OpsWorks, go to Step 3.3: AWS OpsWorks Source Code (p. 16).

To download the source code for Elastic Beanstalk

1. You should already be logged in to the instance that you launched in Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository (p. 10). If not, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
2. From the command prompt on the instance, run the `pwd` command to confirm you are in the `/home/ec2-user/my-demo-repo` directory (also referred to as the `~/my-demo-repo` directory). If a path other than `/home/ec2-user/my-demo-repo` appears in the output, run the `cd ~/my-demo-repo` command.
3. Download the source code for Elastic Beanstalk into the `~/my-demo-repo` directory, and then prepare the source code for deployment by running the following commands, one at a time.

```
wget http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/samples/php-v1.zip
```

```
unzip php-v1.zip
```

```
rm php-v1.zip
```

4. Explore the source code. Otherwise, go to Step 3.4: Push the Source Code (p. 17).

To explore the source code for Elastic Beanstalk

1. From the `~/my-demo-repo` directory, run these commands to display the contents of the extracted files in read-only mode.

```
less cron.yaml
```

```
less index.php
```

```
less scheduled.php
```

```
less styles.css
```

```
less .ebextensions/logging.config
```

To move up or down one line at a time, press the k or j key, respectively. To move up or down one page at a time, press b or the space bar, respectively. To exit, press the q key.

The cron.yaml file contains the definition of a periodic background task for Elastic Beanstalk to run in conjunction with the scheduled.php page. The logging.config file contains application logging settings. For more information, see Periodic Tasks and Advanced Environment Customization with Configuration Files (.ebextensions) in the AWS Elastic Beanstalk Developer Guide.

2. After you are finished exploring the contents, go to Step 3.4: Push the Source Code (p. 17).

Step 3.3: Download the Source Code for AWS OpsWorks

In this step, you will download the source code that you will deploy to an AWS OpsWorks deployment target as part of Step 7.3: AWS OpsWorks Pipeline (p. 34).

If you want to download the source code for AWS CodeDeploy, go to Step 3.1: AWS CodeDeploy Source Code (p. 13).

If you want to download the source code for AWS Elastic Beanstalk, go to Step 3.2: Elastic Beanstalk Source Code (p. 15).

To download the source code for AWS OpsWorks

1. You should already be logged in to the instance that you launched in Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository (p. 10). If not, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
2. From the instance's command prompt, run the pwd command to confirm you are in the /home/ec2-user/my-demo-repo directory (also referred to as the ~/my-demo-repo directory). If a path other than /home/ec2-user/my-demo-repo appears in the output, run the cd ~/my-demo-repo command.
3. Download the source code for AWS OpsWorks into the ~/my-demo-repo directory, and then prepare the source code for deployment by running the following commands, one at a time.

```
wget https://github.com/awslabs/opsworks-demo-php-simple-app/archive/ver
sion1.zip
```

```
unzip version1.zip
```

```
rm version1.zip
```

```
cp -r opsworks-demo-php-simple-app-version1/* .
```

```
rm -rf opsworks-demo-php-simple-app-version1
```

Because the directory structure of the https://github.com/awslabs/opsworks-demo-php-simple-app/archive/version1.zip file is not suitable for deployment as is, the preceding commands prepare the source code for deployment by:

- Extracting the contents of the GitHub version of the ZIP file and then deleting that ZIP file.

- Copying the contents of the `opsworks-demo-php-simple-app-version1` directory in the original source code to the parent directory and then deleting the `opsworks-demo-php-simple-app-version1` directory.

4. Explore the source code. Otherwise, go to Step 3.4: Push the Source Code (p. 17).

To explore the source code for AWS OpsWorks

1. From the `~/my-demo-repo` directory, run these commands to display the contents of the extracted files in read-only mode.

```
less index.php
```

```
less assets/css/bootstrap.min.css
```

```
less assets/js/bootstrap.min.js
```

   To move up or down one line at a time, press the `k` or `j` key, respectively. To move up or down one page at a time, press `b` or the space bar, respectively. To exit, press the `q` key.
2. After you are finished exploring the contents, go to Step 3.4: Push the Source Code (p. 17).

Step 3.4: Push the Source Code

In this step, you will push the source code to the AWS CodeCommit repository. (This source code will be deployed to the deployment target later, as part of Step 7: Create and Run the Pipeline (p. 30).)

To push the source code to the AWS CodeCommit repository

1. You should already be logged in to the instance that you launched in Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository (p. 10). If not, then follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
2. From the command prompt on the instance, run the `pwd` command to confirm that you are in the `~/my-demo-repo` directory. If a path other than `/home/ec2-user/my-demo-repo` appears in the output, run the `cd ~/my-demo-repo` command.
3. Run the following Git commands, one at a time, to add, commit, and then push the source code files to the AWS CodeCommit repository.

```
git add .
```

```
git commit -m "Added source code files"
```

```
git push
```

4. After the push is successful, go to one of the following:

   - Step 4: AWS CodeDeploy Setup (p. 19).
   - Step 5: Elastic Beanstalk Setup (p. 23).

- Step 6: AWS OpsWorks Setup (p. 26).

Step 4: Set Up for AWS CodeDeploy

In this step, you will create AWS resources that AWS CodeDeploy will use to deploy the source code to the deployment target (an Amazon EC2 instance running Amazon Linux).

If you want to deploy with AWS Elastic Beanstalk, go to Step 5: Elastic Beanstalk Setup (p. 23).

If you want to deploy with AWS OpsWorks, go to Step 6: AWS OpsWorks Setup (p. 26).

To log in to and explore the deployment target, you will need an Amazon EC2 key pair. You can use the key pair you created or identified in Step 2.2: Create a Key Pair (p. 10).

The following procedure uses an AWS CloudFormation template to create an AWS CodeDeploy application, deployment group, service role, and an IAM instance profile, and to launch an Amazon Linux instance. (To view the contents of the AWS CloudFormation template, see CodeDeploySetup.template.) To learn how to create these resources in other ways, such as with the AWS Management Console or the AWS Command Line Interface (AWS CLI), see the following topics in the AWS CodeDeploy User Guide:

- Create an Application with AWS CodeDeploy
- Create a Deployment Group with AWS CodeDeploy
- Create a Service Role for AWS CodeDeploy
- Create an IAM Instance Profile for Your Amazon EC2 Instances
- Working with Instances for AWS CodeDeploy

To create the AWS CodeDeploy resources

1. Use the AWS CloudFormation console to create a stack in the US East (N. Virginia) region based on the following Amazon S3 template URL:
   `https://s3.amazonaws.com/aws-for-devops/cfn-templates/CodeDeploySetup.template`

   On the **Specify Details** page, the setttings in the **Parameters** area are used to:

   - Create an AWS CodeDeploy default service role and IAM instance profile. The service role enables AWS CodeDeploy to interact with dependent AWS services on the user's behalf. The IAM instance profile enables associated Amazon EC2 instances to interact with dependent AWS services.
   - Create an AWS CodeDeploy application and deployment group, and associate the new service role with the new deployment group.

- Launch an Amazon EC2 instance running Amazon Linux and install the AWS CodeDeploy agent on the instance. This instance allows users to log in to it by using the SSH protocol and to access it by using the HTTP protocol. The new instance profile is attached to the instance.

To accept these default settings, type your Amazon EC2 key pair name (for example, the one you created in Step 2.2: Create a Key Pair (p. 10)) into **KeyPair**, and then go to step 2 of this procedure.

Alternatively, you may want to experiment with these default settings by running this template multiple times to create AWS CodeDeploy, IAM, and Amazon EC2 resources for this walkthrough. For example, later on you may want to create more AWS CodeDeploy deployment groups or launch more Amazon Linux instances that are compatible with AWS CodeDeploy.

The following table shows which settings to choose in the **Parameters** area.

I want to create a new AWS CodeDeploy application.	Leave **Application** set to the default value of **New**. For **ApplicationName**, type the name for the new application. **Note** Set **Application** to **Skip** if you only want to create a new AWS CodeDeploy service role, instance profile, or instance.
I want to create a new AWS CodeDeploy deployment group.	Leave **DeploymentGroup** set to the default value of **New**. For **DeploymentGroupName**, type the name of the new application. A deployment group must be associated with an application. Set **Application** to **New** or **Existing**, and in **ApplicationName**, type the name of the new or existing AWS CodeDeploy application to associate it with the new deployment group. A deployment group must have an Amazon EC2 tag filter key and value so that AWS CodeDeploy knows which instances to deploy to. Type the key in **TagKey**, and type the value in **TagValue**. A deployment group must have an AWS CodeDeploy service role. Set **ServiceRole** to **New** to create a new service role and associate it with the new deployment group, or set it to **Existing** to associate an existing service with the new deployment group. If you choose **Existing**, type the Amazon Resource Name (ARN) of the service role in **ServiceRoleARN**. **Note** Set **DeploymentGroup** to **Skip** if you only want to create a new AWS CodeDeploy service role, instance profile, or instance.

I want to create a new AWS CodeDeploy service role.	Leave **ServiceRole** set to the default value of **New**. **Note** Set **ServiceRole** to **Skip** if you only want to launch a new Amazon Linux instance that is compatible with AWS CodeDeploy. If you choose **Existing** and specify the service role's ARN in **ServiceRoleARN**, the service role must be compatible with AWS CodeDeploy. For more information, see Step 3: Create a Service Role for AWS CodeDeploy in the AWS CodeDeploy User Guide.
I want to launch a new Amazon Linux instance that is compatible with AWS CodeDeploy.	Leave **CreateInstance** set to the default value of **New**. For **InstanceType**, type the value of the Amazon EC2 instance type to launch. For **TagKey**, type the Amazon EC2 tag filter key to associate with the instance. For **TagValue**, type Amazon EC2 tag filter value to associate with the instance. For **KeyPair**, type the name of the existing Amazon EC2 key pair you will use to log in to the launched instance. For **InstanceProfile**, if you choose **Existing**, type the name (not ARN) of the IAM instance profile in **InstanceProfileName**. The IAM instance profile must be compatible with AWS CodeDeploy. For more information, see Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances in the AWS CodeDeploy User Guide.

2. When **CREATE_COMPLETE** is displayed for **Status** for the stack, you can view the resources created by the stack. Otherwise, go to Step 7.1: Create and Run the Pipeline for AWS CodeDeploy (p. 30).

To explore the AWS CodeDeploy resources created by the stack

1. Open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
2. In the AWS region selector, choose **US East (N. Virginia)**.
3. On the **Applications** page, choose the AWS CodeDeploy application.
4. On the **Application details** page, in the **Deployment groups** area, expand the deployment group's name.

To explore the IAM resources created by the stack

1. In the AWS CloudFormation console, choose the **Resources** tab for the stack.
2. If you created an IAM instance profile, for **CodeDeployInstanceProfileRole**, choose the **Physical ID** link.
3. If you created a service role, for **CodeDeployServiceRole**, choose the **Physical ID** link.

To explore the Amazon EC2 resources created by the stack

1. In the AWS CloudFormation console, choose the **Resources** tab for the stack.
2. If you launched an instance, for **CodeDeployInstance**, choose the **Physical ID** link to view details about the instance.
3. To view details about the security group for the launched instance, in the Amazon EC2 console, on the **Description** tab for the instance, choose the **Security groups** link.
4. To log in to the launched instance, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
5. To confirm the AWS CodeDeploy agent is running on the launched instance, log in to the instance and then run the `sudo service codedeploy-agent status` command. If the agent is running, the output should start with `The AWS CodeDeploy agent is running`.
6. Go to Step 7.1: Create and Run the Pipeline for AWS CodeDeploy (p. 30).

Step 5: Set Up for AWS Elastic Beanstalk

In this step, you will you will create AWS resources that Elastic Beanstalk will use to deploy the source code to the deployment target (an Amazon EC2 instance running Amazon Linux).

If you want to deploy with AWS CodeDeploy, go to Step 4: AWS CodeDeploy Setup (p. 19).

If you want to deploy with AWS OpsWorks, go to Step 6: AWS OpsWorks Setup (p. 26).

To log in to and explore the deployment target, you will need an Amazon EC2 key pair. You can use the key pair you created or identified in Step 2.2: Create a Key Pair (p. 10).

The following procedure uses an AWS CloudFormation template to create an Elastic Beanstalk application, environment, service role, and an IAM instance profile, and to launch an Amazon Linux instance. (To view the contents of the AWS CloudFormation template, see ElasticBeanstalkSetup.template.) To learn how to create these resources in other ways, such as with the AWS Management Console or the AWS Command Line Interface (AWS CLI), see the following topics in the AWS Elastic Beanstalk Developer Guide:

- Create an Application
- The New Environment Wizard
- Managing Elastic Beanstalk Instance Profiles
- Managing Elastic Beanstalk Service Roles

To create the Elastic Beanstalk resources

1. Use the AWS CloudFormation console to create a stack in the US East (N. Virginia) region based on the following Amazon S3 template URL:
   `https://s3.amazonaws.com/aws-for-devops/cfn-templates/ElasticBeanstalkSetup.template`

   On the **Specify Details** page, the settings in the **Parameters** area are used to:

   - Create a new Elastic Beanstalk application and environment, a service role, and an IAM instance profile. The service role enables Elastic Beanstalk to interact with dependent AWS services on the user's behalf. The IAM instance profile enables associated Amazon EC2 instance to interact with dependent AWS services.
   - Associate the new service role and IAM instance profile with the new environment.

- Launch into the environment an Amazon EC2 instance running Amazon Linux. This instance allows users to log in to it by using the SSH protocol and to access it by using the HTTP protocol. The new IAM instance profile is attached to the instance.

To accept these default settings, type your Amazon EC2 key pair name (for example, the one you created in Step 2.2: Create a Key Pair (p. 10)) into **InstanceKeyPair**, and then go to step 2 of this procedure.

Alternatively, you may want to experiment with these default settings by running this template multiple times to create Elastic Beanstalk, IAM, and Amazon EC2 resources for this walkthrough. For example, later on you may want to create additional Elastic Beanstalk applications, environments, or both.

The following table shows which settings to choose in the **Parameters** area.

I want to create a new Elastic Beanstalk application.	Leave **Application** set to the default value of **New**. For **ApplicationName**, type the name of the new application. **Note** Set **Application** to **Existing** if you only want to create a new Elastic Beanstalk environment for an existing application. For **ApplicationName**, type the name of the existing application.
I want to create a new IAM instance profile for the Amazon Linux instance that will be launched into the new Elastic Beanstalk environment.	Leave **InstanceProfile** set to the default value of **New**. **Note** Set **InstanceProfile** to **Existing** if you already have an IAM instance profile you want to use. For **InstanceProfileARN**, type the Amazon Resource Name (ARN) of the existing IAM instance profile. The IAM instance profile must be compatible with Elastic Beanstalk. For more information, see Elastic Beanstalk Instance Profile in the AWS Elastic Beanstalk Developer Guide.
I want to create a new Elastic Beanstalk service role for the new Elastic Beanstalk environment.	Leave **ServiceRole** set to the default value of **New**. **Note** Set **ServiceRole** to **Existing** if you already have a service role you want to use. For **ServiceRoleARN**, type the ARN of the existing service role. The service role must be compatible with Elastic Beanstalk. For more information, see Elastic Beanstalk Service Role in the AWS Elastic Beanstalk Developer Guide.

2. When **CREATE_COMPLETE** is displayed for **Status** for this stack and for the stack named **awseb-e-*`random-ID`*-stack**, you can view the resources created by the stacks. Otherwise, go to Step 7.2: Create and Run the Pipeline for AWS Elastic Beanstalk (p. 32).

Note

This stack creates an additional AWS CloudFormation stack for the Elastic Beanstalk environment with a name similar to `awseb-e-`*`random-ID`*`-stack`. Do not modify the stack. When you delete this stack in Step 11.3: Clean Up Elastic Beanstalk (p. 50), AWS CloudFormation will delete the additional stack, too.

To explore the Elastic Beanstalk resources created by the stack

1. In the AWS CloudFormation console, choose the **Resources** tab for the stack.
2. If you created a new application, for **ElasticBeanstalkApplication**, choose the **Physical ID** link.
3. On the **All Applications** > ***`environment name`*** page, choose the name of the new environment.

To explore the IAM resources created by the stack

1. In the AWS CloudFormation console, choose the **Resources** tab for the stack.
2. If you created a new IAM instance profile, for **ElasticBeanstalkInstanceProfileRole**, choose the **Physical ID** link.
3. If you created a new service role, for **ElasticBeanstalkServiceRole**, choose the **Physical ID** link.
4. Go to Step 7.2: Create and Run the Pipeline for AWS Elastic Beanstalk (p. 32).

Step 6: Set Up for AWS OpsWorks

In this step, you will create AWS resources that AWS OpsWorks will use to deploy the source code to the deployment target (an Amazon EC2 instance running Amazon Linux).

If you want to deploy with AWS CodeDeploy, go to Step 4: AWS CodeDeploy Setup (p. 19).

If you want to deploy with AWS Elastic Beanstalk, go to Step 5: Elastic Beanstalk Setup (p. 23).

To log in to and explore the deployment target, you will need an Amazon EC2 key pair. You can use the key pair you created or identified in Step 2.2: Create a Key Pair (p. 10).

The following procedure uses an AWS CloudFormation template to create an AWS OpsWorks stack, layer, app, service role, and an IAM instance profile and to launch an Amazon Linux instance. (To view the contents of the AWS CloudFormation template, see OpsWorksSetup.template.) To learn how to create these resources in other ways, such as with the AWS Management Console or the AWS Command Line Interface (AWS CLI), see the following topics in the AWS OpsWorks User Guide:

- Create a New Stack
- Creating an AWS OpsWorks Layer
- Allowing AWS OpsWorks to Act on Your Behalf
- Specifying Permissions for Apps Running on Amazon EC2 instances
- Adding an Instance to a Layer
- Adding an App

To create the AWS OpsWorks resources

1. Use the AWS CloudFormation console to create a stack in the US East (N. Virginia) region based on the following Amazon S3 template URL:
   `https://s3.amazonaws.com/aws-for-devops/cfn-templates/OpsWorksSetup.template`

   On the **Specify Details** page, the settings in the **Parameters** area are used to:

   - Create an AWS OpsWorks stack that launches, by default, Amazon EC2 instances running Amazon Linux and Chef 11.10.
   - In the AWS OpsWorks stack, create an AWS OpsWorks layer that is optimized for PHP apps.
   - In the AWS OpsWorks stack, create an AWS OpsWorks app that is based on a simple PHP solution.
   - Create a default AWS OpsWorks service role that enables AWS OpsWorks to interact with dependent AWS services on the user's behalf.

- In the AWS OpsWorks layer, launch an Amazon EC2 instance running Amazon Linux that is compatible with AWS OpsWorks and Chef 11.10. This instance allows users to log in to it by using the SSH protocol and to access it by using the HTTP protocol. The AWS OpsWorks default IAM instance profile is attached to the instance. This IAM instance profile enables the instance to take actions related to AWS OpsWorks and its dependent AWS services for AWS resources across the AWS account.

To accept these default settings, type your Amazon EC2 key pair name (for example, the one you created in Step 2.2: Create a Key Pair (p. 10)) into **KeyPair**, and then go to step 2 of this procedure.

Alternatively, you may want to experiment with these default settings by running this template multiple times to create AWS OpsWorks, IAM, and Amazon EC2 resources for this walkthrough. For example, later on you may want to create more AWS OpsWorks stacks, layers, or apps. Or you may want to launch more Amazon EC2 instances that are compatible with AWS OpsWorks.

The following table shows which settings to choose in the **Parameters** area.

I want to create a new AWS OpsWorks stack that, by default, launches Amazon EC2 instances running Amazon Linux and Chef 11.10.	Leave **Stack** set to the default value of **New**. For **StackName**, type the name of the new AWS OpsWorks stack. The new stack must have an AWS OpsWorks service role. To create a new default AWS OpsWorks service role, leave **ServiceRole** set to the default value of **New**. **Note** Set **ServiceRole** to **Existing** if you already have a service role you want to use. Type the Amazon Resource Name (ARN) of the service role in **ServiceR-oleARN**. The service role must be compatible with AWS OpsWorks. For more information, see Allowing AWS OpsWorks to Act on Your Behalf in the AWS OpsWorks User Guide. The new stack must also have a default IAM in-stance profile to attach to all associated Amazon EC2 instances. To create a default IAM instance profile, leave **InstanceProfile** set to the default value of **New**. **Note** Set **InstanceProfile** to **Existing** if you already have an IAM instance profile you want to use. Type the ARN of the IAM instance profile in **InstancePro-fileARN**. The IAM instance profile must be compatible with AWS OpsWorks. For more information, see Specifying Permissions for Apps Running on Amazon EC2 Instances in the AWS OpsWorks User Guide.

I want to create a new AWS OpsWorks layer that is optimized for PHP apps.	Leave **Layer** set to the default value of **New**. Optionally, you can associate an existing Amazon EC2 security group with the layer. This security group will be associated with all Amazon EC2 instances launched into the layer. Set **CustomSecurityGroup** to **Yes**, and type the security group ID in **CustomSecurityGroupID**.
I want to launch a new Amazon EC2 instance running Amazon Linux that is compatible with AWS OpsWorks and Chef 11.10.	Leave **Instance** set to the default value of **Yes**. Enter the instance type into **Type**. Type your Amazon EC2 key pair name into **KeyPair**. The instance must have an associated AWS OpsWorks stack. Set **Stack** to **New** or **Existing**. If you set **Stack** to **New**, type the name of the new AWS OpsWorks stack into **StackName**. If you set **Stack** to **Existing**, type the AWS OpsWorks ID of the stack into **StackID**. Be sure to set **ServiceRole**, **ServiceRoleARN**, **Instance-Profile**, and **InstanceProfileARN** as needed. The instance must also have an AWS OpsWorks layer associated with the new or existing AWS OpsWorks stack. Set **Layer** to **New** or **Existing**. If you set **Layer** to **New**, type the name of the new AWS OpsWorks layer into **LayerName**. If you set **Layer** to **Existing**, type the AWS OpsWorks ID of the existing AWS OpsWorks layer into **LayerID**.
I want to create a new AWS OpsWorks app that is based on a simple PHP solution.	Leave **App** set to the default value of **Yes**. For **AppName**, type the name of the new AWS OpsWorks app. The AWS OpsWorks app must have an associated AWS OpsWorks stack. Set **Stack** to **New** or **Existing**. If you set **Stack** to **New**, type the name of the new AWS OpsWorks stack into **StackName**. If you set **Stack** to **Existing**, type the AWS OpsWorks ID of the existing AWS OpsWorks stack into **StackID**. Be sure to set **ServiceRole**, **ServiceRoleARN**, **InstancePro-file**, and **InstanceProfileARN** as needed.

2. When **CREATE_COMPLETE** is displayed for **Status** for the stack, choose the **Resources** tab. Make a note of the following **Physical ID** values. You will need them for Step 7.3: Create and Run the Pipeline for AWS OpsWorks (p. 34).

   - **OpsWorksApp**
   - **OpsWorksLayer**
   - **OpsWorksStack**

3. Continue on to explore the resources created by the stack. Otherwise, go to Step 7.3: Create and Run the Pipeline for AWS OpsWorks (p. 34).

To explore the AWS OpsWorks resources created by the AWS CloudFormation stack

1. Open the AWS OpsWorks console at https://console.aws.amazon.com/opsworks/.
2. On the **OpsWorks Dashboard** page, choose the AWS OpsWorks stack.
3. On the AWS OpsWorks stack's overview page, choose **Stack Settings**.
4. Choose **Layers**.
5. On the **Layers** page, for the layer, choose **Settings**.
6. Choose **Apps**.
7. On the **Apps** page, choose the AWS OpsWorks app.
8. Choose **Instances**.
9. On the **Instances** page, choose the **Hostname** link to view the settings for the instance. You'll find more information by choosing the **EC2 Instance ID** link on the **Details** page.
10. To log in to the instance, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.

To explore the IAM resources created by the AWS CloudFormation stack

1. In the AWS CloudFormation console, choose the **Resources** tab.
2. To explore the IAM instance profile, for **OpsWorksInstanceProfileRole**, choose the **Physical ID** link.
3. To explore the service role, for **OpsWorksServiceRole**, choose the **Physical ID** link.
4. Go to Step 7.3: Create and Run the Pipeline for AWS OpsWorks (p. 34).

Step 7: Create and Run the Pipeline in AWS CodePipeline

In this step, you will create a pipeline in AWS CodePipeline to automate the deployment of the source code in AWS CodeCommit to your deployment target.

Choose the link that corresponds to the AWS service you selected in Step 3: Download and Push the Source Code (p. 13) and the AWS resources you created in steps 4, 5, or 6.

Topics

Step 7.1: Create and Run the Pipeline for AWS CodeDeploy

In this step, you will create a pipeline in AWS CodePipeline to automate the deployment of the source code in AWS CodeCommit to your deployment target in AWS CodeDeploy. After you create the pipeline, AWS CodePipeline will immediately start the deployment.

If you want to create a pipeline for Elastic Beanstalk instead, go to Step 7.2: Elastic Beanstalk Pipeline (p. 32).

If you want to create a pipeline for AWS OpsWorks instead, go to Step 7.3: AWS OpsWorks Pipeline (p. 34).

The following procedure uses an AWS CloudFormation template to create a pipeline in AWS CodePipeline. (To view the contents of the AWS CloudFormation template, see CodeDeployPipelineSetup.template.) To learn how to create the pipeline in other ways, such as with the AWS Management Console or the AWS Command Line Interface (AWS CLI), see the following topics in the AWS CodePipeline User Guide:

- Create a Pipeline in AWS CodePipeline
- Create a Policy for an Amazon S3 Bucket to Use as the Artifact Store for AWS CodePipeline
- Edit a Policy for an IAM Service Role

To create and run the pipeline for AWS CodeDeploy

1. Use the AWS CloudFormation console to create a stack in the US East (N. Virginia) region based on the following Amazon S3 template URL: **`https://s3.amazonaws.com/aws-for-devops/cfn-templates/CodeDeployPipelineSetup.template`**

   On the **Specify Details** page, the settings in the **Parameters** area are used to:

   - Create a pipeline in AWS CodePipeline, specifying the AWS CodeCommit repository and branch to deploy from and the AWS CodeDeploy application and deployment group to deploy to.
   - Create an Amazon S3 bucket used by AWS CodePipeline to store and manage the files to be deployed, apply to the new bucket an Amazon S3 bucket policy that allows AWS CodePipeline to access the bucket, and associate the bucket with the new pipeline.
   - Create an AWS CodePipeline service role for the pipeline to use to interact with dependent AWS services, and associate the service role with the new pipeline.

   To accept these default settings, type a unique bucket name for **ArtifactBucketName**, and then go to step 2 of this procedure.

   Alternatively, you may want to experiment with these default settings by running this template multiple times to create AWS CodePipeline, Amazon S3, and IAM resources for this walkthrough. For example, later on you may want to create more pipelines for use with AWS CodeDeploy and associate them with an existing bucket and an existing service role.

   The following table shows which settings to choose in the **Parameters** area.

I want to create an Amazon S3 bucket for the pipeline to use to store and manage the files to be deployed.	Leave **ArtifactBucket** set to the default value of **New**. For **ArtifactBucketName**, type the name of the new bucket. **Note** Set **ArtifactBucket** to **Existing** if you already have a bucket you want the pipeline to use. For **ArtifactBucket-Name**, type the name of the existing bucket. If you use an existing bucket, the bucket must have versioning enabled and a specific bucket policy attached. For more information, see Create a Policy for an Amazon S3 Bucket to Use as the Artifact Store for AWS CodePipeline in the AWS CodePipeline User Guide.

I want to create an AWS CodePipeline service role for the pipeline to use to interact with dependent AWS services.	Leave **ServiceRole** set to the default value of **New**. **Note** Set **ServiceRole** to **Existing** if you already have an AWS CodePipeline service role you want the pipeline to use. For **ServiceRoleARN**, type the Amazon Resource Name (ARN) of the service role. The service role must be compatible with AWS CodePipeline. For more information, see Edit a Policy for an IAM Service Role in the AWS CodePipeline User Guide.

2. When **CREATE_COMPLETE** is displayed for **Status** for the stack, open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
3. In the AWS region selector, choose **US East (N. Virginia)**.
4. On the **All Pipelines** page, choose the new pipeline.
5. When **Succeeded** is displayed for the **Source** and **Deploy** stages, go to Step 8.1: Verify AWS CodeDeploy Results (p. 36).

Step 7.2: Create and Run the Pipeline for AWS Elastic Beanstalk

In this step, you will create a pipeline in AWS CodePipeline to automate the deployment of the source code in AWS CodeCommit to your deployment target in Elastic Beanstalk. After you create the pipeline, AWS CodePipeline will immediately start the deployment.

If you want to create a pipeline for AWS CodeDeploy, go to Step 7.1: AWS CodeDeploy Pipeline (p. 30).

If you want to create a pipeline for AWS OpsWorks, go to Step 7.3: AWS OpsWorks Pipeline (p. 34).

The following procedure uses an AWS CloudFormation template to create a pipeline in AWS CodePipeline. (To view the contents of the AWS CloudFormation template, see ElasticBeanstalkPipelineSetup.template.) To learn how to create the pipeline in other ways, such as with the AWS Management Console or the AWS Command Line Interface (AWS CLI), see the following topics in the AWS CodePipeline User Guide:

- Create a Pipeline in AWS CodePipeline
- Create a Policy for an Amazon S3 Bucket to Use as the Artifact Store for AWS CodePipeline
- Edit a Policy for an IAM Service Role

To create and run the pipeline for Elastic Beanstalk

1. Use the AWS CloudFormation console to create a stack in the US East (N. Virginia) region based on the following Amazon S3 template URL:

   ```
   https://s3.amazonaws.com/aws-for-devops/cfn-templates/ElasticBeanstalkPipelineSetup.template
   ```

   On the **Specify Details** page, the settings in the **Parameters** area are used to:

   - Create a pipeline in AWS CodePipeline, specifying the AWS CodeCommit repository and branch to deploy from and the Elastic Beanstalk application and environment to deploy to.

- Create an Amazon S3 bucket used by AWS CodePipeline o store and manage the files to be deployed, apply to the bucket an Amazon S3 bucket policy that allows AWS CodePipeline to access the bucket, and associate the bucket with the pipeline.
- Create an AWS CodePipeline service role for the pipeline to use to interact with dependent AWS services, and associate the service role with the pipeline.

To accept these default settings, type a unique bucket name for **ArtifactBucketName**, and then go to step 2 of this procedure.

Alternatively, you may want to experiment with these default settings by running this template multiple times to create AWS CodePipeline, Amazon S3, and IAM resources for this walkthrough. For example, later on you may want to create more pipelines for use with Elastic Beanstalk and associate them with an existing bucket and service role.

The following table shows which settings to choose in the **Parameters** area.

I want to create an Amazon S3 bucket that will be used by the pipeline to store and manage the files to be deployed.	Leave **ArtifactBucket** set to the default value of **New**. For **ArtifactBucketName**, type the name of the new bucket. **Note** Set **ArtifactBucket** to **Existing** if you already have a bucket you want the pipeline to use. For **ArtifactBucket-Name**, type the name of the bucket. If you use an existing bucket, the bucket must have versioning enabled and a specific bucket policy attached. For more information, see Create a Policy for an Amazon S3 Bucket to Use as the Artifact Store for AWS CodePipeline in the AWS CodePipeline User Guide.
I want to create an AWS CodePipeline service role for the pipeline to use to interact with depend-ent AWS services.	Leave **ServiceRole** set to the default value of **New**. **Note** Set **ServiceRole** to **Existing** if you already have an AWS CodePipeline service role you want the pipeline to use. For **ServiceRoleARN**, type the Amazon Resource Name (ARN) of the service role. The service role must be compatible with AWS CodePipeline. For more information, see Edit a Policy for an IAM Service Role in the AWS CodePipeline User Guide.

2. When **CREATE_COMPLETE** is displayed for **Status** for the stack, open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
3. In the AWS region selector, choose **US East (N. Virginia)**.
4. On the **All Pipelines** page, choose the new pipeline.
5. When **Succeeded** is displayed for the **Source** and **Deploy** stages, go to Step 8.2: Verify Elastic Beanstalk Results (p. 37).

Step 7.3: Create and Run the Pipeline for AWS OpsWorks

In this step, you will create a pipeline in AWS CodePipeline to automate the deployment of the source code in AWS CodeCommit to your deployment target in AWS OpsWorks. After you create the pipeline, AWS CodePipeline will immediately start the deployment.

If you want to create a pipeline for AWS CodeDeploy, go to Step 7.1: AWS CodeDeploy Pipeline (p. 30).

If you want to create a pipeline for Elastic Beanstalk, go to Step 7.2: Elastic Beanstalk Pipeline (p. 32).

The following procedure uses an AWS CloudFormation template to create a pipeline in AWS CodePipeline. (To view the contents of the AWS CloudFormation template, see OpsWorksPipelineSetup.template.) To learn how to create the pipeline in other ways, such as with the AWS Management Console or the AWS Command Line Interface (AWS CLI), see the following topics in the AWS CodePipeline User Guide:

- Create a Pipeline in AWS CodePipeline
- Create a Policy for an Amazon S3 Bucket to Use as the Artifact Store for AWS CodePipeline
- Edit a Policy for an IAM Service Role

To create and run the pipeline for AWS OpsWorks

1. Use the AWS CloudFormation console to create a stack in the US East (N. Virginia) region based on the following Amazon S3 template URL: **`https://s3.amazonaws.com/aws-for-devops/cfn-templates/OpsWorksPipelineSetup.template`**

   On the **Specify Details** page, the settings in the **Parameters** area are used to:

   - Create a pipeline in AWS CodePipeline, specifying the AWS CodeCommit repository and branch to deploy from and the AWS OpsWorks stack, layer, and app to deploy to.
   - Create an Amazon S3 bucket used by AWS CodePipeline to store and manage the files to be deployed, apply to the new bucket an Amazon S3 bucket policy that allows AWS CodePipeline to access the bucket, and associate the bucket with the pipeline.
   - Create an AWS CodePipeline service role for the pipeline to use to interact with dependent AWS services, and associate the service role with the pipeline.

   To accept these default settings, type the AWS OpsWorks IDs you noted for **StackID**, **LayerID**, and **AppID**. Type a unique bucket name for **ArtifactBucketName**, and then go to step 2 of this procedure.

   Alternatively, you may want to experiment with these default settings by running this template multiple times to create AWS CodePipeline, Amazon S3, and IAM resources for this walkthrough. For example, later on you may want to create more pipelines for use with AWS OpsWorks and associate them with an existing bucket and service role.

The following table shows which settings to choose in the **Parameters** area.

I want to create an Amazon S3 bucket that will be used by the pipeline to store and manage the files to be deployed.	Leave **ArtifactBucket** set to the default value of **New**. For **ArtifactBucketName**, type the name of the new bucket. **Note** Set **ArtifactBucket** to **Existing** if you already have a bucket you want to the pipeline to use. For **ArtifactBucket-Name**, type the name of the bucket. If you use an existing bucket, the bucket must have versioning enabled and a specific bucket policy attached. For more information, see Create a Policy for an Amazon S3 Bucket to Use as the Artifact Store for AWS CodePipeline in the AWS CodePipeline User Guide.
I want to create an AWS CodePipeline service role for the pipeline to use to interact with dependent AWS services.	Leave **ServiceRole** set to the default value of **New**. **Note** Set **ServiceRole** to **Existing** if you already have an AWS CodePipeline service role you want the pipeline to use. For **ServiceRoleName**, type the Amazon Resource Name (ARN) of the service role. The service role must be compatible with AWS CodePipeline. For more information, see Edit a Policy for an IAM Service Role in the AWS CodePipeline User Guide.

2. When **CREATE_COMPLETE** is displayed for **Status**, open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
3. In the AWS region selector, choose **US East (N. Virginia)**.
4. On the **All Pipelines** page, choose the new pipeline.
5. When **Succeeded** is displayed for the **Source** and **Deploy** stages, go to Step 8.3: Verify AWS OpsWorks Results (p. 38).

Step 8: Verify the Deployment Results

In this step, you will verify AWS CodePipeline deployed the source code in AWS CodeCommit to your deployment target.

Choose the link that corresponds to the pipeline you created in Step 7: Create and Run the Pipeline (p. 30).

Topics

Step 8.1: Verify the Results for AWS CodeDeploy

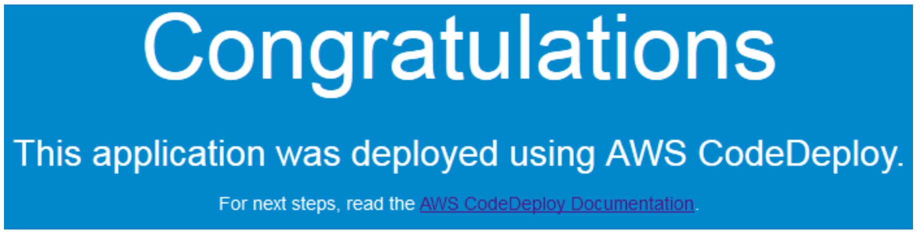
In this step, you will verify AWS CodePipeline deployed the source code in AWS CodeCommit to your deployment target in AWS CodeDeploy.

If you want to verify the results for Elastic Beanstalk, go to Step 8.2: Verify Elastic Beanstalk Results (p. 37).

If you want to verify the results for AWS OpsWorks, go to Step 8.3: Verify AWS OpsWorks Results (p. 38).

To verify the results for AWS CodeDeploy

1. After **Succeeded** is displayed for the **Source** and **Deploy** stages of the pipeline, open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the AWS region selector, choose **US East (N. Virginia)**.
3. In the list of stacks, on the **Resources** tab for the stack you created in Step 4: AWS CodeDeploy Setup (p. 19), for **CodeDeployInstance**, choose the **Physical ID** link.
4. In the Amazon EC2 console, on the **Description** tab, make a note of the **Public IP** value.
5. In a web browser, type `http://` followed by the **Public IP** value. The following web page will be displayed.



Congratulations

This application was deployed using AWS CodeDeploy.

For next steps, read the AWS CodeDeploy Documentation.

6. Go to Step 9.1: Change AWS CodeDeploy Source Code (p. 39).

Step 8.2: Verify the Results for AWS Elastic Beanstalk

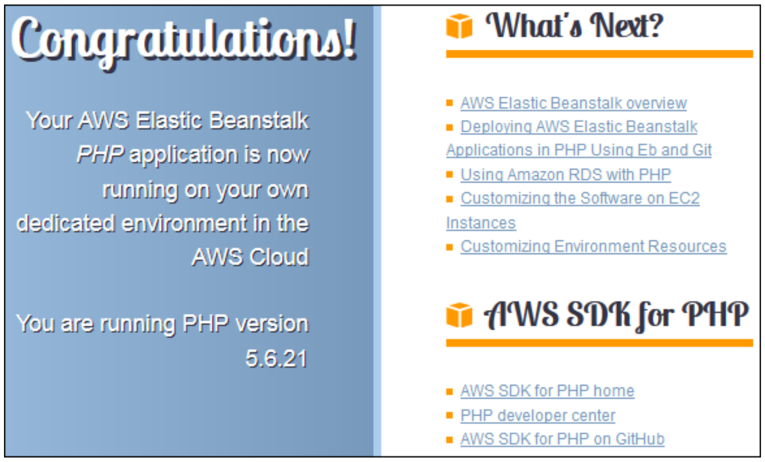
In this step, you will verify AWS CodePipeline deployed the source code in AWS CodeCommit to your deployment target in Elastic Beanstalk.

If you want to verify the results for AWS CodeDeploy, go to Step 8.1: Verify AWS CodeDeploy Results (p. 36).

If you want to verify the results for AWS OpsWorks, go to Step 8.3: Verify AWS OpsWorks Results (p. 38).

To verify the results for Elastic Beanstalk

1. After **Succeeded** is displayed for the **Source** and **Deploy** stages of the pipeline, open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the AWS region selector, choose **US East (N. Virginia)**.
3. In the list of stacks, on the **Resources** tab for the stack you created in Step 5: Elastic Beanstalk Setup (p. 23), for **ElasticBeanstalkEnvironment**, choose the **Physical ID** link.
4. In the Elastic Beanstalk console, choose the link to the newly created environment.
5. On the environment overview page, choose the **URL** link.
6. The following web page will be displayed in your web browser.



Congratulations!

Your AWS Elastic Beanstalk *PHP* application is now running on your own dedicated environment in the AWS Cloud

You are running PHP version 5.6.21

What's Next?

- AWS Elastic Beanstalk overview
- Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git
- Using Amazon RDS with PHP
- Customizing the Software on EC2 Instances
- Customizing Environment Resources

AWS SDK for PHP

- AWS SDK for PHP home
- PHP developer center
- AWS SDK for PHP on GitHub

7. Go to Step 9.2: Change Elastic Beanstalk Source Code (p. 41).

Step 8.3: Verify the Results for AWS OpsWorks

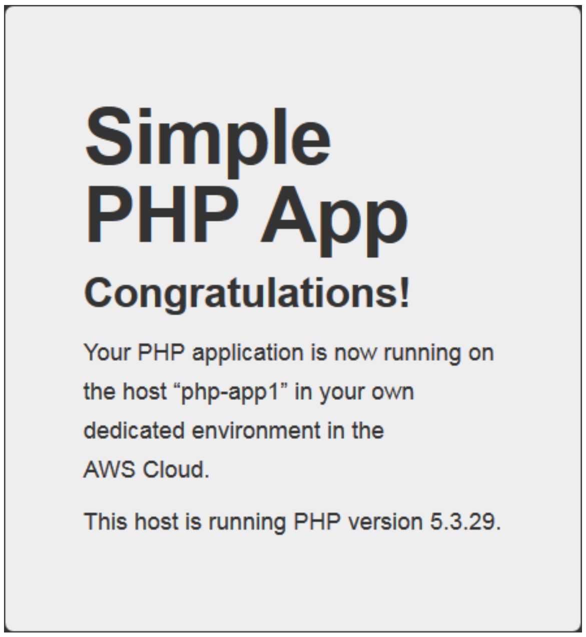
In this step, you will verify AWS CodePipeline deployed the source code in AWS CodeCommit to your deployment target in AWS OpsWorks.

If you want to verify the results for AWS CodeDeploy, go to Step 8.1: Verify AWS CodeDeploy Results (p. 36).

If you want to verify the results for Elastic Beanstalk, go to Step 8.2: Verify Elastic Beanstalk Results (p. 37).

To verify the results for AWS OpsWorks

1. After **Succeeded** is displayed for the **Source** and **Deploy** stages of the pipeline, open the AWS OpsWorks console at https://console.aws.amazon.com/opsworks.
2. On the **OpsWorks Dashboard** page, choose the name of the newly created AWS OpsWorks stack.
3. Choose **Instances**.
4. Choose the **Public IP** link.
5. The following web page will be displayed in your web browser.



Simple PHP App

Congratulations!

Your PHP application is now running on the host “php-app1” in your own dedicated environment in the AWS Cloud.

This host is running PHP version 5.3.29.

6. Go to Step 9.3: Change AWS OpsWorks Source Code (p. 42).

Step 9: Change the Source Code

In this step, you will change the source code in AWS CodeCommit. As soon as you push the code to the repository, AWS CodePipeline will deploy it to the deployment target.

Choose the link that corresponds to the source code you downloaded in Step 3: Download and Push the Source Code (p. 13).

Topics

Step 9.1: Change the Source Code for AWS CodeDeploy

In this step, you will change the source code in AWS CodeCommit. As soon as you push the code to the repository, AWS CodePipeline will deploy it to the deployment target in AWS CodeDeploy.

To change the source code for Elastic Beanstalk, go to Step 9.2: Change Elastic Beanstalk Source Code (p. 41).

To change the source code for AWS OpsWorks, go to Step 9.3: Change AWS OpsWorks Source Code (p. 42).

To change the source code for AWS CodeDeploy

1. You should already be logged in to the instance you launched in Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository (p. 10). If not, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
2. From the command prompt on the instance, run the `pwd` command to confirm you are in the `/home/ec2-user/my-demo-repo` directory. If a path other than `/home/ec2-user/my-demo-repo` appears in the output, run the `cd ~/my-demo-repo` command.
3. From the `~/my-demo-repo` directory, run the `vi index.html` command to open the `index.html` file in the vi editor.
4. Change the following line of code:

```
background-color: #0188cc
```

to:

```
background-color: #66cc00
```

This will change the background color of the page from blue to green.

Note
If you are unfamiliar with the vi editor, you can make the change by typing `:%s/#0188cc/#66cc00` and pressing Enter.

5. Change the following line of code:

```
<h2>This application was deployed using AWS CodeDeploy</h2>
```

to:

```
<h2>This application was redeployed using AWS CodeDeploy</h2>
```

This will change the text that appears on the web page.

Note
If you are unfamiliar with the vi editor, you can make the change by typing `:%s/deployed/redeployed` and pressing Enter.

6. To save your changes, type the `:wq` command, and then press Enter.
7. Run the following Git commands, one at a time, to add, commit, and then push the changed code to the AWS CodeCommit repository.

```
git add .
```

```
git commit -m "Changed source code file"
```

```
git push
```

8. After the push is successful, open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
9. In the AWS region selector, choose **US East (N. Virginia)**.
10. If the pipeline is not displayed, then on the **All Pipelines** page, choose the name of the pipeline you created in Step 7.1: AWS CodeDeploy Pipeline (p. 30).
11. When **Succeeded** is displayed for the **Source** and **Deploy** stages, go to Step 10.1: Verify AWS CodeDeploy Changes (p. 45). (It may take several minutes for AWS CodePipeline to detect the push to the repository.)

Step 9.2: Change the Source Code for AWS Elastic Beanstalk

In this step, you will change the source code in AWS CodeCommit. As soon as you push the code to the repository, AWS CodePipeline will deploy it to the deployment target in Elastic Beanstalk.

To change the source code for AWS CodeDeploy, go to Step 9.1: Change AWS CodeDeploy Source Code (p. 39).

To change the source code for AWS OpsWorks, go to Step 9.3: Change AWS OpsWorks Source Code (p. 42).

To change the source code for Elastic Beanstalk

1. You should already be logged in to the instance you launched in Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository (p. 10). If not, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
2. From the command prompt on the instance, run the `pwd` command to confirm you are in the `/home/ec2-user/my-demo-repo` directory. If a path other than `/home/ec2-user/my-demo-repo` appears in the output, run the `cd ~/my-demo-repo` command.
3. From the `~/my-demo-repo` directory, run the `vi styles.css` command to open the `styles.css` file in the vi editor.
4. Change line 36 of `styles.css`:

   ```
   background-color: #fff
   ```

   to:

   ```
   background-color: #0e0
   ```

   This will change the background color of the word **Congratulations!** on the `index.php` page from white to green.

   Note
   If you are unfamiliar with the vi editor, you can make this change by typing `vi styles.css` followed by two commands. To go to line 36, type `:36`, and then press Enter. To change `#fff` on line 36 to `#0e0`, type `:s/#fff/#0e0`, and then press Enter.
5. To save your changes, type the `:wq` command, and then press Enter.
6. Run the `vi index.php` command to open the `index.php` file in the vi editor.
7. Change the following line of code:

   ```
   <p>Your AWS Elastic Beanstalk <em>PHP</em> application is now running on
   your own dedicated environment in the AWS&nbsp;Cloud</p>
   ```

   to:

   ```
   <p>Your redeployed AWS Elastic Beanstalk <em>PHP</em> application is now
   running on your own dedicated environment in the AWS&nbsp;Cloud</p>
   ```

> **Note**
> If you are unfamiliar with the vi editor, you can make the change by typing `:%s/Your/Your redeployed` and then pressing Enter.

8. To save your changes, type the `:wq` command, and then press Enter.
9. Run the following Git commands, one at a time, to add, commit, and then push the changed code to the AWS CodeCommit repository.

```
git add .
```

```
git commit -m "Changed source code files"
```

```
git push
```

10. After the push is successful, open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
11. In the AWS region selector, choose **US East (N. Virginia)**.
12. If the pipeline is not displayed, then on the **All Pipelines** page, choose the name of the pipeline you created in Step 7.2: Elastic Beanstalk Pipeline (p. 32).
13. When **Succeeded** is displayed for the **Source** and **Deploy** stages, go to Step 10.2: Verify Elastic Beanstalk Changes (p. 46). (It may take several minutes for AWS CodePipeline to detect the push to the repository.)

Step 9.3: Change the Source Code for AWS OpsWorks

In this step, you will change the source code in AWS CodeCommit. As soon as you push the code to the repository, AWS CodePipeline will deploy it to the deployment target in AWS OpsWorks.

To change the source code for AWS CodeDeploy, go to Step 9.1: Change AWS CodeDeploy Source Code (p. 39).

To change the source code for Elastic Beanstalk, go to Step 9.2: Change Elastic Beanstalk Source Code (p. 41).

To change the source code for AWS OpsWorks

1. You should already be logged in to the instance that you launched in Step 2.3: Launch an Amazon EC2 Instance to Access the AWS CodeCommit Repository (p. 10). If not, follow the instructions in Connect to Your Linux Instance in the Amazon EC2 User Guide for Linux Instances.
2. From the command prompt on the instance, run the `pwd` command to confirm you are in the `/home/ec2-user/my-demo-repo` directory. If a path other than `/home/ec2-user/my-demo-repo` appears in the output, run the `cd ~/my-demo-repo` command.
3. From the `~/my-demo-repo` directory, run the `vi assets/css/bootstrap.min.css` command to open the `bootstrap.min.css` file in the vi editor.
4. Change the following code:

```
.hero-unit{padding:60px;margin-bottom:30px;font-size:18px;font-
weight:200;line-height:30px;color:inherit;background-color:#eee;-webkit-
border-radius:6px;-moz-border-radius:6px;border-radius:6px}
```

to:

```
.hero-unit{padding:60px;margin-bottom:30px;font-size:18px;font-
weight:200;line-height:30px;color:inherit;background-color:#0e0;-webkit-
border-radius:6px;-moz-border-radius:6px;border-radius:6px}
```

This will change the background color of the page from gray to green (`background-color: #eee` to `background-color: #0e0`).

Note
The `bootstrap.min.css` file is minimized. In other words, whitespace has been removed from the file for performance reasons. For this reason, it can be difficult to find the code to be changed. If you are unfamiliar with the vi editor, you can make the change by doing the following:

1. In the `bootstrap.min.css` file, type `/.hero-unit`, and then press Enter.
2. With the cursor on the first instance of `/.hero-unit`, press the `l` key or right arrow key repeatedly to move the cursor toward `background-color:#eee` just after `.hero-unit`.
3. Continue to repeatedly press the `l` key or right arrow key to move the cursor onto the first e character immediately after `background-color:#`.
4. Press the `x` key three times to delete the `eee` characters.
5. To insert the `0e0` characters, press the `i` key to enter insert mode. Type `0e0`, and then press the Esc key to go back to command mode.

5. To save your changes, type the `:wq` command, and then press Enter.
6. Run the `vi index.php` command to open the `index.php` file in the vi editor.
7. Change the the following line of code:

```
<p>Your PHP application is now running in the host &ldquo;<?php echo geth
ostname(); ?>&rdquo; in your own dedicated environment in the
AWS&nbsp;Cloud.</p>
```

to:

```
<p>Your redeployed PHP application is now running in the host &ldquo;<?php
 echo gethostname(); ?>&rdquo; in your own dedicated environment in the
AWS&nbsp;Cloud.</p>
```

Note
If you are unfamiliar with the vi editor, you can make the change by typing the `:%s/Your/Your redeployed` command and then pressing Enter.

8. To save your changes, type the `:wq` command, and then press Enter.
9. Run the following Git commands, one at a time, to add, commit, and then push the changed code to the AWS CodeCommit repository.

```
git add .
git commit -m "Changed source code files"
```

```
git push
```

10. After the push is successful, open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline/.
11. In the AWS region selector, choose **US East (N. Virginia)**.
12. If the pipeline is not displayed, then on the **All Pipelines** page, choose the name of the pipeline you created in Step 7.3: AWS OpsWorks Pipeline (p. 34).
13. When **Succeeded** is displayed for the **Source** and **Deploy** stages, go to Step 10.3: Verify AWS OpsWorks Changes (p. 47). (It may take several minutes for AWS CodePipeline to detect the push to the repository.)

Step 10: Verify the Changes in AWS CodePipeline

In this step, you will verify that AWS CodePipeline deployed the source code changes in AWS CodeCommit to your deployment target.

Choose the link that corresponds to the source code you changed in Step 9: Change the Source Code (p. 39).

Topics

Step 10.1: Verify the Changes for AWS CodeDeploy

In this step, you will verify AWS CodePipeline deployed the source code changes in AWS CodeCommit to your deployment target in AWS CodeDeploy.

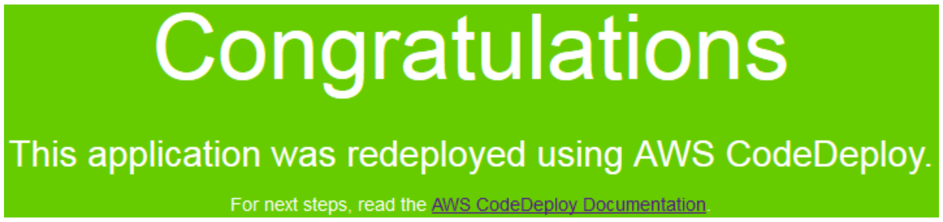
If you want to verify the results for Elastic Beanstalk, go to Step 10.2: Verify Elastic Beanstalk Changes (p. 46).

If you want to verify the results for AWS OpsWorks, go to Step 10.3: Verify AWS OpsWorks Changes (p. 47).

To verify the changes for AWS CodeDeploy

1. After **Succeeded** is displayed for the **Source** and **Deploy** stages of the pipeline, refresh the web browser tab you opened in Step 8.1: Verify AWS CodeDeploy Results (p. 36), and then skip to step 7 of this procedure. If you closed your web browser, continue with the next step of this procedure.
2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. In the AWS region selector, choose **US East (N. Virginia)**.
4. In the list of stacks, on the **Resources** tab for the stack you created in Step 4: AWS CodeDeploy Setup (p. 19), for **CodeDeployInstance**, choose the **Physical ID** link.

5. In the Amazon EC2 console, on the **Description** tab, make a note of the **Public IP** value.
6. In a web browser, type **`http://`** followed by the **Public IP** value.
7. The following web page will be displayed.



Congratulations

This application was redeployed using AWS CodeDeploy.

For next steps, read the AWS CodeDeploy Documentation.

8. Go to Step 11: Clean Up (p. 49).

Step 10.2: Verify the Changes for AWS Elastic Beanstalk

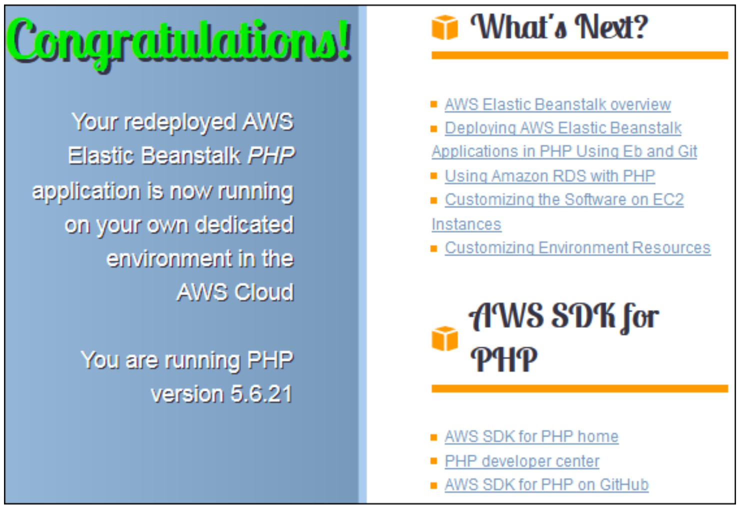
In this step, you will verify that AWS CodePipeline deployed the source code changes in AWS CodeCommit to your deployment target in Elastic Beanstalk.

If you want to verify the results for AWS CodeDeploy, go to Step 10.1: Verify AWS CodeDeploy Changes (p. 45).

If you want to verify the results for AWS OpsWorks, go to Step 10.3: Verify AWS OpsWorks Changes (p. 47).

To verify the changes for Elastic Beanstalk

1. After **Succeeded** is displayed for the **Source** and **Deploy** stages of the pipeline, refresh the web browser tab you opened in Step 8.2: Verify Elastic Beanstalk Results (p. 37), and then skip to step 7 of this procedure. If you have closed your web browser, continue to the next step of this procedure.
2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. In the AWS region selector, choose **US East (N. Virginia)**.
4. In the list of stacks, on the **Resources** tab for the stack you created in Step 5: Elastic Beanstalk Setup (p. 23), for **ElasticBeanstalkEnvironment**, choose the **Physical ID** link.
5. In the Elastic Beanstalk console, choose the link to the corresponding environment.
6. In the overview page, choose the **URL** link.
7. The following web page will be displayed in your browser.


Congratulations!

Your redeployed AWS Elastic Beanstalk *PHP* application is now running on your own dedicated environment in the AWS Cloud

You are running PHP version 5.6.21

What's Next?

- AWS Elastic Beanstalk overview
- Deploying AWS Elastic Beanstalk Applications in PHP Using Eb and Git
- Using Amazon RDS with PHP
- Customizing the Software on EC2 Instances
- Customizing Environment Resources

AWS SDK for PHP

- AWS SDK for PHP home
- PHP developer center
- AWS SDK for PHP on GitHub

8. Go to Step 11: Clean Up (p. 49).

Step 10.3: Verify the Changes for AWS OpsWorks

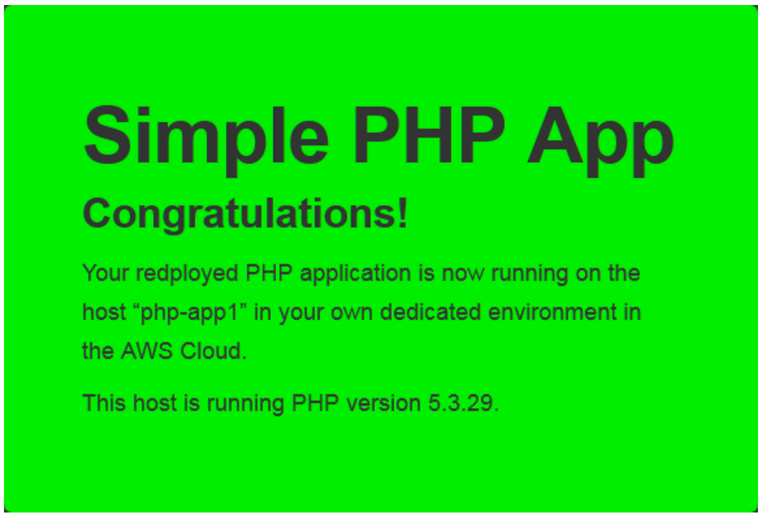
In this step, you will verify that AWS CodePipeline deployed the source code changes in AWS CodeCommit to your deployment target in AWS OpsWorks.

If you want to verify the results for AWS CodeDeploy, go to Step 10.1: Verify AWS CodeDeploy Changes (p. 45).

If you want to verify the results for Elastic Beanstalk, go to Step 10.2: Verify Elastic Beanstalk Changes (p. 46).

To verify the changes for AWS OpsWorks

1. After **Succeeded** is displayed for the **Source** and **Deploy** stages in the pipeline, refresh the web browser tab you opened in Step 8.3: Verify AWS OpsWorks Results (p. 38), and then skip to step 6 of this procedure. If you have closed your web browser, continue to the next step of this procedure.
2. Open the AWS OpsWorks console at https://console.aws.amazon.com/opsworks.
3. On the **OpsWorks Dashboard** page, choose the name of your stack.
4. Choose **Instances**.
5. Choose the **Public IP** link.
6. The following web page will be displayed in your browser.



Simple PHP App

Congratulations!

Your redployed PHP application is now running on the host “php-app1” in your own dedicated environment in the AWS Cloud.

This host is running PHP version 5.3.29.

7. Go to Step 11: Clean Up (p. 49).

Step 11: Clean Up

In this step, you will clean up the AWS resources you created for this walkthrough. This will help you avoid possible ongoing charges to your AWS account.

Note
If you do not want to delete these resources, and you are comfortable with the charges you may incur, then you have reached the end of this walkthrough. For more information, see Additional Resources (p. 53).

To clean up the AWS resources you created for this walkthrough, go to Step 11.1: Clean Up AWS CodePipeline (p. 49).

Topics

Step 11.1: Clean Up AWS CodePipeline Resources

In this step, you will clean up the AWS CodePipeline resources you created. Before you can clean up the pipeline, you must empty the Amazon S3 bucket used by the pipeline.

To empty the Amazon S3 bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the list of buckets, choose the icon next to the bucket for the pipeline (for example, **codepipeline-us-east-1-*`your-AWS-account-ID`*-artifact-bucket**).
3. Choose **Actions**, and then choose **Empty Bucket**.
4. In the **Empty bucket** dialog box, type the name of the bucket, and then choose **Empty bucket**.

To clean up other AWS CodePipeline resources

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. In the AWS region selector, choose **US East (N. Virginia)**.
3. Choose one of the following stacks from the list:
    - For AWS CodeDeploy, choose the stack you created in Step 7.1: AWS CodeDeploy Pipeline (p. 30).
    - For Elastic Beanstalk, choose the stack you created in Step 7.2: Elastic Beanstalk Pipeline (p. 32).
    - For AWS OpsWorks, choose the stack you created in Step 7.3: AWS OpsWorks Pipeline (p. 34).
4. Choose **Actions**, choose **Delete Stack**, and then choose **Yes, Delete**.
5. After the stack disappears from the list, go to one of the following:
    - For AWS CodeDeploy, go to Step 11.2: Clean Up AWS CodeDeploy (p. 50).
    - For Elastic Beanstalk, go to Step 11.3: Clean Up Elastic Beanstalk (p. 50).
    - For AWS OpsWorks, go to Step 11.4: Clean Up AWS OpsWorks (p. 51).

Step 11.2: Clean Up AWS CodeDeploy Resources

In this step, you will clean up the AWS CodeDeploy resources you created.

To clean up Elastic Beanstalk resources, go to Step 11.3: Clean Up Elastic Beanstalk (p. 50).

To clean up AWS OpsWorks resources, go to Step 11.4: Clean Up AWS OpsWorks (p. 51).

To clean up the AWS CodeDeploy resources

1. In the AWS CloudFormation console, delete the stack you created in Step 4: AWS CodeDeploy Setup (p. 19).
2. After the stack disappears from the list, go to Step 11.5: Clean Up AWS CodeCommit (p. 51).

Step 11.3: Clean Up AWS Elastic Beanstalk Resources

In this step, you will clean up the Elastic Beanstalk resources you created.

To clean up AWS CodeDeploy resources, go to Step 11.2: Clean Up AWS CodeDeploy (p. 50).

To clean up AWS OpsWorks resources, go to Step 11.4: Clean Up AWS OpsWorks (p. 51).

To clean up the Elastic Beanstalk resources

1. In the AWS CloudFormation console, delete the stack you created in Step 5: Elastic Beanstalk Setup (p. 23).

> **Note**
> When AWS CloudFormation deletes DevOpsElasticBeanstalkStack, it also deletes the associated stack named awseb-e-*`random-ID`*-stack.

2. After the two stacks disappear from the list, go to Step 11.5: Clean Up AWS CodeCommit (p. 51).

Step 11.4: Clean Up AWS OpsWorks Resources

In this step, you will clean up the AWS OpsWorks resources you created.

To clean up AWS CodeDeploy resources, go to Step 11.2: Clean Up AWS CodeDeploy (p. 50).

To clean up Elastic Beanstalk resources, go to Step 11.3: Clean Up Elastic Beanstalk (p. 50).

To clean up the AWS OpsWorks resources

1. In the AWS CloudFormation console, delete the stack you created in Step 6: AWS OpsWorks Setup (p. 26).
2. After the stack disappears from the list, go to Step 11.5: Clean Up AWS CodeCommit (p. 51).

Step 11.5: Clean Up AWS CodeCommit Resources

In this step, you will delete the AWS CodeCommit repository you used to store the source code. You will also delete the Amazon Linux instance you used to manage a copy of the source code in the repository.

To delete the AWS CodeCommit repository

1. Open the AWS CodeCommit console at https://console.aws.amazon.com/codecommit.
2. In the AWS region selector, choose **US East (N. Virginia)**.
3. On the **Dashboard** page, choose **MyDemoRepo**.
4. Choose **Settings**.
5. Choose the **Delete repository** button.
6. In the **Delete** dialog box, type **`MyDemoRepo`**, and then choose **Delete**.

To delete the Amazon Linux instance

1. In the AWS CloudFormation console, delete the stack you created in Step 2.3: Launch an Instance (p. 10).
2. After the stack disappears from the list, go to Step 11.6: Clean Up IAM (p. 51).

Step 11.6: Clean Up IAM Resources

In this step, you will clean up the IAM resources you created.

This step deletes the user you used to sign in to the AWS Management Console. Therefore, to complete this step, you must be signed in to the AWS Management Console using the credentials of the AWS root account or an administrative IAM user in the account.

To clean up the IAM resources

1. In the AWS CloudFormation console, delete the stack you created in Step 1.2: Create IAM Resources (p. 5).
2. After the stack disappears from the list, you are done with this walkthrough.

For more information, see Additional Resources (p. 53).

Additional Resources for AWS for DevOps

AWS CodeCommit

- Product Overview
- Product Details
- Developer Resources
- Documentation
- Product and Service Integrations
- Forum (sign-in required)

AWS CodeDeploy

- Product Overview
- Product Details
- Developer Resources
- Documentation
- Product and Service Integrations
- Forum (sign-in required)

AWS CodePipeline

- Product Overview
- Product Details
- Documentation
- Product and Service Integrations
- Forum (sign-in required)

AWS Elastic Beanstalk

- Product Overview
- Product Details
- Developer Resources
- Documentation
- Supported Platforms
- Forum (sign-in required)

AWS OpsWorks

- Product Overview
- Product Details
- Developer Resources
- Documentation
- Using AWS OpsWorks with Other AWS Services
- Forum (sign-in required)

General

- DevOps and AWS
- AWS DevOps Blog
- Tools for Amazon Web Services
- Start Developing with Amazon Web Services
- AWS Training and Certification
- AWS User Groups
- AWS Partner Directory

Document History for the AWS for DevOps Walkthrough

The following table describes the documentation for this release of the AWS for DevOps Walkthrough.

- **Latest documentation update:** July 22, 2016

Change	Description	Date
Initial release	This is the initial release of the AWS for DevOps Walkthrough.	July 22, 2016

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.